This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3071531, IEEE Internet of Things Journal

Enabling Efficient Scheduling in Large-Scale UAV-Assisted Mobile Edge Computing via Hierarchical Reinforcement Learning

Tao Ren, *Member, IEEE,* Jianwei Niu, *Senior Member, IEEE,* Bin Dai, Xuefeng Liu, Zheyuan Hu, Mingliang Xu, *Member, IEEE* and Mohsen Guizani, *Fellow, IEEE*

Abstract—Due to the high maneuverability and flexibility, unmanned-aerial-vehicles (UAVs) have been considered as a promising paradigm to assist mobile edge computing (MEC) in many scenarios including disaster rescue and field operation. Most existing research focuses on the study of trajectory and computation-offloading scheduling for UAV-assisted MEC in stationary environments, and could face challenges in dynamic environments where the locations of UAVs and mobile devices (MDs) vary significantly. Some latest research attempts to develop scheduling policies for dynamic environments by means of reinforcement learning. However, as these need to explore in high-dimensional state and action space, they may fail to cover in large-scale networks where multiple UAVs serve numerous MDs. To address this challenge, we leverage the idea of 'divide-and-conquer' and propose HT3O, a scalable scheduling approach for large-scale UAV-assisted MEC. First, HT3O is built with neural networks via deep reinforcement learning to obtain real-time scheduling policies for MEC in dynamic environments. More importantly, to make HT3O more scalable, we decompose the scheduling problem into two-layered sub-problems and optimize them alternately via hierarchical reinforcement learning. This not only substantially reduces the complexity of each sub-problem, but also improves the convergence efficiency. Experimental results show that HT3O can achieve promising performance improvements over state-of-the-art approaches.

Index Terms—Unmanned aerial vehicle, mobile edge computing, computation offloading, trajectory optimization, hierarchical reinforcement learning.

Corresponding author: Mohsen Guizani
T. Ren is with the Hangzhou Innovation Institute, Beihang University, Hangzhou 310051, China, and the State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing 100191, China (e-mail: taotao_1982@126.com).
J. Niu, B. Dai and X. Liu are with the State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing 100191, China, and Hangzhou Innovation Institute, Beihang University, Hangzhou 310051, China (e-mail: niujianwei@buaa.edu.cn; daibin@buaa.edu.cn; liu_xuefeng@buaa.edu.cn).
J. Niu is also with the Zhengzhou University Research Institute of Industrial Technology, School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China.
Z. Hu is with the School of Computer Science and Engineering, Beihang University, Beijing 100191, China (e-mail: huzheyuan18@buaa.edu.cn).
M. Xu is with the School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China (e-mail: iexumingliang@zzu.edu.cn).
M. Guizani is with the Department of Computer Science and Engineering, Qatar University, Doha, Qatar. (e-mail: mguizani@ieee.org).

1 Introduction

With the rapid development of mobile computing and wireless communication techniques, mobile devices (MDs) and mobile applications are becoming more and more popular [1]. Since cloud computing is difficult to satisfy the low-latency requirement of MDs, mobile edge computing (MEC) has emerged as an effective complementary paradigm to support the growing popularity of mobile applications with approximate capabilities as cloud computing [2], [3]. Due to the high maneuverability and flexibility, unmanned aerial vehicles (UAVs) have been proposed and envisioned as a promising technique to assist mobile edge computing in both military and civilian applications [4], [5]. For instance, when network infrastructures are either unavailable (e.g., disaster rescue) or sparely distributed (e.g., field operation), or the temporary increase of mobile devices goes far beyond the network service capability (e.g., football match) [6], UAVs can serve as communication relay stations or edge computing platforms. Especially when UAVs are deployed with computing resources, lots of benefits can be acquired for UAV-assisted MEC, such as reduced network overhead, low execution latency, better quality of experience (QoE), and longer battery life of MDs.

In the field of UAV-assisted MEC, UAVs' trajectories and tasks' computation-offloading need to be scheduled appropriately to obtain desirable performances. Most existing research focuses on the study of UAV-assisted MEC scheduling in stationary environments, where motionless MDs are served by mobile UAVs during the whole task execution time. To obtain efficient scheduling policies, these studies solve the scheduling optimization problem primarily based on heuristic methods, e.g., block coordinate descent (BCD) [7], [8], genetic algorithm (GA), and particle swarm optimization (PSO) [9]. Once the locations of UAVs and MDs change significantly, the optimization needs to be resolved [10]. In dynamic environments, MDs moves within near ranges in each time-slot, which will give rise to frequent re-optimization and high computation burdens. This could make heuristic-based methods unsuitable for real-time applications where the available time in each time-slot is not sufficient for resolving the optimization.

More recently, some research has tried to enable real-time UAV-assisted MEC in dynamic environments by developing scheduling policies via reinforcement learning (RL) [11]–[14]. RL explores the dynamic environments of MEC and learns efficient scheduling policies from the experience, which eliminates the need for resolving optimization problems. However, when the number of UAVs or MDs increases, the system state (e.g., UAV locations, MD locations, arrived tasks) space and action (e.g., UAV movements, MD offloading decisions) space of the MEC will grow exponentially. This problem, called the curse of dimensionality [15], dramatically reduces the convergence efficiency of these methods. As a result, it is difficult for these methods to obtain convergent scheduling policies for large-scale MEC where multiple UAVs serve numerous MDs.

To address the challenge of efficient scheduling for large-scale UAV-assisted MEC (LU-MEC) in dynamic environments, this paper leverages the idea of divide-and-conquer and proposes a scalable scheduling approach named hierarchical trajectory optimization and offloading optimization (HT3O). First, to achieve real-time scheduling of UAVs' trajectories and offloading decisions according to the dynamic system state, we build HT3O with neural networks via deep reinforcement learning. More importantly, to make HT3O scalable, we decompose the original optimization problem into two-layered sub-problems and optimize them alternately via hierarchical reinforcement learning (HRL). This not only substantially reduces the complexity of each sub-problem but also improves the overall learning efficiency by reusing lower-layered policies. Extensive experiments are conducted via numerical simulation, and the results show that the learned HT3O model can achieve satisfactory performance improvements over state-of-the-art approaches in terms of learning efficiency, computation efficiency, and average task delays, respectively.

In particular, the main contributions of this paper are summarized as follows.

- We adopt deep reinforcement learning to obtain real-time scheduling policies for LU-MEC in dynamic environments, which generates efficient UAVs' movements and task offloading decisions according to dynamic system states.
- We decompose the optimization problem into two-layered sub-problems and optimize them alternately via hierarchical reinforcement learning, which not only reduces the complexity of each sub-problem but also improves the learning efficiency.
- We conduct extensive experiments via numerical simulation with various network settings, and demonstrate the substantial performance improvements of the proposed approach over state-of-the-art approaches.

The rest of the paper is organized as follows. In the next section, we discuss the related work about trajectory and computation-offloading optimization in MEC networks. Section 3 describes the system model and then formulates the optimization problem. In Section 4, the optimization problem is decomposed into two-layered sub-problems and solved by the proposed HT3O approach. In Section 5, the performance of HT3O is evaluated via numerical simulation with various network settings. Finally, Section 6 discusses conclusions and future work of the paper.

2 Related Work

This section discusses the related work about mobile edge computing from the perspective of two popular methods: heuristic-based approaches and learning-based approaches.

2.1 MEC Scheduling via Heuristic Methods

In MEC networks, multiple types of variables (e.g., UAVs' trajectories, task offloading decisions, computing resource allocation) need to be optimized to achieve the desired scheduling objective, a popular method of which is heuristic optimization. In [7], the minimum throughput over all MDs in a multi-UAV assisted MEC network was maximized by jointly optimizing task allocation and UAVs' trajectories via the BCD and successive convex approximation (SCA) methods. In [8], an alternative optimization approach based on BCD and SCA techniques was proposed to minimize the overall energy-consumption of all MDs' tasks. Different from the above BCD and SCA based methods, Guo *et al.* [16] designed a sub-optimal approach based on hierarchical GA and PSO to obtain energy-efficient computation offloading policies. Bi *et al.* [17] proposed a novel heuristic-based method by means of particle swarm optimization to produce a near-optimal computation-offloading policy to minimize energy consumption. More studies about heuristic-based approaches can be found in [18]–[20].

2.2 MEC Scheduling via Learning Methods

Since plenty of iterations are required to obtain a (near) optimal solution, heuristic-based approaches are unsuitable for real-time MEC applications in dynamic environments. With the wide application of big data [21] and deep learning [22], [23] in networking, lots of researchers are exploring learning-based MEC scheduling approaches [24]–[26]. Wang *et al.* [12] proposed an approach based on RL for the joint optimization of user-association and resource-allocation to achieve the energy-consumption minimization of a UAV-assisted MEC network, whereas the scalability could be intractable due to the adoption of Q-table in their work. Different from [12], Chen *et al.* [27] proposed a novel method based on deep Q network (DQN) to obtain near-optimal computation-offloading policies of a MEC network; Liu *et al.* [28] adopted a double deep Q-network to develop a QoS-based action selection model to optimize UAVs' trajectories and UAV-MD associations in a UAV-assisted MEC network. Mukherjee *et al.* [13] built a distributed scheduling model consisting of multiple deep neural networks (DNNs) which are trained with the same play experience, and selected the lowest-loss DNN to perform the computation offloading of MDs' tasks. More research about learning-based approaches can be found in [11], [14], [29], [30].

Despite the great progress of existing work, further investigations are still required for learning-based approaches to make them be more applicable for large-scale UAV-assisted MEC in dynamic environments. Therefore, this paper attempts to obtain scalable and efficient scheduling policies for LU-MEC by solving the trajectory and offloading optimization problem in a hierarchical reinforcement learning way.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3071531, IEEE Internet of Things Journal

3 System Model and Problem Formulation

In this section, we first describe the system model of the LU-MEC network, and then formulate the optimization problem for the system. The main notations used in the following parts are summarized in Table I.

TABLE 1
List of Main Notations

Notation	Description
$\mathcal{M}$	The set of MDs' index
M	The number of MDs
m	The index of MD
$\mathcal{U}$	The set of UAVs' index
U	The number of UAVs
u	The index of UAV
$\mathcal{T}$	The set of time-slots
N	The number of time-slots
n	The index of time-slot (TS)
TS^n	The n-th time-slot
$\mathbf{c}_u^{\text{U}}(n)$	Horizontal coordinate of UAV u at TS^n
E_u^{f}	Energy consumption for the flying of UAV u
$h_m^u(n)$	Channel gain from UAV u to MD m at TS^n
$I_m(n)$	Computation task arrived at MD m at TS^n
$D_m(n)$	Data size of task $I_m(n)$
$C_m(n)$	Required CPU cycles to compute one bit data of $I_m(n)$
$\Gamma_m(n)$	Maximum admissible delay of task $I_m(n)$
$a_m^u(n)$	Offloading indicator of task $I_m(n)$
$R_m^u(n)$	Transmission rate from MD m to UAV u at TS^n
f_m	Local computing capability of MD m
f_u	Computing capability of UAV u

3.1 System Model

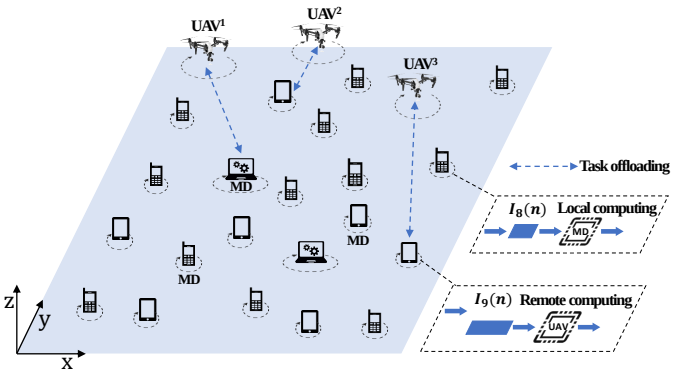


Fig. 1. Schematic of the LU-MEC network.

The LU-MEC network considered in this paper is presented as shown in Fig. 1. The LU-MEC network consists of M MDs denoted as

$$\mathcal{M} \triangleq \{1, 2, ..., M\}$$

and U UAVs denoted as

$$\mathcal{U} \triangleq \{1, 2, ..., U\}.$$

The task execution time is denoted as T, which is further discretized into N time-slots, i.e., the slot-length $\tau = T/N$. The τ should be small enough to guarantee the approximately unchanged distances between UAVs and MDs during each time-slot. For the sake of convenience, the set of time-slots is denoted as

$$\mathcal{T} \triangleq \{1, 2, ..., N\}.$$

Similar to prior studies, this paper adopts a 3D Cartesian coordinate system to model the stereoscopic space of the LU-MEC network. The MDs and UAVs are assumed to move at the fixed heights 0 (i.e., the height of the ground) and H, respectively.

However, different from most existing studies, the MDs are assumed to be able to move away from its horizontal position to a new position during each time-slot, and the distance follows a normal distribution $\mathcal{N}(0, \iota^2)$. ι is small so that MDs could only move to very close horizontal space at most cases. For convenience, the MDs are only allowed to move long four directions (east, south, west, and north), which could also be easily extended to multiple directions. Suppose the horizontal coordinate of MD $m \in \mathcal{M}$ at time-slot $n \in \mathcal{T}$ is denoted as

$$\mathbf{c}_m^{\text{M}}(n) = [x_m^{\text{M}}(n), y_m^{\text{M}}(n)]^T,$$

then the probability ρ^{M} of the horizontal coordinate $\mathbf{c}_m^{\text{M}}(n+1)$ being one of

$$\{[x_m^{\text{M}}(n) + d, y_m^{\text{M}}(n)]^T, [x_m^{\text{M}}(n), y_m^{\text{M}}(n) - d]^T, \\ [x_m^{\text{M}}(n) - d, y_m^{\text{M}}(n)]^T, [x_m^{\text{M}}(n), y_m^{\text{M}}(n) + d]^T\}$$

could be denoted as

$$\rho^{\text{M}} = \frac{1}{4\sqrt{2\pi}\iota} e^{-\frac{d^2}{2}\iota^2}, \tag{1}$$

where d is the distance of $\mathbf{c}_m^{\text{M}}(n+1)$ from $\mathbf{c}_m^{\text{M}}(n)$.

The horizontal coordinate of UAV $u \in \mathcal{U}$ at time-slot n is denoted as

$$\mathbf{c}_u^{\text{U}}(n) = [x_u^{\text{U}}(n), y_u^{\text{U}}(n)]^T.$$

To reduce energy consumption, UAVs are supposed to move in a limited frequency (move up to once at the first slot of every Δ time-slots). The maximum horizontal flying speed of UAVs is given by V^{U}, then the following constraint could be derived:

$$\begin{cases} 0 \le \dfrac{||\mathbf{c}_u^{\text{U}}(n+1) - \mathbf{c}_u^{\text{U}}(n)||}{\tau} \le V^{\text{U}}, & if \quad n\%\Delta = 0 \\ \mathbf{c}_u^{\text{U}}(n+1) = \mathbf{c}_u^{\text{U}}(n), & else \end{cases}. \tag{2}$$

Since the UAV could only move at a limited speed ($\le V^{\text{U}}$) and within a little time length (τ), the channel-gain could be approximately sampled within the time-slot. According to the simplified energy-consumption model adopted in existing research [31], [32], the energy E_u^{f} consumed for the flying of UAV u is represented as

$$E_u^{\text{f}} \triangleq \begin{cases} \mathcal{K}_u ||v_u(n)||^2, & if \quad n \bmod \Delta = 0 \\ 0, & else \end{cases}, \tag{3}$$

where

$$||v_u(n)|| \triangleq \frac{||\mathbf{c}_u^{\text{U}}(n+1) - \mathbf{c}_u^{\text{U}}(n)||}{\tau}$$

is the horizontal flying speed of UAV u at slot n, $\mathcal{K}_u = 0.5 M_{\text{g}} \tau$ and M_{g} is the workload of UAV u. Since the MDs move randomly at each time-slot, the UAVs' movements should be appropriately optimized at the first slot of each Δ slots so that optimum computation offloading performances

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3071531, IEEE Internet of Things Journal

could be achieved.

In UAV-assisted networks, the line-of-sight wireless channels between UAVs and MDs are more dominant than other channel impairments due to the high altitudes of UAVs. Hence, the channel-gain $h_m^u(n)$ from UAV u to MD m at time-slot n could be denoted by the free-space path loss model [33] as follows:

$$h_m^u(n) = g_0(d_m^u(n))^{-2} = g_0\left(H^2 + ||\mathbf{c}_u^{\mathrm{U}} - \mathbf{c}_m^{\mathrm{M}}||^2\right)^{-1}, \quad (4)$$

where g_0 stands for the received power for the 1 m reference distance and 1 W transmission power, and $d_m^u(n)$ denotes the Euclidean distance between UAV u and MD m at time-slot n.

The computation task arrived at MD m at time-slot n is expressed by $I_m(n) = \{D_m(n), C_m(n), \Gamma_m(n)\}$, where $D_m(n)$ denotes the data size of task $I_m(n)$ measured in bits, $C_m(n)$ denotes the required CPU cycles for the computation of one bit data, and $\Gamma_m(n)$ is the maximum admissible delay of task $I_m(n)$. In this paper, the values of $D_m(n)$, $C_m(n)$, and $\Gamma_m(n)$ are selected randomly from predefined sets, while the local computing time of the most computation-intensive and delay-critical task at the MD with the lowest computing capability is less than the time-slot length τ.

The MDs are assumed to adopt binary offloading policies, which means task $I_m(n)$ is either performed locally at MD m or offloaded remotely to a UAV for execution as a whole. The UAV is assumed to be able to provide computation service to up to one MD at each time-slot. Therefore, the binary task scheduling variables between UAVs and MDs are denoted as

$$\mathcal{A} = \{\alpha_m^u(n) \mid u \in \mathcal{U}, m \in \mathcal{M}, n \in \mathcal{T}\},$$

where $\alpha_m^u(n) = 1$ denotes that the task $I_m(n)$ of MD m is offloaded to UAV u at time-slot n and $\alpha_m^u(n) = 0$ denotes the local computing. Then the following constraints about α could be derived as

$$\sum_{m=1}^{M} \alpha_m^u(n) \leq 1, \quad \forall u \in \mathcal{U}, \forall n \in \mathcal{T}, \quad (5)$$

$$\sum_{u=1}^{U} \alpha_m^u(n) \leq 1, \quad \forall m \in \mathcal{M}, \forall n \in \mathcal{T}, \quad (6)$$

$$\alpha_m^u(n) \in \{0, 1\}, \quad \forall u \in \mathcal{U}, \forall m \in \mathcal{M}, \forall n \in \mathcal{T}. \quad (7)$$

3.2 Communication Model

To improve the signal to noise ratio, MDs are assumed to transmit at their maximum power $\hat{P}_m,\ m \in \mathcal{M}$. As each UAV can accept computation offloading for at most one MD in each time-slot, the communication interference between channels can be neglected. Hence, the instantaneous transmission rate (bps/Hz) between UAV u and MD m at time-slot n could be given as

$$R_m^u(n) = log_2(1 + \frac{\hat{P}_m h_m^u(n)}{\sigma^2}),\ u \in \mathcal{U}, m \in \mathcal{M}, n \in \mathcal{T}, \quad (8)$$

where σ^2 represents the noise power at UAV u. Similar to [34], orthogonal frequencies are allocated to UAVs with overlapped coverage areas, which could result in noninterference among UAVs.

Then, the time delay resulted from transmitting $I_m(n)$ from MD u to UAV u at time-slot t is shown as

$$\hat{t}_m^u(n) = \frac{\alpha_m^u(n) D_m(n)}{R_m^u(n) B}, \quad (9)$$

where B refers to the channel bandwidth. Consequently, the energy consumed to offload the task can be given by

$$\hat{E}_m^u(n) = \hat{P}_m \hat{t}_m^u(n) = \hat{P}_m \frac{\alpha_m^u(n) D_m(n)}{R_m^u(n) B}. \quad (10)$$

Considering the outputs of various computation-intensive tasks (e.g., speech processing, image classification) are much smaller than the inputs, the time and energy spent to transmit the outputs from UAVs to MDs could be neglected.

3.3 Computation Model

Each task can be either carried out locally on the MD (where it arrives) or remotely on a UAV via computation offloading, therefore, the computation model should be formulated from two perspectives:

3.3.1 Local Computing

If the task $I_m(n)$ of MD m at time-slot n is not scheduled for edge computing ($\alpha_m^u(n) = 0, \forall u \in \mathcal{U}$), then $I_m(n)$ should be performed locally on MD m and the required computation (CPU cycles) can be derived as $(1 - \sum_{u=1}^{U} \alpha_m^u(n)) D_m(n) C_m(n)$. Let f_m represents the local computing capability (CPU cycles per second) of MD m, then the local executing time of $I_m(n)$ can be calculated as

$$\bar{t}_m(n) = \frac{(1 - \sum_{u=1}^{U} \alpha_m^u(n)) D_m(n) C_m(n)}{f_m}. \quad (11)$$

Similar to prior work [35]–[37], this paper models the power consumption of MD's CPU as a function related to the computing capability:

$$\bar{P}_m = \gamma_m (f_m)^3,$$

where γ_m is the coefficient associated with the CPU architecture of MD m. Then the energy consumption of locally computing task $I_m(n)$ on MD m at time-slot n is denoted as the product of the power consumption and computing time

$$\bar{E}_m(n) = \bar{P}_m \bar{t}_m(n) = \gamma_m (1 - \sum_{u=1}^{U} \alpha_m^u(n)) D_m(n) C_m(n) (f_m)^2. \quad (12)$$

3.3.2 Edge Computing

If the task $I_m(n)$ is scheduled to offload to a UAV for computation ($\alpha_m^u(n) \neq 0, \exists u \in \mathcal{U}$), the time and energy spent to executing $I_m(n)$ on the UAV can be given by

$$\check{t}_m^u(n) = \frac{\alpha_m^u(n) D_m(n) C_m(n)}{f_u}, \quad (13)$$

and

$$\check{E}_m^u(n) = \check{P}^u \check{t}_m^u(n) = \gamma^u \alpha_m^u(n) D_m(n) C_m(n) (f_u)^2, \quad (14)$$

respectively, where f_u denotes the computing capability (CPU cycles per second) of UAV u, γ^u is the computation coefficient associated with the CPU architecture of UAV u.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3071531, IEEE Internet of Things Journal

3.4 Problem Formulation

Based on the above models, the time delay $t_m(n)$ for executing the task $I_m(n)$ of MD m at time-slot n can be denoted as

$$t_m(n) = \begin{cases} \bar{t}_m(n), & if \quad \sum_{u=1}^{U} \alpha_m^u(n) = 0 \\ \max\limits_{u \in \mathcal{U}} \left\{ \hat{t}_m^u(n) + \check{t}_m^u(n) \right\}, & else \end{cases}. \tag{15}$$

Consequently, the average delay of MD m can be represented as

$$t_m^{\text{avg}} = \frac{1}{N} \sum_{n=1}^{N} t_m(n). \tag{16}$$

Under the objective of improving the QoE of computation-intensive and delay-critical applications, we aim to minimize the average delay t_m^{avg} of all MDs during the period T by jointly optimizing the movements of all MDs and the offloading variables:

$$\mathcal{P}1: \min_{\{\mathbf{c}_u^{\text{U}}(n), \alpha_m^u(n)\}} \frac{1}{M} \sum_{m \in \mathcal{M}} t_m^{\text{avg}}, \tag{17}$$

$$\begin{aligned}
s.t.\ \ C1:&\ 0 \le \frac{||\mathbf{c}_u^{\text{U}}(n+1) - \mathbf{c}_u^{\text{U}}(n)||}{\tau} \le V^{\text{U}}, && \forall u \in \mathcal{U}, n\%\Delta = 0, \\
C2:&\ \mathbf{c}_u^{\text{U}}(n+1) = \mathbf{c}_u^{\text{U}}(n), && \forall u \in \mathcal{U}, n\%\Delta \neq 0, \\
C3:&\ \sum_{m=1}^{M} \alpha_m^u(n) \le 1, && \forall u \in \mathcal{U},\ \forall n \in \mathcal{T}, \\
C4:&\ \sum_{u=1}^{U} \alpha_m^u(n) \le 1, && \forall m \in \mathcal{M},\ \forall n \in \mathcal{T}, \\
C5:&\ \alpha_m^u(n) \in \{0,1\}, && \forall u \in \mathcal{U}, \forall m \in \mathcal{M}, \forall n \in \mathcal{T}, \\
C6:&\ \sum_{n'=1}^{n} \left\{ \bar{E}_m(n) + \sum_{u=1}^{U} \hat{E}_m^u(n) \right\} \le \Phi^{\text{M}}, && \forall n \in \mathcal{T}, \forall m \in \mathcal{M}, \\
C7:&\ \sum_{n'=1}^{n} \sum_{m=1}^{M} \check{E}_m^u(n) \le \Phi^{\text{U}}, && \forall n \in \mathcal{T}, \forall u \in \mathcal{U}, \\
C8:&\ t_m(n) \le \min\{\Gamma_m(n), \tau\}, && \forall m \in \mathcal{M},\ \forall n \in \mathcal{T},
\end{aligned}$$

where the constraints $C1$ and $C2$ ensure the limited flying speed and frequency of UAVs, the constraints $C3$, $C4$, and $C5$ refer to the binary scheduling limitation between UAVs and MDs, $C6$ and $C7$ are the energy budget constraints of MDs and UAVs, where Φ^{M} and Φ^{U} represent the stored energy in each MD and UAV, respectively, and the constraint $C8$ guarantees that each task should be completed within both its maximum admissible delay and the time-slot length.

In general, it is hard to solve the mixed integer nonlinear programming problem $\mathcal{P}1$ due to the discrete binary scheduling variables which are further highly coupled with UAV's movement variables $\mathbf{c}_u^{\text{U}}(n)$. In addition, the optimization difficulty also arises from the following aspects:

- the scheduling variables $\alpha_m^u(n)$ for computation offloading is binary and of large-scale ($O(U \times M \times N)$);
- the distribution of MDs changes dynamically in each time-slot;
- the UAVs could only move with low frequency (up to once every Δ time-slots);
- the computation offloading decision must be made in real time (every time-slot).

4 HRL-Based Computation Offloading

In this section, we first describe the hierarchical decomposition of the original optimization problem, then discuss the HRL-based scheduling algorithm to solve the decomposed sub-problems, last give the performance analysis of the proposed scheduling algorithm.

4.1 Hierarchical Decomposition of the Optimization Problem

At the first time-slot of every Δ time-slots, each UAV should make a promising decision about its new location based on the observed system state (e.g., the positions of MDs and UAVs, the arrived tasks, the consumed energies) so as to obtain satisfying delay performance of MDs during the period T. When UAVs have made decisions and moved to new locations at the first time-slot, the UAVs will stay at their positions for the upcoming Δ slots and perform efficient scheduling policies about $\alpha_m^u(n)$ at each time-slot so as to minimize the delays of MDs for the Δ slots. The two steps alternate until the end of T.

Considering the characteristic of the decision actions (the locations $\mathbf{c}_u^{\text{U}}(n)$ of UAVs every Δ slots and the offloading decisions $\alpha_m^u(n)$ in each slot), we decompose the whole optimization problem into two sub-problems hierarchically according to the two types of actions:

- Location optimization ($\bar{\mathcal{P}}1$). Since the UAVs could move to new locations (at the first slot of every Δ slots) within a wide range (under the speed constraints) while the MDs could move (in every slot) within a limited range, the UAVs should properly choose the new locations in view of the predictions of the trajectories $\mathbf{c}_m^{\text{M}}$ of MDs in the next Δ slots so as to acquire the desired distances $d_m^u(n)$ between UAVs and MDs, because $d_m^u(n)$ is the key factor of channel gain $h_m^u(n)$ which will further affect the wireless transmission rate $R_m^u(n)$ and finally affect the time delay and energy consumption of wireless transmission.
- Offloading optimization ($\bar{\mathcal{P}}2$). During the Δ slots, the offloading scheduling variables $\alpha_m^u(n)$ should be determined in view of the UAVs' unchanging locations and the predictions of MDs' varying locations $\mathbf{c}_m^{\text{M}}$ and upcoming tasks $I_m(n)$ in each slot of the current Δ slots so as to minimize the average delay of all MDs.

Hence, we transform problem $\mathcal{P}1$ into the location optimization problem $\bar{\mathcal{P}}1$ and the offloading optimization problem $\bar{\mathcal{P}}2$ as follows,

$$\bar{\mathcal{P}}1: \min_{\{\mathbf{c}_u^{\text{U}}(\dot{n})\}} \{\bar{\mathcal{P}}2(\dot{n}) \mid \dot{n}\%\Delta = 0, \dot{n} \in \mathcal{T}\}, \tag{18}$$
$$s.t. \quad C1 \quad and \quad C2,$$

and

$$\bar{\mathcal{P}}2(\dot{n}): \min_{\{\alpha_m^u(n)\}} \frac{1}{M} \sum_{m \in \mathcal{M}} \left(\frac{1}{\Delta} \sum_{n=\dot{n}+1}^{\dot{n}+\Delta} t_m(n) \right), \tag{19}$$
$$s.t. \quad C3 - C8.$$

Based on the above decomposition, we propose the HT3O framework to obtain efficient computation offloading in the LU-MEC network, which will be elaborated in the following subsections.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3071531, IEEE Internet of Things Journal

4.2 Reinforcement Learning Framework of HT3O

We model the dynamic computation offloading process of the LU-MEC network as a discrete-time, finite Markov Decision Process (MDP) based on the assumption that the new state of the system only depends on the current system state and action.

In MDP, as shown in the left part of Fig. 2, there are typically an agent and an environment, who interact with each other iteratively. In some state, the agent sends an action with some probability to the environment; In return, the environment generates an instantaneous reward to the agent and turns into a new state, with some probability. The right part of Fig. 2 is an example of the interaction and transmission process.

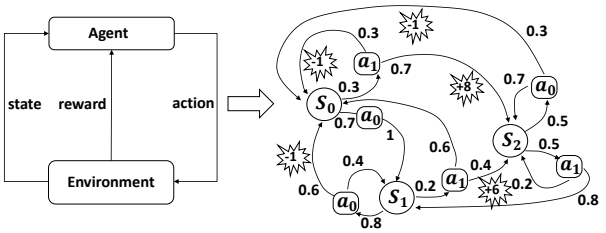


Fig. 2. Schematic of the Markov decision process.

The MDP is denoted by a 4-tuple: $\{S, A, \Pr, R\}$ where

- S is the set of the system states of the LU-MEC network. In time-slot n, the element

$$s_n = \{\mathbf{c}_m^{\mathrm{M}}(n), I_m(n), \bar{\Phi}_m^{\mathrm{M}}(n), \bar{\Phi}_u^{\mathrm{U}}(n)\},$$

where

$$\bar{\Phi}_m^{\mathrm{M}}(n) = \Phi^{\mathrm{M}} - \sum_{n'=1}^{n} \left\{ \bar{E}_m(n') + \sum_{u=1}^{U} \hat{E}_m^u(n') \right\}$$

and

$$\bar{\Phi}_u^{\mathrm{U}}(n) = \Phi^{\mathrm{U}} - \sum_{n'=1}^{n} \sum_{m=1}^{M} \check{E}_m^u(n')$$

are the left energies of MD m and UAV u in time-slot n, respectively.

- A is the set of the scheduling actions of the LU-MEC network, and $a_n = \{\mathbf{u}_u^{\mathrm{U}}(n), \alpha_m^u(n)\}$ is the scheduling action of the LU-MEC network in time-slot n.
- Pr is the transition function of the LU-MEC network, where $\Pr(s'|s,a)$ is the probability of the LU-MEC network transiting from state s to s' under taking action a.
- R is the reward function of the LU-MEC network, where $r_n(s'|s,a)$ is the immediate reward generated from the LU-MEC network after transiting from s to s' under taking a in time-slot n.

Since the objective of $\mathcal{P}1$ is to minimize

$$\begin{aligned} \frac{1}{M}\sum_{m\in\mathcal{M}} t_m^{\mathrm{avg}} &= \frac{1}{M}\sum_{m\in\mathcal{M}} \frac{1}{N}\sum_{n=1}^{N} t_m(n) \\ &= \frac{1}{N}\sum_{n=1}^{N} \frac{1}{M}\sum_{m\in\mathcal{M}} t_m(n) \\ &= \frac{1}{N}\sum_{n=1}^{N} t_n^{\mathrm{mean}}, \end{aligned} \tag{20}$$

where

$$t_n^{\mathrm{mean}} = \frac{1}{M}\sum_{m\in\mathcal{M}} t_m(n),$$

the immediate reward r_n in time-slot n is given by

$$r_n = (t_n^{\mathrm{mean}})^{-1}, \tag{21}$$

where the parameters s', s, and a of $r_n(\cdot)$ and $t_n^{\mathrm{mean}}(\cdot)$ are omitted for clarity. A negative value will be generated as the immediate penalty to punish any violation of constraints $C1-C8$, which are consistent with the $\mathcal{P}1$ problem of equation (17).

In each time-slot n, the agent for trajectory and offloading optimization receives a state s_n from the LU-MEC network (also denoted as Environment) and performs an action a_n according to its scheduling policy π, which brings about a new state s_{n+1} and an immediate reward $r_{a_n}^{s_n,s_{n+1}}$. The objective of the agent is to maximize the expected cumulative rewards received from the Environment during the task execution time T (N time-slots), which can be expressed by a value function V^π as follows

$$V^\pi = \sum_{n\in[1,N]} \gamma^{n-1} r_n, \tag{22}$$

where $\gamma \in [0,1]$ is the discount factor of future rewards.

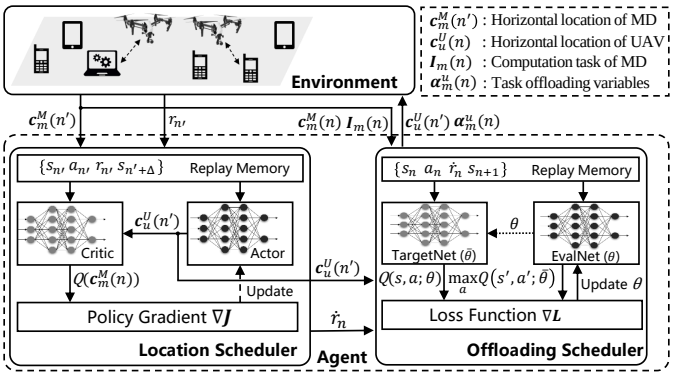


Fig. 3. Schematic of the HT3O framework for the LU-MEC network.

In view of the different scheduling frequency of $\{\mathbf{c}_u^{\mathrm{U}}(\dot{n}) \mid \dot{n}\%\Delta = 0, \dot{n} \in \mathcal{T}\}$ and $\{\alpha_m^u(n) \mid n \in \mathcal{T}\}$, we adopt a two-stage hierarchical reinforcement learning framework (HT3O) to optimize $\mathbf{c}_u^{\mathrm{U}}(\dot{n})$ and $\alpha_m^u(n)$ in different temporal scales, as shown in Fig. 3. The framework of HT3O consists of two parts: the scheduling agent and the environment. The scheduling agent is further divided into Location Scheduler (LS) and Offloading Scheduler (OS), which are developed with the aim to obtain desirable policies ($\mathbf{c}_u^{\mathrm{U}}$ and α_m^u) of the problem $\mathcal{P}1$ and $\mathcal{P}2$, respectively. LS and OS act in different levels:

- The top-level LS generates a location action $\mathbf{c}_u^{\mathrm{U}}(n')$ under the observation of the environment state ($\mathbf{c}_m^{\mathrm{M}}(n')$) every Δ time-slots ($n'\%\Delta = 0$) and receives a corresponding external reward $r_{n'}$.
- In each of the Δ time-slots, OS generates an offloading action $\alpha_m^u(n)$ based on the observation ($\mathbf{c}_m^{\mathrm{M}}(n)$ and $I_m(n)$) and the UAV locations $\mathbf{c}_u^{\mathrm{U}}(n')$ predetermined by LS.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3071531, IEEE Internet of Things Journal

- LS performs like a goal-maker in the way that LS determines the UAV locations $\mathbf{c}_u^{\mathrm{U}}(n')$ under which OS tries to optimize $\mathcal{P}2$, and like a critic in the way that LS sends an internal reward $\dot{r}_n$ to OS to indicate the effect of OS's action in each time-slot.

4.3 Location Optimization of HT3O

Since the horizontal locations of UAVs are continuous within the task area, we develop the LS module to perform trajectory optimization of problem $\mathcal{P}1$ based on Deep Deterministic Policy Gradient (DDPG) [38], which is an off-line RL algorithm suitable for environments with continuous action spaces. DDPG concurrently learns a Q value function that plays the part of a critic and a policy function that plays the part of an actor.

Specifically, the optimal Q value function of the critic can be derived from V^{π} of equation (22) based on Bellman equation and denoted as

$$\begin{aligned} Q^*(s,a) &= \mathop{\mathbb{E}}_{s'\sim \mathrm{Pr}}\left[\sum_{n\in[1,N]}\gamma^{n-1}r_n\right] \\ &= \mathop{\mathbb{E}}_{s'\sim \mathrm{Pr}}\left[r_{n'}+\gamma r_{n''}+\gamma^2 r_{n'''}+...\right] \\ &= \mathop{\mathbb{E}}_{s'\sim \mathrm{Pr}}\left[r_{n'}+\gamma(r_{n''}+\gamma r_{n'''}+...)\right] \\ &= \mathop{\mathbb{E}}_{s'\sim \mathrm{Pr}}\left[r_{n'}+\gamma Q^*(s'')\right] \\ &= \mathop{\mathbb{E}}_{s'\sim \mathrm{Pr}}\left[r_{n'}(s'\,|\,s_{n'},a_{n'})+\gamma\max_{a'}Q^*(s',a')\right], \end{aligned} \tag{23}$$

where

$$s_{n'}=\{\mathbf{c}_m^{\mathrm{M}}(n),\bar{\Phi}_m^{\mathrm{M}}(n),\bar{\Phi}_u^{\mathrm{U}}(n)\}$$

is the environment state observed by LS at time-slot n' ($n'\%\Delta=0$), $a_{n'}=\{\mathbf{c}_u^{\mathrm{U}}(n')\mid u\in\mathcal{U}\}$ is the set of movement actions of UAVs in time-slot n', and $s'=s_{n'+\Delta}$ is the environment state at the beginning of the next Δ slots obeying the transition function Pr of the environment.

The critic is built via a neural network (denoted as parameters θ^{C}) to approximate the Q value function, which now can be represented as $Q(s,a;\theta^{\mathrm{C}})$. Therefore, we can train $Q(s,a;\theta^{\mathrm{C}})$ with a set of interaction experiences $\mathcal{E}^{\mathrm{u}}$ (each one of which is denoted as a state-transition tuple $\chi=\{s_{n'},a_{n'},r_{n'},s_{n'+\Delta}\}$) under the following loss function:

$$L(\theta^{\mathrm{C}};\mathcal{E}^{\mathrm{u}})=\mathop{\mathbb{E}}_{\chi\sim\mathcal{E}}\left[\left(Q(s_{n'},a_{n'};\theta^{\mathrm{C}})-(r_{n'}+\gamma\max_{a'}Q^*(s_{n'+\Delta},a')\right)^2\right]. \tag{24}$$

The policy function of each actor is denoted by $u(s_{n'};\theta^{\mathrm{A}})$, which receives the environment state $s_{n'}$ as input and outputs a deterministic action at time-slot $s_{n'}$. Similarly, $u(s_{n'};\theta^{\mathrm{A}})$ is approximated via a neural network with parameters θ^{A}. The gradient of the policy function can be derived as

$$\begin{aligned} \nabla_{\theta^{\mathrm{A}}}u &\approx \mathbb{E}\left[\nabla_{\theta^{\mathrm{A}}}Q(s,a;\theta^{\mathrm{C}})|_{s=s_{n'},a=u(s_{n'};\theta^{\mathrm{A}})}\right] \\ &= \mathbb{E}\left[\nabla_a Q(s,a;\theta^{\mathrm{C}})|_{s=s_{n'},a=u(s_{n'})}\nabla_{\theta^{\mathrm{A}}}u(s;\theta^{\mathrm{A}})|_{s=s_{n'}}\right]. \end{aligned} \tag{25}$$

Similar to most DRL approaches, two target networks (Q' and u') with parameters ($\theta^{\mathrm{C}'}$ and $\theta^{\mathrm{A}'}$) are built to make the training process more stable, and $\theta^{\mathrm{C}'}$ and $\theta^{\mathrm{A}'}$ are periodically updated by θ^{C} and θ^{A}. When $\mathcal{M}$ transitions are sampled from $\mathcal{E}^{\mathrm{u}}$, $L(\theta^{\mathrm{C}};\mathcal{E}^{\mathrm{u}})$ and $\nabla_{\theta^{\mathrm{A}}}u$ can be given by

$$L(\theta^{\mathrm{C}};\mathcal{M})=\frac{1}{\mathcal{M}}\sum_{n'\in\mathcal{M}}(y_{n'}-Q(s_{n'},a_{n'};\theta^{\mathrm{C}})^2), \tag{26}$$

$$\nabla_{\theta^{\mathrm{A}}}u\,|_{s_{n'}}\approx\frac{1}{\mathcal{M}}\sum_{n'\in\mathcal{M}}\nabla_a Q(s,a;\theta^{\mathrm{C}})|_{s=s_{n'},a=u(s_{n'})}\nabla_{\theta^{\mathrm{A}}}u(s;\theta^{\mathrm{A}})|_{s=s_{n'}}, \tag{27}$$

where $y_{n'}=r_{n'}+\gamma Q'(s_{n'+\Delta},u'(s_{n'+\Delta};\theta^{\mathrm{A}'});\theta^{\mathrm{C}'})$.

4.4 Offloading Optimization of HT3O

Once the LS has determined $\mathbf{c}^{\mathrm{U}}(n)$ at the beginning of time-slot n ($n\%\Delta=0$), OS will generate $\boldsymbol{\alpha}(n+\eta)$ at each of the next Δ time-slots ($\eta\in[0,\Delta-1]$) based on $\mathbf{c}^{\mathrm{U}}(n)$. Since $\boldsymbol{\alpha}(n+\eta)$ are discrete binary variables, DQN [39] is adopted to develop OS where the action-value function $\mathcal{Q}(s_{n+\eta},\boldsymbol{\alpha}(n+\eta);\theta^{\alpha},\mathbf{c}^{\mathrm{U}}(n))$ (expected cumulative rewards of action $\boldsymbol{\alpha}(n+\eta)$ in state $s_{n+\eta}$) is approximated by a neural network with parameters θ^{α}.

Similarly, a target network $\mathcal{Q}'$ with parameters $\theta^{\alpha'}$ and a replay buffer $\mathcal{E}^{\alpha}$ are leveraged to facilitate the training process of OS. The learning experiences, each one of which can be expressed as

$$\kappa=\{s_{n+\eta},\boldsymbol{\alpha}(n+\eta),\dot{r}_{n+\eta},s_{n+\eta+1}\},$$

are sampled as mini-batches to update the $\mathcal{Q}$ value function

$$\begin{aligned} \mathcal{Q}_{n+\eta+1}(s_{n+\eta},\boldsymbol{\alpha}(n+\eta)) &= \mathcal{Q}_{n+\eta}(s_{n+\eta},\boldsymbol{\alpha}(n+\eta))+\beta[\dot{r}_{n+\eta} \\ &+\gamma\max_{\boldsymbol{\alpha}'}\mathcal{Q}'_{n+\eta}(s_{n+\eta+1},\boldsymbol{\alpha}')-\mathcal{Q}_{n+\eta}(s_{n+\eta},\boldsymbol{\alpha}(n+\eta))], \end{aligned} \tag{28}$$

where β is the learning rate of the neural network, based on the loss function

$$L(\theta^{\alpha};\mathcal{E}^{\alpha})=\mathop{\mathbb{E}}_{\kappa\in\mathcal{E}^{\alpha}}\left[\left(\dot{r}_{n+\eta}+\gamma\max_{\boldsymbol{\alpha}'}\mathcal{Q}'(s_{n+\eta+1},\boldsymbol{\alpha}';\theta^{\alpha'})-\mathcal{Q}(s_{n+\eta},\boldsymbol{\alpha}(n+\eta);\theta^{\alpha})\right)^2\right]. \tag{29}$$

The learning objective of OS's value function $\mathcal{Q}$ (parameterized by $\theta^{\alpha'}$) is to obtain desired policies of $\mathcal{P}2$, which is guided by the immediate internal reward $\dot{r}_{n+\eta}$ ($n\%\Delta=0,\eta\in[0,\Delta-1]$). Therefore, $\dot{r}_{n+\eta}$ is given by

$$\dot{r}_{n+\eta}=\left(\frac{1}{\eta+1}\sum_{\eta'\in[0,\eta]}t_{n+\eta'}^{\mathrm{mean}}\right)^{-1}. \tag{30}$$

4.5 Learning Algorithm of HT3O

The training algorithm of HT3O is formalized as shown in Algorithm 1, where two hierarchical stages (line 8 for the outer stage and line 12 for the inner stage) are comprised in each of the training episode (line 4).

Algorithm 1 Learning algorithm for HT3O

1: Initialize a critic $Q(s,a;\theta^{\mathrm{C}})$ and an actor $u(s;\theta^{\mathrm{A}})$ with random parameters θ^{C} and θ^{A} for LS; Initialize an action-value approximator $\mathcal{Q}(s,\boldsymbol{\alpha};\theta^{\alpha})$ with random parameters θ^{α};
2: Initialize the target network Q', u', and $\mathcal{Q}'$ with parameters $\theta^{\mathrm{C}'}\leftarrow\theta^{\mathrm{C}}$, $\theta^{\mathrm{A}'}\leftarrow\theta^{\mathrm{A}}$, and $\theta^{\alpha'}\leftarrow\theta^{\alpha}$;
3: Initialize replay buffer $\mathcal{E}^{\mathrm{u}}$ and $\mathcal{E}^{\alpha}$;

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3071531, IEEE Internet of Things Journal

for episode $= 1$ to M **do**
  $n = 1$;
  Initialize the exploration threshold $\epsilon^u = 0.9$, $\epsilon^\alpha = 0.9$;
  Initialize the LU-MEC network;
  while $n \leq N$ **do**
    Get the current state s_n;
    $\mathbf{c}^{\mathrm{U}}(n) = \begin{cases} u(s_n;\theta^{\mathrm{A}}) & \text{random()} < \epsilon^{\mathrm{u}} \\ \text{choose } u \text{ randomly} & \text{else} \end{cases}$;
    $R = 0, \quad \eta = 0$;
    while $\eta < \Delta$ **do**
      $\boldsymbol{\alpha}(n+\eta) = \begin{cases} \max\limits_{\boldsymbol{\alpha}} \mathcal{Q}(s_{n+\eta}, \boldsymbol{\alpha}; \theta^\alpha, \mathbf{c}^{\mathrm{U}}(n)) & \text{random()} < \epsilon^\alpha \\ \text{choose } \boldsymbol{\alpha} \text{ randomly} & \text{else} \end{cases}$;
      The LU-MEC network execute $\boldsymbol{\alpha}(n+\eta)$ and generate next state s' and external immediate reward $r_{n+\eta}$;
      $R = R + r_{n+\eta}$;
      Get internal immediate reward $\dot{r}_{n+\eta}$ from LS;
      Store transition $\{s_{n+\eta}, \boldsymbol{\alpha}(n+\eta), \dot{r}_{n+\eta}, s'\}$ in $\mathcal{E}^\alpha$;
      Update network parameters θ^α via gradient descent on $L(\theta^\alpha; \mathcal{E}^\alpha)$ of equation (29);
      $\eta = \eta + 1$;
      $s_{n+\eta} = s'$;
      Anneal ϵ^α;
    end while
    Store transition $\{s_n, \mathbf{c}^{\mathrm{U}}(n), R, s_{n+\eta}\}$ in $\mathcal{E}^{\mathrm{u}}$;
    Sample $\mathcal{M}$ transitions $\{s_j, \mathbf{c}^{\mathrm{U}}(j), R_j, s_{j+1}\}$ from $\mathcal{E}^{\mathrm{u}}$;
    $y_j = R_j + \gamma Q'(s_{j+1}, u'(s_{j+1}; \theta^{\mathrm{A}'}); \theta^{\mathrm{C}'})$;
    Update the critic network parameters θ^{C} by minimizing $L(\theta^{\mathrm{C}}; \mathcal{M})$ of equation (26);
    Update the actor network parameters θ^{A} with the sampled gradient $\nabla_{\theta^{\mathrm{A}}} u\,|_{s_j}$ of equation (27);
    Update target network $\theta^{\mathrm{C}'}$ and $\theta^{\mathrm{A}'}$ with θ^{C} and θ^{A} periodically;
    $n = n + \eta$;
    Anneal ϵ^{u};
  end while
end for

During the task execution time (N time-slots), LS and OS work alternately for $\lfloor N/\Delta \rfloor$ steps. In each step, LS and OS work in two different hierarchical stages as shown in Fig. 4.

- In the outer stage, LS receives the state s_n and generates a movement action $\mathbf{c}^{\mathrm{U}}(n)$ output by the policy network $u(s_n; \theta^{\mathrm{A}})$. $\mathbf{c}^{\mathrm{U}}(n)$ will be performed by UAVs at the first slot of the current Δ slots and input to OSs as the precondition for their offloading scheduling.
- In the inner stage, OS receives the network state and generates an offloading policy under the determined UAVs' locations $\mathbf{c}^{\mathrm{U}}(n)$, in each of the following Δ time-slots.

4.6 Performance Analysis

In this subsection, the convergence performance of HT3O and the lower bound of the computation delay are discussed, respectively.

Proposition 1 (Convergence of OS): If each offloading-action $\boldsymbol{\alpha}$ is visited in each inner state infinitely often, OS converges to the optimal offloading value function $\mathcal{Q}^*_{\boldsymbol{\alpha}}$.

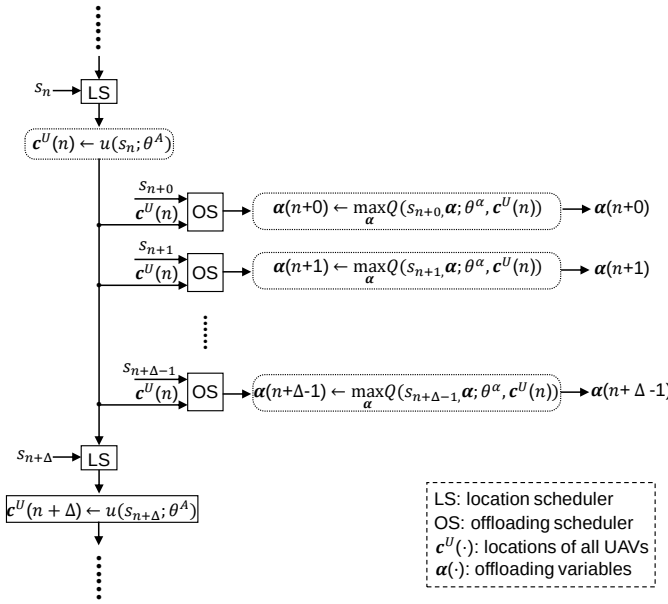


Fig. 4. Workflow of the two hierarchical stages.

Proof. According to equation (28), we can rewrite

$$\mathcal{Q}_{t+1}(s_t, \boldsymbol{\alpha}(t)) = (1-\beta)\mathcal{Q}_t(s_t, \boldsymbol{\alpha}(t)) + \beta[\dot{r}_t + \gamma \max_{\boldsymbol{\alpha}'} \mathcal{Q}_t(s_{t+1}, \boldsymbol{\alpha}')],$$

where $t = n + \eta$. Subtracting $\mathcal{Q}^*(s_t, \boldsymbol{\alpha}(t))$ from both sides, we can get

$$\begin{aligned}\mathcal{Q}_{t+1}(s_t,\boldsymbol{\alpha}(t)) - \mathcal{Q}^*(s_t,\boldsymbol{\alpha}(t)) &= (1-\beta)(\mathcal{Q}_t(s_t,\boldsymbol{\alpha}(t)) - \mathcal{Q}^*(s_t,\boldsymbol{\alpha}(t)) \\ &\quad + \beta[\dot{r}_t + \gamma \max_{\boldsymbol{\alpha}'} \mathcal{Q}_t(s_{t+1},\boldsymbol{\alpha}') - \mathcal{Q}^*(s_t,\boldsymbol{\alpha}(t))].\end{aligned}$$

Letting $\Psi_t(s, \boldsymbol{\alpha}) = \mathcal{Q}_t(s, \boldsymbol{\alpha}) - \mathcal{Q}^*(s, \boldsymbol{\alpha})$, then it yields

$$\begin{aligned}\Psi_t(s_t, \boldsymbol{\alpha}_t) &= (1-\beta)\Psi_t(s_t, \boldsymbol{\alpha}_t) \\ &\quad + \beta[\dot{r}_t + \gamma \max_{\boldsymbol{\alpha}'} \mathcal{Q}_t(s_{t+1}, \boldsymbol{\alpha}') - \mathcal{Q}^*(s_t, \boldsymbol{\alpha}(t))].\end{aligned}$$

Let the gap between the Q value updating and the optimal Q value be denoted as

$$G_t(s, \boldsymbol{\alpha}) = \dot{r}_t + \gamma \max_{\boldsymbol{\alpha}'} \mathcal{Q}_t(s, \boldsymbol{\alpha}') - \mathcal{Q}^*(s, \boldsymbol{\alpha}),$$

then we can get the expected gap $\mathbb{E}[G]$ based on random sample states $\mathcal{K}$ as

$$\begin{aligned}\mathbb{E}[G_t(s, \boldsymbol{\alpha})] &= \sum_{\{s,\boldsymbol{\alpha},\dot{r},s'\} \in \mathcal{K}} \Pr(s'|s, \boldsymbol{\alpha})[\dot{r}_t + \gamma \max_{\boldsymbol{\alpha}'} \mathcal{Q}_t(s', \boldsymbol{\alpha}') - \mathcal{Q}^*(s, \boldsymbol{\alpha})] \\ &= (\mathbf{K}\mathcal{Q}_t(s, \boldsymbol{\alpha})) - \mathcal{Q}^*((s, \boldsymbol{\alpha})),\end{aligned}$$

where

$$\mathbf{K} = \sum \Pr(s'|s, \boldsymbol{\alpha})[\dot{r}_t + \gamma \max_{\boldsymbol{\alpha}'} \mathcal{Q}_t(s', \boldsymbol{\alpha}')]$$

is the value contraction operator (Bertsekas & Tsitsiklis, 1989) and satisfies

$$\|\mathbf{K}Q_1 - \mathbf{K}Q_2\|_\infty \leq \gamma \|Q_1 - Q_2\|_\infty.$$

Given the fact that $\mathcal{Q}^* = \mathbf{K}\mathcal{Q}^*$, we have

$$\begin{aligned}\|\mathbb{E}[G_t(s, \boldsymbol{\alpha})]\|_\infty &= \|(\mathbf{K}\mathcal{Q}_t(s, \boldsymbol{\alpha})) - \mathbf{K}\mathcal{Q}^*((s, \boldsymbol{\alpha}))\|_\infty \\ &\leq \gamma \|\mathcal{Q}_t(s, \boldsymbol{\alpha}) - \mathcal{Q}^*(s, \boldsymbol{\alpha})\|_\infty \\ &= \gamma \|\Psi_t(s, \boldsymbol{\alpha})\|_\infty.\end{aligned}$$

Similarly, we can get

$$\mathbf{Var}[G_t(s, \boldsymbol{\alpha})] \leq C(1 + \|\Psi_t(s, \boldsymbol{\alpha})\|^2_\infty),$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3071531, IEEE Internet of Things Journal

for some constant C. Therefore, according to Theorem 1 of [40], the gap $G_t(s, \boldsymbol{\alpha})$ could converge to zero with probability one, which further indicates that OS could converge to $\mathcal{Q}^*_{\boldsymbol{\alpha}}$ under given conditions. □

Proposition 2 (Convergence of LS): If each movement-action $\mathbf{c}^{\mathrm{U}}$ is visited in each outer state infinitely often, LS converges to the optimal location scheduling policy u^*.

Proof. The proof is similar to Proposition 1.

Theorem 1 (Convergence of HT3O): If each movement-action $\mathbf{c}^{\mathrm{U}}$ and offloading-action $\boldsymbol{\alpha}$ are visited in each state infinitely often, HT3O converges to the optimal computation offloading policy Π^*_a ($a = \{\mathbf{c}^{\mathrm{U}}, \boldsymbol{\alpha}\}$).

Proof. If each offloading-action $\boldsymbol{\alpha}$ is visited in each state infinitely often, OS will converge to the optimal offloading value $\mathcal{Q}^*_{\boldsymbol{\alpha}}$, which indicates OS could converge to a deterministic offloading policy in a predetermined location. Therefore, according to Proposition 2, HT3O could converge to the optimal computation offloading if each location is visited in each state infinitely often and lead to the convergence of deterministic offloading policies. □

Proposition 3 : The optimal offloading policy of MD m at time-slot n is $\alpha_m^{u^*}(n) = 1$, if

$$f_m \leq ((R_m^{u^*}(n)BC_m(n))^{-1} + f_{u^*}^{-1})^{-1},$$

where

$$u^* = \arg\min_{u \in \mathcal{U}} d_m^u(n).$$

Proof. By (15), the local computing delay is

$$\bar{t}_m(n) = \frac{1}{f_m} D_m(n) C_m(n),$$

and by (9) and (13), the optimal edge computing delay is

$$\begin{aligned} t_m^{\text{edge}}(n) &= \max_{u \in \mathcal{U}} \left\{ \hat{t}_m^u(n) + \check{t}_m^u(n) \right\} \\ &= \hat{t}_m^{u^*}(n) + \check{t}_m^{u^*}(n) \\ &= \left(\frac{1}{R_m^{u^*}(n)BC_m(n)} + \frac{1}{f_{u^*}}\right) \alpha_m^{u^*}(n) D_m(n) C_m(n). \end{aligned}$$

Due to the fact that $\alpha_m^{u^*}(n) = 1$ when MD m chooses the nearest UAV $u^* = \arg\min_{u \in \mathcal{U}} d_m^u(n)$ to obtain the optimal edge computing delay,

$$t_m^{\text{edge}}(n) = (1/R_m^{u^*}(n)BC_m(n) + 1/f_{u^*}) D_m(n) C_m(n).$$

Therefore, the optimal offloading policy of MD m at time-slot n should be $\alpha_m^{u^*}(n) = 1$ when

$$\frac{1}{f_m} \geq \frac{1}{R_m^{u^*}(n)BC_m(n)} + \frac{1}{f_{u^*}},$$

i.e., $f_m \leq ((R_m^{u^*}(n)BC_m(n))^{-1} + f_{u^*}^{-1})^{-1}$. □

Theorem 2: The lowest computing delay at time-slot n is

$$t^* = \min_{m \in \mathcal{M}} \min\{f_m^{-1}, (R_m^{u^*}(n)BC_m(n))^{-1} + f_{u^*}^{-1}\} D_m(n) C_m(n),$$

if the optimal offloading policy could be performed without violating the constraints, where

$$u^* = \arg\min_{u \in \mathcal{U}} d_m^u(n).$$

Proof. The optimal delay for MD m at time-slot n should be

$$t_m^*(n) = \min\{f_m^{-1}, (R_m^{u^*}(n)BC_m(n))^{-1} + f_{u^*}^{-1}\} D_m(n) C_m(n),$$

if MD m could choose the optimal computation policy (offloading or local computing) according to Proposition 3, where $u^* = \arg\min_{u \in \mathcal{U}} d_m^u(n)$. Therefore, the lowest computing delay at time-slot n is $t^* = \min_{m \in \mathcal{M}} t_m^*(n)$. □

5 Simulation Results

In this section, the performance of HT3O is evaluated via numerical simulation. With primary network parameters as summarized in Table 2, experiments are performed in a GPU-based server (Xeon Gold 6148, 128G Memory, Tesla V100 GPU, Python3.7, Tensorflow-gpu 1.15) from two aspects: one is the performance impacts of different network settings on HT3O, the other is the performance comparisons (w.r.t. the learning efficiency, computation efficiency, and average task delay) between HT3O and other approaches which include:

TABLE 2
Network parameters of numerical simulations

Parameters (Notations)	Values
The ground area of the network	1×1 km^2
The fixed height of UAVs (H)	50 m
The number of UAVs (U)	$\{4, 8, 16\}$
The number of MDs (M)	64
Task execution time (T)	$\{50, 100, 200\}$ s
The number of time-slots (N)	100
The scale of the normal distribution (ι)	$\{0.5, 1, 2\}$
The move cycles of UAVs (Δ)	10
The maximum speed of UAVs (V^{U})	30 m/s
The workload of UAVs (M_{g})	9.65 kg
Channel power gain (g_0)	-50 dB
Task size of MD m at slot n ($D_m(n)$)	$[1 \times 10^5, 2 \times 10^5]$ bits
CPU cycles for one bit of task ($C_m(n)$)	$[100, 1000]$
Maximum admissible delay of tasks ($\Gamma_m(n)$)	$[0.01, 0.1]$ s
The noise power of the UAV receiver (σ^2)	10^{-9} W
Transmission bandwidth (B)	1 MHz
Local computation capability of MDs (f_m)	$[0.5, 1]$ GHz
Edge computation capability of UAVs (f_u)	$[5, 10]$ GHz

- Baseline approaches. (1) Non-Offloading (labeled as NO): all tasks are performed locally at individual MDs. (2) Random-Offloading (labeled as RO): U MDs are randomly selected to offload tasks to UAVs.
- BCD-SCA approaches. By combining BCD (for alternate optimization among block variables) [41] with SCA (for approximated convex optimization of non-convex block variables) [42], BCD-SCA has been widely applied as a heuristic-based approach for computation offloading in MEC networks [7], [8], [31].
- RL approaches. RL based approaches are attracting growing attention for computation offloading of MEC (e.g., DROO [43], DDDQ [44], RLAA [12]), due to their

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3071531, IEEE Internet of Things Journal

suitability for real-time online scheduling for systems with time-varying network dynamics compared with heuristic-based methods.

MDs are randomly initialized at the ground area, while UAVs are initially located via K-Means to speed up the convergence of HT3O. The initialization of UAVs' locations works as follows:

- Set the number of K-Means clusters as U;
- Randomly set the centroids of U clusters;
- Iterate the following operations until the update of any centroid is less than a specified threshold: (1) Compute the distance between each MD and centroid; (2) Designate each MD to the closest centroid (cluster); (3) Update the centroid of each cluster according to the newly designated MDs.

In addition, the HT3O model is trained via deep reinforcement learning, whose hyper-parameters are tuned via evaluations with different settings and fixed with values as shown in Table 3.

TABLE 3
KEY TRAINING PARAMETERS OF HT3O

Hyper-Parameter	Value
Layer type	fully connected
Number of hidden layers	128, 128, 64
Learning rate (LS)	Critic=0.001, Actor=0.0001
Learning rate (OS)	TargetNet=0.0001, EvalNet=0.0001
Optimizer	Adam
ϵ_{greedy}	$0.9 \rightarrow 1.0$
Discount factor	0.9
Batch size	32
Max iteration	20000
Gradient descent	Adam

5.1 Impacts of Different Network Settings on HT3O

In this part, we evaluate performances of HT3O with different network configurations (U, T, and ι), where the default values of U, T, and ι are 8, 100s, and 1, respectively, if not specified.

In Fig. 5, we evaluate the performances of HT3O with different numbers of UAVs ($U = 4, 8, 16$). It can be noticed that the convergence of HT3O will become slow when the number of UAVs changes from 4, 8, to 16: HT3O converges at about time-slot 2100 when $U = 4$, while HT3O converges at about time-slot 3500 and 5100 when $U = 8$ and 16, respectively. It indicates that the convergence performance of HT3O is inversely relative to the number of UAVs, which could be resulted from the reason that more number of UAVs give rise to higher difficulties for LS to acquire desired UAV locations. However, when U changes from 4 to 16, the total utility of each episode (after convergence) increases from about 1700 to almost 2400, which indicates that more number of UAVs could provide more opportunities for MDs to perform computation offloading so as to obtain lower task delays.

In Fig. 6, we evaluate the performance of HT3O with different task execution time ($T = 50, 100, 200s$). It can be found that the convergence time of HT3O increases from about time-slot 2700 to 3500 and 4300, when T decreases from 200 s to 100 s and 50 s, respectively. It indicates that short task execution time brings troubles for HT3O to converge, which could be caused by the following reasons: short T results in short time-slot length τ, since the number of time-slots (N) is fixed; hence, the constraints $C1$ and $C8$ are more difficult to be satisfied during the policy-search process of LS and OS, respectively, which gives rise to longer convergence time. In addition, it can be seen that short T results in low total utilities, which could be caused due to the reason that the more strict is $C1$ the more difficult is it for UAVs to move close enough (with sufficient small h and large R to achieve low $\hat{t}$ to satisfy $C8$) to delay-critical MDs to provide edge computing services.

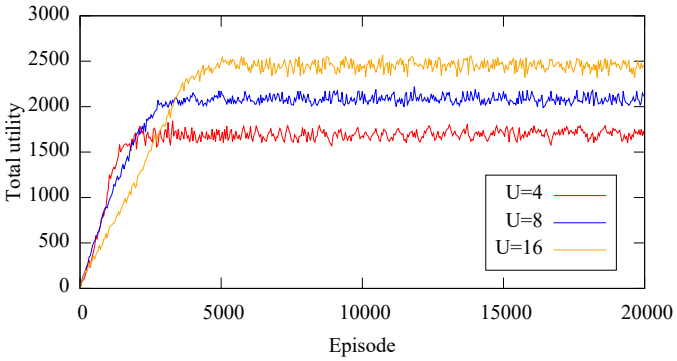


Fig. 5. Performance evaluation of HT3O with different numbers of UAVs.

In Fig. 7, we evaluate the performance of HT3O with different normal-distribution scales of MDs' movements. It can be observed that the convergence time of HT3O becomes long when the move scale ι increases, which demonstrates that the larger is ι the harder is it for HT3O to converge. This could be because that larger ι gives rise to larger ranges of MDs' movements which consequently results in more difficulties for LS to obtain promising positions of UAVs. Furthermore, it can be found that the utility increases obviously when ι decreases. This could be resulted from the following reasons: the movement ranges of MDs become small when ι is small, which brings more choices for UAVs to select desirable MDs for computation offloading; therefore, edge computing ability could be utilized more appropriately to reduce task delays.

In Fig. 8, we further show the movement decisions of UAVs generated by trained HT3O at different time-slots. It can be found that the UAVs tend to serve different clusters of MDs and move towards the centroids of individual clusters at the beginning of each Δ time-slots. Since MDs could move within near ranges at each time-slot and UAVs could only move at the beginning of each Δ time-slots, the LS of HT3O tends to guide UAVs moving to different clusters dynamically at different time-slots. The consequent benefits could be that the computation offloading can be balanced among all UAVs and the MDs of each cluster can be chosen by OS of HT3O for computation offloading according to the dynamical network states and task requirements.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3071531, IEEE Internet of Things Journal

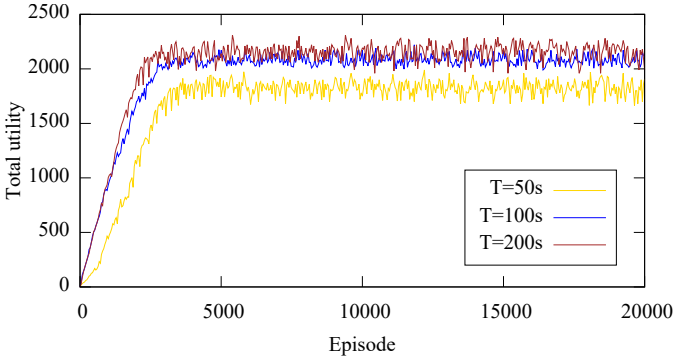


Fig. 6. Performance evaluation of HT3O with different task execution times.

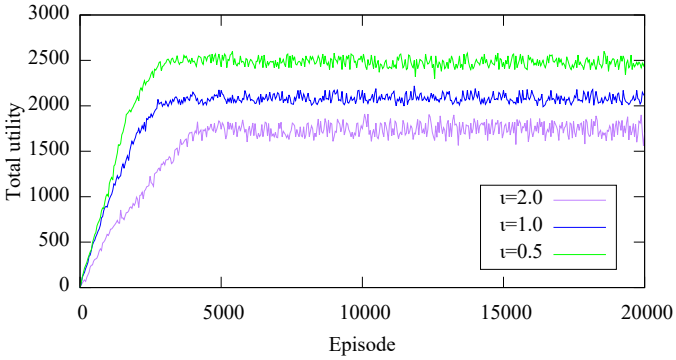
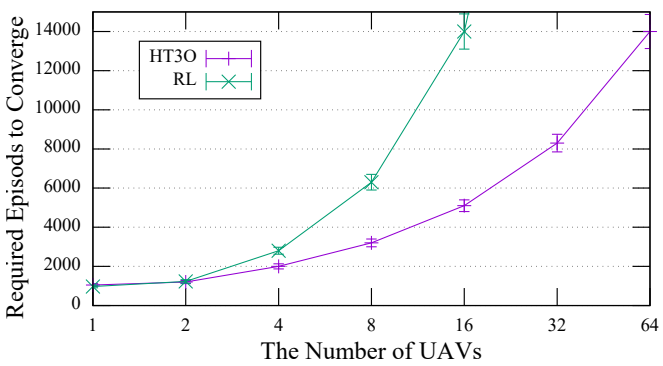


Fig. 7. Performance evaluation of HT3O with different move-scales of MDs.

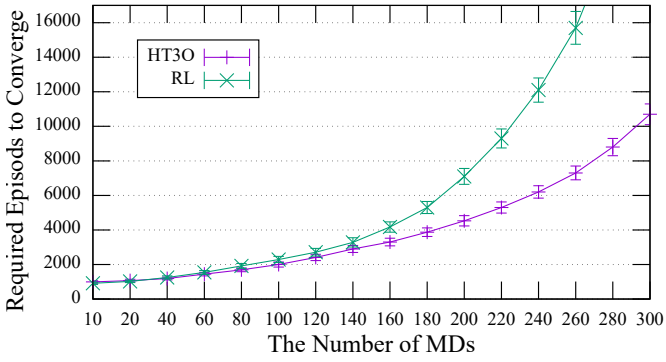
5.2 Performance Comparisons of Learning Efficiency

In this part, we compare the learning efficiencies of learning-based approaches (HT3O and RL) with respect to different numbers of UAVs and MDs.

As shown in Fig. 9a, the required number of episodes (RNE) to achieve convergence for RL is a little lower than that for HT3O when the number of UAVs is 1. This could be resulted from the reason that the cost of hierarchical scheduling for HT3O is higher than that of joint scheduling for RL when there is only one single UAV in the LU-MEC network. The RNE of RL grows dramatically when U increases from 4 to 16, whereas the RNE of HT3O is still under six thousand. When U increases from 32 to 64, RL even can not converge with limited time, whereas HT3O still converges around about ten thousand. It indicates that



(a) Different numbers of UAVs.

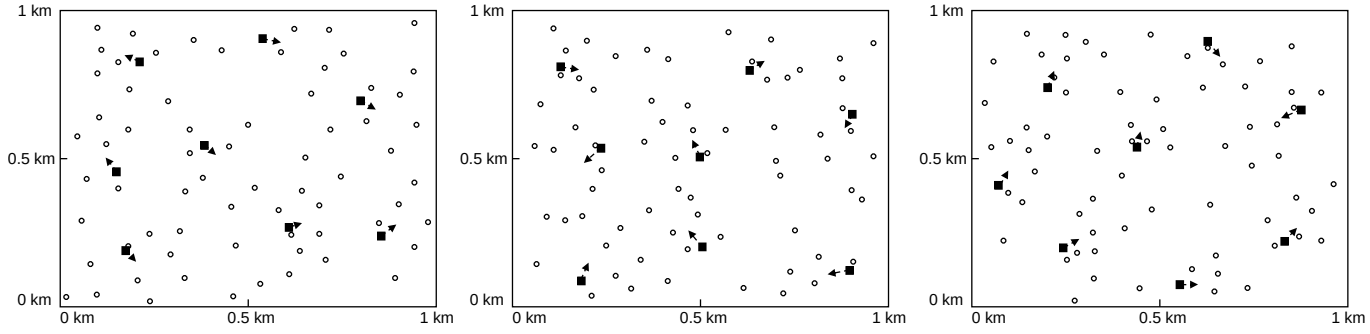


(b) Different numbers of MDs.

Fig. 9. Performance comparison between learning approaches w.r.t. the required episodes of convergence for LU-MEC.

HT3O could benefit a lot from the problem decomposition and hierarchical scheduling method which not only reduces the complexity of each sub-problem but also improves the learning efficiency by reusing lower-layer policies.

A similar trend could be found in Fig. 9b, where the RNE of RL is a little lower than that of HT3O when the number of MDs is small and become much higher than that of HT3O when the number of MDs increases. When the number of MDs increases from 200 to 300, RL converges at more than 10000 episodes and even unfeasibly within limited time while HT3O can still converge within 10000 episodes. This demonstrates that HT3O achieves much higher learning efficiency than RL does when the LU-MEC network scale grows w.r.t. the number of MDs.



(a) 10 time-slots. (b) 50 time-slots. (c) 90 time-slots.

Fig. 8. Movement decisions of UAVs at different time-slots (the circles, squares and arrows represent MDs, UAVs and movement decisions of UAVs, respectively).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3071531, IEEE Internet of Things Journal

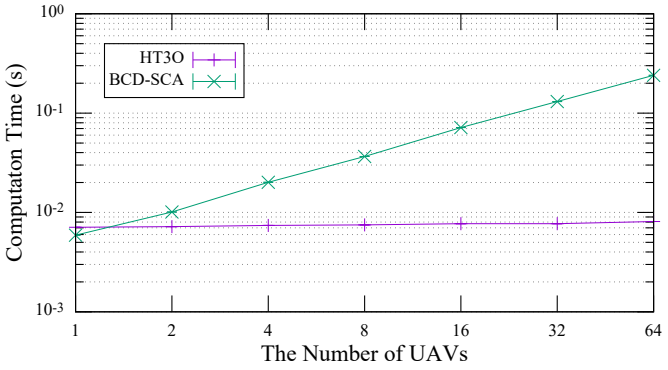
5.3 Performance Comparison of Computation Efficiency

We also evaluate the computation efficiencies of HT3O and BCD-SCA during the running of LU-MEC networks with different scales (considering the number of UAVs and MDs), where HT3O is tested using the models after the convergence of training.

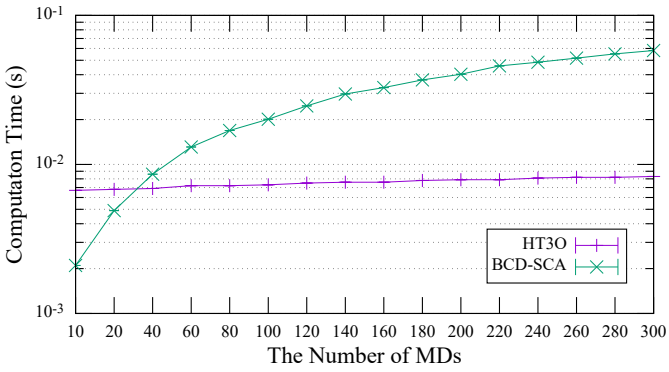
As shown in Fig. 10a, BCD-SCA generates a scheduling action in less than 6×10^{-2} s when there is only one single UAV enabling the LU-MEC network, while HT3O spends a little higher time (more than 7×10^{-2} s). Nevertheless, the time spent by BCD-SCA grows dramatically when U increases from 2 to 64, while HT3O can still keep a much lower time cost around 8×10^{-2} s. When U is larger than 32, the time cost of BCD-SCA becomes higher than 0.1 s, which could lead to the computational infeasibility of BCD-SCA in real-time scheduling environments.

Similarly, HT3O can achieve much higher computation efficiency than BCD-SCA do when the number of MDs increases as shown in Fig. 10b. When U is less than 40, BCD-SCA spends less time (2×10^{-3} s and 5×10^{-3} s) than HT3O does to generate one scheduling action. However, when U increases from 60 to 300, the time cost of BCD-SCA climbs rapidly towards 6×10^{-2} s while the time cost of HT3O is still lower than 9×10^{-3} s.

Overall, it indicates that HT3O is more suitable than conventional BCD methods for real-time applications in LU-MEC networks with dynamic system states.



(a) Different numbers of UAVs.



(b) Different numbers of MDs.

Fig. 10. Performance comparison between HT3O and BCD-SCA w.r.t. the required computation time (s) to generate one scheduling action in each time-slot of the LU-MEC network.

5.4 Performance Comparison of Average Task Delay

In Fig. 11, we compare the average delay of different approaches with different numbers of UAVs. It can be noticed that HT3O can achieve the lowest delay, while NO achieves the highest delay, compared with other approaches. It indicates the substantial benefits of adopting HT3O for computation offloading in LU-MEC networks. In addition, when the number of UAVs increases from 4 to 8 (8 to 16), the average delay of NO stays nearly unchanged (around 100) due to its constant local computing policy, and the average delay of HT3O decreases about 17.02% (14.14%) while the average delay of RO, BCD-SCA, and RL only decreases 5.2% (3.69), 9.52% (6.74%), and 10.36% (7.87%), respectively. This implies that HT3O could achieve higher performance improvements than other approaches do with respect to the increment of the number of UAVs.

In Fig. 12, we compare the average delays of different approaches with different lengths of experiment time. It can be found that HT3O can achieve the lowest delay compared with other approaches, which implies the promising performance improvements of HT3O over other approaches with different lengths of experiment time. In addition, the experiment doesn't see large variations of the average delays for NO and RO when the length of experiment time changes from 50 s to 200 s, which could be caused by the reason that no matter what the length of the experiment time (time-slot) is, NO (RO) remains performing local computing (random offloading) with no substantial impact on average delay. It can also be seen that when the length of experiment time changes from 50 s to 150 s, the average delay of HT3O declines more desirably than those of BCD-SCA and RL do,

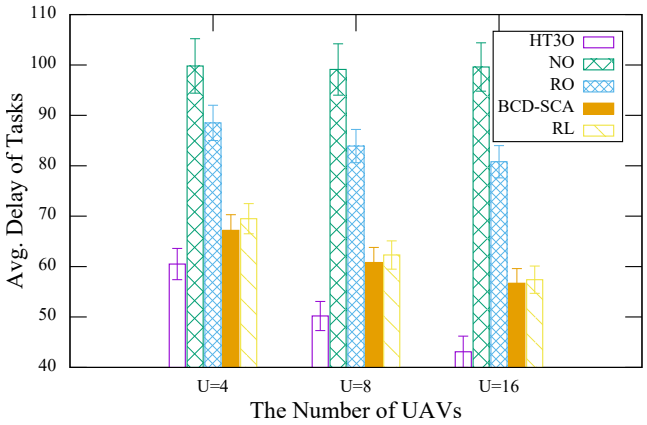


Fig. 11. Performance comparison of different approaches in terms of average delays w.r.t. different numbers of UAVs.

which indicates that LS of HT3O could benefit more from the longer time-slot to schedule UAVs to desired locations for computation offloading than BCD-SCA and RL do.

In Fig. 13, we compare the average delay of different approaches with different scales of MDs' movements. It can be seen that HT3O achieves a task delay that is much lower than those of other approaches for any scale of MDs' movements. In addition, it can also be noticed that when the scale increase from 1.0 to 2.0, the task delays of BCD-SCA and RL increase remarkably while the delay of HT3O can still keep at a low level. This demonstrates that HT3O is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3071531, IEEE Internet of Things Journal

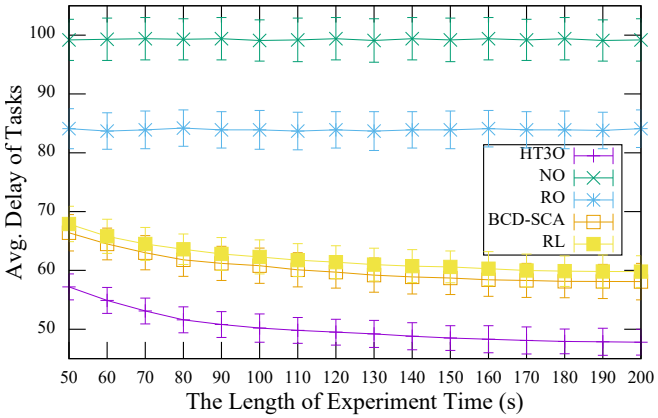


Fig. 12. Performance comparison of different approaches in terms of average delays w.r.t. different lengths of experiment time.

more adaptable to the changes of MDs' movement-scales than other approaches are. Furthermore, the experiment also sees similar delays of NO and RO as those in Fig. 12, which indicates the insensitivity of NO and RO to MDs' movement-scales.

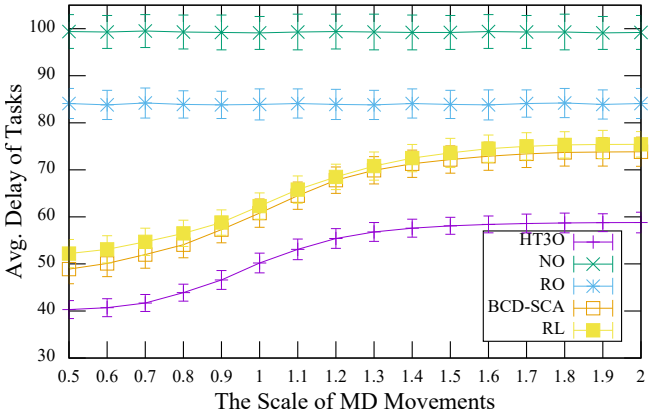


Fig. 13. Performance comparison of different approaches in terms of average delays w.r.t. different scales of MDs' movements.

6 Conclusions and Future Work

This paper has investigated a large-scale multi-UAV assisted MEC network where numerous mobile MDs can either offload tasks to UAVs or execute them locally. Specifically, the movements of UAVs and computation-offloading are jointly scheduled to minimize the average task delay (i.e., maximize the average computation rate) of all MDs. The optimization problem is hierarchically decomposed into two-layered sub-problems (the trajectory optimization sub-problem and offloading optimization sub-problem). To reduce the complexity of each sub-problem and improve the learning efficiency by reusing lower-layer policies, we propose a hierarchical optimization approach (called HT3O) based on hierarchical reinforcement learning, where the policies of two sub-problems are alternately optimized and learned via interaction with the MEC network. Finally, the HT3O model is built with neural networks and trained via deep reinforcement learning. Compared to conventional approaches, the proposed HT3O approach can relieve the heavy burden of hardly solving the mixed integer nonlinear programming problem and generate real-time scheduling decisions in large-scale UAV-assisted MEC networks. Simulation results show that HT3O can achieve satisfactory performance improvements over baseline approaches and state-of-the-art approaches with various network settings in terms of convergence efficiency, computation efficiency, and average task delay, respectively.

However, further investigations are still needed in future work to address the challenge of how to guarantee convergence when MDs move within large ranges. In addition, it would also be of interest to explore other research scenarios to extend the work of this paper, e.g., partial offloading, wireless energy harvesting, collaboration between UAV-BSs and ground-BSs.

References

[1] 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.

[2] Y. Zhang, B. Di, P. Wang, J. Lin, and L. Song, "Hetmec: Heterogeneous multi-layer mobile edge computing in the 6 g era," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4388–4400, 2020.

[3] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4804–4814, 2018.

[4] C. Zhan, H. Hu, X. Sui, Z. Liu, and D. Niyato, "Completion time and energy optimization in the uav-enabled mobile-edge computing system," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7808–7822, 2020.

[5] F. Luo, C. Jiang, S. Yu, J. Wang, Y. Li, and Y. Ren, "Stability of cloud-based uav systems supporting big data acquisition and processing," *IEEE Transactions on Cloud Computing*, vol. 7, no. 3, pp. 866–877, 2017.

[6] F. Zhou, R. Q. Hu, Z. Li, and Y. Wang, "Mobile edge computing in unmanned aerial vehicle networks," *IEEE Wireless Communications*, vol. 27, no. 1, pp. 140–146, 2020.

[7] 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, 2018.

[8] H. Guo and J. Liu, "Uav-enhanced intelligent offloading for internet of things at the edge," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2737–2746, 2019.

[9] X. Hou, Z. Ren, J. Wang, S. Zheng, and H. Zhang, "Latency and reliability oriented collaborative optimization for multi-uav aided mobile edge computing system," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 150–156.

[10] 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.

[11] J. Li, Q. Liu, P. Wu, F. Shu, and S. Jin, "Task offloading for uav-based mobile edge computing via deep reinforcement learning," in *2018 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, 2018, pp. 798–802.

[12] L. Wang, P. Huang, K. Wang, G. Zhang, L. Zhang, N. Aslam, and K. Yang, "Rl-based user association and resource allocation for multi-uav enabled mec," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 2019, pp. 741–746.

[13] M. Mukherjee, V. Kumar, A. Lat, M. Guo, R. Matam, and Y. Lv, "Distributed deep learning-based task offloading for uav-enabled mobile edge computing," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2020, pp. 1208–1212.

[14] 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*, 2020.

[15] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[16] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2651–2664, 2018.

[17] J. Bi, H. Yuan, S. Duanmu, M. C. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet of Things Journal*, 2020.

[18] S. Zhu, L. Gui, J. Chen, Q. Zhang, and N. Zhang, "Cooperative computation offloading for uavs: A joint radio and computing resource allocation approach," in *2018 IEEE International Conference on Edge Computing (EDGE)*. IEEE, 2018, pp. 74–79.

[19] L. Yang, H. Zhang, M. Li, J. Guo, and H. Ji, "Mobile edge computing empowered energy efficient task offloading in 5g," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 7, pp. 6398–6409, 2018.

[20] Y. Liu, Y. Li, Y. Niu, and D. Jin, "Joint optimization of path planning and resource allocation in mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 19, no. 9, pp. 2129–2144, 2020.

[21] S. Yu, M. Liu, W. Dou, X. Liu, and S. Zhou, "Networking for big data: A survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 531–549, 2016.

[22] Z. Ning, P. Dong, X. Wang, M. S. Obaidat, X. Hu, L. Guo, Y. Guo, J. Huang, B. Hu, and Y. Li, "When deep reinforcement learning meets 5g-enabled vehicular networks: A distributed offloading framework for traffic big data," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1352–1361, 2019.

[23] G. Li, B. Feng, H. Zhou, Y. Zhang, K. Sood, and S. Yu, "Adaptive service function chaining mappings in 5g using deep q-learning," *Computer Communications*, vol. 152, pp. 305–315, 2020.

[24] A. Shakarami, M. Ghobaei-Arani, and A. Shahidinejad, "A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective," *Computer Networks*, p. 107496, 2020.

[25] S. Vimal, M. Khari, N. Dey, R. G. Crespo, and Y. H. Robinson, "Enhanced resource allocation in mobile edge computing using reinforcement learning based moaco algorithm for iiot," *Computer Communications*, vol. 151, pp. 355–364, 2020.

[26] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, S. U. Khan, and P. Li, "A double deep q-learning model for energy-efficient edge scheduling," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 739–749, 2018.

[27] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Performance optimization in mobile-edge computing via deep reinforcement learning," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. IEEE, 2018, pp. 1–6.

[28] Q. Liu, L. Shi, L. Sun, J. Li, M. Ding, and F. Shu, "Path planning for uav-mounted mobile edge computing with deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5723–5728, 2020.

[29] 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*, 2020.

[30] B. Yang, X. Cao, J. Bassey, X. Li, and L. Qian, "Computation offloading in multi-access edge computing: A multi-task learning approach," *IEEE Transactions on Mobile Computing*, 2020.

[31] 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, 2017.

[32] 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, 2018.

[33] S. K. Islam and M. R. Haider, *Sensors and low power signal processing*. Springer Science & Business Media, 2009.

[34] 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, pp. 4576–4589, 2019.

[35] J. Zhang, L. Zhou, Q. Tang, E. C.-H. Ngai, X. Hu, H. Zhao, and J. Wei, "Stochastic computation offloading and trajectory scheduling for uav-assisted mobile edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3688–3699, 2018.

[36] Y. Liu, K. Xiong, Q. Ni, P. Fan, and K. B. Letaief, "Uav-assisted wireless powered cooperative mobile edge computing: Joint offloading, cpu control, and trajectory optimization," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2777–2790, 2019.

[37] W. Yuan and K. Nahrstedt, "Energy-efficient cpu scheduling for multimedia applications," *ACM Transactions on Computer Systems (TOCS)*, vol. 24, no. 3, pp. 292–331, 2006.

[38] 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," *CoRR*, vol. abs/1509, no. 02971, pp. 1–14, 2015.

[39] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[40] T. Jaakkola, M. I. Jordan, and S. P. Singh, "Convergence of stochastic iterative dynamic programming algorithms," in *Advances in neural information processing systems*, 1994, pp. 703–710.

[41] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *Journal of optimization theory and applications*, vol. 109, no. 3, pp. 475–494, 2001.

[42] M. Razaviyayn, "Successive convex approximation: Analysis and applications," Ph.D. dissertation, University of Minnesota, 2014.

[43] L. Huang, S. Bi, and Y. J. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, 2019.

[44] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 44–55, 2017.

Tao Ren received the B.S., M.S., and Ph.D. degrees in computer science from the PLA Information Engineering University, Zhengzhou, China, in 2004, 2007, and 2011, respectively. He is currently an associate research fellow with the Hangzhou Innovation Institute of Beihang University, Hangzhou, China. His research interests include mobile edge computing, reinforcement learning and artificial intelligence in optimal control of industrial IoT.

Jianwei Niu (SM'07) received the M.S. and Ph.D. degrees in computer science from Beihang University, Beijing, China, in 1998 and 2002, respectively. He was a Visiting Scholar with the School of Computer Science, Carnegie Mellon University, USA, from 2010 to 2011. He is currently a Professor with the School of Computer Science and Engineering, BUAA. His current research interests include mobile and pervasive computing, and mobile video analysis.

Bin Dai received the B.Eng. & M.S. degree in computer science and engineering from Beihang University, Beijing, China, in 2010 & 2013, respectively. He is pursuing the Ph.D. degree in computer science and engineering at Beihang University, where he works on mobile computing, wireless sensor networks, and industrial IoT.

Xuefeng Liu received the MS and PhD degrees from the Beijing Institute of Technology, and the University of Bristol, United Kingdom, in 2003 and 2008, respectively. He was an associate professor at the School of Electronics and Information Engineering in the HuaZhong University of Science and Technology, China from 2008 to 2018. He is currently an associate professor at the School of Computer Science and Engineering, Beihang University, China. His research interests include wireless sensor networks, distributed computing and in-network processing.

Zheyuan Hu received the B.S. degree in computer science and engineering from Northeastern University, Shenyang, China, in 2017, where he is currently pursuing the M.S. degree with the School of Computer Science and Engineering, Beihang University, Beijing, China. His research interests include mobile edge computing and robot operating system.

Mingliang Xu received the Ph.D. degree from the State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China. He is currently a Professor with the School of Information Engineering, Zhengzhou University, Zhengzhou, China. He has authored more than 60 journal articles and conference papers in the areas, including the ACM Transaction on Graphics (TOG), the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI), IEEE TRANSACTIONS ON IMAGE PROCESSING (TIP), IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (TCSVT), the ACM Special Interest Group on Computer Graphics (ACM SIGGRAPH) Asia/ACM Multimedia Conference (MM), and the IEEE International Conference on Computer Vision (ICCV). His current research interests include computer graphics, multimedia, and artificial intelligence.

Mohsen Guizani (Fellow, IEEE) received the B.S. (Hons.) and M.S. degrees in electrical engineering and the M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively. He served in different academic and administrative positions at the University of Idaho, Western Michigan University, the University of West Florida, the University of Missouri–Kansas City, the University of Colorado at Boulder, and Syracuse University. He is currently a Professor with the CSE Department, Qatar University, Qatar. His research interests include wireless communications and mobile computing, computer networks, mobile cloud computing, security, and smart grid. He has authored 9 books and more than 500 publications in refereed journals and conferences. He is a Senior Member of ACM. He also served as a member, chair, and general chair of a number of international conferences. Throughout his career, he received three teaching awards and four research awards. He also received the 2017 IEEE Communications Society WTC Recognition Award and the 2018 AdHoc Technical Committee Recognition Award for his contribution to outstanding research in wireless communications and Ad-Hoc Sensor networks. He was the Chair of the IEEE Communications Society Wireless Technical Committee and the Chair of the TAOS Technical Committee. He served as the IEEE Computer Society Distinguished Speaker. He is currently the IEEE ComSoc Distinguished Lecturer. He guest edited a number of special issues in IEEE journals and magazines. He is currently the Editor-in-Chief of the IEEE Network Magazine, serves on the editorial boards of several international technical journals, and the Founder and Editor-in-Chief of Wireless Communications and Mobile Computing (Wiley).