Deep Reinforcement Learning Driven UAV-assisted Edge Computing

Liang Zhang, *Member, IEEE*, Bijan Jabbari, *Fellow, IEEE*, and Nirwan Ansari, *Fellow, IEEE*

***Abstract*—Unmanned aerial vehicles (*UAV*s) are playing a critical role in provisioning instant connectivity and computational needs of Internet of Things devices (*IoTD*s), especially in crisis and disaster management. In this work, we focus on optimizing trajectories of UAVs along which IoTDs are served with communication and computing resources in multiple time slots. The quality of experience (*QoE*) of an IoTD depends on its latency performance; we thus aim to maximize the average aggregate QoE of all IoTDs over all time slots. However, this is a non-convex, nonlinear and mixed discrete optimization problem, which is difficult to solve and obtain the optimal solution. We thus propose two deep reinforcement learning algorithms to solve this problem by considering UAV path planning, user assignment, bandwidth and computing resource assignment. We compare the performance of our proposed algorithms through simulations with three baseline cases: 1) with fixed UAV locations, 2) without UAVs, and 3) the fixed UAV trajectories. We demonstrate that the deep reinforcement learning algorithms perform better than all baseline cases.**

***Index Terms*—Machine learning, unmanned aerial vehicle (*UAV*), edge computing, computation offloading, path planning, joint optimization.**

I. Introduction

Advances in Internet of Things (*IoT*) are empowering many emerging applications such as autonomous driving vehicles, virtual and augmented reality, face recognition, video processing, and online gaming, which all require low latency and perhaps large computing resources [1], [2]. Since many IoT devices (*IoTD*s) have limited power, computing and storage resources, mobile edge computing (*MEC*) has been proposed to alleviate the mismatch of the computing resource and reduce the latency by deploying computing and storage facilities at the cellular edge to serve IoTDs [3], [4]. While MEC may reduce the long round-trip communication delay, it presents challenges to network operations in terms of the computing resource assignment and the inherent small scale of computing facility as compared to the cloud [4].

Unmanned aerial vehicles (*UAV*s) with their high agility and mobility are becoming increasingly important in providing communications and edge-computing facilities on board to serve IoTDs, especially for the rapid deployment and disaster service recovery. UAVs have been widely studied in wireless networks owing to their agility, high mobility and flexible deployment features [5]. Moreover, the performance and the coverage area of the network can be improved and enlarged by multi-UAVs cooperation [6]. Traditional MEC cannot handle scenarios in which the network resources are sparsely distributed or the network workload fluctuates greatly while UAV-aided MEC can tackle these challenges due to the unique characteristics of UAVs [7], [8].

Many works about MEC/UAV-aided MEC have been studied. A UAV with computing facility was proposed to provide service to users in the coverage of a damaged base station with the objective of optimizing the total weighted cost of the energy consumption and the latency of users in [9]. References [10]–[12] studied UAV-aided MEC; the objectives are to minimize the processing delay of users, optimize the energy consumption of a UAV and minimize the average latency of users, respectively. Yao and Ansari [13] minimized the average task executing time of users in a fog IoT network. Peng and Sheng [14] proposed a multi-agent deep reinforcement learning algorithm to solve the joint spectrum resource, computing resource, and caching resource management problem in a UAV-assisted hotspot. Zhang *et al.* [15] studied how to extend the service time of a UAV while the service time may be longer than its maximum flight duration, and they proposed an algorithm with two-UAV substitution and another joint trajectory optimization and power control algorithm to overcome this challenge. Zhao *et al.* [16] designed a distributed MEC framework and studied user task offloading with the target to optimize the throughput and the service level of the network. Chen *et al.* [5] studied the joint base station association and UAV trajectory design problem to optimize the completion time in providing communication service to users, and proposed a deep reinforcement learning algorithm to obtain the optimal trajectory. Hu *et al.* [17] studied the trajectory problem of one UAV in a hotspot to minimize the maximum delay of all the users. Yang *et al.* [18] investigated the computing offloading problem in a multi-UAV-assisted MEC system for IoT devices; the target is to optimize the deployment of UAVs and task scheduling for UAVs; the sub-optimal user association is achieved by converting the studied problem into the generalized assignment problem and a deep reinforcement learning is also proposed to increase the task execution efficiency.

Although numerous works have been reported in UAV-aided edge computing in the literature, there are limited works that studied edge computing in a UAV-aided hierarchical network, where each user can tap on the computing resource from the UAV or the base station (*BS*) in each time slot and UAVs

This work was supported in part by National Science Foundation under Grant NSF OAC-2029221 and CNS-1814748.

Liang Zhang and Bijan Jabbari are with the Communications and Networks Laboratory, Department of Electrical and Computing Engineering, George Mason University, Fairfax, VA 22030 USA (email: lzhang36@gmu.edu; bjabbari@gmu.edu).

Nirwan Ansari is with the Advanced Networking Laboratory, Department of Electrical and Computing Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (email: nirwan.ansari@njit.edu).

can also work as relay nodes between users and the BS. In this work, we propose a UAV-aided hierarchical network where each edge node (UAV/BS) is equipped with both communication and computing facilities, each UAV acts as an edge node as well as a relay node, and each user has a task to be served by an edge node directly or by the BS via a UAV. We consider multiple time slots; a UAV may move to a different location in different time slots but constrained by its maximum reachable distance. Here, we study the uplink communications and each user needs to transmit its data to an edge node and obtain the computing service from this edge node in each time slot.

We then formulate the joint Resource Allocation and UAV Trajectory planning for Edge computing (*RATE*) problem in the proposed UAV-aided hierarchical network to maximize the average total quality-of-experience (*QoE*) of all users over all time slots. However, unlike the case of a single time slot, the case of multiple time slots becomes extremely difficult to analyze due to the dynamic nature of the problem.

It is very challenging to solve the RATE problem because 1) the position of each UAV needs to be determined in every time slot, and this location is bounded by the position of the neighboring time slot and the flying speed of a UAV; 2) an edge node is required to serve an IoTD and one IoTD can only be served by one edge node in each time slot; 3) the served QoE level of an IoTD needs to be determined in each time slot; 4) the communication and computing resources need to be assigned to IoTDs in each time slot; note that the requirement of each IoTD may vary in different time slots; 5) the connection between an edge node and an IoTD is unique, and this connection varies in different time slots because of different positions of UAVs.

The main contributions of this work are summarized below.

- We study the IoTD service provisioning problem in a UAV-aided hierarchical network and present a QoE model that can map different latency of each IoTD to different QoE.
- We formulate the RATE problem in the UAV-aided hierarchical network to maximize the average total QoE of all users over all time slots.
- We design an algorithm to obtain the optimal resource assignment based on given locations of UAVs.
- We propose two deep reinforcement learning methods to solve the UAV trajectory problem.
- The sub-optimal solution for the RATE problem is obtained based on the solution for the resource assignment and the UAV trajectory.
- The performance of the proposed deep reinforcement learning algorithms is evaluated via extensive simulations and compared with three baseline algorithms.

II. System Model

Figure 1 shows a UAV-assisted edge computing network, which consists of UAVs, BS, and IoTDs. The BS and UAVs are equipped with communications resources as well as the computing facilities, and a UAV can also work as a relay node for the communications between an IoTD and the BS. This work focuses on the uplink communications from IoTDs

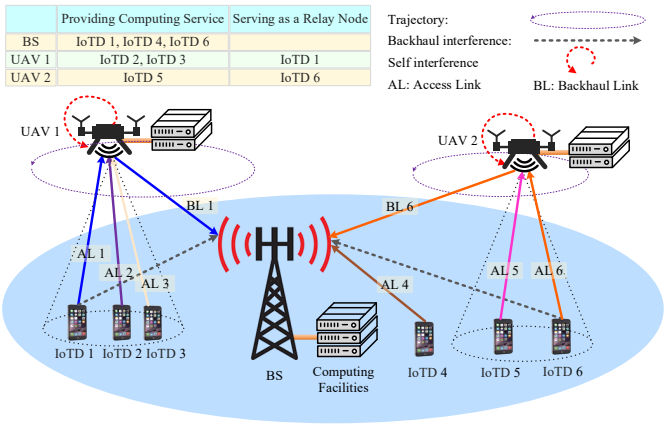


Fig. 1: A UAV-aided edge computing network.

to edge nodes as well as obtaining computing services from edge nodes. Each IoTD has a task to be executed by an edge node (a UAV or the BS), and it can obtain the computing service from the edge nodes directly or from the BS via a relay node. All IoTDs and edge nodes share the radio resources in an OFDM manner; each IoTD as well as each edge node uses unique frequency spectra for communications. Assume UAVs are full-duplex enabled and IoTDs are half-duplex enabled, implying that UAVs can utilize the frequency spectra for reception and transmission. When an IoTD receives the service from the BS via a UAV, the BS is susceptible to the backhaul interference and the UAV is subject to the self interference (*SI*). For example, IoTD 1 and IoTD 6 are provisioned by the BS via relay nodes UAV 1 and UAV 2; IoTD 1 and IoTD 6 are assigned with different frequency spectra for communications; the same frequency spectra are used from IoTD 1 to UAV 1 and from UAV 1 to the BS; IoTD 1 incurs the backhaul interference to the BS and the SI exists in UAV 1 because the same frequency spectra are used for transmission and reception. Since we focus on uplink communications, the bits of data of the task of an IoTD needs to be transmitted to an edge node; the edge node is required to provide computing service to the IoTD and the computation result may need to return to the IoTD. Here, we assume the computation result is small and negligible in our work.

Let $\mathscr{B}$ and $\mathscr{U}$ be the set of edge nodes and the set of IoTDs. The task of IoTD i to be executed in time slot t is denoted as $u_i(t) = \{d_i(t), c_i(t), L_i^{th}(t)\}$, where $d_i(t)$, $c_i(t)$ and $L_i^{th}(t)$ are the bits of data requirement, the required computation resource, and the delay threshold.

Assume $q_i(t)$ is the received QoE of IoTD i at time slot t. In our conference publication [19], $q_i(t)$ was defined as $d_i(t)/d^{total}(t)$ when the delay of IoTD i is smaller than the delay threshold $T_i^{th}(t)$; otherwise, it is 0. Note that the latency performance is crucial to the edge network and the latency of the IoTDs can be mapped to different QoE such as various levels of satisfaction of the service, e.g., excellent, fair, poor, and dissatisfied [20]. In order to better evaluate the latency performance of the IoTDs, we define four levels of service: excellent, fair, poor and dissatisfied (not served), implying that each $q_{i,j}(t)$ has up to four levels of QoE based on the delay

performance, as expressed in Eq. (1) below. Here, $q_{i,j}(t)$ is the QoE of IoTD i if served by edge node j in time slot t; $\tau_{i,j}^{total}(t)$ is the total delay of IoTD i if served by edge node j; $T_i^f(t)$ and $T_i^{max}(t)$ are the latency threshold for the fair service and the maximum latency threshold; $0 < \mu_k \leq 1,\ k = 1, 2, 3$, are constants, which represent the achieved QoE for the poor service, the fair service and the excellent service, respectively. For example, μ_1, μ_2 and μ_3 can be set as 1/3, 2/3, and 1, respectively.

$$q_{i,j}(t) = \begin{cases} \mu_3, & \tau_{i,j}^{total}(t) \leq T_i^{th}(t), \\ \mu_2, & T_i^{th}(t) < \tau_{i,j}^{total}(t) \leq T_i^f(t), \\ \mu_1, & T_i^f(t) < \tau_{i,j}^{total}(t) \leq T_i^{max}(t), \\ 0, & \tau_{i,j}^{total}(t) > T_i^{max}(t). \end{cases} \tag{1}$$

A. Communications Model

Assume $r_{i,j}$, $r_{i,j}^O$, $r_{i,j}^R$ are the received data rate, the one-hop data rate and the two-hop (relay) data rate of IoTD i towards edge node j, respectively. Then, the relationship among $r_{i,j}$, $r_{i,j}^O$ and $r_{i,j}^R$ are described in Eqs. (2)–(3):

$$r_{i,j} = \begin{cases} \max(r_{i,j}^O, r_{i,j}^R), & \forall i \in \mathcal{U}, j = 1, \\ r_{i,j}^O, & \forall i \in \mathcal{U}, j \in \mathscr{B}, j > 1. \end{cases} \tag{2}$$

$$\begin{cases} r_{i,j}^O = f(\beta_{i,j}, s_{i,j}), & \forall i, j = 1, \\ f(\beta_{i,j}, s_{i,j}) = \beta_0 \beta_{i,j} \log_2(1 + s_{i,j}) & \\ r_{i,j}^R = \min\{r_{i,j'}^{OA}, r_{j',1}^{OR}\}, & \forall i, j = 1. \end{cases} \tag{3}$$

Here, $f(\cdot)$ represents the function to calculate the data rate; $s_{i,j}$ and $\beta_{i,j}$ represent the signal to interference plus noise ratio (*SINR*) and the assigned bandwidth of IoTD i towards edge node j, as expressed in Eq. (4), respectively; β_0 stands for the bandwidth of one resource block (*RB*); $r_{i,j'}^{OA}$ and $r_{j',1}^{OR}$ are defined as the data rate of the access link (from IoTD i to UAVj') and the backhaul link (from UAV j' to the BS), respectively.

Assume P^I and P^U are the transmission power of an IoTD and a UAV, $\Gamma_{i,j}^G$ is the path loss between between IoTD i and the BS, and $\Gamma_{i,j}$ is the path loss between between IoTD i and UAV j, which consists of the line-of-sight (*LoS*) path loss and the non-line-of-sight (*NLoS*) path loss [21], [22]. Then, $s_{i,j}$ is computed as:

$$s_{i,j} = \begin{cases} \frac{P^I \Gamma_{i,j}^G}{\sigma_{i,j}^2}, & \forall i, j = 1, \\ \frac{P^I \Gamma_{i,j}}{\sigma_{i,j}^2}, & \forall i, j > 1. \end{cases} \tag{4}$$

Here, $\sigma_{i,j}^2 = \beta_0 \beta_{i,j} N_0$ is the thermal noise power and N_0 is the thermal noise power spectral density.

The path loss between a UAV and an IoTD [21], [22] is defined as:

$$\Gamma_{i,j} = \phi_{i,j}^L \zeta^L + \phi_{i,j}^N \zeta^N + \Gamma_0, \qquad \forall i, j > 1. \tag{5}$$

Here, $\phi_{i,j}^L$ and $\phi_{i,j}^N$ stand for the line-of-sight (*LoS*) path loss and non-line-of-sight (*NLoS*) path loss ($\phi_{i,j}^L + \phi_{i,j}^N = 1$), respectively; ζ^L and ζ^N are the extra LoS and NLoS path loss, respectively; $\Gamma_0 = 20\log(4\pi\varsigma_1 d_{i,j}/\varsigma_0)$ is the free space path loss, ς_1 is the carrier frequency and ς_0 is the speed of light [21]–[23]. Note that $\phi_{i,j}^L = [1 + \zeta_1 \exp(-\zeta_2(\frac{180\Omega_{i,j}}{\pi} - \zeta_1)]^{-1}$; ζ_1 and ζ_2 are environment parameters and $\Omega_{i,j} = \arctan(\frac{h_j}{d_{i,j}})$ [21]–[23].

Assume $s_{i,j'}^R$ and $s_{j',1}^R$ are the SINR of the access link of IoTD i towards edge node j' and the backhaul link from edge node j' to the BS for two-hop communications, respectively. Here, UAV j' incurs SI owing to the same frequency spectra utilized for transmission and reception; the BS is susceptible to the backhaul interference as IoTD i and UAV j' both use the same frequency spectra for communications. Then, $r_{i,j'}^{OA}$ and $r_{j',1}^{OR}$ are calculated according to Eqs. (6)–(7).

$$r_{i,j'}^{OA} = f(\beta_{i,j'}, s_{i,j'}^R), \qquad \forall i, j' > 1. \tag{6}$$

$$r_{j',1}^{OR} = f(\beta_{j',1}, s_{j',1}^R), \qquad \forall i, j' > 1. \tag{7}$$

Note that $\beta_{j',1} = \beta_{i,j'}$. Then, $s_{i,j'}^R$ and $s_{j',1}^R$ are expressed as Eqs. (8)–(9):

$$s_{i,j'}^R = \frac{P^I \Gamma_{i,j'}}{p_{j',1}/G_0 + \sigma_{i,j'}^2}, \quad \forall i \in \mathcal{U}, j' \in \mathscr{B}, j' > 1. \tag{8}$$

$$s_{j',1}^R = \frac{p_{j',1} \Gamma_{j',1}^{'}}{P^I \Gamma_{i,1} + \sigma_{j',1}^2}, \quad \forall i \in \mathcal{U}, j' \in \mathscr{B}, j' > 1. \tag{9}$$

Here, $p_{j',1}$ is the power required for transmitting data from UAV j' to the BS; $p_{j',1}/G_0$ is the SI; G_0 is the SI cancellation

TABLE I: Important Notations and Variables

Symbol	Definition
$\mathcal{U}$	set of IoTDs.
$\mathcal{T}$	set of time slots.
$\mathscr{B}$	set of edge nodes, including the BS and all UAVs.
$d_i(t)$	data requirement of IoTD i in time slot t.
$c_i(t)$	computing requirement of IoTD i in time slot t.
$q_{i,j}(t)$	obtained QoE of IoTD i when served by edge node j in time slot t.
$q_i(t)$	achieved QoE of IoTD i in time slot t.
$T_i^{th}(t)$	latency threshold for excellent service of IoTD i.
$T_i^{fair}(t)$	latency threshold for fair service of IoTD i.
$T_i^{max}(t)$	maximum latency threshold of IoTD i.
$d^{total}(t)$	data rate requirement of all IoTDs in time slot t.
P^U	maximum transmission power of a UAV.
P^I	maximum transmission power of an IoTD.
C_j	maximum computing resource of edge node j.
$r_{i,j}(t)$	data rate of IoTD i towards edge node j in time slot t.
β_j^{max}	total bandwidth of edge node j in terms of RBs.
$\tau_{i,j}^T(t)$	transmission time of edge node j to serve IoTD i in time slot t.
$\tau_{i,j}^C(t)$	computing time of edge node j to serve IoTD i in time slot t.
Θ	set of candidate positions for UAVs in the horizontal plane.
Δt	time duration of a time slot.
$\omega_{i,j}(t)$	IoTD-Edge Node indicator in time slot t.
$\alpha_{i,j}(t)$	used computing resource of IoTD i in edge node j in time slot t.
$\beta_{i,j}(t)$	used bandwidth of IoTD i in edge node j in time slot t.
$v_j(t)$	position of UAV j in the horizontal plane in time slot t, $j > 1$.
h_j	position of the jth BS in the vertical plane, $j > 1$.

capability [24], [25]; $\Gamma_{j',1}$ is the path loss between UAV j' and the BS; $\sigma^2_{i,j'}$ and $\sigma^2_{j',1}$ are the thermal noise power.

B. Computing Model

Each IoTD has one task to be served by an edge node. We assume the maximum number of edge nodes used to serve one task is one and no partial offloading is allowed. An IoTD can receive the service directly from UAVs and the BS, or it can receive the service from the BS via a UAV. Let $\tau^C_{i,j}(t)$ be the computing time of the task from IoTD i to edge node j in time slot t. Then, the required computing time $\tau^C_{i,j}(t)$ is given by:

$$\tau^C_{i,j}(t) = \frac{c_i(t)}{\alpha_{i,j}(t)}, \forall i \in \mathscr{U}, j \in \mathscr{B}. \tag{10}$$

Assume $\tau^T_{i,j}(t)$ is the transmission delay for the task from IoTD i to edge node j in time slot t, which can be obtained by $\tau^T_{i,j}(t) = d_i(t)/r_{i,j}(t)$. Then, the total delay is calculated as $\tau^{total}_{i,j}(t) = \tau^C_{i,j}(t) + \tau^T_{i,j}(t)$.

III. Problem formulation

Let $\omega_{i,j}(t)$ be the IoTD-edge-node assignment indicator and it equals to 1 if IoTD i is served by edge node j in time slot t; otherwise, it is 0. Let $\alpha_{i,j}(t)$ be the assigned computing resource to IoTD i by edge node j in time slot t. Denote $q_i(t)$ as the achieved QoE of IoTD i in time slot t, $q_i(t) = \sum_j \omega_{i,j}(t) q_{i,j}(t)$. We assume the number of deployed UAVs is given and the altitudes of all UAVs are the same and fixed. Important notations and variables are summarized in Table I. To make the paper concise, the time slot t associated with variables may be dropped throughout the manuscript.

We then formulate the RATE problem to maximize the average total QoE of all IoTDs in all time slots as follows.

$$\begin{aligned}
\mathscr{P}_0 : & \max_{\omega_{i,j}(t), \alpha_{i,j}(t), \beta_{i,j}(t), v_j(t)} \quad \frac{1}{|\mathscr{T}|} \sum_i \sum_t q_i(t) \\
s.t. : & \\
& C1 : \sum_j \omega_{i,j}(t) \le 1, \quad \forall i \in \mathscr{U}, t \in \mathscr{T}, \\
& C2 : \sum_i \omega_{i,j}(t) \beta_{i,j}(t) \le \beta^{max}_j, \quad \forall j \in \mathscr{B}, \\
& C3 : \sum_i \alpha_{i,j}(t) \le C_j, \quad \forall j, \\
& C4 : v_j(t) \in \Theta, \quad \forall j \in \mathscr{B}, j > 1, \\
& C5 : v_j(t) \cap v_{j'}(t) = \varnothing, \quad \forall j, j' > 1, \\
& C6 : f(v_j(t), v_j(t+1)) \le v^{max} \Delta t, \quad j > 1, \\
& C7 : \omega_{i,j}(t) \in \{0, 1\}, \quad \forall i, j, \\
& C8 : q_i(t) = \sum_j \omega_{i,j}(t) q_{i,j}(t), \quad \forall i, j.
\end{aligned} \tag{11}$$

Here, C1 and C7 impose one IoTD to be served by no more than one edge node. C2 ensures that the total utilized frequency spectra not to exceed the frequency spectra capacity of each edge node. C3 imposes the total consumed computing resources of all IoTDs not to exceed the computing resource capacity of each edge node. C4 is the UAV placement constraint to ensure each UAV placed in a pre-defined candidate location in the horizontal plane in any time slot. C5 imposes that any two UAVs cannot be placed in one candidate location in order to avoid collision. C6 is the trajectory constraint to ensure that a UAV can reach the location of the next time slot within its maximum reachable distance. Here, $f(\cdot)$ is the function to obtain the distance between two UAVs. C8 is used to compute the QoE of each IoTD in every time slot.

IV. Analysis and Algorithms

The RATE problem is NP-hard because it is a non-convex, nonlinear and mixed discrete optimization problem even for one time slot ($|\mathscr{T}| = 1$) [26]. In this work, we first find the solution for the user association and resource allocation and then propose two deep reinforcement learning methods to obtain sub-optimal solutions for the RATE problem as machine learning is a good approach for solving the dynamic optimization problems [27], [28].

A. User Association and Resource Allocation

Assuming the locations of UAVs in all time slots ($v_j(t)$) are known, problem $\mathscr{P}_0$ can be transformed into problem $\mathscr{P}_1$.

$$\begin{aligned}
\mathscr{P}_1 : & \max_{\omega_{i,j}(t), \alpha_{i,j}(t), \beta_{i,j}(t)} \quad \frac{1}{|\mathscr{T}|} \sum_i \sum_j \sum_t q_i(t) \\
s.t. \quad & C1, C2, C3, C7, \quad in \quad \mathscr{P}_0.
\end{aligned} \tag{12}$$

Note that $|\mathscr{T}|$ is a constant; the user association and resource allocation in time slot t do not depend on those in time slot t' ($t' < t$) because the requirements of IoTDs vary in different time slots. Then, we remove $|\mathscr{T}|$ and focus on the optimization of the one time slot scenario. When the results of user association and resource allocation for one time slot are achieved, they can be easily applied to the remaining time slots. Problem $\mathscr{P}_1$ can be further transformed into problem $\mathscr{P}_2$. Here, all "(t)" are removed to differentiate the formulation between one time slot and all time slots.

$$\begin{aligned}
\mathscr{P}_2 : & \max_{\omega_{i,j}, \alpha_{i,j}, \beta_{i,j}} \quad \sum_i \sum_j \omega_{i,j} q_{i,j} \\
& C1 : \sum_j \omega_{i,j} \le 1, \quad \forall i \in \mathscr{U}, t \in \mathscr{T}, \\
& C2 : \sum_i \omega_{i,j} \beta_{i,j} \le \beta^{max}_j, \quad \forall j \in \mathscr{B}, \\
& C3 : \sum_i \alpha_{i,j} \le C_j, \quad \forall j, \\
& C4 : \omega_{i,j} \in \{0, 1\}, \quad \forall i, j.
\end{aligned} \tag{13}$$

For the communications and computing resources assignment, both $\beta_{i,j}$ and $\alpha_{i,j}$ are required to successfully receive the service; the QoE of an IoTD depends on the total latency $\tau^{total}_{i,j}$, which depends on the resource assignment $\alpha_{i,j}$ and $\beta_{i,j}$. If the QoE of an IoTD is not 1, we can increase $\beta_{i,j}$ or $\alpha_{i,j}$, or increase both $\beta_{i,j}$ and $\alpha_{i,j}$ to improve the obtained QoE. To efficiently utilize the communications and computing resources, the computing resource assignment is assumed to be proportional to the communications resource assignment, $\alpha_{i,j} = \beta_{i,j} \zeta_j$. Here, ζ_j represents the ratio of the computing

resource to the communications resource in an edge node, $\zeta_j = C_j/\beta_j^{max}$. Then, we can transform problem $\mathscr{P}_2$ into problem $\mathscr{P}_3$.

$$\begin{aligned}\mathscr{P}_3: \max_{\omega_{i,j},\beta_{i,j}} \quad & \sum_i \sum_j \omega_{i,j} q_{i,j} \\ C1: & \sum_j \omega_{i,j} \le 1, \quad \forall i \in \mathscr{U}, t \in \mathscr{T}, \\ C2: & \sum_i \omega_{i,j}\beta_{i,j} \le \beta_j^{max}, \quad \forall j \in \mathscr{B}, \\ C3: & \omega_{i,j} \in \{0,1\}, \quad \forall i,j. \end{aligned} \tag{14}$$

If an IoTD is not served, the obtained QoE is 0; otherwise, the achieved QoE is bigger than 0, which is differentiated into three levels: μ_1, μ_2 and μ_3, respectively. Assuming $\omega_{i,j}^k$ is the IoTD-Edge-node-QoE indicator and k is the indicator of the obtained QoE level, $\omega_{i,j} = \sum_k \omega_{i,j}^k$; $\omega_{i,j}^k = 1$ represents the obtained QoE of IoTD i, μ_k, being served by edge node j. Let $\beta_{i,j}^k$ represent the minimum required corresponding resources for receiving QoE μ_k of IoTD i when served by edge node j, $\beta_{i,j} = \sum_k \omega_{i,j}^k \beta_{i,j}^k$. Since the minimum required frequency spectra to obtain the three levels of services are linear, $\mu_1/\beta_{i,j}^1 = \mu_k/\beta_{i,j}^k$, $k = 1,2,3$.

A Joint communication and computing Resources Assignment (*JCA*) algorithm is then proposed to solve problem $\mathscr{P}_3$, as expressed in *Algorithm* 1. *Steps* $1-7$ initialize parameters and calculate the required resource to serve each IoTD; *Steps* $8-17$ assign the radio resource and computing resource to IoTDs; *Step* 10 determines the index of the IoTD that needs to be served; *Step* 11 obtains $\omega_{i,j}^k$, j and k that can maximize the total QoE of the network; *Steps* $12-17$ update the network information after IoTD i has been served; *Steps* $18-19$ update the parameters for output based on the service status. Here, q^{total} is the total received QoE based on the current service status and k indicates the achieved QoE level; $q_{i,j}^k$ is the obtained QoE of IoTD i served by edge node j, $q_{i,j}^k = \mu_k$.

Algorithm 1: JCA

Input : $\mathscr{B}$, $\mathscr{U}$, β_j^{max}, and C_j;
Output: $\omega_{i,j}$, $\tilde{\omega}_{m,j}$, $\beta_{i,j}$ and $\alpha_{i,j}$;
1 $\beta_j^R = \beta_j^{max}$, $\mathscr{U}^R = \mathscr{U}$ and $\mathscr{U}^S = \varnothing$;
2 **for** *edge node j in* $\mathscr{B}^R$ **do**
3 **for** *IoTD i in* $\mathscr{U}^R$ **do**
4 **for** *each QoE level k* **do**
5 calculate $\beta_{i,j}^k$;
6 set $q_{i,j}^k = \mu_k$;
7 Initialize $\beta_j^{temp} = \arg\min_{i,k} \beta_{i,j}^k$;
8 $q^{total} = 0$, $\omega_{i,j} = 0$ and $\omega_{i,j}^k = 0$;
9 **while** $(\beta_j^R \ge \beta_j^{temp})$ & $(\mathscr{U}^R \ne \varnothing)$ **do**
10 find $i = \arg\min_i \beta_{i,j}^k$, $i \in \mathscr{U}^R$;
11 get $(j,k,\omega_{i,j}^k) = \arg\max_{j,k,\omega_{i,j}^k} (q^{total} + \omega_{i,j}^k \mu_k)$, $j \in \mathscr{B}^R$;
12 set $\omega_{i,j}^k = 1$;
13 calculate $q^{total} = q^{total} + \omega_{i,j}^k \mu_k$;
14 compute $\beta_j^R = \beta_j^R - \beta_{i,j}^k$;
15 update $\beta_j^{temp} = \arg\min_{i,k} \beta_{i,j}^k$;
16 $\mathscr{U}^S = \mathscr{U}^S \cup \{i\}$ and $\mathscr{U}^R = \mathscr{U}^R \setminus \{i\}$;
17 update $\omega_{i,j} = \sum_k \omega_{i,j}^k$;
18 update $\beta_{i,j}$ based on $\omega_{i,j}^k$ and $\beta_{i,j}^k$;
19 update $\alpha_{i,j}$ according to $\beta_{i,j}$;

Theorem 1. Algorithm *1 achieves the maximum total QoE.*

Proof. 1) When all IoTDs are served and the highest QoE is received, $\omega_{i,j} = 1$ and the obtained total QoE $q^{total} = \sum_i \sum_j \omega_{i,j} q_{i,j} = \sum_i \sum_j q_{i,j} = |\mathscr{U}|$. Since $q_{i,j} \le 1$, $q^{total} \le |\mathscr{U}|$. Then, the maximum total QoE is achieved.
2) When not all IoTDs receive the highest QoE, $q^{total} = \sum_i \sum_j \omega_{i,j} q_{i,j} < |\mathscr{U}|$. In *Algorithm* 1, the required resource $\beta_{i,j}^k$ is calculated to serve each IoTD and IoTD i is determined by $i = \arg\min_i \beta_{i,j}^k$, which requires the least resource. For IoTD i and i', if $\beta_{i,j}^k < \beta_{i',j}^k$, then we have $q_{i,j}^k/\beta_{i,j}^k > q_{i',j}^k/\beta_{i',j}^k$ because $q_{i,j}^k = \mu_k = q_{i',j}^k$. That is, the IoTD requiring the least resource is served first. After that, edge node j and the obtained QoE level is determined by $(j,k,\omega_{i,j}^k) = \arg\max_{j,k,\omega_{i,j}^k} (q^{total} + \omega_{i,j}^k \mu_k)$, which ensures that the maximum QoE of IoTD i is obtained in the current iteration. Since the minimum required resource to obtain the three levels of services are linear, $q_{i,j}^1/\beta_{i,j}^1 = q_{i,j}^k/\beta_{i,j}^k$, the maximum increment of QoE is received after IoTD i is served based on the current network status, $q_{i,j} = \max_k (\omega_{i,j}^k q_{i,j}^k)$. Note that $q_{i,j} = q_{i,j}^3 = \mu_3$ if the communication and computing resources are sufficient. Here, we first determine the IoTD which requires the least resource and then find the maximum QoE level for this IoTD based on the current service status; the maximum QoE of each IoTD is the same, and more IoTDs can be served if the IoTDs with less resource requirements are chosen to be served first; then, the highest total QoE is achieved in the end. In summary, *Algorithm* 1 obtains the largest increment of the total QoE $q^{total} = \sum_i \sum_j \omega_{i,j} q_{i,j}$ in each iteration of assigning the communication and computing resources to IoTDs; the maximum total QoE is thus achieved by *Algorithm* 1. □

B. Deep Reinforcement Learning Algorithms

A typical fully connected neural network formed by neurons consists of three layers: the input layer, the hidden layer and the output layer. Each link in the neural network has a weight and the weights of all links are randomly initialized, which will be updated according to the environment. The state $s(t)$ and action $a(t)$ are the input and output of the neural network, and the reward is obtained based on the action for the environment. The weights of all links can approximate the environment after efficient training with the maximum reward.

To proceed with deep reinforcement learning, we need to define state, action and reward, as described below.
1) State: $s_t = \{X(\mathscr{U}(t)), Y(\mathscr{U}(t)), D(t), W(t), V(t)\}$. Here, $X(\mathscr{U}(t))$ and $Y(\mathscr{U}(t))$ are cartesian coordinates, which represent the locations of all IoTDs in the horizontal plane in time slot t; $D(t) = \cup\{d_i(t)\}$ is the set of data rate requirements of

all IoTDs in time slot t; $W(t) = \cup\{\omega_{i,j}(t)\}$ is the set of user association indicators of all IoTDs; $V(t) = \cup\{v_j(t)\}$ represents the positions of all UAVs in time slot t.

2) Action: $a(t) = \{X(V(t)), Y(V(t))\}$. $X(V(t))$ and $Y(V(t)$ are cartesian coordinates, which represent the target locations of UAVs in the horizontal plane in time slot t. If $|X(V(t)) - X(V(t+1))| > dis\_min$ or $|Y(V(t)) - Y(V(t+1))| > dis\_min$, we set $|X(V(t)) - X(V(t+1))| = dis\_min$ or $|Y(V(t)) - Y(V(t+1))| = dis\_min$. Here, dis_min represents the length between two neighboring candidate positions in the horizontal plane of UAVs.

3) Reward: $g(t) = \sum_i \sum_j q_{i,j}(t)$ is the summation of received QoE of all IoTDs in time slot t.

Deep Q-learning (*DQL*) is a proven algorithm, which was proposed by Google Deepmind [29] and has been popularly used by researchers. We thus leverage DQL to solve the RATE problem. DQL is a model-free reinforcement learning algorithm and employs two neural networks: the fixed neural network and the target neural network; the fixed neural network is used to estimate the Q value; the Q value is employed to generate the action and a reward is returned based on this action; the target neural network is created to help DQL converge and it is updated slowly using the same parameters from the fixed neural network; experience buffer (replay buffer), which stores previous samples, is leveraged to enhance convergence of DQL. The proposed DQL tailored to solve the RATE problem is referred to as *DQL-RATE*, as delineated as *Algorithm* 2. Here, $R(t)$ represents the number of samples in the replay buffer in time slot t and R^f stands for the time duration to update the target network; $\tilde{g}(e)$ denotes the final average total QoE of epoch e. *Steps* $1-2$ initialize the fixed network, the target network and replay buffer; *Steps* $7-13$ calculate state $s(t+1)$, action $a(t)$; *Step* 14 compute the reward $g(t)$; *Steps* $17-21$ train the fixed network and the target network; *Step* 22 obtain the average QoE of all time slots.

The fixed network is updated by minimizing the loss $(y(t) - \varphi(s(t), a(t)|\theta^{\varphi}))^2$. The loss is defined as the difference between the target Q-value and the predicted Q-value. The target network is updated by $\theta^{\varphi^-} \leftarrow \eta\theta^{\varphi} + (1-\eta)\theta^{\varphi^-}$. Here, the target value $y(t)$ is estimated as:

$$y(t) = \begin{cases} g(t), & \text{if terminates in time slot } (t+1), \\ g(t) + \gamma \max_a \varphi(s(t+1), a(t)|\theta^{\varphi^-}), & \text{otherwise.} \end{cases} \tag{15}$$

Since DQL is not easy to converge, we propose to tailor another deep reinforcement learning method named deep deterministic policy gradient (*DDPG*) [30], [31] for solving the RATE problem, and the resulting algorithm is referred to as the DDPG-RATE algorithm. DDPG is a combination of the actor-critic approach and the architecture of DQL [30], and the structure of the DDPG-RATE algorithm is shown in Fig. 2. The actor network is utilized to generate the action $a(t)$ for the environment based on the policy gradient method; the critic network is employed to estimate how good the action of the actor network is; similar to the target network of DQL, the target networks (target actor network and target critic network) are leveraged to minimize the loss function that can also improve the stability of DDPG, and they are updated slowly from the actor-critic networks. Moreover, the input of the actor network is $s(t)$ and the inputs of the critic network are $s(t)$ and $a(t)$. Note that the IoTD assignment can be obtained based on the action $a(t)$; the state $s(t)$, action $a(t)$, reward $g(t)$, and next state $s(t+1)$ constitute a sample, which is added to the replay buffer.

Algorithm 2: DQL-RATE

Input : $\mathscr{B}$, $\mathscr{U}$, $u_i(t)$, $\varphi(s(t)|\theta^{\varphi})$ and $\varphi(s(t)|\theta^{\varphi^-})$;
Output: $\tilde{g}(e)$ and $g(t)$;
1 initialize the fixed network and the target network with the weights θ^{φ} and θ^{φ^-};
2 initialize the replay buffer η;
3 **for** *epoch* e **do**
4 initialize positions of all UAVs;
5 initialize $s(t)$ and $g(t)$;
6 **for** *time slot* t **do**
7 obtain $X(\mathscr{U}(t))$, $Y(\mathscr{U}(t))$, $D(t)$, $W(t)$, and $V(t)$;
8 obtain $s(t+1)$;
9 generate a random variable $\epsilon' \in (0, 1)$;
10 **if** $\epsilon' \geq \epsilon$ **then**
11 randomly generate an action $a(t)$;
12 **else**
13 use the maximum Q-value to obtain action $a(t)$;
14 calculate $g(t)$ based on $a(t)$ and *Algorithm* 1;
15 add sample $(s(t), a(t), g(t), s(t+1))$ to replay buffer;
16 set $s(t) = s(t+1)$;
17 **if** $R(e,t) \geq R^b$ **then**
18 estimate target output y_i by Eq. (15);
19 update θ^{φ} of the fixed network by minimizing $(y(t) - \varphi(s(t), a(t)|\theta^{\varphi}))^2$;
20 **if** $R(e,t)\%R^f == 0$ **then**
21 update the target network by the weight of the neural network;
22 obtain the results $\tilde{g}(e) = \frac{1}{|\mathscr{T}|}\sum_t g(t)$;
23 return $\tilde{g}(e)$ and $g(t)$.

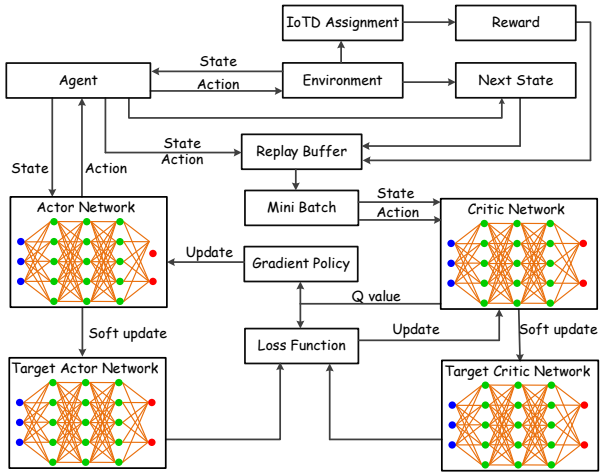


Fig. 2: The structure of the DDPG-RATE algorithm.

In the DDPG algorithm, two pairs of neural networks are utilized: the actor neural network $a(t) = \varphi(s(t)|\theta^{\varphi})$ and the critic neural network $Q(s(t), a(t)|\theta^{Q})$, the target actor

Algorithm 3: DDPG-RATE

Input : $\mathscr{B}$, $\mathscr{U}$, $u_i(t)$, $\varphi(s(t)|\theta^{\varphi})$, $Q(s(t),a(t)|\theta^{Q})$, $\varphi(s(t)|\theta^{\varphi-})$ and $Q(s(t),a(t)|\theta^{Q-})$;
Output: $\tilde{g}(e)$ and $g(t)$;
1 initialize the actor network $\varphi(s(t)|\theta^{\varphi})$ and the critic network $Q(s(t),a(t)|\theta^{Q})$ with parameters θ^{φ} and θ^{Q};
2 initialize the target actor network $\varphi(s(t)|\theta^{\varphi-})$ and the target critic network $Q(s(t),a(t)|\theta^{Q-})$ with parameters $\theta^{\varphi-}=\theta^{\varphi}$ and $\theta^{Q-}=\theta^{Q}$;
3 set $R(e,t)=0$;
4 initialize the replay buffer R;
5 **for** *epoch* e **do**
6 initialize the position of all UAVs;
7 initialize $s(t)$ and $g(t)$;
8 **for** *time slot* t **do**
9 get $X(\mathscr{U}(t))$, $Y(\mathscr{U}(t))$, $D(t)$, $W(t)$, and $V(t)$;
10 obtain $s(t+1)$;
11 generate action $a(t)$;
12 add noise to action $a(t)=a(t)+a_0$;
13 map the action $a(t)$ to positions;
14 update positions of all UAVs;
15 calculate $g(t)$ based on $a(t)$ and *Algorithm* 1;
16 add sample $(s(t),a(t),g(t),s(t+1))$ to replay buffer;
17 set $s(t)=s(t+1)$;
18 **if** $R(e,t)\geq R^{b}$ **then**
19 update θ^{Q} of the critic network by Eq. (16);
20 update θ^{φ} of the actor network by Eq. (17);
21 update target networks by Eq. (18);
22 $R(e,t)=R(e,t)+1$;
23 obtain the results $\tilde{g}(e)=\frac{1}{|\mathscr{T}|}\sum_t g(t)$;
24 return $\tilde{g}(e)$ and $g(t)$.

network $a(t)=\varphi(s(t)|\theta^{\varphi-})$ and the target critic network $Q(s(t),a(t)|\theta^{Q-})$.

For Algorithm 3, *Steps* $1-3$ initialize the actor-critic networks and the target actor-critic networks and replay buffer; *Steps* $5-15$ calculate state $s(t+1)$, action $a(t)$ and reward $g(t)$, form one sample and add the sample to the replay buffer; *Steps* $17-20$ updates the actor-critic networks and the target actor-critic networks based on Eqs. (16) – (17); *Step* 22 obtains the average QoE of all time slots.

The critic network is updated through the loss function:

$$L(\theta^{Q})=\frac{1}{M}\sum_{m=1}^{M}(A_1+\gamma A_2)^2. \tag{16}$$

Here, m is the index of a sample in the batch and M stands for the maximum number of samples in a batch, $1\leq m\leq M$; $g(m)$ is the reward of sample m; $A_1=g(m)-Q(s(m),a(m)|\theta^{Q})$; $A_2=Q(s(m+1),a(m+1)|\theta^{Q-})$, which is the output result of sample $(m+1)$ from the target critic network [31].

The actor network is updated according to the gradient policy:

$$\nabla_{\theta^{\varphi}}J(\theta^{\varphi})=\frac{1}{M}\sum_{m=1}^{M}(A_3\cdot A_4). \tag{17}$$

Here, $\nabla_{\theta}J(\theta)$ is the gradient function of $J(\theta)$; $J(\theta)$ is the policy objective function; $A_3=\nabla_{\theta^{\varphi}}\varphi(s|\theta^{\varphi})|_{s=s(m)}$ and $A_4=\nabla_a Q(s,a|\theta^{Q})\big|_{s=s(m),a=\varphi(s(m))}$.

The target networks are updated as:

$$\begin{cases}\theta^{\varphi-}\leftarrow\eta\theta^{\varphi}+(1-\eta)\theta^{\varphi-},\\ \theta^{Q-}\leftarrow\eta\theta^{Q}+(1-\eta)\theta^{Q-}.\end{cases} \tag{18}$$

Here, η is the updating rate, which is a small constant and utilized to smoothen the input.

C. Computational Complexity

Two baseline algorithms are employed to evaluate the performance of the machine learning algorithms. One is named *Fixed-UAV*, in which all UAVs are placed in the fixed locations and the best SINR strategy is employed to determine the IoTD assignment. The other one is referred to as *No-UAV*, i.e., no UAV is deployed and all IoTDs are served by the BS.

The computational complexity of the *Fixed-UAV* algorithm is $O(|\mathscr{T}|(|\mathscr{B}||\mathscr{U}|K+\log(|\mathscr{B}|+|\mathscr{U}|+K)))$ and that of the *No-UAV* algorithm is $O(|\mathscr{T}|(|\mathscr{U}|K+\log(|\mathscr{U}|+K)))$; the complexity of user association and resource allocation is $O(|\mathscr{B}||\mathscr{U}|K+\log(|\mathscr{B}|+|\mathscr{U}|+K))$ and $|\mathscr{B}|=1$ for the *No-UAV* algorithm; K is the number of QoE levels for each IoTD.

For machine learning algorithms, the computational complexity of finding actions is $O(\sum_{l=1}^{L}N_lN_{l-1})$ [31]. Here, L is the number of layers of the neural networks and N_l is the number of neurons of layer l. The complexity of the JCR algorithm (user association and resource allocation) is $O(|\mathscr{B}||\mathscr{U}|K+\log(|\mathscr{B}|+|\mathscr{U}|+K))$. The overall complexity of the machine learning algorithms is $O((\sum_{l=1}^{L}N_lN_{l-1}+|\mathscr{B}||\mathscr{U}|K+\log(|\mathscr{B}|+|\mathscr{U}|+K))\mathscr{T}E)$, where E is the number of epoches used in the training.

V. Performance Evaluation

Python 3.6 is employed to run the simulations and Tensorflow 2.6.2 (tf.keras.optimizers.Adam) is utilized to run machine learning algorithms. The coverage area is set as $500\ m\times 500\ m$, which is evenly divided into $25\times 25=625$ sub-areas. Each sub-area is $20\ m\times 20\ m$ and it is also called a block. The BS is fixed at the center of the coverage area and every UAV is placed at the center of a block. Two UAVs are deployed; the height and the maximum flying speed of each UAV are set as $30\ m$ and $30\ m/s$. In each time slot, a UAV can move to one of the neighbouring blocks in both X-axis and Y-axis in the horizontal plane in the next time slot. In other words, each UAV has 9 (3×3) candidate blocks to be placed in the next time slot. Here, constraint C6 in problem $P0$ is satisfied ($30\ m>500/25\ m$). We consider 30 time slots in the simulation and the duration of each time slot is $1\ s$. We assume the locations of IoTDs are fixed within each time slot; IoTDs are generated according to a Matérn cluster process, and the communication and computing resources requirements of IoTDs in different time slots may vary. For the communication parameters setting, $(\zeta^1,\zeta^2,\zeta^L,\zeta^N)$ is set as $(9.6,0.16,1,20)$ [21]–[23]; the SI cancellation capability G_0 is set as 130 db [24]; the path loss between the IoTD and BS (the edge node on the ground) is $131.1+42.8\log_{10}(d_{i,j})$ according to the path loss model from 3GPP [32]. For machine learning configurations, all neural networks are initialized with the same parameters: each has three layers with 800 neurons, 600

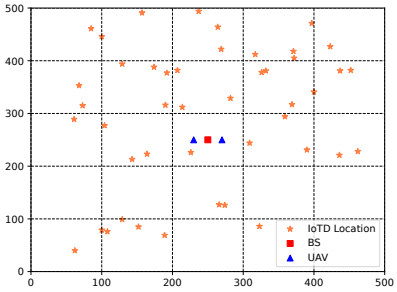


Fig. 3: A sample positions of IoTDs and edge nodes.

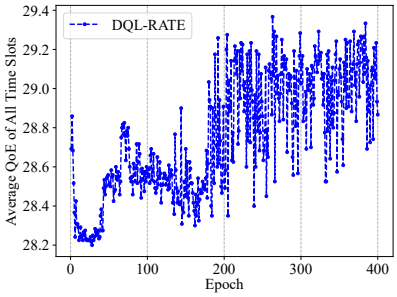


Fig. 4: Convergence results of DQL-RATE.

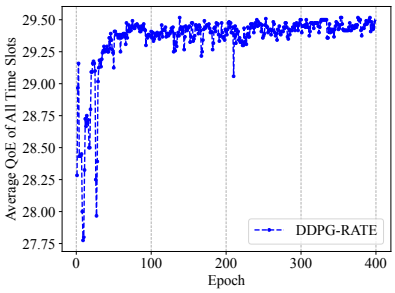


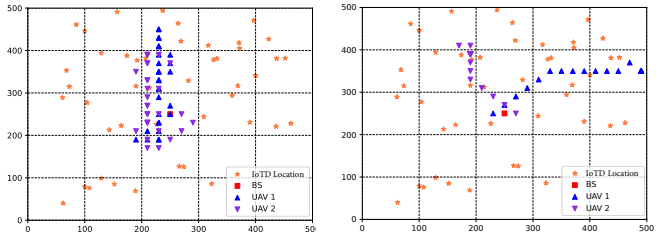
Fig. 5: Convergence results of DDPG-RATE.

neurons and 300 neurons, respectively. Important simulation parameters are summarized in Table II.

TABLE II: **Simulation Parameters**

Parameters	**Value**	**Parameters**	**Value**
coverage	$500\ m \times 500\ m$	time slot	30
$\|\mathcal{U}\|$	$\{20, 25, \cdots, 45\}$	d_i	$[0.1, 1]$ Mb
c_i	$[1, 10] \times 10^7$	β_0	180 kHz
C_0 (BS)	2×10^{10} CPU cycle/s	P^V	27 dBm
C_j (UAV)	1×10^{10} CPU cycle/s	P^E	20 dBm
v^{max}	$30\ m/s$	Δt	1s
rayleigh	8 dB	N_0	-174 dBm/Hz
β^{max}	100 RB (20 MHz)		
$\Gamma^G_{i,1}$		$131.1 + 42.8 \log_{10}(d_{i,j})$ [32]	
Parameters for DQL-RATE		**Value**	
learning rate		5×10^{-3}	
γ, discount factor		0.85	
η, soft target update		0.005	
replay buffer		10^6	
Parameters for DDPG-RATE		**Value**	
learning rate of actor		5×10^{-5}	
learning rate of critic		5×10^{-4}	
γ, discount factor		0.99	
η, soft target update		0.001	
replay buffer		10^6	

We set the latency parameters as follows: $T_i^{th}(t)$, $T_i^f(t)$, $T_i^{max}(t)$ are set as 100 ms, 200 ms and 400 ms; $\mu_1 = 0.25$, $\mu_2 = 0.5$ and $\mu_3 = 1$. Fig. 4 and Fig. 5 show the convergence results under 45 IoTDs with batch size $R^b = 32$ of the DQL-RATE algorithm and DDPG-RATE algorithm. Here, the batch is a set of samples, $(s(t), a(t), g(t), s(t+1))$, which is utilized to train neural networks; the batch size is the number of samples in a batch. Each epoch has 30 time slots; 100 epochs represent $30 \times 100 = 3000$ training steps; the average QoE of all time slots is calculated in each epoch. For both DQL-RATE algorithm and DDPG-RATE algorithm, the actions for all UAVs are generated by the machine learning agent and a sample is added to the buffer in each time slot. When the buffer is full, the oldest sample will be discarded. The machine learning agent starts to update the weights of the neural networks when the number of samples in the buffer is more than $R^b = 32$. Then, the best positions of UAVs can be achieved after efficient training based on the knowledge of the environment. The results show that the DQL-RATE algorithm and DDPG-RATE algorithm can converge after 300 and 200 epochs, respectively. The trajectories of all time slots in the last epoch of the DQL-RATE algorithm and DDPG-RATE algorithm are shown in Fig. 6(a) and Fig. 6(b).



(a) Trajectory of DQL-RATE. (b) Trajectory of DDPG-RATE.

Fig. 6: UAV trajectory results with 45 IoTDs and $R^b = 32$.

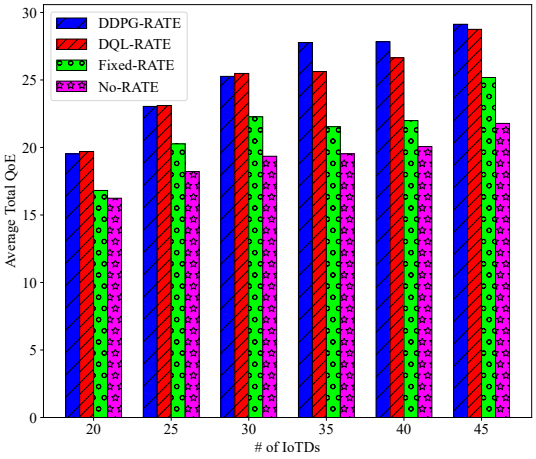


Fig. 7: Average QoE versus number of IoTDs.

Fig. 7 shows the average total QoE in all time slots versus different workloads (IoTDs). The average total QoE of all algorithms increases as the workload increases. Less communications resources are used in serving the same number of IoTDs as compared to a heavy workload scenario because it is easier to obtain IoTDs with higher weight and better SINR under a heavy workload. Then, the total communications and computing resources in all edge nodes can be utilized to serve more IoTDs, thus incurring a higher average total QoE. The average total QoE results of the DQL-RATE algorithm increase by up to 21% and 32% as compared to those of Fixed-RATE and No-RATE, while those of the DDPG-RATE

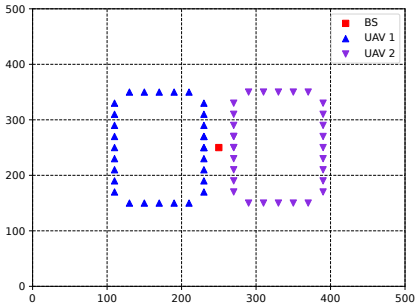


Fig. 8: Positions of UAVs of the Fixed-Trajectory algorithm.

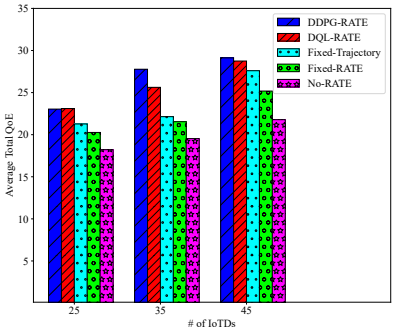


Fig. 9: Average QoE versus the number of IoTDs.

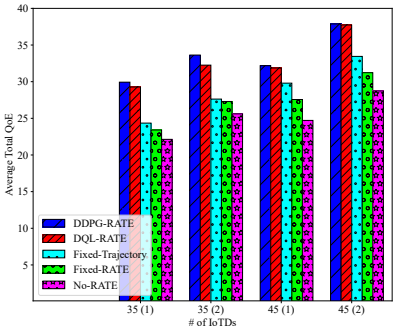


Fig. 10: Average QoE under different scales.

algorithm increase by up to 29% and 42% as compared to those of Fixed-RATE and No-RATE. The average total QoE of the Fixed-RATE algorithm is better than that of the No-RATE algorithm because more computing resources are available and UAVs also have better connection to remote IoTDs as compared to the BS.

We have added one more algorithm to evaluate the performance of the proposed algorithm. It is named *Fixed-Trajectory*, which lets each UAV fly in given positions, as shown in Fig. 8. The complexity of the Fixed-trajectory algorithm is similar to the Fix-RATE, which is $O(|\mathcal{T}||\mathcal{B}|(|\mathcal{B}||\mathcal{U}|K+\log(|\mathcal{B}|+|\mathcal{U}|+K)))$. The average QoE results of the DDPG-RATE algorithm and the DQL-RATE algorithm have up to 25.4% and 15.7% enhancement as compared to those of Fixed-Trajectory, as shown in Fig. 9. The Fixed-trajectory algorithm does not perform well because its UAV trajectories do not consider the positions and the resource requirements of IoTDs.

To better evaluate the QoE performance of the machine learning algorithms, we have considered two more cases: the scale of computing resources (case 1) and the scale of the delay requirements of QoE (case 2). For case 1, each edge node's computing capacity is increased by 50%; for case 2, all latency thresholds of achieving different levels of QoE are increased by 50%. Then, the average QoE results under different scales with 35 and 45 IoTDs are shown in Fig. 10. Here, the number in a bracket represents the case information, e.g., "35(2)" represents case 2 with 35 IoTDs. The average QoE results of all algorithms have been increased for both case 1 and case 2. Note that the QoE increment of case 2 for every algorithm is more than case 1 because case 1 only adds computing resources to the network but case 2 relaxes both communication and computing resources. For case 1, the DDPG-RATE algorithm and the DQL-RATE algorithm have achieved up to 35.2% and 32.3% improvement of the average QoE as compared to those of baseline algorithms. For case 2, the DDPG-RATE algorithm and the DQL-RATE algorithm have achieved up to 31.3% and 26% improvement of the average QoE as compared to those of baseline algorithms.

VI. Conclusion

In this work, we have studied the RATE problem with consideration of the UAV path planning, user assignment, and communication and computing resources assignment; the target is to maximize the average total QoE of all IoTDs over multiple time slots. As the RATE problem is NP-hard, two deep reinforcement learning algorithms have been proposed to achieve the sub-optimal solution. Three baseline strategies, Fixed-RATE (locations of all UAVs are fixed), No-RATE (no UAV is deployed) and Fixed-Trajectory (every UAV flies in given positions), are utilized to evaluate the performance of the deep reinforcement learning algorithms. Simulation results show that the average total QoE result of the deep reinforcement learning algorithms have increased up to 32% and 42% as compared to those of baseline strategies.

The future research directions attempt to address the following questions: 1) The current flying time of commercial UAVs is around 30 minutes or up to two hours; how does one provision seamless services in the long duration event? 2) As new applications such as AR and VR with large data rate requirements emerge, how does one satisfy these new applications in the UAV-aided MEC network? 3) Many research works focus on one dimension of resource allocation in the UAV-aided MEC networks, how does one efficiently utilize different types of computing resources (CPU, memory, storage, etc.) in the UAV-aided MEC network?

References

[1] P. Porambage *et al.*, "Survey on multi-access edge computing for internet of things realization," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2961–2991, Fourthquarter 2018.

[2] J. Ji, K. Zhu, C. Yi, and D. Niyato, "Energy consumption minimization in UAV-assisted mobile-edge computing systems: Joint resource allocation and trajectory design," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 8570–8584, May 2021.

[3] N. Ansari and X. Sun, "Mobile edge computing empowers Internet of Things," *IEICE Trans. Commun.*, vol. E101-B, no. 3, pp. 604–619, Mar. 2018.

[4] X. Cao *et al.*, "Joint computation and communication cooperation for energy-efficient mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4188–4200, Jun. 2019.

[5] Y. J. Chen and D. Y. Huang, "Joint trajectory design and BS association for cellular-connected UAV: An imitation-augmented deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 9, no. 4, pp. 2843–2858, Feb. 2022.

[6] J. Wang *et al.*, "Taking drones to the next level: Cooperative distributed unmanned-aerial-vehicular networks for small and mini drones," *IEEE Veh. Technol. Mag*, vol. 12, no. 3, pp. 73–82, Sept. 2017.

[7] J. Zhang, L. Zhou, Q. Tang, E. C. . Ngai, X. Hu, H. Zhao, and J. Wei, "Stochastic computation offloading and trajectory scheduling for UAV-assisted mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3688–3699, Apr. 2019.

[8] Z. Yu, Y. Gong, S. Gong, and Y. Guo, "Joint task offloading and resource allocation in UAV-enabled mobile edge computing," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3147–3159, Apr. 2020.

[9] K. Zhang *et al.*, "Energy-latency tradeoff for computation offloading in UAV-assisted multiaccess edge computing system," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6709–6719, Apr. 2021.

[10] S. R. Sabuj *et al.*, "Delay optimization in mobile edge computing: Cognitive UAV-assisted eMBB and mMTC services," *IEEE Trans. Cogn. Commun. Netw., early access*, Feb. 2022.

[11] C. Sun, W. Ni, and X. Wang, "Joint computation offloading and trajectory planning for UAV-assisted edge computing," *IEEE Trans. Wireless Commun*, vol. 20, no. 8, pp. 5343–5358, Mar. 2021.

[12] L. Zhang and N. Ansari, "Latency-aware IoT service provisioning in UAV-aided mobile edge computing networks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10 573–10 580, Jun. 2020.

[13] J. Yao and N. Ansari, "Task allocation in fog-aided mobile IoT by lyapunov online reinforcement learning," *IEEE Trans. Green Commun. Netw.*, vol. 4, no. 2, pp. 556–565, Jun. 2020.

[14] H. Peng and X. Shen, "Multi-agent reinforcement learning based resource management in MEC- and UAV-assisted vehicular networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 131–141, Jan. 2021.

[15] G. Zhang *et al.*, "Cooperative UAV enabled relaying systems: Joint trajectory and transmit power optimization," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 1, pp. 543–557, Mar. 2022.

[16] Z. Zhao *et al.*, "Predictive UAV base station deployment and service offloading with distributed edge learning," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 3955–3972, Dec. 2021.

[17] Q. Hu *et al.*, "Joint offloading and trajectory design for UAV-enabled mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1879–1892, Apr. 2019.

[18] L. Yang *et al.*, "Multi-UAV-enabled load-balance mobile-edge computing for IoT networks," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6898–6908, Feb. 2020.

[19] L. Zhang, B. Jabbari, and N. Ansari, "Machine learning driven UAV-assisted edge computing," in *Proc. of IEEE WCNC*, Apr. 2022, pp. 2220–2225.

[20] X.-Q. Pham, T.-D. Nguyen, V. Nguyen, and E.-N. Huh, "Joint service caching and task offloading in multi-access edge computing: A qoe-based utility optimization approach," *IEEE Commun. Lett.*, vol. 25, no. 3, pp. 965–969, Mar. 2021.

[21] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal LAP altitude for maximum coverage," *IEEE Wireless Commun. Lett.*, vol. 3, no. 6, pp. 569–572, Dec. 2014.

[22] R. I. Bor-Yaliniz, A. El-Keyi, and H. Yanikomeroglu, "Efficient 3-D placement of an aerial base station in next generation cellular networks," in *Proc.of IEEE ICC*, Jul. 2016, pp. 1–5.

[23] L. Zhang and N. Ansari, "Approximate algorithms for 3-D placement of IBFD enabled drone-mounted base-stations," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7715–7722, Aug. 2019.

[24] U. Siddique, H. Tabassum, and E. Hossain, "Downlink spectrum allocation for in-band and out-band wireless backhauling of full-duplex small cells," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3538–3554, Aug. 2017.

[25] M. S. Elbamby *et al.*, "Resource optimization and power allocation in in-band full duplex-enabled non-orthogonal multiple access networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 12, pp. 2860–2873, Dec. 2017.

[26] R. M. Nauss, "Solving the generalized assignment problem: An optimizing and heuristic approach," *INFORMS Journal on Computing*, vol. 15, no. 3, pp. 249–266, 2003.

[27] N. Rastegardoost and B. Jabbari, "A machine learning algorithm for unlicensed LTE and WiFi spectrum sharing," in *Proc. of IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2018, pp. 1–6.

[28] Z. Chang *et al.*, "Machine learning-based resource allocation for multi-UAV communications system," in *Proc. of ICC Workshops*, 2020, pp. 1–6.

[29] V. Minh *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[30] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," arXiv preprint arXiv:1509.02971, Sept. 2015.

[31] L. Wang *et al.*, "Deep reinforcement learning based dynamic trajectory control for UAV-assisted mobile edge computing," *IEEE Trans. Mobile Comput., doi:10.1109/TMC.2021.3059691*, 2021.

[32] *3GPP TR 36.828 version 11.0.0, release 11*. 3GPP Tech. Rep., Jun. 2012. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2507