Research Article

Deep Reinforcement Learning-Based Dynamic Offloading Management in UAV-Assisted MEC System

Kang Tian ¹, Yameng Liu ¹, Haojun Chai ¹ and Boyang Liu ²

¹China Mobile System Integration Co., Ltd., Xi'an 710000, China

²College of Communications and Information Engineering, Xi'an University of Posts and Telecommunications, Xi'an 710000, China

Correspondence should be addressed to Boyang Liu; boyangliulby@163.com

Received 10 January 2022; Accepted 21 January 2022; Published 16 February 2022

Academic Editor: Xin Ning

Copyright © 2022 Kang Tian et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Unmanned aerial vehicles (UAVs) have been envisioned as a promising technique to provide relaying and mobile edge computing (MEC) services for ground user equipment (UE). In this paper, we propose a UAV-assisted MEC architecture in dynamic environment, where a UAV flies with a fixed trajectory and may act as a MEC server to process the tasks offloaded from the UE or act as a relay to help the UE to offload their tasks to the ground base station (BS). The objective of this work is to maximize the long-term number of completed tasks of the UE. An optimization problem is formulated to optimize the task offloading decisions of the UE. Considering the random demands of the UE, a deep reinforcement learning- (DRL-) based algorithm is proposed to solve the formulated nonconvex optimization problem. Simulation results verify the effectiveness and correctness of the proposed algorithm.

1. Introduction

The smart city paradigm brings massive computation-intensive and latency-sensitive applications, such as augmented reality (AR), virtual reality (VR), and video compression [1–3]. Due to the limited computing capability, it is hard for the user equipment (UE) to complete above applications in time without extra computing service. Although the mobile cloud computing can provide powerful computing service, the computation tasks needed to be transmitted to the remote cloud server, which may cause severe transmission delay and it is difficult for it to meet the delay demands of the applications. To address this issue, mobile edge computing (MEC) technique emerges.

In MEC systems, MEC servers are deployed at the edge of the network to establish direct connection between the UE and the servers to provide real-time computing service for the UE [4]. The UE offloads computation tasks to the MEC servers through wireless channels. In traditional MEC architectures, MEC servers are typically deployed in fixed locations, such as being equipped at base stations (BS) or at

the access points (AP). The quality of the communication links between the UE and MEC servers is easily influenced by dense buildings, which may lead to weak transmission signals or even interruption. This circumstance is especially serious in 5G scenarios due to the transmission characteristics of millimeter waves.

Equipping the MEC server at unmanned aerial vehicle (UAV) can alleviate the problem stated above with its advantages of easy deployment, high flexibility, and maneuverability. The UAVs fly with predetermined trajectories in the air and have higher probability to establish line-of-sight (LoS) communication links with the ground UE. Hence, it is a great attempt to incorporate the UAVs in MEC systems to enlarge the MEC service coverage range and enhance the service capacity at the same time.

To prolong the endurance of the UAV, the authors in [5] studied an energy consumption minimization problem; the scheduling, resource allocation, and the hovering time of the UAV were jointly designed. The authors in [6] jointly optimized the bit allocation and communications for uplink and downlink and the trajectory of the UAV to minimize the

total mobile energy consumption. In [7], a UAV enabled MEC network was considered; the UAV position, time slot allocation, and computation task partition were jointly optimized to minimize the sum of energy consumption of ground users. In [8], the computing scheduling, 3D trajectory of the UAV, bandwidth, and transmission power allocation were jointly optimized to maximize the computation efficiency. In [9], the sum of the maximum delay among all the users was minimized by jointly optimizing the trajectory of the UAV, offloading task partition, and computing scheduling of the users.

In [10], the computation rate was maximized by jointly optimizing the CPU frequencies, the length of the offloading times, the transmit power, and the trajectory of the UAV under both partial and binary computation offloading modes. A computation efficiency maximization problem was formulated in [11], and the user association, CPU frequency, spectrum resources, and the UAV trajectory were jointly optimized.

Efforts in [5–11] focused on how to adopt the UAVs to provide computation services to the ground UE. In [12,13], the UAV was utilized not only as a MEC server but also as a relay to establish high-quality LoS wireless links between the UE and the remote ground high-performance MEC servers. The authors in [12] designed the trajectory of the UAV and the transmit power of the UE and the UAV to minimize the outage probability of the considered UAV-assisted MEC network. In [13], the computation resource scheduling parameters and the trajectory of the UAV were jointly optimized to minimize the weighted sum energy consumption subject to the task and the information causality, the bandwidth allocation, and the UAV's trajectory constraints. In [14], the bits allocation, time slot scheduling, power allocation, and the trajectory of the UAV were jointly optimized to minimize the total energy consumption of the MEC network.

The works mentioned above mainly concerned the resource allocation and trajectory design problems in static environment. However, the task arrivals or the locations of the ground UE may be randomly changed in practical scenarios [15–19]. Therefore, the dynamic environment should be taken into account during the design process of the UAV-assisted MEC system.

In [15], the motion of the UE was considered, and a double Q-network based deep reinforcement learning (DRL) algorithm was proposed to maximize the system reward and meet the QoS constraint by optimizing the trajectory of the UAV and the association of the UE. In [16], a multi-UAV-assisted MEC system was studied, where multiple UAVs helped the BS for computing the tasks offloaded from the UE. An average mission response time minimization problem was formulated and solved by a conceived multi-agent reinforcement learning algorithm. In [17], the task scheduling was optimized by a DRL-based method to balance the workload among multiple UAVs while guaranteeing the coverage constraint and the quality of service (QoS) demands of the UE. In [18], a three-layer online data processing network was proposed; the bottom layer was composed of sensors to generate raw data. The middle layer included multiple UAVs to collect and preprocess the data. The top layer was responsible for relieving the processed

results and conducted further evaluation. Lyapunov optimization and DRL-based algorithms were proposed to schedule edge processing and plan the path online. In [19], the multiagent deep deterministic policy gradient method was adopted to design the trajectory of UAVs to jointly optimize the fairness among all the UE, the UAV's UE-load, and the total energy consumption of the UE.

It is worth noting that, due to the short battery life and the limited physical size, the UAVs cannot equip too powerful MEC servers. Hence, it is not practical to just rely on the UAVs to process all the tasks of the UE. The role of the UAVs in MEC systems should be a helper of the UE. The UAVs and the ground MEC servers (BS or AP) with strong computing power should coexist in UAV-assisted MEC systems. The main function of the UAVs should be establishing a communication link between the UE and the ground MEC servers. The secondary function of the UAVs is to provide limited computing service for the UE. Motivated by all facts analysed above, we investigate the task allocation problem in UAV-assisted MEC system in dynamic environment. The main contributions are summarized as follows:

- (1) We propose a UAV-assisted MEC system in dynamic environment, where a UAV is deployed to provide relaying and edge computing service to the ground UE with randomly computation task arrivals at the beginning of each time slot.
- (2) We formulate a long-term number of completed tasks maximization problem for the proposed UAV-assisted MEC system, which is nonconvex and difficult to solve. A DRL-based algorithm is proposed to solve the formulated problem.
- (3) Simulation results are presented to evaluate the performance of the proposed algorithm. The convergence and effectiveness of the proposed algorithm are demonstrated by the simulation results.

The remainder of this paper is organized as follows. Section 2 presents the system model and formulates the optimization problem. The DRL-based optimization algorithm is described in Section 3. Simulation results are presented and discussed in Section 4. Finally, we conclude this study in Section 5.

2. System Model

In this section, we describe the system model and formulate a long-term number of completed tasks' maximization problem. The key notations used in this paper are summarized in Table 1.

2.1. Network Model. As shown in Figure 1, we consider a UAV-assisted MEC system with one ground BS, one UAV, and N ground UE. A MEC server with strong computing power is equipped at the BS. Let $\mathcal{N} = \{1, 2, \dots, N\}$ denote the set of UE. The system is assumed to be operated in a time-slotted manner with time slot length Δ . We are concerned about the long-term utility during T consecutive time slots. The set of time slots is denoted as $\mathcal{T} = \{1, 2, \dots, T\}$.

TABLE 1: Key notations used in this paper.

Notion	Description
$\theta_n^{\text{UE}}(t)$	Computation task arrived at the UE n at time slot $t, t \in \mathcal{T}$
$L(x)$	Input data size of computation task x
$\tau(x)$	Maximal tolerable latency of computation task x
$C(x)$	The number of CPU cycles required for computing 1 bit of the task x
$\theta_n^{\text{UE}}(t_{n,b}^{\text{Comp}})$	The first task processed at the UE n
$\theta_n^{\text{UE}}(t_{n,b}^{\text{Tr}})$	The first task to be offloaded by the UE n
$\theta_n^{\text{UE}}(t_{\text{UAV},n,b}^{\text{Comp}})$	The first task of the UE n processed by the UAV
$\theta_n^{\text{UE}}(t_{\text{UAV},n,b}^{\text{Tr}})$	The first task of the UE n offloaded by the UAV
$\mu_{n,\text{Tr}}^{\text{UE}}(x)$	The index of time slot that the task x begins to be offloaded to the UAV by the UE n
$\mu_{\text{Tr}}^{\text{UAV}}(x)$	The index of time slot that the task x of the UE n begins to be offloaded to the BS by the UAV
$T_{n,l}^{\text{Comp}}(x)$	The number of time slots required for computing the task x of the UE n at local
$T_{\text{UAV},n}^{\text{Comp}}(x)$	The number of time slots required for computing the task x of the UE n by the UAV
$T_n^{\text{Tr}}(x)$	The number of time slots required for offloading the task x to the UAV by the UE n
$T_{\text{UAV},n}^{\text{Tr}}(x)$	The number of time slots required for offloading the task x of the UE n from the UAV to the BS
$T_{n,\text{Comp}}^{\text{Wait}}(x)$	The number of time slots the task x of the UE n will wait for local processing
$T_{n,\text{Tr}}^{\text{Wait}}(x)$	The number of time slots the task x will wait for offloading at the UE n
$T_{\text{UAV},n,\text{Comp}}^{\text{Wait}}(x)$	The number of time slots that the task x of the UE n will wait for processing at the UAV
$T_{\text{UAV},n,\text{Tr}}^{\text{Wait}}(x)$	The number of time slots the task x of the UE n will wait for offloading at the UAV
$T_{n,\text{end}}^{\text{Comp}}(x)$	The index of time slot that the computation of the task x of the UE n is completed by local computing
$T_{\text{UAV},n,\text{end}}^{\text{Comp}}(x)$	The index of time slot that the computation of the task x of the UE n is completed at the UAV
$T_{n,\text{end}}^{\text{Tr}}(x)$	The index of time slot that the offloading of the task x is completed at the UE n
$T_{\text{UAV},n,\text{end}}^{\text{Tr}}(x)$	The index of time slot that the offloading of the task x of the UE n is completed by the UAV
$T_{n,l,\text{total}}^{\text{Comp}}(x)$	The number of total time slots required for completing the task x 's computation by the UE n
$T_{\text{UE},n,\text{total}}^{\text{Tr}}(x)$	The number of total time slots required for completing the offloading of the task x by the UE n
$T_{\text{UAV},\text{total}}^{\text{Comp}}(x)$	The number of total time slots required for completing the computation of task x of the UE n by the UAV
$T_{\text{BS},\text{total}}^{\text{Comp}}(x)$	The number of total time slots required for completing the computation of task x of the UE n by the BS
$T_{n,l,\text{total},\text{true}}^{\text{Comp}}(x)$	The number of total time slots consumed for completing task x by the UE n
$T_{\text{UAV},\text{total},\text{true}}^{\text{Comp}}(x)$	The number of total time slots consumed for completing the computation of the task x of the UE n by the UAV
$T_{\text{BS},\text{total},\text{true}}^{\text{Comp}}(x)$	The number of total time slots consumed for completing the computation of the task x of the UE n by the BS

In this paper, the three-dimensional Cartesian coordinate system is adopted. The BS and UE are fixed on the ground with zero altitude. The horizontal locations of the BS and the UE $n, n \in \mathcal{N}$ are denoted as $l^{\text{BS}} = (x_b, y_b)$ and $l_n^{\text{UE}} = (x_n, y_n)$, respectively. The altitude of the UAV is set to H , which is appropriate to avoid buildings in the work terrain. The UAV flies with a predetermined trajectory. The horizontal location of the UAV in time slot $t, t \in \mathcal{T}$ is denoted as $l^{\text{UAV}}(t) = [x_U(t), y_U(t)]$. As the length of each time slot is relatively short, the location of the UAV can be considered unchanged during each time slot, which is similar to [13].

2.2. Channel Model. In this paper, we consider the case that there are no direct links between the UE and the BS owing to the long distances and blockages between them. Similar to [9, 13, 14, 20, 21], we assume the air-ground wireless channels are dominated by LoS links. The channel power gains from the UAV to the ground BS and from the UAV to the UE $n, n \in \mathcal{N}$ at time slot $t, t \in \mathcal{T}$ are, respectively, given as

$$g_{\text{BS}}(t) = \frac{\rho_0}{\|l^{\text{UAV}}(t) - l^{\text{BS}}\|^2 + H^2}, \quad n \in \mathcal{N}, t \in \mathcal{T}, \quad (1a)$$

$$h_n(t) = \frac{\rho_n}{\|l^{\text{UAV}}(t) - l_n^{\text{UE}}\|^2 + H^2}, \quad n \in \mathcal{N}, t \in \mathcal{T}, \quad (1b)$$

where $\rho_n, n = 0, 1, \dots, N$ denote the channel power gain with reference distance 1 m.

2.3. Computation Model. At the beginning of each time slot $t, t \in \mathcal{T}$, each user $n, n \in \mathcal{N}$ has a new task arrival $\theta_n^{\text{UE}}(t)$, which can be represented by a 3-tuple $(L[\theta_n^{\text{UE}}(t)], C[\theta_n^{\text{UE}}(t)], \tau[\theta_n^{\text{UE}}(t)])$, where $L[\theta_n^{\text{UE}}(t)]$ denotes the size (in bits) of task $\theta_n^{\text{UE}}(t)$, $C[\theta_n^{\text{UE}}(t)]$ denotes the computation complexity (i.e., the number of required CPU cycles for computing 1 bit of task $\theta_n^{\text{UE}}(t)$) of the task $\theta_n^{\text{UE}}(t)$, and $\tau[\theta_n^{\text{UE}}(t)]$ is the maximal tolerable latency of the task $\theta_n^{\text{UE}}(t)$. We consider a random task arrival scenario: $L[\theta_n^{\text{UE}}(t)]$ is randomly generated from a discrete set with M_n elements $\Theta_{L_n} = \{L_{n,1}, L_{n,2}, \dots, L_{n,M_n}\}$. The corresponding task complexity set and maximal tolerable latency set of the task

FIGURE 1: The illustration of the UAV-assisted MEC system.

$\theta_n^{\text{UE}}(t)$ are denoted as $\Theta_{C_n} = \{C_{n,1}, C_{n,2}, \dots, C_{n,M_n}\}$ and $\Theta_{\tau_n} = \{\tau_{n,1}, \tau_{n,2}, \dots, \tau_{n,M_n}\}$, respectively.

At the beginning of each time slot, the UE decides to execute the task $\theta_n^{\text{UE}}(t)$ either at local, offloading to the UAV, or relaying to the BS for processing. Let $\alpha_{n,t} \in \{0, -1, 1\}$, $n \in \mathcal{N}$, $t \in \mathcal{T}$ denote the task allocation variable. If $\alpha_{n,t} = 0$, the task $\theta_n^{\text{UE}}(t)$ is processed locally by the UE n , $n \in \mathcal{N}$; if $\alpha_{n,t} = -1$, the task $\theta_n^{\text{UE}}(t)$ is offloaded to the UAV for processing; otherwise, if $\alpha_{n,t} = 1$, the task $\theta_n^{\text{UE}}(t)$ is relayed by the UAV to the BS for processing. Similar to [22], we assume that if the task processing period exceeds its maximal tolerable latency, the task must be dropped immediately.

All the UE and the UAV keep two queues for each UE: computation queue and transmission queue, which are operated with a first-in-first-out (FIFO) manner. If $\alpha_{n,t} = 0$, the task $\theta_n^{\text{UE}}(t)$ will be stored into the computation queue at the UE n , $n \in \mathcal{N}$ and wait for processing. If $\alpha_{n,t} = \pm 1$, the task $\theta_n^{\text{UE}}(t)$ will be stored into the transmission queue at the UE n , $n \in \mathcal{N}$ and wait for offloading to the UAV. If $\alpha_{n,t} = -1$, after receiving task $\theta_n^{\text{UE}}(t)$, the UAV will store $\theta_n^{\text{UE}}(t)$ into the UE n 's computation queue

at the UAV and wait for processing. If $\alpha_{n,t} = 1$, after receiving task $\theta_n^{\text{UE}}(t)$, the UAV stores $\theta_n^{\text{UE}}(t)$ into the UE n , $n \in \mathcal{N}$'s transmission queue at the UAV and wait for relaying to the BS.

2.3.1. Local Computing at the UE n , $n \in \mathcal{N}$. The number of time slots required for computing the task $\theta_n^{\text{UE}}(t)$ at local is expressed as

$$T_{n,l}^{\text{Comp}}[\theta_n^{\text{UE}}(t)] = \lceil \frac{L[\theta_n^{\text{UE}}(t)]C[\theta_n^{\text{UE}}(t)]}{f_n^l \Delta} \rceil, \quad n \in \mathcal{N}, t \in \mathcal{T}, \quad (2)$$

where $\lceil \cdot \rceil$ is the ceiling function, and f_n^l is the CPU frequency of the UE n , $n \in \mathcal{N}$. In practice, the task $\theta_n^{\text{UE}}(t)$ may not be completed within one time slot. The subsequent tasks of the UE n , $n \in \mathcal{N}$ may wait for processing. Let $T_{n,\text{Comp}}^{\text{Wait}}[\theta_n^{\text{UE}}(t)]$ denote the number of time slots the task $\theta_n^{\text{UE}}(t)$ will wait for local processing, which is given as

$$T_{n,\text{Comp}}^{\text{Wait}}[\theta_n^{\text{UE}}(t)] = \left[\max_{t' \in \{0,1,\dots,t-1\}} T_{n,\text{end}}^{\text{Comp}}[\theta_n^{\text{UE}}(t')] - t + 1 \right]^+, \quad n \in \mathcal{N}, t \in \mathcal{T}, \quad (3)$$

where $[x]^+ = \max(0, x)$; $T_{n,\text{end}}^{\text{Comp}}[\theta_n^{\text{UE}}(t')]$ is the index of time slot that the computation of the task $\theta_n^{\text{UE}}(t')$ is completed, which is given as

$$T_{n,\text{end}}^{\text{Comp}}[\theta_n^{\text{UE}}(t')] = T_{n,\text{Comp}}^{\text{Wait}}[\theta_n^{\text{UE}}(t')] + T_{n,l}^{\text{Comp}}[\theta_n^{\text{UE}}(t')] + t' - 1, \quad n \in \mathcal{N}, t' \in \mathcal{T}. \quad (4)$$

Recursive method can be used to obtain the value of $T_{n,\text{end}}^{\text{Comp}}[\theta_n^{\text{UE}}(t')]$ with $T_{n,\text{Comp}}^{\text{Wait}}[\theta_n^{\text{UE}}(t_{n,b}^{\text{Comp}})] = 0$, where $\theta_n^{\text{UE}}(t_{n,b}^{\text{Comp}})$ is the first task processed at the UE $n, n \in \mathcal{N}$. Then, the number of total time slots required for completing the task $\theta_n^{\text{UE}}(t)$ by the UE $n, n \in \mathcal{N}$ is given as

$$T_{n,l,\text{total}}^{\text{Comp}}[\theta_n^{\text{UE}}(t)] = T_{n,\text{Comp}}^{\text{Wait}}[\theta_n^{\text{UE}}(t)] + T_{n,l}^{\text{Comp}}[\theta_n^{\text{UE}}(t)], \quad n \in \mathcal{N}, t \in \mathcal{T}. \quad (5)$$

However, as mentioned above, if the task processing period of $\theta_n^{\text{UE}}(t)$ exceeds the maximal tolerable latency, $\theta_n^{\text{UE}}(t)$ must be dropped immediately. Hence, the number of total time slots consumed for completing task $\theta_n^{\text{UE}}(t), n \in \mathcal{N}$ (including the time for waiting and processing) is given as

$$T_{n,l,\text{total},\text{true}}^{\text{Comp}}[\theta_n^{\text{UE}}(t)] = \min\{T_{n,l,\text{total}}^{\text{Comp}}[\theta_n^{\text{UE}}(t)], \tau[\theta_n^{\text{UE}}(t)]\}, \quad n \in \mathcal{N}, t \in \mathcal{T}. \quad (6)$$

2.3.2. Task Offloaded to the UAV for Computing. We adopt frequency-division multiple access (FDMA) for the UE's task offloading. The total bandwidth of the channel between the UE and the UAV is equally allocated to each UE. Then, the offloading rate of the UE $n, n \in \mathcal{N}$ at time slot $t, t \in \mathcal{T}$ is given as

$$r_n(t) = \frac{W_U}{N} \log_2 \left(1 + \frac{P_n^{\text{UE}} h_n(t)}{\sigma^2} \right), \quad n \in \mathcal{N}, t \in \mathcal{T}, \quad (7)$$

where W_U denotes the total available bandwidth between UE and the UAV, σ^2 denotes the noise power at the UAV, and P_n^{UE} is the transmit power of the UE $n, n \in \mathcal{N}$. Thus, the number of time slots required for offloading the task $\theta_n^{\text{UE}}(t)$ to the UAV by the UE $n, n \in \mathcal{N}$ can be expressed as

$$T_n^{\text{Tr}}[\theta_n^{\text{UE}}(t)] = \arg \min_m \left[\sum_{t'=\mu_{n,\text{Tr}}^{\text{UE}}[\theta_n^{\text{UE}}(t)]}^{\mu_{n,\text{Tr}}^{\text{UE}}[\theta_n^{\text{UE}}(t)]+m-1} r_n(t') \Delta \geq L[\theta_n^{\text{UE}}(t)] \right], \quad m \in Z_{++}, n \in \mathcal{N}, t \in \mathcal{T}, \quad (8)$$

where Z_{++} denotes the set of positive integers; $\mu_{n,\text{Tr}}^{\text{UE}}[\theta_n^{\text{UE}}(t)]$ is the index of the time slot that the task $\theta_n^{\text{UE}}(t)$ begins to be offloaded to the UAV by the UE $n, n \in \mathcal{N}$.

The task $\theta_n^{\text{UE}}(t)$ may not be able to be offloaded to the UAV within one time slot. The subsequent tasks of the UE

$n, n \in \mathcal{N}$ may wait for offloading. The number of time slots the task $\theta_n^{\text{UE}}(t)$ will wait for offloading at the UE $n, n \in \mathcal{N}$ is given as

$$T_{n,\text{Tr}}^{\text{Wait}}[\theta_n^{\text{UE}}(t)] = \left[\max_{t' \in \{0,1,\dots,t-1\}} T_{n,\text{end}}^{\text{Tr}}[\theta_n^{\text{UE}}(t')] - t + 1 \right]^+, \quad n \in \mathcal{N}, t \in \mathcal{T}. \quad (9)$$

$T_{n,\text{end}}^{Tr}[\theta_n^{\text{UE}}(t')]$ denotes the index of time slot that the offloading of the task $\theta_n^{\text{UE}}(t')$ is completed at the UE $n, n \in \mathcal{N}$, which is given as

$$T_{n,\text{end}}^{Tr}[\theta_n^{\text{UE}}(t')] = T_{n,Tr}^{\text{Wait}}[\theta_n^{\text{UE}}(t')] + T_n^{Tr}[\theta_n^{\text{UE}}(t')] + t' - 1, \quad n \in \mathcal{N}, t' \in \mathcal{T}. \quad (10)$$

Similar to (4), recursive method can be used to obtain the value of $T_{n,\text{end}}^{Tr}[\theta_n^{\text{UE}}(t')]$ with $T_{n,Tr}^{\text{Wait}}[\theta_n^{\text{UE}}(t_{n,b}^{Tr})] = 0$, where $\theta_n^{\text{UE}}(t_{n,b}^{Tr})$ is the first task to be offloaded by the UE $n, n \in \mathcal{N}$. The number of total time slots required for completing the task $\theta_n^{\text{UE}}(t)$'s offloading by the UE $n, n \in \mathcal{N}$ is given as

$$T_{UE,n,\text{total}}^{Tr}[\theta_n^{\text{UE}}(t)] = T_{n,Tr}^{\text{Wait}}[\theta_n^{\text{UE}}(t)] + T_n^{Tr}[\theta_n^{\text{UE}}(t)], \quad n \in \mathcal{N}, t \in \mathcal{T}. \quad (11)$$

After the task $\theta_n^{\text{UE}}(t)$'s offloading is finished, it will be stored in UE n 's computation queue at the UAV. We adopt a simple resource allocation scheme: the UAV equally

allocates its computing resource to each UE. Then, the number of time slots required for computing the task $\theta_n^{\text{UE}}(t)$ by the UAV is given as

$$T_{\text{UAV},n}^{\text{Comp}}[\theta_n^{\text{UE}}(t)] = \lceil \frac{L[\theta_n^{\text{UE}}(t)]C[\theta_n^{\text{UE}}(t)]}{f_{\max}/N} \rceil, \quad n \in \mathcal{N}, t \in \mathcal{T}, \quad (12)$$

where $f_{\max}$ is maximal CPU frequency of the UAV. The number of time slots that the task $\theta_n^{\text{UE}}(t)$ will wait for processing at the UAV is given as

$$T_{\text{UAV},n,\text{Comp}}^{\text{Wait}}[\theta_n^{\text{UE}}(t)] = \left[\max_{t' \in \{0,1,\dots,t-1\}} T_{\text{UAV},n,\text{end}}^{\text{Comp}}[\theta_n^{\text{UE}}(t')] - t + 1 \right]^+, \quad n \in \mathcal{N}, t \in \mathcal{T}, \quad (13)$$

where $T_{\text{UAV},n,\text{end}}^{\text{Comp}}[\theta_n^{\text{UE}}(t')]$ is the index of time slot that the computation of the task $\theta_n^{\text{UE}}(t')$ is completed at the UAV, which is given as

$$T_{\text{UAV},n,\text{end}}^{\text{Comp}}[\theta_n^{\text{UE}}(t')] = T_{\text{UAV},n,\text{Comp}}^{\text{Wait}}[\theta_n^{\text{UE}}(t')] + T_{\text{UAV},n}^{\text{Comp}}[\theta_n^{\text{UE}}(t')] + t' - 1, \quad n \in \mathcal{N}, t' \in \mathcal{T}. \quad (14)$$

We have $T_{\text{UAV},n,\text{Comp}}^{\text{Wait}}[\theta_n^{\text{UE}}(t_{\text{UAV},n,b}^{\text{Comp}})] = 0$, where $\theta_n^{\text{UE}}(t_{\text{UAV},n,b}^{\text{Comp}})$ is the first task of the UE $n, n \in \mathcal{N}$ processed by the UAV. The number of total time slots required for completing the computation of the task $\theta_n^{\text{UE}}(t)$ by the UAV (including the time for task offloading from the UE $n, n \in \mathcal{N}$ to the UAV and the waiting and processing time at the UAV) is given as

$$\begin{aligned} T_{\text{UAV},\text{total}}^{\text{Comp}}[\theta_n^{\text{UE}}(t)] &= T_{UE,n,\text{total}}^{Tr}[\theta_n^{\text{UE}}(t)] \\ &\quad + T_{\text{UAV},n,\text{Comp}}^{\text{Wait}}[\theta_n^{\text{UE}}(t)] \\ &\quad + T_{\text{UAV},n}^{\text{Comp}}[\theta_n^{\text{UE}}(t)], \quad n \in \mathcal{N}, t \in \mathcal{T}. \end{aligned} \quad (15)$$

The number of total time slots consumed for completing the computation of the task $\theta_n^{\text{UE}}(t)$ by the UAV is given as

$$T_{\text{UAV},\text{total},\text{true}}^{\text{Comp}}[\theta_n^{\text{UE}}(t)] = \min\{T_{\text{UAV},\text{total}}^{\text{Comp}}[\theta_n^{\text{UE}}(t)], \tau[\theta_n^{\text{UE}}(t)]\}, \quad n \in \mathcal{N}, t \in \mathcal{T}. \quad (16)$$

2.3.3. Task Offloaded to the BS for Computing. We adopt time division multiple access (TDMA) scheme for the UAV to offload UE's tasks to the BS. Each time slot is further

divided into N equal durations with length $\delta = \Delta/N$. The transmission rate from the UAV to the BS at time slot $t, t \in \mathcal{T}$ is given as

$$r_{BS}(t) = W_U \log_2 \left(1 + \frac{P^{UAV} g_{BS}(t)}{\sigma_b^2} \right), \quad n \in \mathcal{N}, t \in \mathcal{T}, \quad (17)$$

where σ_b^2 is the power of noise at the BS; P^{UAV} is the transmit power of the UAV. The number of time slots required for offloading the task $\theta_n^{UE}(t)$ from the UAV to the BS is given as

$$T_{UAV,n}^{Tr}[\theta_n^{UE}(t)] = \arg \min_m \left[\sum_{t'=\mu_{Tr}^{UAV}[\theta_n^{UE}(t)]}^{\mu_{Tr}^{UAV}[\theta_n^{UE}(t)]+m-1} r_{BS}(t') \delta \geq L[\theta_n^{UE}(t)] \right], \quad m \in \mathbb{Z}_{++}, n \in \mathcal{N}, t \in \mathcal{T}, \quad (18)$$

where $\mu_{Tr}^{UAV}[\theta_n^{UE}(t)]$ is the index of time slot that the task $\theta_n^{UE}(t)$ begins to be offloaded to the BS by the UAV. The

number of time slots the task $\theta_n^{UE}(t)$ will wait for offloading at the UAV is given as

$$T_{UAV,n,Tr}^{Wait}[\theta_n^{UE}(t)] = \left[\max_{t' \in \{0,1,\dots,t-1\}} T_{UAV,n,end}^{Tr}[\theta_n^{UE}(t')] - t + 1 \right]^+, \quad n \in \mathcal{N}, t \in \mathcal{T}, \quad (19)$$

where $T_{UAV,n,end}^{Tr}[\theta_n^{UE}(t')]$ is the index of time slot when the offloading of the task $\theta_n^{UE}(t')$ is completed by the UAV, which is given as

$$T_{UAV,end}^{Tr}[\theta_n^{UE}(t')] = T_{UAV,Tr}^{Wait}[\theta_n^{UE}(t')] + T_{UAV}^{Tr}[\theta_n^{UE}(t')] + t' - 1, \quad n \in \mathcal{N}, t' \in \mathcal{T}. \quad (20)$$

We have $T_{UAV,Tr}^{Wait}[\theta_n^{UE}(t_{UAV,b}^{Tr})] = 0$ where $\theta_n^{UE}(t_{UAV,b}^{Tr})$ is the first task of the UE $n, n \in \mathcal{N}$ offloaded by the UAV.

Similar to [23–25], we assume that the computing power of the BS is strong enough and the computing time of UE's tasks can be neglected. Then, the number of total time slots

required for completing the computation of the task $\theta_n^{UE}(t)$ by the BS (including the time required for task offloading from the UE $\theta_n^{UE}(t)$ to the UAV, the waiting time, and the offloading time from the UAV to the BS) is given as

$$T_{BS,total}^{Comp}[\theta_n^{UE}(t)] = T_{UE,n,total}^{Tr}[\theta_n^{UE}(t)] + T_{UAV,Tr}^{Wait}[\theta_n^{UE}(t)] + T_{UAV}^{Tr}[\theta_n^{UE}(t)], \quad n \in \mathcal{N}, t \in \mathcal{T}. \quad (21)$$

Similar to (6), the number of total time slots consumed for completing the computation of the task $\theta_n^{UE}(t)$ by the BS is given as

$$T_{BS,total,true}^{Comp}[\theta_n^{UE}(t)] = \min\{T_{BS,total}^{Comp}[\theta_n^{UE}(t)], \tau[\theta_n^{UE}(t)]\}, \quad n \in \mathcal{N}, t \in \mathcal{T}. \quad (22)$$

2.4. Problem Formulation. Our objective is to maximize the long-term number of completed tasks of the UE; i.e., maximize the number of the completed tasks of the UE

during T consecutive time slots. Towards this end, we formulate the optimization problem as

$$\begin{aligned} P_1: \quad & \max_{\alpha_{n,t}, n \in \mathcal{N}, t \in \mathcal{T}} \sum_{t=1}^T \sum_{n=1}^N \gamma^{t-1} I_{n,t}[\theta_n^{UE}(t)] \\ & \text{s.t. } \alpha_{n,t} \in \{-1, 0, 1\}, n \in \mathcal{N}, t \in \mathcal{T}, \end{aligned} \quad (23)$$

where $\gamma \in [0, 1]$ is the discount factor, which denotes the difference on importance between future rewards and present reward [26]. $I_{n,t}[\theta_n^{\text{UE}}(t)]$ is the indicator function that $I_{n,t}[\theta_n^{\text{UE}}(t)] = 0$ if the task $\theta_n^{\text{UE}}(t)$ is completed within its maximal tolerable latency, and $I_{n,t}[\theta_n^{\text{UE}}(t)] = -1$ if the task $\theta_n^{\text{UE}}(t)$ is failed. Obviously, the problem P_1 is nonconvex and difficult to solve by traditional optimization methods. In order to solve this problem, we propose a DRL-based algorithm.

3. A Reinforce with Baseline Optimization Framework

As we adopt FDMA and TDMA and the computing resource of the UAV is equally allocated to all UE, problem P_1 can be decomposed into N subproblems. The subproblem of the UE $n, n \in \mathcal{N}$ is given as

$$P_n: \max_{\alpha_{n,t}, t \in \mathcal{T}} \sum_{t=1}^T \gamma^{t-1} I_{n,t}[\theta_n^{\text{UE}}(t)] \quad (24)$$

s.t. $\alpha_{n,t} \in \{-1, 0, 1\}, t \in \mathcal{T}$.

We adopt the parameterized policy gradient approach with baseline method to obtain the optimal task allocation of the UE $n, n \in \mathcal{N}$. In order to apply the DRL algorithm, we first give the state, action, reward function, and the policy of the UE $n, n \in \mathcal{N}$ as follows:

(1) *System state* $S_n(t)$: the state of the UE $n, n \in \mathcal{N}$ in time slot $t, t \in \mathcal{T}$ is given as

$$S_n(t) = (s_1(t), s_2(t), s_3(t), s_4(t), s_5(t), s_6(t), s_7(t), s_8(t), s_9(t)), \quad (25)$$

$$\begin{aligned} \text{where } s_1(t) &= \max_{t' \in \{0, 1, \dots, t-1\}} T_{n, \text{end}}^{\text{Comp}}[\theta_n^{\text{UE}}(t')], s_2(t) = \\ & \max_{t' \in \{0, 1, \dots, t-1\}} T_{\text{UAV}, n, \text{end}}^{\text{Comp}}[\theta_n^{\text{UE}}(t')], s_3(t) = \\ & \max_{t' \in \{0, 1, \dots, t-1\}} T_{\text{UAV}, n, \text{end}}^{\text{Tr}}[\theta_n^{\text{UE}}(t')], \\ s_4(t) &= L[\theta_n^{\text{UE}}(t)], s_5(t) = C[\theta_n^{\text{UE}}(t)], s_6(t) = \tau \end{aligned}$$

$[\theta_n^{\text{UE}}(t)], s_7(t) = T_{n, \text{total}, \text{true}}^{\text{Comp}}[\theta_n^{\text{UE}}(t)], s_8(t) = T_{\text{UAV}, \text{total}, \text{true}}^{\text{Comp}}[\theta_n^{\text{UE}}(t)], s_9(t) = T_{\text{BS}, \text{total}, \text{true}}^{\text{Comp}}[\theta_n^{\text{UE}}(t)].$

(2) *Action* $A_n(t)$: $A_n(t)$ is the task allocation variable of the UE $n, n \in \mathcal{N}$, namely,

$$A_n(t) = \alpha_{n,t}, n \in \mathcal{N}, t \in \mathcal{T}. \quad (26)$$

(3) *Reward function* $R_n(t)$: reward function $R_n(t), n \in \mathcal{N}, t \in \mathcal{T}$ is the immediate reward when selecting action $A_n(t)$ in state $S_n(t)$, which is directly related with the objective of the optimization problem P_n . Hence, the reward function is defined as

$$R_n(t) = \begin{cases} 0, T_{i, \text{total}}^{\text{Comp}}[\theta_n^{\text{UE}}(t)] \leq \tau[\theta_n^{\text{UE}}(t)], & i = n, \text{UAV, BS}, \\ -1, T_{i, \text{total}}^{\text{Comp}}[\theta_n^{\text{UE}}(t)] > \tau[\theta_n^{\text{UE}}(t)], & i = n, \text{UAV, BS}. \end{cases} \quad (27)$$

(4) *Policy* π_{n, θ_n} : the policy π_{n, θ_n} denotes the mapping from the state $S_n(t)$ to the action $A_n(t)$ of the UE $n, n \in \mathcal{N}$, i.e., $\pi_{n, \theta_n}: S_n(t) \rightarrow A_n(t)$. θ_n is the parameter of the policy.

Our goal is to find an optimal policy for the UE $n, n \in \mathcal{N}$; the problem P_n can be transformed into

$$P_{n, \text{DRL}}: \max_{\pi_{n, \theta_n}} \sum_{t=1}^T \gamma^{t-1} I_{n,t}[\theta_n^{\text{UE}}(t)] \quad (28)$$

s.t. (23a).

The parameterized policy gradient approach adopts a parameterized policy and optimizes the parameters through gradient iteration algorithm [26]. A typical scalar performance measure $U(\theta_n)$ is given as $v_{\pi_{n, \theta_n}}[S_n(0)]$, which is the value function for policy π_{n, θ_n} starting from initial state $S_n(0)$. In the episodic case, an analytic expression for the gradient of $U(\theta_n)$ is provided by policy gradient theorem [26], which is given as

$$\nabla U(\theta_n) \propto \sum_{S_n(t)} \mu[S_n(t)] \times \sum_{A_n(t)} q_{\pi_{n, \theta_n}}[S_n(t), A_n(t)] \nabla \pi_{n, \theta_n}[A_n(t)|S_n(t)], \quad (29)$$

where $\mu[S_n(t)]$ is the on-policy distribution over states; $q_{\pi_{n, \theta_n}}[S_n(t), A_n(t)]$ is the value of taking action $A_n(t)$ in state $S_n(t)$ under policy π_{n, θ_n} .

In order to reduce the variance during the learning process, an action-independent baseline $b[S_n(t)]$ can be introduced, and the policy gradient with baseline is denoted as [26]

$$\nabla U(\theta_n) \propto \sum_{S_n(t)} \mu[S_n(t)] \times \sum_{A_n(t)} \left\{ q_{\pi_{n, \theta_n}}[S_n(t), A_n(t)] - b[S_n(t)] \right\} \nabla \pi_{n, \theta_n}[A_n(t)|S_n(t)]. \quad (30)$$

- (1) **Initialization:**
- (2) $\theta, \mathbf{w}$, a differentiable policy π_{n,θ_n} , maximal number of iterations $K_{\max}$, discount factor γ , the learning rate of policy $\alpha_p > 0$, the learning rate of the baseline $\alpha_b > 0$;
- (3) **for** $k = 1: K_{\max}$:
- (4) According to the policy π_{n,θ_n} , (24)–(26) to generate trajectory: $S_n(0), A_n(0), R_n(1), S_n(1) \dots S_n(T-1), A_n(T-1), R_n(T)$;
- (5) $G \leftarrow 0$;
- (6) **for** $t = T-1, T-2, \dots, 0$:
- (7) Update return: $G \leftarrow \gamma G + R_n(t)$;
- (8) Update $\mathbf{w}$: $\mathbf{w} \leftarrow \mathbf{w} + \alpha_b [G - v[S_n(t); \mathbf{w}]] \nabla v[S_n(t); \mathbf{w}]$;
- (9) Update θ_n : update θ_n by (30) with $\alpha = \alpha_p$;
- (10) **end for**
- (11) **end for**

ALGORITHM 1: The DRL-based algorithm.

The estimate of the state value $v[S_n(t); \mathbf{w}]$ is often selected as the baseline, where $\mathbf{w}$ is the weight vector of the state value function. Considering the baseline, θ_n is updated as [26]

$$\theta_{n,t+1} = \theta_{n,t} + \alpha \gamma^t (G - v[S_n(t); \mathbf{w}]) \nabla \ln \pi_{n,\theta_{n,t}}[A_n(t)|S_n(t)], \quad (31)$$

where α is the learning rate, G is the return following time slot t , $t \in \mathcal{T}$, and $\theta_{n,t}$ is the estimate of θ_n at time slot t , $t \in \mathcal{T}$. The details of the DRL-based algorithm are given in Algorithm 1.

4. Simulation Results

In this section, the performances of the proposed algorithm under different parameters are provided. According to the analysis above, solving problem P_1 is equivalent to solving each subproblem P_n . Therefore, we focus on the analysis of the performance for the UE $n, n \in \mathcal{N}$ instead of for all the UE. The default simulation parameters are set as follows, if not specified. The computation data size (in bits) of the task $\theta_n^{\text{UE}}(t)$ is randomly selected from set $\Theta_{I_n} = \{1.5e^4 \text{ bits}, 3e^4 \text{ bits}, 4.5e^4 \text{ bits}, 6e^4 \text{ bits}\}$. The corresponding maximal tolerable latency set of the task $\theta_n^{\text{UE}}(t)$ is set as $\Theta_{\tau_n} = \{3\Delta, 5\Delta, 7\Delta, 9\Delta\}$. Without loss of generality, we set $C[\theta_n^{\text{UE}}] = \beta, \beta = 1000$, which can be ignored in the system state. The UAV flies around in circles with center coordinates $(0, 0)$ and radius 20 m. The UE is uniformly distributed in a $100 \text{ m} \times 100 \text{ m}$ square area with center coordinate $(0, 0)$ m, which is the coordinate of the BS. The maximal CPU frequency of the UAV and the UE $n, n \in \mathcal{N}$ are set as 200 MHz and 10 MHz, respectively. The bandwidth of the channel between the UE $n, n \in \mathcal{N}$ and the UAV and between the UAV and the BS is set as 40 kHz and 200 kHz, respectively. T is set as 100. The length of the time slot Δ and sub-time slot δ is set as 100 ms and 20 ms, respectively. We conduct the simulations through Python

3.8 and Tensorflow 2.5.0. Fully connected hidden layer with 17 and 16 neurons in the policy and the baseline networks are employed. The learning rates of policy and baseline networks change in different figures to guarantee the convergence of the proposed algorithm.

Figures 2 and 3 show the convergence performance of proposed algorithm versus different learning rate α_p . The learning rate is important in iterative algorithms. As shown in Figure 2, the larger the learning rate is, the faster the convergence speed of the algorithm is. However, if the learning rate α_p is too large, the variance of the learning curve may become high, which can be observed in Figure 3.

Figure 4 shows the convergence performance of proposed algorithm versus different learning rate α_b . As can be observed in Figure 4 the convergence speed of the algorithm is not monotonically increasing in the learning rate α_b . The learning rate $\alpha_b = 5e^{-5}$ has faster convergence speed than the cases $\alpha_b = 5e^{-2}$ and $\alpha_b = 5e^{-4}$. $\alpha_b = 5e^{-2}$ has faster convergence speed than the case $\alpha_b = 5e^{-4}$. Although the convergence speed of the different learning rate α_b cases is different, they will converge to the same order of accumulated reward.

Figure 5 shows the performance of the proposed algorithm versus the complexity of the tasks of the UE $n, n \in \mathcal{N}$. As can be seen from Figure 5 the complexity of the task directly affects the performance of the proposed algorithm. The higher the complexity is, the lower the accumulated reward is, which agrees with intuitions.

Figure 6 shows the performance of the proposed algorithm versus the computing power of the UE $n, n \in \mathcal{N}$. As shown in Figure 6 the accumulated reward increases with the computing power of the UE $n, n \in \mathcal{N}$. The stronger the computing power of the UE $n, n \in \mathcal{N}$ is, the higher the accumulated reward of the UE $n, n \in \mathcal{N}$ is.

Figure 7 shows the effects of the length of time slot Δ . It can be observed that the accumulated reward increases with the length of time slot Δ . When Δ is too short, most of the tasks of the UE $n, n \in \mathcal{N}$ may fail.

FIGURE 2: The impact of learning rate α_p on the accumulated reward, where $\alpha_p = 1e^{-4}, 1e^{-5}, 2e^{-5}, 3e^{-5}$, and $\alpha_b = 5e^{-5}$.

FIGURE 3: The impact of learning rate α_p on the accumulated reward, where $\alpha_p = 1e^{-3}$ and $\alpha_b = 5e^{-5}$.

FIGURE 4: The impact of learning rate α_b on the accumulated reward, where $\alpha_b = 5e^{-2}, 5e^{-4}, 5e^{-5}$, and $\alpha_p = 3e^{-5}$.

FIGURE 5: The impact of complexity on the accumulated reward.

FIGURE 6: The impact of computing power of the UE $n, n \in \mathcal{N}$ on the accumulated reward. FIGURE 7: The impact of the length of time slot Δ on the accumulated reward.

5. Conclusion

We studied the problem of maximizing the long-term number of completed tasks of the UE in a proposed UAV-assisted MEC system. The formulated nonconvex problem was decomposed and solved by our proposed DRL-based algorithm. The validity and the convergence of the proposed algorithm were demonstrated through simulation results.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The research reported in this article was supported by the Research Program of China Mobile System Integration Co., Ltd. under Grant ZYJC-Shaanxi-202110-B-CB-001.

References

- [1] Y. Dong, S. Guo, J. Liu, and Y. Yang, "Energy-efficient fair cooperation fog computing in mobile edge networks for smart city," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7543–7554, 2019.
- [2] M. Li, P. Si, and Y. Zhang, "Delay-tolerant data traffic to software-defined vehicular networks with mobile edge computing in smart city," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 10, pp. 9073–9086, 2018.
- [3] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 242–253, 2021.
- [4] 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.
- [5] Y. Du, K. Wang, K. Yang, and G. Zhang, "Energy-efficient resource allocation in UAV based MEC system for IoT devices," *IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2018.
- [6] 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, 2018.
- [7] J. Hu, M. Jiang, Q. Zhang, Q. Li, and J. Qin, "Joint optimization of UAV position, time slot allocation, and computation task partition in multiuser aerial mobile-edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 7, pp. 7231–7235, 2019.
- [8] Z. Hu, F. Zeng, Z. Xiao, B. Fu, H. Jiang, and H. Chen, "Computation efficiency maximization and QoE-provisioning in UAV-enabled MEC communication systems," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1630–1645, 2021.
- [9] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao, and G. Y. Li, "Joint offloading and trajectory design for UAV-enabled mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1879–1892, 2019.
- [10] 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.
- [11] J. Zhang, L. Zhou, F. Zhou et al., "Computation-efficient offloading and trajectory scheduling for multi-UAV assisted mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2114–2125, 2020.
- [12] S. Zhang, H. Zhang, Q. He, K. Bian, and L. Song, "Joint trajectory and power optimization for UAV relay networks," *IEEE Communications Letters*, vol. 22, no. 1, pp. 161–164, 2018.
- [13] X. Hu, K.-K. Wong, K. Yang, and Z. Zheng, "UAV-assisted relaying and edge computing: scheduling and trajectory optimization," *IEEE Transactions on Wireless Communications*, vol. 18, no. 10, pp. 4738–4752, 2019.
- [14] T. Zhang, Y. Xu, J. Loo, D. Yang, and L. Xiao, "Joint computation and communication design for UAV-assisted mobile edge computing in IoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5505–5516, 2020.
- [15] Q. Liu, L. Shi, L. Sun, J. Li, M. Ding, and F. S. Shu, "Path planning for UAV-mounted mobile edge computing with deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5723–5728, 2020.
- [16] S. Zhu, L. Gui, D. Zhao, N. Cheng, Q. Zhang, and X. Lang, "Learning-based computation offloading approaches in UAVs-assisted edge computing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 928–944, 2021.
- [17] L. Yang, H. Yao, J. Wang, C. Jiang, A. Benslimane, and Y. Liu, "Multi-UAV-enabled load-balance mobile-edge computing for IoT networks," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6898–6908, 2020.
- [18] S. Wan, J. Lu, P. Fan, and K. B. Letaief, "Toward big data processing in IoT: path planning and resource management of UAV base stations in mobile-edge computing system," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5995–6009, 2020.
- [19] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and L. Hanzo, "Multi-agent deep reinforcement learning-based trajectory planning for multi-UAV assisted mobile edge computing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 73–84, 2021.
- [20] J. Xiong, H. Guo, and J. Liu, "Task offloading in UAV-aided edge computing: bit allocation and trajectory optimization," *IEEE Communications Letters*, vol. 23, no. 3, pp. 538–541, 2019.
- [21] L. Xiao, Y. Liang, C. Weng, D. Yang, and Q. Zhao, "Uav-enabled data collection: multiple access, trajectory optimization, and energy trade-off," *Wireless Communications and Mobile Computing*, vol. 201914 pages, Article ID 9647539, 2019.
- [22] M. Tang and V. W. S. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Transactions on Mobile Computing*, early access, Nov., 2020.
- [23] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784–1797, 2018.
- [24] S. Mao, S. Leng, K. Yang, X. Huang, and Q. Zhao, "Fair energy-efficient scheduling in wireless powered full-duplex mobile-edge computing systems," in *Proceedings of the GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–6, Singapore, December 2017.

- [25] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4177–4190, 2018.
- [26] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, The MIT Press, Cambridge, MA, USA, 2nd edition, 2018.