Deep Reinforcement Learning Based Dynamic Trajectory Control for UAV-assisted Mobile Edge Computing

Liang Wang, Kezhi Wang, Cunhua Pan, Wei Xu, Nauman Aslam and Arumugam Nallanathan, *Fellow, IEEE*

***Abstract*—In this paper, we consider a platform of flying mobile edge computing (F-MEC), where unmanned aerial vehicles (UAVs) serve as equipment providing computation resource, and they enable task offloading from user equipment (UE). We aim to minimize energy consumption of all the UEs via optimizing the user association, resource allocation and the trajectory of UAVs. To this end, we first propose a Convex optimizAtion based Trajectory control algorithm (CAT), which solves the problem in an iterative way by using block coordinate descent (BCD) method. Then, to make the real-time decision while taking into account the dynamics of the environment (i.e., UAV may take off from different locations), we propose a deep Reinforcement leArning based Trajectory control algorithm (RAT). In RAT, we apply the Prioritized Experience Replay (PER) to improve the convergence of the training procedure. Different from the convex optimization based algorithm which may be susceptible to the initial points and requires iterations, RAT can be adapted to any taking off points of the UAVs and can obtain the solution more rapidly than CAT once training process has been completed. Simulation results show that the proposed CAT and RAT achieve the considerable performance and both outperform traditional algorithms.**

***Index Terms*—Deep Reinforcement Learning, Mobile Edge Computing, Unmanned Aerial Vehicle (UAV), Trajectory Control, User Association**

I. Introduction

With the popularity of computationally-intensive tasks, e.g., smart navigation and augmented reality, people are expecting to enjoy more convenient life than ever before. However, current smart devices and user equipments (UEs), due to small size and limited resource, e.g., computation and battery, may not be able to provide satisfactory Quality of Service (QoS) and Quality of Experience (QoE) in executing those highly demanding tasks.

Mobile edge computing (MEC) has been proposed by moving the computation resource to the network edge and it has been proved to greatly enhance UE's ability in executing computation-hungry tasks [1]. Recently, flying mobile edge computing (F-MEC) has been proposed, which goes one step further by considering that the computing resource can be carried by unmanned aerial vehicles (UAVs) [2]. F-MEC inherits the merits of UAV and it is expected to provide more flexible, easier and faster computing service than traditional fixed-location MEC infrastructures. However, the F-MEC also brings several challenges: 1) how to minimize the long-term energy consumption of all UEs by choosing proper user association (i.e., whether UE should offload the tasks and if so, which UAV to offload to, in the case of multiple flying UAVs); 2) how much computations the UAV should allocate to each offloaded UE by considering the limited amount of on-board resource; 3) how to control each UAV's trajectory in real time (namely, flying direction and distance), especially considering the dynamic environment (i.e., the UAV may serve UEs from different taking off points). Traditional approaches like exhaustive search are hardly to tackle the above problems due to the fact that the decision variable space of F-MEC, e.g., deciding the optimal trajectory and resource allocation, is continuous instead of discrete. In [3], the authors propose a quantized dynamic programming algorithm to address the resource allocation problem of MEC. However, the complexity of this approach is very high as the flying choice of UAV is nearly infinite (as continues variables). Moreover, the authors in [4] discretize the UAV trajectory into a sequence of UAV locations and make their proposed problem tractable. Similarly, in [5], the authors assume that the UAV's trajectory can be approximated by using the discrete variables and then they deal with it by using the traditional convex optimization approaches. However, the above treatment may decrease the control accuracy of the UAV and also is not flexible. Furthermore, the above contributions only considered a single UAV case. In practice, one UAV may not have enough resource to serve all the users. If the served area is very large, more than one UAV are normally needed, which will undoubtedly increase the decision space and make it very difficult for the traditional convex optimization based approaches to obtain the optimal control strategies of each UAV. In [6], Liu *et al*. propose a deep reinforcement learning based DRL-EC^3 algorithm, which can control the trajectory of multiple UAVs but did not consider the user association and resource allocation.

Inspired by the challenges mentioned above, in this paper, we first propose a Convex optimizAtion based Trajectory control algorithm (CAT) to minimize the energy consumption of all the UEs, by jointly optimizing user association, resource allocation and UAV trajectory. Specifically, by applying block coordinate descent (BCD) method, CAT is divided into two parts, i.e., subproblems for deciding UAV trajectories and for deciding user association and resource allocation. In each

Liang, Kezhi and Nauman are with the Department of Computer and Informantion Science, Northumbria University, Newcastle upon Tyne, UK, NE1 8ST. Cunhua and Arumugam are with School of Electronic Engineering and Computer Science, Queen Mary University of London, E1 4NS, U.K. Wei is with National Mobile Communications Research Lab, Southeast University, China.

iteration, we solve each part separately while keep the other part fixed, until the convergence is achieved.

Next, we propose a deep Reinforcement leArning based Trajectory control algorithm (RAT) to facilitate the real-time decision making. In RAT, two deep Q networks (DQNs), i.e., actor and critic networks are applied, where the actor network is responsible for deciding the direction and flying distance of the UAV, while the critic network is in charge of evaluating the actions generated by the actor network. Then, we propose a low-complexity matching algorithm to decide the user association and resource allocation with the UAVs. We choose the overall energy consumption of all the UEs as a reward of the RAT. In addition, we deploy a mini-batch to collect samples from the experience replay buffer by using a Prioritized Experience Replay (PER) scheme.

Different from the traditional optimization based algorithms which normally need iterations and are susceptible to the initial points, the proposed RAT can be adapted to any taking off points of the UAVs and can obtain the solutions very rapidly once the training process has been completed. In other words, if the taking off points of the UAV are input to the RAT, the trajectories of the UAVs will be determined by the proposed RAT with only some simple algebraic calculations instead of solving the original optimization problem through traditional high-complexity optimization algorithms. This attributes to the fact that during the training stages, excessive randomly taking off points of UAV are generated and used to train the networks until the networks are converged. Also, with the help of prioritized experience reply (PER), the convergence speed will be increased significantly. RAT can be applied to the practical scenarios where the UAVs needs to act and fly swiftly such as the battlefields. By inputting the current coordinates as the taking off points to the networks, the trajectories of the UAVs will be immediately obtained and then all the UAVs can take off and fly according to the obtained trajectories. Also, the resource allocation and user association are determined by the proposed low-complexity matching algorithm. This is particularly useful to some emergence scenarios (e.g., battlefields, earthquake, large fires), as fast decision making is crucial in these areas.

In the simulation, we can see that the proposed RAT can achieve the similar performance as the convex-based solution CAT. They both have considerable performance gain over other traditional algorithms. In addition, we can see that during the learning procedure, the proposed RAT is less sensitive to the hyperparameters, i.e., the size of mini-batch and the experience replay buffer, when comparing to tradtional reinforcement learning where PER is not applied.

The remainder of this paper is organized as follows. Section II presents the related work. Section III describes the system model. Section IV introduces the proposed CAT algorithm, whereas Section V gives the proposed RAT algorithm including the preliminaries of DRL. Section VI shows the application of proposed RAT algorithm in 3-D UAV trajectory and 3-D channel model scenario. The simulation results are reported in Section VII. Finally, conclusions are given in Section VIII.

II. Related Work

There are many related works that study UAV, MEC and DRL separately, but only a very few consider them holistically. For UAV aided wireless communications, several scenarios have been studied, such as in areas of relay transmissions [7], [8], [9], cellular system [10], data collection [11], [12], [13], [14], wireless power transfer [15], caching networks [16], and D2D communication [17]. In [18], the authors presented an approach to optimize the altitude of UAV to guarantee the maximum radio coverage on the ground. In [19], the authors presented a fly-hover-and-communicate protocol in a UAV-enabled multiuser communication system. They partitioned the ground terminals into disjoint clusters and deployed the UAV as a flying base station. Then, by jointly optimizing the UAV altitude and antenna beamwidth, they optimized the throughput in UAV-enabled downlink multicasting, downlink broadcasting, and uplink multiple access models. In [4], to maximize the minimum average throughput of covered users in OFDMA system, the authors proposed an efficient iterative algorithm based on block coordinate descent and convex optimization techniques to optimize the UAV trajectory and resource allocation. Furthermore, UAV trajectory optimization research were also investigated. For instance in [20], Zeng *et al.* proposed an efficient design by optimizing UAV's flight radius and speed for the sake of maximizing the energy efficiency of UAV communication. In order to maximize the minimum throughput of all mobile terminals in cellular networks, Lyu *et al.* [13] developed a new hybrid network architecture by deploying UAV as an aerial mobile base station. Different from [18], [19], [4], [20] with the single UAV system, a multi-UAV enabled wireless communication system was considered to serve a group of users in [21]. Also, in [22], resource allocation between communication and computation has been investigated in multi-UAV systems. In [23], Mozaffari *et al.* investigated the application of UAVs in Internet of Things (IoT) network, and they optimized the mobility of UAVs, the device-UAV association and uplink power control, for minimizing the overall transmit power of ground IoT devices.

In addition, some recent literature made efforts to mobile edge computing (MEC), which is considered to be a promising technology for bringing computing resource to the edge of the wireless networks [24], where UEs can benefit from offloading their intensive tasks to MEC servers. In [25], partial computation offloading was studied. The computation tasks can be divided into two parts, where one part is executed locally and the other part is offloaded to MEC servers. In [26], binary computation offloading was studied, where the computation tasks can either be executed locally or offloaded to MEC servers.

By taking the advantage of the mobility of UAVs, UAV-enabled MEC has also been studied in [27], [28]. In [27], the authors minimized the overall mobile energy consumption by jointly optimizing UAV trajectory and bit allocation, while satisfying QoS requirements of the offloaded mobile application. In [28], the authors studied UAV-enabled MEC, where wireless power transfer technology is applied to power the Internet of

things devices and collect data from them. In [29], Zhou *et al.* investigated an UAV-enabled MEC wireless-powered system, and they tackled the computation maximization problem through optimizing the UAV's speed, partial and binary computation offloading modes. In [30], Asheralieva *et al.* studied the network operation problem in UAV-enabled MEC network, and they developed a framework based on hierarchical game-theoretic and reinforcement learning. In [31], Zhang *et al.* established a communication and computation optimization model in an MEC-enabled UAV network, where the successful transmission probability was derived through using stochastic geometry.

For most of the above works, optimization theory are mainly applied in order to obtain the optimal and / or suboptimal solutions, e.g., trajectory design and resource allocation. However, solving such optimization problems normally requires plenty of computational resources and take much time. To address this problem, DRL has been applied and attracted much attention recently. In [32], the authors proposed a RL framework that uses DQN as the function approximator. In addition, two important ingredients experience replay and target network are used for improving the convergence performance. In [33], the authors pointed out that the classical DQN algorithm may suffer from substantial overestimations in some scenarios, and proposed a double Q-learning algorithm. In order to solve control problems with continuous state and action space, Lillicrap *at al.* [34] proposed a policy gradient based algorithm. For the purpose of obtaining faster learning and state-of-art performance, in [35], the authors proposed a more robust and scalable approach named prioritized experience replay. Although DRL has achieved remarkable successes in game-playing scenarios, it is still an open research area in UAV-enabled MEC.

III. System Model

As shown in Fig. 1, we consider a scenario that there are N UEs with the set denoted as $\mathcal{N} = \{1, 2, ..., N\}$ and M UAVs with the set denoted as $\mathcal{M} = \{1, 2, ..., M\}$, which form an F-MEC platform. To make it clear, the main notations used in this paper are listed in Table. I.

We assume that the i-th UE constantly generates one task $I_i(t)$ in the t-th time slot and lasting for T time slots. Note that each of time slot has a maximal time duration $T^{\max}$. Then, T tasks will be generated for each UE and one has $t \in \mathcal{T} = \{1, 2, ..., T\}$ and

$$I_i(t) = \{D_i(t),\ F_i(t)\},\ \forall i \in \mathcal{N}, t \in \mathcal{T}, \tag{1}$$

where $D_i(t)$ denotes the size of data required to be transmitted to a UAV if the UE chooses to offload the task, and $F_i(t)$ denotes the total number of CPU cycles needed to execute this task. Assume that each UE can choose either to offload the task to one of the UAVs or execute the task locally. Then one can have

$$a_{ij}(t) = \{0, 1\}, \forall i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T}, \tag{2}$$

where $a_{ij}(t) = 1$, $j \neq 0$ implies that the i-th UE decides to offload the task to the j-th UAV in the t-th time slot, while $a_{ij}(t) = 1$, $j = 0$ means that the i-th UE executes the task itself in the t-th time slot, and otherwise, $a_{ij}(t) = 0$. Define a new set $j \in \mathcal{M}' = \{0, 1, 2, ..., M\}$ to represent the possible place where the tasks from UEs can be executed, where $j = 0$ indicates that UE conducts its own task locally without offloading.

TABLE I: Main Notations.

Notation	Definition
$i, N, \mathcal{N}$	index, number, set of of UEs.
$j, M, \mathcal{M}$	index,number,set of UAVs.
$t, T, \mathcal{T}$	index, number, set of time slots.
$I_i(t), D_i(t), F_i(t)$	i-th UEs' task in t-th time slot.
$a_{ij}(t)$	user association between i-th UE and j-th UAV.
$R_j^{\max}$	maximal horizontal coverage radius of j-th UAV.
$\theta_j^h(t), \theta_j^v(t), d_j(t)$	flying action j-th UAV.
$d^{\max}, v_j(t)$	maximal distance, velocity of j-th UAV.
$[X_j(t), Y_j(t), Z_j(t)]$	coordinate of j-th UAV.
$X^{\max}, Y^{\max}$	side length of rectangle-shaped area.
$T^{\max}$	maximal time duration.
$V^{\max}, f^{\max}$	maximal number of tasks, computation resource.
$[x_i, y_i]$	coordinate of i-th UE.
$R_{ij}(t)$	horizontal distance between UE and UAV.
B, P^{Tr}	channel bandwidth, transmitting power.
g_0, σ^2	channel power gain, noise power.
$T_{ij}^{\mathrm{O}}(t), T_{ij}^{\mathrm{Tr}}(t), T_{ij}^{\mathrm{C}}(t)$	time for task completion, offloading, executing.
$E_{ij}^{\mathrm{Tr}}(t), E_{ij}^{\mathrm{L}}(t)$	energy for offloading, local execution.
$\boldsymbol{U}, \boldsymbol{G}$	set of UAV trajectory, UAV coordinates.
$\boldsymbol{A}, \boldsymbol{F}$	set of user association, resource allocation.
$s(t), a(t), z(t)$	state, action and reward.
$\pi(\cdot), Q(\cdot), L(\cdot)$	policy function, Q function, loss function.
K, X	size of mini-batch, experience replay buffer.
ϕ, δ, J	network parameter, TD-error, policy gradient.
$Z^{\min}, Z^{\max}$	minimal, maximal altitude value.
$d_{ij}(t)$	distance between the j-th UAV and i-th UE.

In addition, we assume that each UE can only be served by at most one UAV or itself, and each task only has one place to execute. Then, it follows

$$\sum_{j=0}^{M} a_{ij}(t) = 1, \forall i \in \mathcal{N}, t \in \mathcal{T}. \tag{3}$$

Additionally, in this paper, the OFDM is applied, which means that each UAV can only accept $V^{\max}$ tasks in each time slot, due to the number of limited sub-carriers. Thus, one has

$$\sum_{i=1}^{N} a_{ij}(t) \leq V^{\max}, \forall j \in \mathcal{M}, t \in \mathcal{T}. \tag{4}$$

A. UAV Movement

Assume that the j-th UAV flies at a fixed altitude like [19], and it has a maximal horizontal coverage, which depends on the azimuth angle of antennas and the flying altitude. Also, assume that in the t-th time slot, the j-th UAV can fly with a horizontal direction as

$$0 \leq \theta_j^h(t) \leq 2\pi, \forall j \in \mathcal{M}, t \in \mathcal{T}, \tag{5}$$

and distance as

$$0 \leq d_j(t) \leq d^{\max}, \forall j \in \mathcal{M}, t \in \mathcal{T}, \tag{6}$$

where $d^{\max}$ is the maximal flying distance in each time slot. We also denote the coordinate of the j-th UAV in the

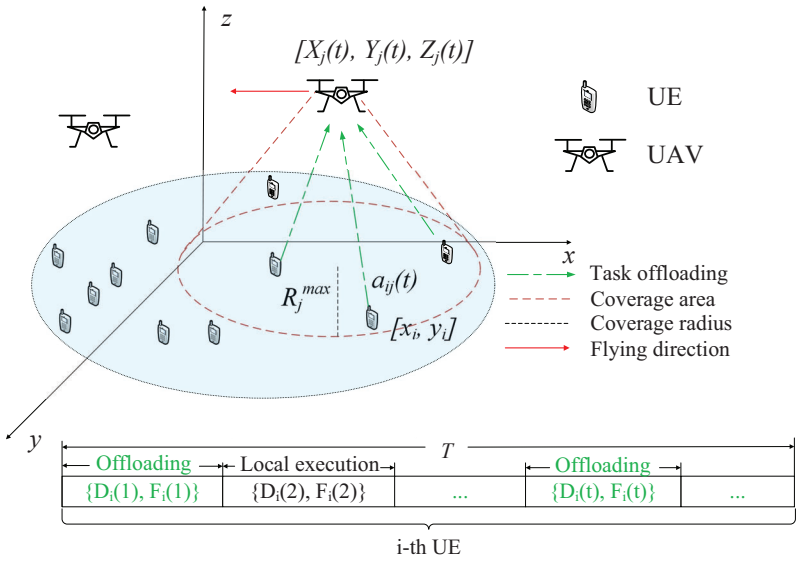


Fig. 1: Multi-UAV enabled F-MEC architecture.

t-th time slot as $[X_j(t), Y_j(t), Z_j]$, where $X_j(t) = X_j(0) + \sum_{l=1}^{t} d_j(l)\cos(\theta_j^h(l))$, $Y_j(t) = Y_j(0) + \sum_{l=1}^{t} d_j(l)\sin(\theta_j^h(l))$ and $[X_j(0), Y_j(0), Z_j]$ is the initial coordinate of the j-th UAV.

In this paper, each UAV can only move within a rectangle-shaped area with the side length of $X^{\max}$ and $Y^{\max}$. Thus, it has

$$0 \le X_j(t) \le X^{\max}, \tag{7}$$

and

$$0 \le Y_j(t) \le Y^{\max}. \tag{8}$$

Then, the flying velocity of the j-th UAV in the t-th time slot can be expressed as

$$v_j(t) = \frac{d_j(t)}{T^{\max}}. \tag{9}$$

In this paper, we ignore the communication related energy, including communication circuitry and signal processing.

B. Task Execution

If the i-th UE decides to offload the task to the j-th UAV in the t-th time slot, then the horizontal distance $R_{ij}(t)$ can be written as

$$R_{ij}(t) = \sqrt{(X_j(t) - x_i)^2 + (Y_j(t) - y_i)^2}, \tag{10}$$

where $[x_i, y_i]$ is the coordinate of the i-th UE. Additionally, we assume that each UAV has a maximal azimuth angle $\theta^{\max}$ [1]. Thus, in each time slot, the maximal horizontal coverage of the j-th UAV $R^{\max}$ can be obtained as follows

$$R^{\max} = Z_j \tan(\theta^{\max}). \tag{11}$$

[1]We define the azimuth angle with respect to a 3-D reference axis, such as x axis, y axis, z axis.

Thus, it has

$$a_{ij}(t) R_{ij}(t) \le R^{\max}, \forall i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T}. \tag{12}$$

In this paper, the free space channel model is applied. Thus, the uplink data rate is given by

$$r_{ij}(t) = B \log_2 \left(1 + \frac{\alpha P^{\mathrm{Tr}}}{Z_j^2 + R_{ij}^2(t)}\right), \ \forall i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T}, \tag{13}$$

where B is the bandwidth for each communication channel; P^{Tr} is the transmitting power of the i-th UE; $\alpha = \frac{g_0 G_0}{\sigma^2}$ with $G_0 \approx 2.2846$ [19], [2], [22], [36]; g_0 is the channel power gain at the reference distance 1 m and σ^2 is the noise power. Note that we consider each user applies orthogonal frequency division multiplexing (OFDM) channel and there is no interference among them.

If the i-th UE decides to offload its task to the j-th UAV in the t-th time slot, the total task completion time is given by

$$T_{ij}^{\mathrm{O}}(t) = T_{ij}^{\mathrm{Tr}}(t) + T_{ij}^{\mathrm{C}}(t), \ \forall t \in \mathcal{T}, \tag{14}$$

where $T_{ij}^{\mathrm{Tr}}(t)$ is the time to offload the data from the i-th UE to the j-th UAV in the t-th time slot, given by

$$T_{ij}^{\mathrm{Tr}}(t) = \frac{D_i(t)}{r_{ij}(t)}, \ \forall t \in \mathcal{T}, \tag{15}$$

and $T_{ij}^{\mathrm{C}}(t)$ is the time required to execute the task at the UAV as

$$T_{ij}^{\mathrm{C}}(t) = \frac{F_i(t)}{f_{ij}^{\mathrm{C}}(t)}, \ \forall t \in \mathcal{T}, \tag{16}$$

where $f_{ij}^{\mathrm{C}}(t)$ is the computation resource that the j-th UAV can provide to the i-th UE in the t-th time slot.

Note that the time needed for returning the results back to UE from UAV is ignored, similar to [37]. The overall energy

consumption of the i-th UE to the j-th UAV in the t-th time slot is given by

$$E_{ij}^{\mathrm{Tr}}(t) = P^{\mathrm{Tr}} T_{ij}^{\mathrm{Tr}}(t),\ \forall t \in \mathcal{T}. \tag{17}$$

If the UE decides to execute the task locally, the power consumption can be evaluated as $k_i(f_{ij}^{\mathrm{L}}(t))^{v_i}$ [2], [22], [36], where $k_i \geq 0$ is the effective switched capacitance, v_i is typically set to 3, and $f_{ij}^{\mathrm{L}}(t)$ is the computation resource that the i-th UE applies to execute the task. The overall time for local execution can be given by

$$T_{ij}^{\mathrm{L}}(t) = \frac{F_i(t)}{f_{ij}^{\mathrm{L}}(t)}. \tag{18}$$

Thus, the total energy consumption for local execution equals

$$E_{ij}^{\mathrm{L}}(t) = k_i(f_{ij}^{\mathrm{L}}(t))^{v_i} T_{ij}^{\mathrm{L}}(t),\ t \in \mathcal{T}. \tag{19}$$

To sum up, the overall energy consumption for task execution $E_{ij}(t)$ is given by

$$E_{ij}(t) = \begin{cases} E_{ij}^{\mathrm{L}}(t), & \text{local execution,} \\ E_{ij}^{\mathrm{Tr}}(t), & \text{offloading,} \end{cases} \tag{20}$$

and the time to complete the task $T_{ij}(t)$ is expressed as

$$T_{ij}(t) = \begin{cases} T_{ij}^{\mathrm{L}}(t), & \text{local execution,} \\ T_{ij}^{\mathrm{O}}(t), & \text{offloading.} \end{cases} \tag{21}$$

Without loss of generality, we assume that each task has to be completed within the maximal time duration $T^{\max}$, which is consistent with the maximal flying time in each time slot. Then, one has

$$T_{ij}(t) \leq T^{\max},\ \forall i \in \mathcal{N}, j \in \mathcal{M}', t \in \mathcal{T}. \tag{22}$$

In each time slot, since the computation resource that each UAV can provide is limited, we have

$$\sum_{i=1}^{N} a_{ij}(t) f_{ij}^{\mathrm{C}}(t) \leq f^{\max},\ \forall j \in \mathcal{M}, t \in \mathcal{T}, \tag{23}$$

where $f^{\max}$ is the maximal computation resource that the j-th UAV can provide in the t-th time slot. Next, we show our proposed problem formulation.

C. Problem Formulation

Denote $\boldsymbol{U} = \{\theta_j^h(t), d_j(t), \forall j \in \mathcal{M}, t \in \mathcal{T}\}$, $\boldsymbol{A} = \{a_{ij}(t), \forall i \in \mathcal{N}, j \in \mathcal{M}', t \in \mathcal{T}\}$, $\boldsymbol{F} = \{f_{ij}(t), \forall i \in \mathcal{N}, j \in \mathcal{M}', t \in \mathcal{T}\}$. Then, the energy minimization for all UEs is formulated as

$$\mathcal{P}1: \min_{\boldsymbol{U},\boldsymbol{A},\boldsymbol{F}} \sum_{i=1}^{N}\sum_{j=0}^{M}\sum_{t=1}^{T} a_{ij}(t) E_{ij}(t) \tag{24a}$$

subject to:

$$a_{ij}(t) = \{0,1\}, \forall i \in \mathcal{N}, j \in \mathcal{M}', t \in \mathcal{T}, \tag{24b}$$

$$\sum_{j=0}^{M} a_{ij}(t) = 1, \forall i \in \mathcal{N}, t \in \mathcal{T}, \tag{24c}$$

$$\sum_{i=1}^{N} a_{ij}(t) \leq V^{\max}, \forall j \in \mathcal{M}, t \in \mathcal{T}, \tag{24d}$$

$$0 \leq \theta_j^h(t) \leq 2\pi, \forall j \in \mathcal{M}, t \in \mathcal{T}, \tag{24e}$$

$$0 \leq d_j(t) \leq d^{\max}, \forall j \in \mathcal{M}, t \in \mathcal{T}, \tag{24f}$$

$$0 \leq X_j(t) \leq X^{\max}, \forall j \in \mathcal{M}, t \in \mathcal{T}, \tag{24g}$$

$$0 \leq Y_j(t) \leq Y^{\max}, \forall j \in \mathcal{M}, t \in \mathcal{T}, \tag{24h}$$

$$a_{ij}(t) R_{ij}(t) \leq R^{\max}, \forall i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T}, \tag{24i}$$

$$T_{ij}(t) \leq T^{\max},\ \forall i \in \mathcal{N}, j \in \mathcal{M}', t \in \mathcal{T}, \tag{24j}$$

$$\sum_{i=1}^{N} a_{ij}(t) f_{ij}^{\mathrm{C}}(t) \leq f^{\max},\ \forall j \in \mathcal{M}, t \in \mathcal{T}. \tag{24k}$$

One can see that the above problem $\mathcal{P}1$ is a mixed integer nonlinear programming (MINLP), as it includes both integer variable, $\boldsymbol{A}$ and continuous variables, $\boldsymbol{F}$ and $\boldsymbol{U}$, which is very difficult to solve in general. We first propose a convex optimization based algorithm CAT to address it iteratively. Then, we propose a Deep Reinforcement Learning (DRL) based RAT to facilitate fast decision-making, which can be applied in dynamic environment. Note that in practice, if the i-th UE does not generate the tasks in the t-th time slot and then the corresponding $D_i(t)$ and $F_i(t)$ can be set to zero.

IV. Proposed CAT Algorithm

In this section, a convex optimization based CAT is proposed to solve the above problem $\mathcal{P}1$. We first define a set of new variables to denote the trajectories of UAVs as $\boldsymbol{G} = \{G_j(t), \forall j \in \mathcal{M}, t \in \mathcal{T}\}$, where the coordinate is $G_j(t) = [X_j(t), Y_j(t)]$, $X_j(t) = X_j(0) + \sum_{l=1}^{t} d_j(l)\cos\left(\theta_j^h(l)\right)$ and $Y_j(t) = Y_j(0) + \sum_{l=1}^{t} d_j(l)\sin\left(\theta_j^h(l)\right)$. Thus, the optimization problem $\mathcal{P}1$ can be reformulated as

$$\mathcal{P}2: \min_{\boldsymbol{G},\boldsymbol{A},\boldsymbol{F}} \sum_{i=1}^{N}\sum_{j=0}^{M}\sum_{t=1}^{T} a_{ij}(t) E_{ij}(t) \tag{25a}$$

subject to: (24b), (24c), (24d), (24g), (24h), (24j), (24k),

$$a_{ij}(t)||G_j(t) - q_i||^2 \leq (R^{\max})^2, \forall i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T}, \tag{25b}$$

$$||G_j(t+1) - G_j(t)||^2 \leq (d^{\max})^2, \forall t \in \{0, 1, ..., T-1\}, \tag{25c}$$

where $q_i = [x_i, y_i]$. In order to solve $\mathcal{P}2$, we divide it into two subproblems and apply the block coordinate descent (BCD) method to address it. To this end, we first optimize the user association $\boldsymbol{A}$ and resource allocation $\boldsymbol{F}$ given the UAV trajectory $\boldsymbol{G}$. Then, we optimize the UAV trajectory $\boldsymbol{G}$ given the user association $\boldsymbol{A}$ and resource allocation $\boldsymbol{F}$. We solve the two optimization problems iteratively, until the convergence is achieved.

A. User Association and Resource Allocation

Given the UAV trajectory $\boldsymbol{G}$, the subproblem to decide user association $\boldsymbol{A}$ and resource allocation $\boldsymbol{F}$ can be formulated as

$$\min_{\boldsymbol{A},\boldsymbol{F}} \sum_{i=1}^{N}\sum_{j=0}^{M}\sum_{t=1}^{T} a_{ij}(t)E_{ij}(t) \tag{26a}$$
subject to: (24b), (24c), (24d), (24j), (24k), (25b).

One can see that (24j) can be written as

$$f_{ij}^{\mathrm{C}}(t) \geq \frac{F_i(t)}{T^{\max} - \frac{D_i(t)}{r_{ij}(t)}},\ \forall j \in \mathcal{M}, t \in \mathcal{T}, \tag{27}$$

if the i-th UE chooses to offload the task, and

$$f_{ij}^{\mathrm{L}}(t) \geq \frac{F_i(t)}{T^{\max}},\ j = 0, \forall t \in \mathcal{T}, \tag{28}$$

if the i-th UE decides to execute the task locally. It is readily to see that equality holds for both (27) and (28).

Then, (26) can be re-written as

$$\min_{\boldsymbol{A},\boldsymbol{F}} \sum_{i=1}^{N}\sum_{j=0}^{M}\sum_{t=1}^{T} \Big(a_{ij}(t)E_{ij}^{\mathrm{Tr}}(t) + (1 - a_{ij}(t))E_{ij}^{\mathrm{L}}(t)\Big) \tag{29a}$$
subject to: (24b), (24c), (24d), (25b),
$$f_{ij}^{\mathrm{L}}(t) = \frac{F_i(t)}{T^{\max}},\ j = 0, \forall t \in \mathcal{T}, \tag{29b}$$
$$\sum_{i=1}^{N} a_{ij}(t)\frac{F_i(t)}{T^{\max} - \frac{D_i(t)}{r_{ij}(t)}} \leq f^{\max},\ \forall j \in \mathcal{M}, t \in \mathcal{T}. \tag{29c}$$

It is readily to find that (29) is a Multiple-Choice Multi-Dimensional 0-1 Knapsack Problem (MMKP), which is NP-hard in general. Fortunately, it can be solved by applying Branch and Bound method via a standard Python package PULP [38].

B. UAV Trajectory Optimization

Given the user association and resource allocation from (29) and removing the constant, $\mathcal{P}2$ can be simplified as

$$\min_{\boldsymbol{G}} \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t=1}^{T} a_{ij}(t)\frac{P^{\mathrm{Tr}}D_i(t)}{B\log_2(1 + \frac{\alpha P^{\mathrm{Tr}}}{Z_j^2 + ||G_j(t) - q_i||^2})} \tag{30a}$$
subject to: (24g), (24h), (25b), (25c),
$$\frac{D_i(t)}{B\log_2(1 + \frac{\alpha P^{\mathrm{Tr}}}{Z_j^2 + ||G_j(t) - q_i||^2})} + \frac{F_i(t)}{f_{ij}^{\mathrm{C}}(t)} \leq T^{\max},$$
$$\forall i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T}. \tag{30b}$$

It is easy to see that the above optimization problem is non-convex with respect to $G_j(t)$. Next, we introduce a set $\boldsymbol{\eta} = \{\eta_{ij}(t), \forall i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T}\}$, where $\eta_{ij}(t) = a_{ij}(t)\frac{P^{\mathrm{Tr}}D_i(t)}{B\log_2(1 + \frac{\alpha P^{\mathrm{Tr}}}{Z_j^2 + ||G_j(t) - q_i||^2})}$, then, problem (30) can be transformed into

$$\min_{\boldsymbol{G},\boldsymbol{\eta}} \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t=1}^{T} \eta_{ij}(t) \tag{31a}$$
subject to: (24g), (24h), (25b), (25c),
$$B\log_2\big(1 + \frac{\alpha P^{\mathrm{Tr}}}{Z_j^2 + ||G_j(t) - q_i||^2}\big) \geq \frac{a_{ij}(t)P^{\mathrm{Tr}}D_i(t)}{\eta_{ij}(t)},$$
$$\forall i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T}, \tag{31b}$$
$$B\log_2\big(1 + \frac{\alpha P^{\mathrm{Tr}}}{Z_j^2 + ||G_j(t) - q_i||^2}\big) \geq \frac{D_i(t)}{T^{\max} - \frac{F_i(t)}{f_{ij}^{\mathrm{C}}(t)}},$$
$$\forall i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T}. \tag{31c}$$

One observes that (31b) and (31c) are convex with respect to $||G_j(t) - q_i||$, respectively. Thus, (31b) and (31c) are non-convex constraints. Then, similar to [4], [5], we apply the successive convex approximation (SCA) to solve this problem. Specifically, for any given local point $G_j^r(t)$ in $\boldsymbol{G}^r = \{G_j^r(t), \forall j \in \mathcal{M}, t \in \mathcal{T}\}$, one can have the following inequality as

$$\begin{aligned} w_{ij}(t) &= B\log_2\big(1 + \frac{\alpha P^{\mathrm{Tr}}}{Z_j^2 + ||G_j(t) - q_i||^2}\big) \\ &\geq K_{ij}^r(t)(||G_j(t) - q_i||^2 - ||G_j^r(t) - g_i||^2) + B_{ij}^r(t) \\ &\triangleq w_{ij}^{lb,r}(t), \end{aligned} \tag{32}$$

where

$$K_{ij}^r(t) = -\frac{B\alpha P^{\mathrm{Tr}}\log_2(e)}{(Z_j^2 + ||G_j^r(t) - q_i||^2)(Z_j^2 + ||G_j^r(t) - q_i||^2 + \alpha P^{\mathrm{Tr}})}, \tag{33}$$

and

$$B_{ij}^r(t) = B\log_2\big(1 + \frac{\alpha P^{\mathrm{Tr}}}{Z_j^2 + ||G_j^r(t) - q_i||^2}\big). \tag{34}$$

Then, problem (31) can be written as

$$\min_{\boldsymbol{G},\boldsymbol{\eta}} \sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{t=1}^{T} \eta_{ij}(t) \tag{35a}$$
subject to: (24g), (24h), (25b), (25c),
$$w_{ij}^{lb,r}(t) \geq \frac{a_{ij}(t)P^{\mathrm{Tr}}D_i(t)}{\eta_{ij}(t)}, \forall i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T}, \tag{35b}$$
$$w_{ij}^{lb,r}(t) \geq \frac{D_i(t)}{T^{\max} - \frac{F_i(t)}{f_{ij}^{\mathrm{C}}(t)}}, \forall i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T}. \tag{35c}$$

The above problem is a convex quadratically constrained quadratic program (QCQP) and it can be solved by a standard Python package CVXPY [39].

C. Overall Algorithm Design

In this section, a convex optimization algorithm based CAT is proposed to solve Problem $\mathcal{P}2$, where we optimize user association and resource allocation subproblem iteratively with the UAV trajectory subproblem until the convergence is achieved. We describe the pseudo code of proposed CAT in Algorithm 1.

Algorithm 1 CAT Algorithm

1: Set $r = 0$, and initialize $\boldsymbol{G}^r$;
2: **repeat**
3: Solve Problem (29) by Branch and Bound method for given $\boldsymbol{G}^r$, and denote the optimal solution as $\boldsymbol{A}^{r+1}$ and $\boldsymbol{F}^{r+1}$;
4: Solve Problem (35) for given $\boldsymbol{A}^{r+1}$ and $\boldsymbol{F}^{r+1}$, and denote the solution as $\boldsymbol{G}^{r+1}$;
5: $r = r + 1$;
6: **until** the convergence is achieved.

Discussions: Algorithm 1 needs to run once the initial taking off locations of the UAVs change. However, the complexity of Algorithm 1 is high as the solutions are iteratively obtained and each subproblem involves a huge number of optimization variables especially when the total number of time slots is high. Precisely, as shown in Algorithm 1, assume that the overall iteration number is K^r. In each iteration, Problem (29) has $N(M+1)T$ variables, and it can be solved by Branch and Bound method, in which the Simplex technique for solving linear programs is used. Thus, the computational complexity is $O\big(2^{N(M+1)T}\big)$ in the worst case. Furthermore, Problem (35) has $3NMT$ variables and $(N+2)MT+T-1$ constraints. The computational complexity is $O\big((3NMT)^2\sqrt{3NMT}\log_2(\frac{1}{\epsilon_1})((N+2)MT+T-1)\big)$, i.e., $O\big((NMT)^{3.5}\log_2(\frac{1}{\epsilon_1})\big)$, where ϵ_1 is the accuracy of SCA for solving Problem (35). Overall, the total complexity of CAT algorithm is $O\big(K^r\big(2^{N(M+1)T}+(NMT)^{3.5}\log_2(\frac{1}{\epsilon_1})\big)\big)$. Hence, Algorithm 1 is not suitable for some emergence scenarios (e.g., battlefields, earthquake, large fires), where fast decision making is highly demanded. This motivates the algorithm developed based on DRL in the following section.

V. Proposed RAT Algorithm

To facilitate the fast decision making, the DRL-based RAT algorithm is proposed in this section. We first give some preliminaries as follows.

A. Preliminaries

1) DQN: In a standard reinforcement learning, an agent is assumed to interact with the environment and select the optimal actions that can maximize the accumulated reward. In [32], a Deep Q Network (DQN) structure developed by Google Deepmind, integrates the deep neural networks with traditional reinforcement learning. The DQN is used to estimate the well-known Q-value defined as

$$Q(s(t),c(t)) = \mathbb{E}[Z(t)|s(t),c(t)], \tag{36}$$

where $s(t)$ and $c(t)$ denote the state and action respectively, $\mathbb{E}[\cdot]$ denotes the expectation, whereas $Z(t) = \sum_{t'=t}^{T}\gamma z(t')$ is a reward and $\gamma \in [0,1]$ is the discount factor and $z(t')$ is a reward function in the t'-th time step (or time slot). As the objective is to maximize the reward, a widely used policy is $\pi(s(t)|\phi^Q) = \text{argmax}_{c(t)}Q(s(t),c(t)|\phi^Q)$, where ϕ^Q is the parameter of the deep neural network. Then, the DQN can be trained by minimizing the loss function [32]. Also, since the deep networks are known to be unstable and very difficult to converge, two effective approaches, i.e., target network and experience replay, have been introduced in [32]. The target network has the same structure as the original DQN but the parameters are updated more slowly. The experience replay stores the state transition samples which can help the DQN converge. However, the DQN was originally designed to solve the problem with discrete variables. Although we can adapt the DQN to continuous problems by discretizing the action space, it may unfortunately result in a huge searching space and therefore intractable to deal with.

2) DDPG: To deal with the problem with continuous variables, e.g., the trajectory control of UAV, one may apply the actor-critic approach, which was developed in [40]. DeepMind has proposed a deep deterministic policy gradient (DDPG) approach [34] by integrating the actor-critic approach into DRL. DDPG includes two DQNs, one of the DQNs, named actor network with function $\pi(s(t)|\phi^\pi)$ is applied to generate action $c(t)$ for a given state $s(t)$. The other DQN named critic network with function $Q(s(t),c(t)|\phi^Q)$, is used to generate the Q-value, which evaluates the action produced by the actor network. In order to improve the learning stability, two adjacent target networks corresponding to the actor and critic networks, $\pi'(\cdot)$, $Q'(\cdot)$ with respective parameters $\phi^{\pi'}$, $\phi^{Q'}$, are also applied.

Then, the critic network can be updated with the loss function, $L(\phi^Q)$, as

$$L(\phi^Q) = \frac{1}{K}\sum_{k=1}^{K}\delta_k^2, \tag{37}$$

where in each time step, the mini-batch randomly samples K constituting experiences from the experience replay buffer, and δ_k is the temporal difference (TD)-error [41] which is given by

$$\begin{aligned}\delta_k =& z(k) + \gamma Q'(s(k+1), \pi'(s(k+1)|\phi^{\pi'})|\phi^{Q'}) \\ &- Q(s(k), \pi(s(k)|\phi^\pi)|\phi^Q).\end{aligned} \tag{38}$$

On the other hand, the actor network can be updated by applying the policy gradient, which is described as [34].

$$\begin{aligned}\nabla_{\phi^\pi} J \approx& \frac{1}{K}\sum_{k=1}^{K}\nabla_c Q(s,c|\phi^Q)|_{s=s(k),c=\pi(s(k)|\phi^\pi)} = \\ &\frac{1}{K}\sum_{k=1}^{K}\left[\nabla_c Q(s,c|\phi^Q)|_{s=s(k),c=\pi(s(k))}\cdot\nabla_{\phi^\pi}\pi(s|\phi^\pi)|_{s=s(k)}\right].\end{aligned} \tag{39}$$

B. The RAT Algorithm

In this section, we introduce the DRL based RAT algorithm, which includes deep neural networks (i.e., actor and critic networks) and the matching algorithms. In order to apply the DRL, we first define the state, action and reward as follows:

1) **State** $s(t)$: $s(t) = \{[X_j(t), Y_j(t), Z_j],\ \forall j \in \mathcal{M}\}$, $s(t)$ is the set of the coordinates of all UAVs.
2) **Action** $c(t)$: $c(t)$ is the set of the actions of all UAVs, including the horizontal direction $\theta_j^h(t)$ and distance

$d_j(t)$. Then, the action set can be defined as $c(t) = \{[\theta_j^h(t), d_j(t)],\ \forall j \in \mathcal{M}\}$.

3) **Reward** $z(t)$: $z(t)$ is defined as the minus of the overall energy consumption of all the UEs in each time slot as

$$z(t) = -\sum_{i=1}^{N}\sum_{j=0}^{M} a_{ij}(t)E_{ij}(t) - p, \tag{40}$$

where p is the penalty if any of UAV flies out of the target area, which means (24g) or (24h) is not satisfied.

The algorithm framework used in this paper is depicted in Fig. 2, where an agent, which could be deployed in the control center of the base station, is assumed to interact with the environment. An actor network $\pi(s(t)|\phi^\pi)$ is applied to generate the action, which includes the flying direction and distance for each UAV. The critic network $Q(s(t), c(t)|\phi^Q)$ is used to obtain the Q-value of the action (i.e., to evaluate the action generated by actor network). In each time slot, the agent sends the action generated by actor network to each UAV. Then, each UE tries to associate with one UAV in its coverage, i.e., (12) by using a matching algorithm in Algorithm 3. More specifically, each UE tries to connect the UAV which can save more offloading energy. If the minimum offloading energy is larger than the energy of local execution, the UE will decide to conduct the task locally. Note that RAT has the same optimization strategy for resource allocation as CAT.

Also, each UAV selects the UEs based on the following criteria: 1) UE should be within its coverage area; 2) UE could save more energy, i.e., $E_{ij}^{\mathrm{L}}(t) - E_{ij}^{\mathrm{C}}(t)$ will be given higher priority in offloading to this UAV. We will introduce the details of the proposed matching algorithm in Algorithm 3. After the matching algorithm, the reward in (40) can be obtained.

We assume that there is an experience replay buffer for the agent to store the experience $[s(t), c(t), z(t), s(t+1)]$. Once the experience replay buffer is full, the learning procedure starts. A mini-batch with K experiences can be obtained from the experience replay buffer to train the networks.

In the classical DRL algorithms, such as Q-learning [42], SARSA [43] and DDPG [34], the mini-batch uniformly samples experiences from the experience replay buffer. However, since TD-error in (38) is used to update the Q-value network, experience with high TD-error often indicates the successful attempts. Therefore, a better way to select the experience is to assign different weights to samples. Schaul *et al.* [35] developed a prioritized experience replay scheme, in which the absolute TD-error $|\delta_k|$ is used to evaluate the probability of the sampled k-th experience from the mini-batch. Then, the probability of sampling the k-th experience can be given by

$$P(k) = \frac{p_k^\beta}{\sum_{m\in K} p_m^\beta}, \tag{41}$$

where $p_k = |\delta_k| + \epsilon$, $\epsilon = 0.001$ is a positive constant to avoid the edge-case of transitions not being revisited if $|\delta_k|$ is 0, $\beta = 0.6$ is denoted as a factor to determine the prioritization [35].

However, frequently sampling experiences with high $|\delta_k|$ can cause divergence and oscillation. To tackle this issue, the importance-sampling weight [44] is introduced to represent the importance of sampled experience, which can be given by

$$\omega_k = \frac{1}{(X \cdot P(k))^\mu}, \tag{42}$$

where X is the size of experience replay buffer, μ is given as 0.4 [35]. Thus, the loss function $L(\phi^Q)$ in (37) can be updated as

$$L(\phi^Q) = \frac{1}{K}\sum_{k=1}^{K} \omega_k \delta_k^2, \tag{43}$$

which is used in our proposed RAT to train the networks. Next, we describe the pseudo code of the overall RAT framework in Algorithm 2.

Algorithm 2 RAT Algorithm

1: Initialize actor network $\pi(s(t)|\phi^\pi)$ with parameters ϕ^π and critic network $Q(s(t), s(t)|\phi^Q)$ with parameters ϕ^Q;
2: Initialize target networks $Q'(\cdot)$ with parameters $\phi^{Q'} = \phi^Q$ and $\pi'(\cdot)$ with parameters $\phi^{\pi'} = \phi^\pi$;
3: Initialize experience replay buffer $\mathcal{X}$;
4: **for** epoch =1,..., $k^{\max}$ **do**
5: Initialize $s(t)$;
6: **for** time slot t =1,..., T **do**
7: $\pi(s(t)|\phi^\pi) + \rho N'$ where N' is the random noise and ρ decays with t;
8: **for** UAV j=1,..., M **do**
9: Execute $c(t)$;
10: Obtain $s(t+1)$;
11: **end for**
12: Obtain the user association with UAVs using matching algorithm proposed in Algorithm 3;
13: Obtain the reward $z(t)$ from (40);
14: Store experience $[s(t), c(t), z(t), s(t+1)]$ into the replay buffer;
15: **if** the replay buffer is full **then**
16: **for** k = 1,..., K **do**
17: Sample k-th experience with probability $P(k)$ from (41);
18: Calculate $|\delta_k|$ and ω_k from (38) and (42) respectively;
19: **end for**
20: Update parameters of the critic network ϕ^Q by minimizing its loss function according to (43);
21: Update parameters of the actor network ϕ^π by using policy gradient approach according to (39);
22: Update two target networks with the updating rate τ:
23: **end if**
24: **end for**
25: **end for**

We first initialize the actor, critic, two target networks, and experience replay buffer in Line 1 - 3. In the beginning of each epoch, all UAVs start to serve UEs from different taking off points. Note that for better exploration, we add a random noise N' to the action, where N' follows a normal distribution with 0 mean and variance 1, ρ is set to 2 and decays with a rate of 0.9995 in each time step. From Line 8-11, each UAV flies according to the generated action $c(t)$ and enters the next

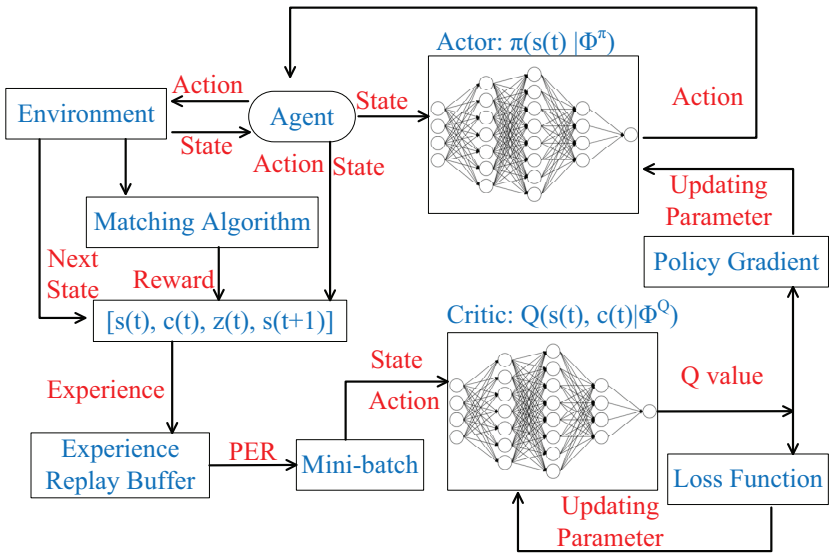


Fig. 2: The structure of RAT algorithm.

state $s(t+1)$. Then, we obtain the user association by using Algorithm 3. Next, the reward $z(t)$ is obtained according to (40) (i.e., Line 13). The experience is also stored in the replay buffer. When the buffer is full, the mini-batch samples K experiences by applying the prioritized experience replay (i.e., Line 16-19). Then, we update the actor and critic networks by using loss function in (43) and policy gradient in (39) respectively. Finally, we update the target networks by using the following equations as (i.e., Line 22)

$$\phi^{Q'} \leftarrow \tau\phi^{Q} + (1-\tau)\phi^{Q'}, \tag{44}$$

and

$$\phi^{\pi'} \leftarrow \tau\phi^{\pi} + (1-\tau)\phi^{\pi'}, \tag{45}$$

where τ is the updating rate.

Next, we introduce the low-complexity matching algorithm which can decide the user association and resource allocation given UAVs' trajectories, as shown in Algorithm 3. First, we denote $\boldsymbol{A}$ with size N to record the user association between UEs and UAVs. If $\boldsymbol{A}(i) = j$, it means the i-th UE matches with the j-th UAV, and if $\boldsymbol{A}(i) = 0$, it denotes that the i-th UE is not matched yet and has to execute its task locally. In addition, we denote a preference list $\boldsymbol{E}_j$ for the j-th UAV to record UEs that can benefit from offloading. Then, from Line 2 to 10, we generate the preference list $\boldsymbol{E}_j$ for the j-th UAV. Precisely, if constraint (12) is met, we obtain $E_{ij}^{\mathrm{L}}(t)$, $E_{ij}^{\mathrm{Tr}}(t)$, and $f_{ij}^{\mathrm{C}}(t)$ according to (19), (17), and (27), respectively. UEs that benefit from offloading will be stored in $\boldsymbol{E}_j$. Since UAVs wish to save as many energy consumption of UEs as possible, we sort the preference list $\boldsymbol{E}_j$ with descending order with respect to $E_{ij}^{\mathrm{L}}(t) - E_{ij}^{\mathrm{Tr}}(t)$, as shown in Line 11. The UE that can save more energy via offloading will be matched with a higher priority. Next, from Line 13 to 23, we conduct the matching process. Each UAV keeps selecting UEs according to its preference list, and constantly checking the constraints (4) and (23) based on $\boldsymbol{A}$. In the meantime, the selected UE will determine whether to match with the UAV or not. Precisely, from Line 17 to 19, if the selected UE is not matched before, or matching with the j-th UAV could save more energy than previous match, the corresponding $\boldsymbol{A}(i)$ will be updated. We do this process until all the UEs in each preference list are checked. Then, the final user association can be obtained from $\boldsymbol{A}$.

Algorithm 3 Matching Algorithm

1: Initialize $\boldsymbol{A}$ and $\boldsymbol{F}_j$, $\forall j \in \mathcal{M}$, $\forall i \in \mathcal{N}$;
2: **for** UAV $j = 1,..., M$ **do**
3: **for** UE $i = 1,..., N$ **do**
4: **if** (12) is met **then**
5: Calculate $E_{ij}^{\mathrm{L}}(t)$, $E_{ij}^{\mathrm{Tr}}(t)$ and $f_{ij}^{\mathrm{C}}(t)$;
6: **if** $E_{ij}^{\mathrm{L}}(t) > E_{ij}^{\mathrm{Tr}}(t)$ **then**
7: Store i into $\boldsymbol{E}_j$;
8: **end if**
9: **end if**
10: **end for**
11: Sort the element in $\boldsymbol{E}_j$ in descending order with respect to $E_{ij}^{\mathrm{L}}(t) - E_{ij}^{\mathrm{Tr}}(t)$;
12: **end for**
13: **repeat**
14: **for** UAV $j = 1,..., M$ **do**
15: $i = GetTopItem(\boldsymbol{E}_j)$;
16: **if** (4), (23) are met **then**
17: **if** $E_{ij}^{\mathrm{Tr}}(t) < E_{i\boldsymbol{A}(i)}^{\mathrm{Tr}}(t)$ or $\boldsymbol{A}(i) = 0$ **then**
18: $\boldsymbol{A}(i) = j$;
19: **end if**
20: $RemoveTopItem(\boldsymbol{E}_j)$;
21: **end if**
22: **end for**
23: **until** Each UE in $\boldsymbol{E}_j$ is checked.
24: Return $\boldsymbol{A}$

According to [34], our RAT algorithm is an offline learning and off-policy DRL-based algorithm as the experience replay mechanism is applied, and the mini-batch will sample several uncorrelated experiences for training networks in each time step. Additionally, the training procedure can be deployed in a simulator, and the RAT model can be easily deployed in reality when the convergence is achieved, which will inevitably reduce the payoff of implementation. Furthermore, once the whole networks are converged, the solutions can be generated very fast with only some simple algebraic calculations instead of solving the original MINLP. This is due to the fact that during the training stages, random taking off points of all the UAVs are generated and the networks are trained to converge.

Discussions: after adequate training process, the RAT model, including the networks is saved for testing. In each time slot, the action of all UAVs is generated together by actor network. In our paper, as the fully-connected hidden layers are applied, the computational complexity for generating action of UAVs is $O\big(\sum_{l=1}^{L} n_l \cdot n_{l-1}\big)$, where L is the number of network layers, n_l is the number of neurons in the l-th layer. Then, the computational complexity of matching algorithm is $O(NM)$. The overall complexity of RAT algorithm in testing process is $O\big((\sum_{l=1}^{L} n_l \cdot n_{l-1} + NM)T\big)$.

VI. Extension to 3-D Channel Model

In this section, in order to consider the impacts of blockage and shadowing, we extend the free-space channel model to 3-D channel model that was proposed in [18], which is more practical. Additionally, we consider to optimize the 3-D UAV trajectories in this paper. Similarly, in each time slot, we assume the UAV can fly with a vertical direction $\theta_j^v(t) \in [0,\pi]$, a horizontal direction $\theta_j^h(t) \in [0,2\pi]$, and a flying distance $d_j(t) \in [0, d^{\max}]$. Thus, we define the coordinate of the j-th UAV in the t-th time slot as $[X_j(t), Y_j(t), Z_j(t)]$, where $X_j(t) = X_j(0) + \sum_{l=1}^{t} d_j(l)\sin\big(\theta_j^v(l)\big)\cos\big(\theta_j^h(l)\big)$, $Y_j(t) = X_j(0) + \sum_{l=1}^{t} d_j(l)\sin\big(\theta_j^v(l)\big)\sin\big(\theta_j^h(l)\big)$, $Z_j(t) = Z_j(0) + \sum_{l=1}^{t} \cos\big(\theta_j^v(l)\big)$, and $[X_j(0), Y_j(0), Z_j(0)]$ is the initial coordinate of the UAV.

For collision avoidance, one has

$$Z^{\min} \le Z_j(t) \le Z^{\max}, \forall t \in \mathcal{T}, \tag{46}$$

where $Z^{\min}$ and $Z^{\max}$ are the minimal and maximal flying altitude of the UAV.

Thus, the distance between the j-th UAV and the i-th UE in t-th time slot is given by

$$d_{ij}(t) = \sqrt{\big(X_j(t) - x_i\big)^2 + \big(Y_j(t) - x_i\big)^2 + Z_j(t)}, \quad \forall j \in \mathcal{M}, i \in \mathcal{N}, t \in \mathcal{T}. \tag{47}$$

The coverage radius of the j-th UAV in the t-th time slot can be given by

$$R_j^{\max}(t) = Z_j(t)\tan(\theta^{\max}). \tag{48}$$

The mean path loss between the j-th UAV and the i-th UE in the t-th time slot can be expressed as

$$L_{ij}(t) = \frac{\eta_{\text{LoS}} - \eta_{\text{NLoS}}}{1 + a\exp(-b(\theta_{ij}(t) - a))} + 20\log_{10}\big(d_{ij}(t)\big) + 20\log_{10}\big(\frac{4\pi f_c}{c}\big) + \eta_{\text{NLoS}}, \tag{49}$$

where η_{LoS}, η_{NLoS} are the path loss of achieving LoS and NLoS links, a and b are constant values that can be obtained in [18], $\theta_{ij}(t) = \arctan\left(\frac{Z_j(t)}{R_{ij}(t)}\right)$ is the elevation angle between the UAV and the UE, f_c is the carrier frequency, and c is the light speed.

Overall, we define the uplink data rate as follows:

$$r_{ij}(t) = B\log_2\left(1 + \frac{P^{\text{Tr}}}{\sigma^2} 10^{-\frac{L_{ij}(t)}{10}}\right). \tag{50}$$

Additionally, we consider to maximize the energy efficiency of UAVs in this paper. More precisely, motivated by [45], we introduce the power consumed by the j-th UAV in the t-th time slot as follows

$$P_j(t) = P_o\left(1 + 3\big(\frac{v_j(t)}{U_b}\big)^2\right) + P_s\left(\sqrt{1 + \frac{1}{4}\big(\frac{v_j(t)}{V_h}\big)^4} - \frac{1}{2}\big(\frac{v_j(t)}{V_h}\big)^2\right)^{\frac{1}{2}} + \frac{\pi}{2} d_0 \rho_a r_s R_r^2 v_j(t)^3 + wg v_j(t)\cos\big(\theta_j^v(t)\big), \tag{51}$$

where P_o and P_s are fixed constants that can be obtained in [45], U_b is the tip speed of the rotor blade, V_h denotes the mean rotor induced velocity when hovering, d_0 is the drag ratio of main body, ρ_a is the air density, r_s is the rotor solidity, R_r means the rotor radius, w is the weight of UAV, and g is the gravity acceleration.

Thus, the remaining energy of the j-th UAV in the t-th time slot is defined as

$$e_j(t) = e^{\max} - \sum_{l=1}^{t} P_j(l) T^{\max}, \tag{52}$$

in which, $e^{\max}$ is the maximal energy of each UAV.

Thus, the optimization problem can be written as follows:

$$\mathcal{P}1: \min_{\boldsymbol{U},\boldsymbol{A},\boldsymbol{F}} \sum_{t=1}^{T}\left(\sum_{j=0}^{M}\sum_{i=1}^{N} a_{ij}(t) E_{ij}(t) + k_z \sum_{j=1}^{M} P_j(t) T^{\max}\right) \tag{53a}$$

subject to: (24b), (24c), (24d), (24e), (24f), (24g), (24h), (24j), (24k),

$$0 \le \theta_j^v(t) \le \pi, \ \forall j \in \mathcal{M}, t \in \mathcal{T}, \tag{53b}$$

$$Z^{\min} \le Z_j(t) \le Z^{\max}, \ \forall j \in \mathcal{M}, t \in \mathcal{T}, \tag{53c}$$

$$a_{ij}(t) R_{ij}(t) \le R_j^{\max}(t), \ \forall i \in \mathcal{N}, j \in \mathcal{M}, t \in \mathcal{T}. \tag{53d}$$

where $\boldsymbol{U} = \{\theta_j^v(t), \theta_j^h(t), d_j(t), \ \forall j \in \mathcal{M}, t \in \mathcal{T}\}$, k_z is the impact factor.

Then, we define the state and action of our proposed RAT algorithm as follows:

1) **State** $s(t)$: $s(t) = \{[X_j(t), Y_j(t), Z_j(t), e_j(t)], \ \forall j \in \mathcal{M}\}$.
2) **Action** $c(t)$: the action set can be defined as $c(t) = \{[\theta_j^v(t), \theta_j^h(t), d_j(t)], \ \forall j \in \mathcal{M}\}$.

3) **Reward** $z(t)$: we define the reward as follows

$$z(t) = -\sum_{j=0}^{M}\sum_{i=1}^{N} a_{ij}(t)E_{ij}(t) - k_z\sum_{j=1}^{M} P_i(t)T^{\max} - p, \quad (54)$$

where p is the penalty if any of UAV flies out of the target area, which means (24g), (24h) or (53c) is not satisfied.

Thus, having defined the state, action and reward, the above problem can be solved by the proposed RAT algorithm.

VII. Simulation Results

In this section, both convex optimization based CAT and DRL based RAT are evaluated with simulations implemented on Intel i5-3450t, NVIDIA GTX 1050Ti, Python 3.6, PULP 1.6.10, CVXPY 1.1.7, and Tensorflow 1.15.0. We deploy three fully-connected hidden layers with 1024, 800 and 600 neurons in both actor and critic networks in RAT. The actor network is trained by applying RMSPropOptimizer with the learning rate 0.001, whereas the critic network is trained by using AdamOptimzer with the learning rate 0.001. In the simulation, we assume there are 60 time slots in each training epoch. There are 100 UEs randomly distributed in a rectangle-shaped area with the side length of $X^{\max}$ = 400 m and $Y^{\max}$ = 400 m. Additionally, there are 2 UAVs deployed to serve UEs within the target area. Note that for RAT, each UAV has 20 different taking off points during the training procedure. Besides, in each time slot, UE generates a task with communication requirement $D_i(t) \in [10, 50]$ KB and computation requirement $F_i(t) \in [2\times10^9, 2\times10^{10}]$ cycles. Other parameters are summarized in Table II. We assume in each time slot, UAVs will send a signal to activate the corresponding UEs, which will either offload the task or execute locally, within the delay requirement.

TABLE II: Simulation Parameters

Parameters	Settings	Parameters	Settings
T	60	N	100
M	2	$V^{\max}$	30
$d^{\max}$	30 m	$T^{\max}$	1 s
$X^{\max}$	400 m	$Y^{\max}$	400 m
$\theta^{\max}$	$\frac{\pi}{4}$	$Z_j(0)$	75 m
v_i	3	g_0	1.42×10^{-4}
P^{Tr}	0.1 W	B	10 MHz
σ^2	-90 dbm	$e^{\max}$	3×10^6 J
k_i	10^{-28}	$f^{\max}$	100 GHz
γ	0.999	p	100
$k^{\max}$	3000	ρ	2
w	2 kg	g	10 m/s^2
τ	0.001	$Z^{\min}$	50 m
$Z^{\max}$	120 m	η_{LoS}	1.6
η_{NLoS}	23	a	12.08
b	0.11	f_c	2.5 GHz
c	3×10^8 m/s	k_z	0.0005
P_o	79.86	U_b	120 m/s
P_s	88.63	V_h	4.03
d_0	0.6	ρ_a	1.25 kg/m^3
r_s	0.05	R_r	0.4 m

In order to evaluate the performance of the proposed CAT and RAT, we present the following three algorithms for comparison purpose.

- **Local Execution (LE):** All tasks are executed locally without offloading.
- **Random moving (RM):** In this setting, each UAV randomly selects the horizontal direction and flying distance to take.
- **Cluster moving (CM):** We group all the UEs into 10 clusters and each UAV flies in the trajectory connecting all the cluster center one by one. Note that it takes $\frac{T}{10}$ time slots for each UAV to move from one cluster center to another one.
- **Deep Deterministic Policy Gradient (DDPG) [34]:** We set the parameter of DDPG the same as actor and critic networks of RAT, but do not apply the prioritized experience replay. In other words, DDPG uniformly samples the experiences from the experience replay buffer in the training procedure.

Note that both RM, CM, DDPG apply the matching algorithm proposed in Algorithm 3 to decide the user association and resource allocation.

A. Convergence Evaluation of CAT and RAT

In this subsection, we show the convergence of proposed CAT and RAT. In Fig. 3, we show the convergence performance of CAT with three different pairs of initial trajectories. Specifically, we group all UEs into one cluster and the UAVs fly in a circle around the cluster center with radius 80 m, 100 m, and 120 m respectively. We denote these three pairs of UAV trajectories as the initial trajectories. As shown in Fig. 3, we can conclude that no matter which the initial trajectory is, the overall energy consumption of UEs achieved by CAT always decreases and finally remains stable after several iteration times. However, one can also observe that the convergent solution achieved by CAT will be influenced by the initial trajectory.

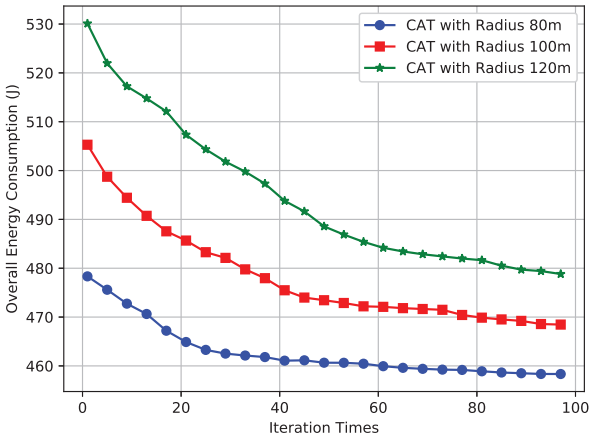
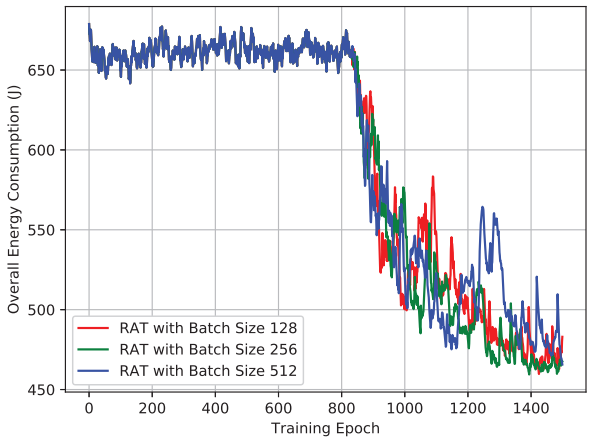
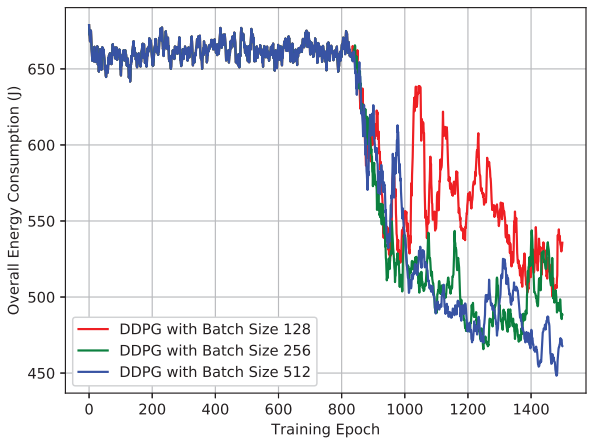


Fig. 3: The convergence performance of proposed CAT.

Then, we show the convergence performance of RAT in training process. From Fig. 4 to Fig. 5, we compare the influence of hyperparameters to both DDPG and RAT. Prioritized experience replay is applied in RAT. Both RAT and DDPG

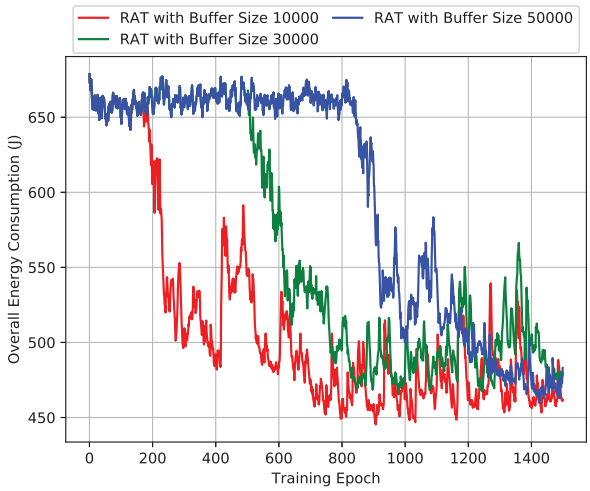


(a) The overall energy consumption of RAT with different batch size.

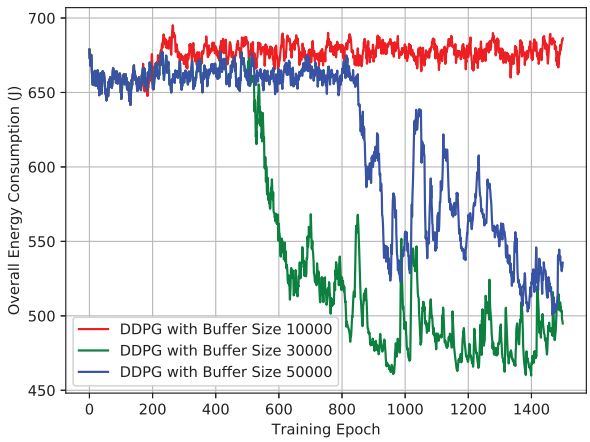


(b) The overall energy consumption of DDPG with different batch size.

Fig. 4: The convergence performance of RAT and DDPG with different size of mini-batch.



(a) The overall energy consumption of RAT with different buffer size.



(b) The overall energy consumption of DDPG with different buffer size.

Fig. 5: The convergence performance of RAT and DDPG with different experience replay buffer.

start the learning procedure once the experience replay buffer is full. In Fig. 4, we depict the overall energy consumption of RAT and DDPG for different size of mini-batches, where the size of experience replay buffer is 50000. To be more specific, from Fig. 4a, we can see that RAT has the similar convergence performance for different size of mini-batches and it becomes more stable during the learning procedure. In Fig. 4b, when the batch size is 128, DDPG has an obvious fluctuation during the learning procedure. When the batch size is 256, the convergence performance of DDPG becomes worse after the 1400-th epoch. While DDPG can only have a promising convergence performance when the batch size is 512. Overall, from Fig. 4, it is clear to see that the RAT is less sensitive to the change of mini-batch than DDPG.

In Fig. 5, we depict the overall energy consumption of RAT and DDPG for different sizes of experience replay buffer, where the size of mini-batch is set as 128. From Fig. 5a and 5b, when the buffer size is 10000, the proposed RAT finally remains stable between 450 J and 500 J, although it has an obvious fluctuation during the learning process. While DDPG has no convergence tendency during the entire learning procedure. When the buffer size is 50000, DDPG becomes worse after 1000-th epoch, and finally reaches 550 J. Overall, we can observe that DDPG can only have a promising performance when the buffer size is 30000, while RAT can always converge and remain stable during the learning procedure, no matter which the buffer size is. Thus, we can conclude that RAT is less sensitive to the size of experience replay buffer than DDPG.

B. Trajectory Evaluation of CAT and RAT

In Fig. 6 and Fig. 7, we show the trajectories obtained by RAT and CAT, respectively. Note that during the training procedure, the UAVs controlled by RAT always start to serve

UEs from 20 different takinkg off points. Additionally, for fairness, the UAVs controlled by CAT have the same taking off points as RAT. We group all the UEs into 6 clusters and each UAV flies in the trajectory connecting all the cluster center one by one. In this setting, we set these trajectories as the initial trajectories. Note that the iteration number of CAT is 10.

As shown in Fig. 6, we randomly select 5 pairs of taking off points for comparison. And we can observe that no matter which the taking off points of the UAVs are, the proposed RAT can guide the UAVs to their certain areas and move around to serve different UEs. This is due to the fact that we train the RAT to converge during the training stage by randomly generating many taking off points of the UAVs. Then, during the testing stage, RAT can intermediately output the best solutions once taking off points are input.

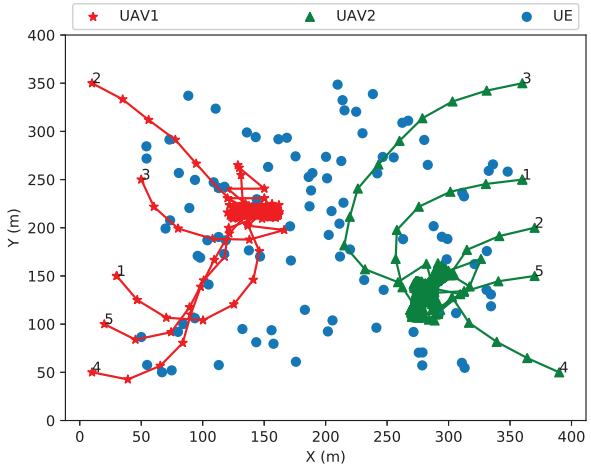


Fig. 6: Multi-UAV enabled F-MEC architecture controlled by RAT.

While in Fig. 7, no matter where the taking off points of the UAVs are, the trajectories obtained by CAT are always similar with the initial trajectories. Also, note that unlike CAT which may fall into the local optimum, the proposed RAT has the global search ability due to the application of reinforcement learning techniques.

C. Energy Consumption Evaluation of CAT and RAT

In Fig. 8, we compare the performance of RAT, CAT, CM, RM, and LE in terms of energy consumption of UEs. As shown in Fig. 8a, we depict the overall energy consumption of UEs achieved by RAT, CAT, CM, RM, and LE with different taking off points. It is obvious to see that LE has the worst performance. This is because all UEs have to execute their tasks locally without offloading, which will inevitably consume more energy. RM outperforms LE but it fluctuates with the index of taking off points. CM has better performance than RM, which always remains between 520 J and 550 J. CAT outperforms LE, RM, and CM, which remains about 500 J. Additionally, one can observe that RAT achieves the optimal performance. One plausible explanation is that the networks in

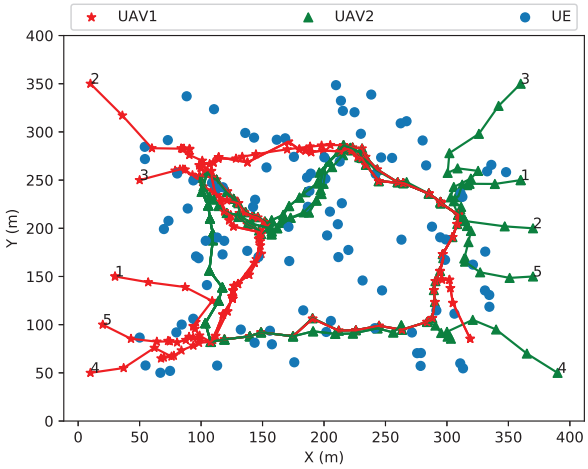


Fig. 7: Multi-UAV enabled F-MEC architecture controlled by CAT.

RAT are convergent after adequate training, and it can output the optimal solutions in real-time, no matter which the taking off points are.

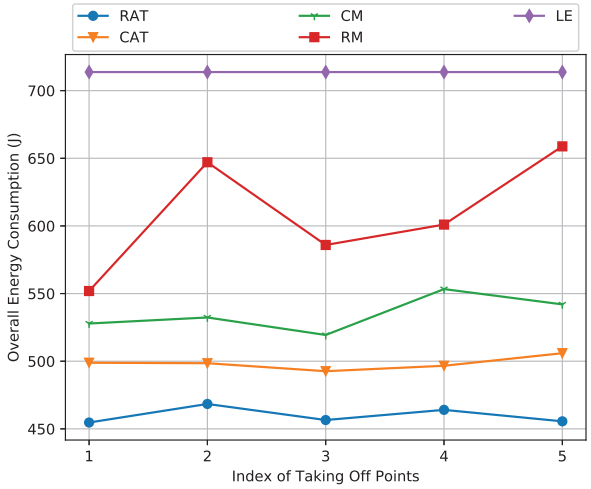
Furthermore, we set the index of taking off points as 1, and depict the overall energy consumption of UEs achieved by RAT, CAT, RM, CM, and LE in different number of time slots in Fig. 8b. It is easy to see that both the energy consumption of RAT, CAT, RM, CM, and LE increase as the number of time slots increases. LE still performs the worst, which consumes above 700 J eventually. Additionally, we can observe that RAT consistently outperforms other algorithms. This is because the UAVs controlled by RAT always serve UEs with the optimal solutions. CAT still has considerable performance, which is only slightly worse than RAT.

In Table III, we show the time consumed by CAT and RAT for each pair of taking off points in Fig. 8. Note that RAT is trained for 3000 epochs, while the iteration number of CAT is 10. One can see that no matter which the index of taking off point is, the proposed CAT takes over 1400 seconds to find solutions, while RAT only takes 1.2 seconds in average, although it takes longer time in training process. This is because the networks of RAT are trained to converge, and the RAT only needs a few number of algebra calculations once the training is completed offline.

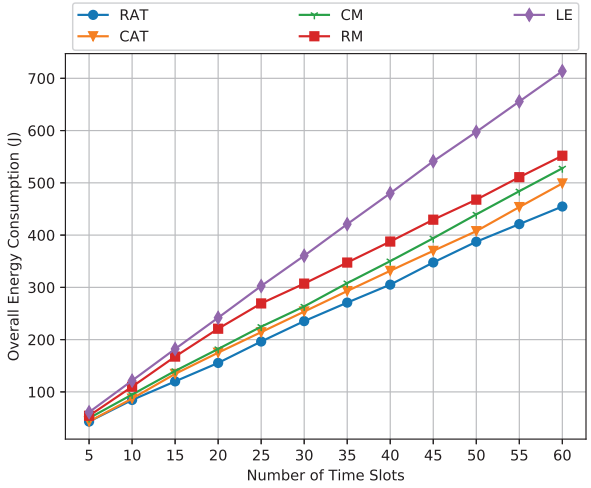
TABLE III: Executed Time of CAT and RAT

Index	**CAT** (s)	**RAT**	
		Training (s)	**Testing** (s)
1	1405.23	10534.88	1.23
2	1491.74		1.22
3	1460.46		1.20
4	1445.11		1.21
5	1402.48		1.21

Furthermore, we analyze the performance of proposed RAT in 3-D UAV trajectory and 3-D channel model scenario. In this setting, we set the number of time slots T as 50, the channel bandwidth as 20 MHz, $D_i(t) \in [5, 10]$ KB, $F_i(t) \in [7.5 \times$



(a) The overall energy consumption of RAT, CAT, RM, CM, LE with different taking off points.



(b) The overall energy consumption of RAT, CAT, RM, CM, LE in different number of time slots.

Fig. 8: The performance comparison of RAT, CAT, RM, CM, and LE.

$10^8, 2\times10^9$] cycles, the size of mini-batch is 512, and the size of experience replay buffer is 100000. In each training epoch, each UAV starts to serve UEs with the altitude of $Z_j(0)$ = 50 m. First of all, we depict the overall energy consumption achieved by the proposed RAT algorithm during the training procedure in Fig. 9. One can observe that the overall energy consumption of UEs firstly remains between 600 J and 700 J. When the learning process starts, the curve decreases and eventually remains above 400 J, which means the convergence is achieved.

Then, we depict the UAV trajectories obtained by RAT during testing phase in Fig. 10. Note that blue dots represent UEs, red stars are UAV1, and green triangles denote UAV2. As shown in Fig. 10, we can observe that no matter which the taking off points are, the UAVs always move from their taking off points to the certain areas, and move around to serve different UEs with the most sufficient distance. In addition,

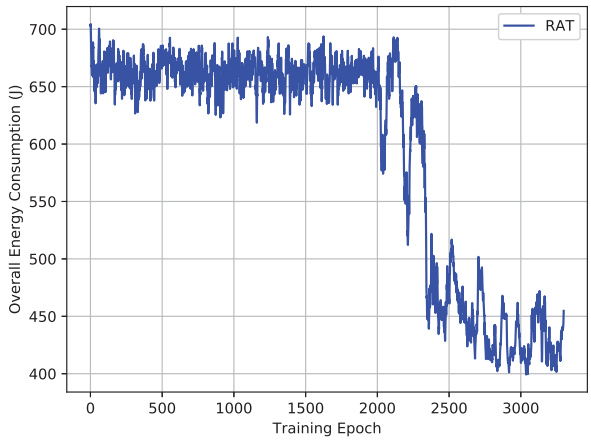


Fig. 9: The convergence performance of proposed RAT in 3-D UAV trajectory and 3-D channel model scenario.

one can observe that each UAV will ascend its altitude in the beginning, this is because higher altitude will increase the coverage radius of the UAV, which will inevitably serve more UEs, although it will also decrease the data rate of offloading.

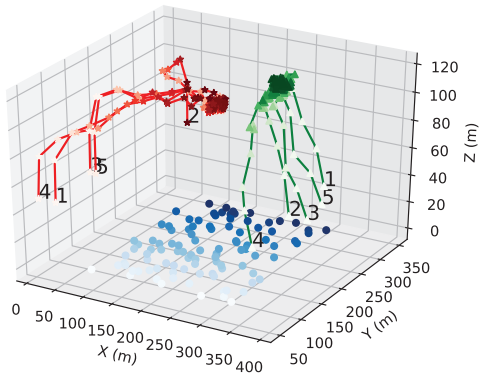
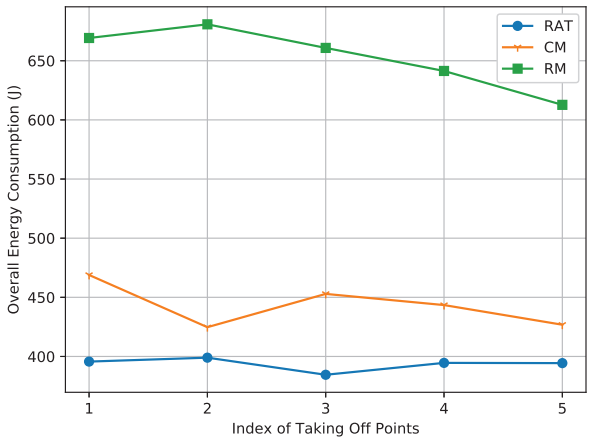


Fig. 10: Multi-UAV enabled F-MEC trajectories obtained by RAT in 3-D channel model scenario (blue dots for UEs, red stars for UAV1, green triangles for UAV2).

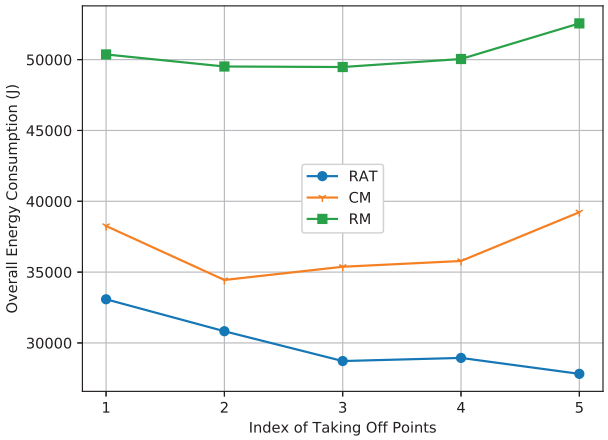
Furthermore, we analyze the overall energy consumption of UEs and UAVs achieved by RAT, CM, and RM in different pair of taking off points in Fig. 11, where the UAVs controlled by CM always fly with the fixed altitude value of 100 m, while RM selects the available flying action for each UAV, including the horizontal flying direction, the vertical flying direction, and the flying distance. More precisely, in Fig. 11a, we can observe that our proposed RAT consistently outperforms CM and RM, whose energy consumption remains under 400 J. While CM performs worse than RAT and better than RM, whose energy consumption keeps at 450 J.

Finally, we show the overall energy consumption of UAVs achieved by RAT, CM and RM in Fig. 11b. We observe that

our proposed RAT algorithm has the optimal performance, no matter which the index of taking off points are. CM has the worse performance than RAT, which consumes 35000 J at least. However RM is the worst.



(a) The overall energy consumption of UEs achieved by RAT, CM, and RM with different taking off points.



(b) The overall energy consumption of UAVs achieved by RAT, CM, and RM with different taking off points.

Fig. 11: The performance comparison of RAT, CM, and RM.

VIII. Conclusion

In this paper, we consider the flying mobile edge computing architecture, by taking advantage of the UAVs to serve as the moving platform. We aim to minimize the energy consumption of all the UEs by optimizing the UAVs' trajectories, user associations and resource allocation. To tackle the multi-UAVs' trajectories control problem, a convex optimization-based CAT was first proposed. Then, in order to conduct fast decision, a DRL-based RAT including a matching algorithm was also proposed. Simulation results show that CAT and RAT have considerable performance. Additionally, RAT also outperforms other traditional algorithms.

References

[1] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[2] Y. Du, K. Wang, K. Yang, and G. Zhang, "Energy-efficient resource allocation in UAV based MEC system for IoT devices," in *IEEE Global Communications Conference*, 2018, pp. 1–6.

[3] X. Lyu, H. Tian, W. Ni, Y. Zhang, P. Zhang, and R. P. Liu, "Energy-efficient admission of delay-sensitive tasks for mobile edge computing," *IEEE Trans. Commun.*, vol. 66, no. 6, pp. 2603–2616, June. 2018.

[4] Q. Wu and R. Zhang, "Common throughput maximization in UAV-enabled OFDMA systems with delay consideration," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6614–6627, Dec. 2018.

[5] Z. Li, M. Chen, C. Pan, N. Huang, Z. Yang, and A. Nallanathan, "Joint trajectory and communication design for secure UAV networks," *IEEE Commun. Lett.*, pp. 1–4, Feb. 2019.

[6] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE J. Select. Areas Commun.*, vol. 36, no. 9, pp. 2059–2070, Sep. 2018.

[7] L. Kong, L. Ye, F. Wu, M. Tao, G. Chen, and A. V. Vasilakos, "Autonomous relay for millimeter-wave wireless communications," *IEEE J. Select. Areas Commun.*, vol. 35, no. 9, pp. 2127–2136, 2017.

[8] P. Zhan, K. Yu, and A. L. Swindlehurst, "Wireless relay communications with unmanned aerial vehicles: Performance and optimization," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 3, pp. 2068–2085, July 2011.

[9] R. Fan, J. Cui, S. Jin, K. Yang, and J. An, "Optimal node placement and resource allocation for UAV relaying network," *IEEE Communications Letters*, vol. 22, no. 4, pp. 808–811, 2018.

[10] U. Challita, A. Ferdowsi, M. Chen, and W. Saad, "Machine learning for wireless connectivity and security of cellular-connected UAVs," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 28–35, 2019.

[11] C. Zhan, Y. Zeng, and R. Zhang, "Energy-efficient data collection in UAV enabled wireless sensor network," *IEEE Wireless Communications Letters*, vol. 7, no. 3, pp. 328–331, June 2018.

[12] J. Gong, T.-H. Chang, C. Shen, and X. Chen, "Aviation time minimization of UAV for data collection from energy constrained sensor networks," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2018, pp. 1–6.

[13] J. Lyu, Y. Zeng, and R. Zhang, "UAV-aided offloading for cellular hotspot," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 3988–4001, 2018.

[14] J. Gu, T. Su, Q. Wang, X. Du, and M. Guizani, "Multiple moving targets surveillance based on a cooperative network for multi-UAV," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 82–89, 2018.

[15] J. Xu, Y. Zeng, and R. Zhang, "UAV-enabled wireless power transfer: Trajectory design and energy optimization," *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5092–5106, Aug 2018.

[16] N. Zhao, F. Cheng, F. R. Yu, J. Tang, Y. Chen, G. Gui, and H. Sari, "Caching UAV assisted secure transmission in hyper-dense networks based on interference alignment," *IEEE Transactions on Communications*, vol. 66, no. 5, pp. 2281–2294, 2018.

[17] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Unmanned aerial vehicle with underlaid device-to-device communications: Performance and tradeoffs," *IEEE Transactions on Wireless Communications*, vol. 15, no. 6, pp. 3949–3963, 2016.

[18] 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.

[19] H. He, S. Zhang, Y. Zeng, and R. Zhang, "Joint altitude and beamwidth optimization for UAV-enabled multiuser communications," *IEEE Commun. Lett.*, vol. 22, no. 2, pp. 344–347, Feb. 2018.

[20] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, June 2017.

[21] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-UAV enabled wireless networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 2109–2121, March 2018.

[22] Z. Yang, C. Pan, K. Wang, and M. Shikh-Bahaei, "Energy efficient resource allocation in UAV-enabled mobile edge computing networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 9, p. 4576–4589, Sep 2019. [Online]. Available: http://dx.doi.org/10.1109/twc.2019.2927313

[23] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Mobile unmanned aerial vehicles (UAVs) for energy-efficient internet of things communications," *IEEE Transactions on Wireless Communications*, vol. 16, no. 11, pp. 7574–7589, 2017.
[24] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.
[25] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 16, no. 8, pp. 4924–4938, Aug 2017.
[26] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.
[27] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 2049–2063, March 2018.
[28] 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 Transactions on Vehicular Technology*, pp. 1–14, 2019.
[29] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 1927–1941, 2018.
[30] A. Asheralieva and D. Niyato, "Hierarchical game-theoretic and reinforcement learning framework for computational offloading in UAV-enabled mobile edge computing networks with multiple service providers," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8753–8769, 2019.
[31] Q. Zhang, J. Chen, L. Ji, Z. Feng, Z. Han, and Z. Chen, "Response delay optimization in mobile edge computing enabled UAV swarm," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3280–3295, 2020.
[32] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529–533, Feb. 2015.
[33] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," 2015.
[34] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
[35] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, Nov. 2015.
[36] F. Jiang, K. Wang, L. Dong, C. Pan, W. Xu, and K. Yang, "Deep learning based joint resource scheduling algorithms for hybrid MEC networks," *IEEE Internet of Things Journal*, 2019.
[37] X. Wang, K. Wang, S. Wu, S. Di, H. Jin, K. Yang, and S. Ou, "Dynamic Resource Scheduling in Mobile Edge Cloud with Cloud Radio Access Network," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 11, pp. 2429–2445, Nov. 2018.
[38] S. Mitchell, M. G. O. Sullivan, and I. Dunning, "Pulp : A linear programming toolkit for python," in *Python*, 2011.
[39] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.
[40] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.
[41] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Thirtieth AAAI Conference on Artificial Intelligence*, Mar. 2016.
[42] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
[43] J. Hamari, J. Koivisto, H. Sarsa *et al.*, "Does gamification work?-a literature review of empirical studies on gamification." in *HICSS*, vol. 14, no. 2014, 2014, pp. 3025–3034.
[44] A. R. Mahmood, H. P. Van Hasselt, and R. S. Sutton, "Weighted importance sampling for off-policy learning with linear function approximation," in *Advances in Neural Information Processing Systems*, 2014, pp. 3014–3022.
[45] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2329–2345, 2019.