Towards Energy-Efficient Scheduling of UAV and Base Station Hybrid Enabled Mobile Edge Computing

Bin Dai, *Member, IEEE*, Jianwei Niu, *Senior Member, IEEE*, Tao Ren, *Member, IEEE*, Zheyuan Hu, *Graduate Student Member, IEEE*, and Mohammed Atiquzzaman, *Senior Member, IEEE*

***Abstract*—Mobile edge computing (MEC) has been considered as a promising paradigm to support the growing popularity of mobile devices (MDs) with similar capabilities as cloud computing. Most existing research focuses on MEC enabled by terrestrial base stations (BSs), which is unable to work in certain scenarios, e.g., disaster rescue and field operation. Some researchers have been making efforts on studying MEC assisted by unmanned-aerial-vehicles (UAVs) and developed lots of efficient scheduling algorithms. However, MEC assisted only by UAVs has limited capability and is unsuitable for heavy-computation applications. To address the issue, this paper proposes a novel UAV-and-BS hybrid enabled MEC system, where multiple UAVs and one BS are deployed to facilitate the provisioning of MEC services either directly from UAVs or indirectly from the BS. Considering maximizing the lifetime of all MDs, the energy-efficient scheduling problem is formulated as minimizing the energy consumption of all MDs by jointly optimizing UAV trajectories, task associations, computing-and-transmitting resource allocations. The optimization problem is further decomposed into three sub-problems and solved by the proposed hybrid heuristic and learning based scheduling algorithms to reduce the complexity. Experimental results show that the proposed algorithm can achieve promising performance improvements over baseline algorithms, including local-computing, random-offloading and greedy-offloading.**

***Index Terms*—Unmanned aerial vehicle, mobile edge computing, computation offloading, task association, deep reinforcement learning.**

Manuscript received July 31, 2021; revised October 14, 2021; accepted November 14, 2021. Date of publication November 19, 2021; date of current version January 20, 2022. The review of this article was coordinated by Prof. Ying-Dar Lin. *(Corresponding author: Tao Ren.)*

Bin Dai is with the State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing 100191, China, and also with Hangzhou Innovation Institute, Beihang University, Hangzhou 310051, China (e-mail: daibin@buaa.edu.cn).

Jianwei Niu is with the State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing 100191, China, with Hangzhou Innovation Institute, Beihang University, Hangzhou 310051, China, and also with the Zhengzhou University Research Institute of Industrial Technology, School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China (e-mail: niujianwei@buaa.edu.cn).

Tao Ren is with the Hangzhou Innovation Institute, Beihang University, Hangzhou 310051, China, and also with 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).

Zheyuan Hu is with the School of Computer Science and Engineering, Beihang University, Beijing 100191, China (e-mail: huzheyuan18@buaa.edu.cn).

Mohammed Atiquzzaman is with the School of Computer Science, University of Oklahoma, Norman 73019, USA (e-mail: atiq@ou.edu).

Digital Object Identifier 10.1109/TVT.2021.3129214

I. Introduction

RECENT years have witnessed the rapid development of smart mobile devices (MDs) and the great popularity of mobile applications [1], [2], e.g., online gaming, face recognition, smart cities. These applications are typically computation-intensive and delay-sensitive, while MDs are in general equipped with limited computing resources and battery energy. Mobile edge computing (MEC) has been proposed as a promising paradigm to ease the tension between mobile applications and devices, by deploying powerful resources at the edge, e.g., cellular base stations (BSs), WiFi access points (APs), of mobile networks [3]–[5]. Mobile applications can choose to offload computation-intensive tasks to and receive computation results from BSs, by which MEC provides approximate computing capabilities as cloud computing while maintaining much lower task latency and energy consumption [6]–[8]. However, BSs are typically terrestrial network infrastructures with fixed locations, which limits the application of MEC in certain scenarios, such as disaster rescue, field operation, etc.

Due to the high maneuverability and flexibility, unmanned aerial vehicles (UAVs) have attracted a great deal of attention from wireless communication researchers [9], [10]. Zeng *et al.* [11] considered mounting wireless communication nodes on a UAV, and maximized the system throughput by optimizing both UAV's trajectory and MDs' and UAV's transmitting power. Wu *et al.* [12] employed multiple UAVs to serve as aerial BSs for ground MDs and achieved the maximization of the minimum MD throughput by jointly optimizing MDs' communication association and UAVs' trajectory and power. More studies about UAV enabled wireless communication could be found in [13]–[15].

More recently, some researchers have been investigating the adoption of UAVs for MEC by deploying edge resources on UAVs [16]–[20]. Hu *et al.* [16] studied UAV-assisted MEC systems where one UAV endowed with computing resources serves multiple MDs, and achieved the minimization of all MDs' maximum latency by optimizing UAV's trajectory, task offloading ratios, and task associations. Guo *et al.* [18] also considered MEC assisted by a single UAV, but aimed at minimizing the energy consumption via the joint optimization of task association, offload ratio, and UAV trajectory. Different from [16], [18], Hou *et al.* [20] considered a multi-UAV enabled

MEC network, and guaranteed both latency and reliability by jointly optimizing computing, caching, and communications. Overall, UAV-enabled MEC can provide more desirable services to MDs due to the following two reasons: first is the better line-of-sight links from high-altitude UAVs to ground MDs, and secondly the shorter communication distance resulting from the UAVs' flexibility.

However, UAV enabled MEC could suffer from the limited deployment of UAVs and the insufficient computing resources to serve MDs. A more promising paradigm is to enable MEC with both UAVs and BSs, the research of which is still scarce. In [21], a UAV is deployed in the areas where MDs can not be served directly by BSs, and MDs are provisioned with MEC services from BSs via the relay of UAVs. This efficiently extends the service-capability of MEC enabled only by UAVs, whereas the two-hop computation offloading could lead to troubles for some delay-sensitive tasks. In [22], a UAV endowed with computing resources collaborates with BSs to provide MEC services for MDs, so that MDs' tasks can either be offloaded to the UAV for immediate execution or further transmitted to BSs for intensive computation. Nevertheless, only one single UAV is considered to play the role as the MEC server or task-offloading relay, which could face challenges in situations where multiple MDs are eager for MEC services simultaneously.

In [23], a binary computation-offloading scheduling approach is proposed for airground integrated networks, where the MDs could choose to offload their tasks either to UAVs or to BSs under the goal of minimizing MDs' energy consumption. In [24], a similar optimization problem is addressed to save the energy consumption of all MDs and UAVs in the MEC, where the MD could not offload its task directly to the BS but to one of the UAVs (where part of the task is further offloaded to the BS). However, the UAV and BS hybrid assisted MEC where the MD could not directly offload indivisible tasks to the BS, has not been thorough investigated, which could play a significant role in the MEC scenario: MDs are deployed to work in the disaster area where inner BSs are all destroyed, and the MDs' tasks (e.g., image classification, natural language process) are indivisible and could only be offloaded to UAVs or further to the outer BS for edge computing. To address the issue, this paper considers computation offloading and resource allocation in the MEC enabled by multiple UAVs and one BS (named mUB-MEC), where each MD's task, as a whole, can be either executed in the MD itself, or offloaded to one of the UAVs for straightforward MEC services, or further transmitted to the BS (through a UAV) for more intensive computation. Since energy is one of the most precious resources for MDs, an energy-efficient scheduling approach is proposed to minimize the energy consumption of all MDs while satisfying the task latency requirement, by jointly optimizing UAVs' trajectories, task associations, MDs' CPU frequencies, and wireless transmitting powers. Due to the high complexity of the optimization problem, it is first decomposed into three sub-problems: UAV trajectory planning, task association scheduling, and computing and transmitting resource allocation. Then, a hybrid heuristic and learning based scheduling (H2LS) algorithm is proposed to solve the three sub-problems (UAV trajectory planning based on long short-term memory and fuzzy c-means, task-association scheduling based on deep deterministic policy gradient, resource allocation based on convex optimization). At last, extensive experiments are conducted via numerical simulation to evaluate the performance of H2LS. Experimental results verify the effectiveness of the proposed approach in mUB-MEC against baseline approaches, including local computing, random offloading, greedy offloading, etc.

The main contributions of this paper are summarized as follows.

- We propose a novel UAV and BS hybrid enabled MEC system where multiple UAVs are deployed to facilitate the provisioning of either straightforward MEC services to MDs or further MEC services from the BS by relaying MDs' tasks.
- Considering maximizing the lifetime of all MDs, we formulate the energy-efficient scheduling problem as minimizing the energy consumption of all MDs of the proposed MEC system, by jointly optimizing UAV trajectories, task associations, MDs' CPU frequencies, and wireless transmitting powers.
- In view of the high complexity of the formulated optimization problem, we decompose it into three sub-problems and propose a hybrid heuristic and learning based scheduling algorithm to solve them with much lower complexity.
- We conduct extensive experiments via numerical simulation to evaluate the proposed hybrid scheduling algorithm, and the experimental results indicate the substantial performance improvements of the proposed scheduling algorithm over baselines in the UAV and BS hybrid enabled MEC.

In addition, it's worth noting that although the authors of the study [22] claim that their research is the first to consider UAV and BS hybrid assisted MEC, there remains a lot of important factors, e.g., the number of UAVs, the binary offloading mode, the MD's transmitting power, that are not taken into consideration and need further investigations. Hence, this paper takes these factors into account and attempts to obtain more insights about the UAV and BS hybrid assisted MEC. Moreover, compared to the UAV trajectory design approach in [25], this paper attempts to achieve dynamic trajectory planning via a hybrid learning and heuristic based method; compared to the data offloading approach in [26] that achieves promising MEC performance by integrating the latest wireless advances (including backscatter, orthogonal frequency division multiple access and wireless power transfer, etc.), this paper also makes efforts to minimize the energy consumption of MEC by adopting dynamic voltage and frequency scaling [27] (i.e., takes scheduling CPU frequency into consideration).

The remainder of the paper is organized as follows. In the next section, the system model is described and the optimization problem is formulated. In Section III, we decompose the optimization problem into three sub-problems and solve them by the proposed H2LS algorithm. Section IV discusses the performance evaluation of H2LS via numerical simulation in terms of efficiency, robustness, scalability, and reliability, followed by the discussion of conclusions and future work in Section V.

TABLE I
KEY NOTATIONS THROUGHOUT THE PAPER

Notation	Description
N	The number of time slots
$\mathcal{N}$	The set of time slots
n	The n-th time slot
M	The number of MDs
$\mathcal{M}$	The set of MDs
m	The MD m
U	The number of UAVs
$\mathcal{U}$	The set of UAVs
u	The UAV u
x, y	The horizontal coordinates
$\boldsymbol{\ell}^{\text{BS}}$	The location of the base station
$\boldsymbol{\ell}_n^m$	The location of MD m at time slot n
$\boldsymbol{\ell}_n^u$	The location of UAV u at time slot n
$\boldsymbol{j}_n^m$	The task arrived at MD m at time slot n
a_n^m	The offloading indicator of $\boldsymbol{j}_n^m$
f_n^m	The actual CPU frequency of MD m at time slot n
$f_n^{u,m}$	The allocated CPU frequency of UAV u to MD m at n
p_n^m	The wireless transmitting power of MD m at n
p_n^u	The wireless transmitting power of UAV u at n
$h_n^{m,u}$	The wireless channel gain from MD m to UAV u at n
$h_n^{u,U+1}$	The wireless channel gain from UAV m to the BS at n
$e_n^{u,\text{fly}}$	The consumed energy of UAV u for its flying during n
$t_n^{m,\text{local}}$	The local computing time of $\boldsymbol{j}_n^m$
$e_n^{m,\text{local}}$	The local energy consumption of $\boldsymbol{j}_n^m$
$t_n^{m,\text{UAV}}$	The computing time of $\boldsymbol{j}_n^m$ at the UAV
$e_n^{m,\text{UAV}}$	The energy consumption of the UAV for $\boldsymbol{j}_n^m$
$t_n^{m,u}$	The cost time to transmit $\boldsymbol{j}_n^m$ from MD m to UAV u
$t_n^{u,U+1}$	The cost time to transmit $\boldsymbol{j}_n^m$ from UAV u to the BS
$e_n^{m,u}$	The consumed energy of MD m to transmit $\boldsymbol{j}_n^m$ to UAV u
$e_n^{u,U+1}$	The consumed energy of UAV u to transmit $\boldsymbol{j}_n^m$ to the BS

II. SYSTEM MODEL AND PROBLEM FORMULATION

This section first describes the network model of mUB-MEC assisted by multiple UAVs and one BS, and then formulates the optimization problem for the system.

A. Notations

For ease of reference, key notations used throughout the paper are listed in Table I, where scalars, vectors and matrices are denoted by italic lowercase letters, boldface italic lowercase letters and boldface italic uppercase letters, respectively. In addition, constants (e.g., the number of UAVs and the number of MDs) are denoted by italic uppercase letters, and sets are denoted by handwritten uppercase letters.

B. System Model

From the perspective of network architecture, the mUB-MEC consists of three layers: the ground MDs, flying UAVs, and remote BS, as shown in Fig. 1. The proposed scheduling approach H2LS works in a centralized way and could be located in the BS or a separate infrastructure with desirable computing and communication conditions. H2LS collects required information (e.g., task size, MD's energy, MD's location, UAV's location and UAV's resource, whose size is typically much smaller than that of the offloaded task) through a separate wireless/wired channel, make scheduling decisions (e.g., the offloading indicator, MD's transmitting power, allocated computing resource) based on the proposed algorithm/model, and send the scheduling decision to the corresponding actors (e.g., the BS, UAVs, MDs) through the separate wireless/wired channel.

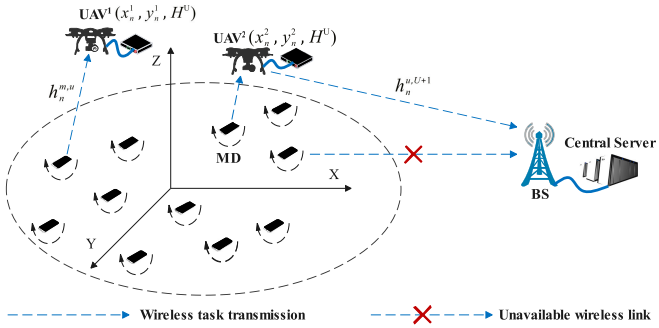

Fig. 1. Schematic of the mUB-MEC network assisted by multiple UAVs and one BS.

The coordinates of MDs, UAVs, and BS are measured by a three-dimensional Cartesian coordinate system. The task execution time is denoted as T, which is divided equally into N time-slots. The set of time-slots is denoted as

$$\mathcal{N} \triangleq \left\{\text{TS}^1, \text{TS}^2, \ldots, \text{TS}^N\right\},$$

where the length τ of each TS satisfies $\tau = T/N$ and is assumed to be small enough to consider the location of each UAV approximately unchanged in each TS. It is assumed that MDs could not communicate with the BS directly due to severe blockage, hence MDs could only offload tasks to the BS with the help of UAVs.

1) Ground MDs: In the mUB-MEC, the set of MDs is denoted as

$$\mathcal{M} \triangleq \left\{\text{MD}^1, \text{MD}^2, \ldots, \text{MD}^M\right\},$$

where M is the number of MDs. The location of MD^m at TS^n is denoted as

$$\boldsymbol{\ell}_n^m \triangleq \{x_n^m, y_n^m, 0\},$$

where x_n^m and y_n^m stand for the horizontal coordinates of MD^m, $m \in \mathcal{M}$, $n \in \mathcal{N}$ (for convenience, $\mathcal{M}$ and $\mathcal{N}$ also denote the sets $\{1, 2, \ldots, M\}$ and $\{1, 2, \ldots, N\}$, respectively).

In TS^n, each MD^m has to perform a task, which can be denoted as

$$\boldsymbol{j}_n^m \triangleq \{I_n^m, C_n^m, T^{\text{req}}\}, \tag{1}$$

where I_n^m represents the input data size (measured in bits) of $\boldsymbol{j}_n^m$, C_n^m represents the required number of CPU cycles to compute one bit of I_n^m, and T^{req} stands for the maximum allowable execution time of $\boldsymbol{j}_n^m$. Without loss of generality, all $\boldsymbol{j}$s are assumed to have the same allowable time requirement T^{req}. Furthermore, T^{req} is selected as a value less than τ which assumes that each task should be accomplished within one time-slot.

Each MD^m is embedded with an onboard CPU, whose maximal computational capacity is denoted as F^{MD}. By dynamically adjusting voltage and frequency, the actual CPU frequency f_n^m of MD^m in TS^n is controlled adaptively to improve energy

efficiency. Thus, f_n^m should satisfy

$$f_n^m \le F^{\text{MD}}, \qquad m \in \mathcal{M},\ n \in \mathcal{N} \tag{2}$$

where all MDs are assumed to have the same maximal computational capacity F^{MD}.

In addition, each MD is assumed to move on the ground within its near ranges at each time-slot, and the moving distances at x and y coordinates follow the Gaussian distributions with parameters $(0, \sigma_x^2)$ and $(0, \sigma_y^2)$, respectively.

2) Flying UAVs: In the mUB-MEC, the set of UAVs is denoted as

$$\mathcal{U} \triangleq \left\{\text{UAV}^1, \text{UAV}^2, \ldots, \text{UAV}^U\right\},$$

where U is the number of UAVs (for convenience, $\mathcal{U}$ also denotes the set $\{1, 2, \ldots, U\}$). The location of UAV^u at TS^n is denoted as

$$\boldsymbol{\ell}_n^u \triangleq \{x_n^u, y_n^u, H\},$$

where x_n^u and y_n^u stand for the horizontal coordinates of UAV^u, $u \in \mathcal{U}$, $n \in \mathcal{N}$, and H is the fixed altitude.

Each UAV is assumed to fly under the maximum speed $V_{\max}$, which indicates that

$$v_n^u = \frac{\|\boldsymbol{\ell}_n^u - \boldsymbol{\ell}_{n-1}^u\|}{\tau} \le V_{\max}, \quad u \in \mathcal{U},\ n \in \mathcal{N}, \tag{C1}$$

where v_n^u denotes the speed of UAV^u at TS^n. In addition, the distance between any two UAVs should be larger than the minimum allowable distance $d_{\min}$, i.e.,

$$\|\boldsymbol{\ell}_n^u - \boldsymbol{\ell}_n^{u'}\| \ge d_{\min},\ u \ne u,' \qquad u \in \mathcal{U},\ n \in \mathcal{N}. \tag{C2}$$

The UAV's energy consumption consists of three components: the computation-related energy, communication-related energy and propulsion-related energy. The computation-related energy is cost to execute offloaded tasks in the UAV and the communication-related energy is cost to transmit tasks to the BS, which will be elaborated in Equations (10) and (20), respectively. According to prior works [25], [28], the propulsion-related energy consumption of UAV u during its flying at TS^t is given by

$$e_n^{u,\text{fly}} = P_0(1 + \alpha_1 v_u^2(n)) + P_1\sqrt{\sqrt{1 + \alpha_2^2 v_u^4(n)} - \alpha_2 v_u^2(n)} + \alpha_3 v_u^3(n), \tag{3}$$

where $P_0 = \frac{\delta}{8}\rho s A \Omega^3 R^3$, $P_1 = (1+I)\frac{W^{3/2}}{\sqrt{2\rho A}}$, $\alpha_1 = \frac{3}{\Omega^2 R^2}$, $\alpha_2 = \frac{1}{2V_R^2}$ and $\alpha_3 = 0.5 a_0 \rho s A$. The three components of Eq. (3) correspond to the energy consumption of the UAV to overcome the blades' profile drag, fuselage drag and induced drag, respectively. Other parameters are the same as those in [28].

Each UAV is deployed with an edge server, whose maximal computational capacity is denoted as F^{UAV}. The computation resource in each UAV is allocated to the MDs whose tasks are offloaded and executed in the UAV, and the allocated CPU frequency of UAV^u to MD^m in TS^n is denoted as $f_n^{u,m}$ and satisfies

$$\sum_{m=1}^{M} f_n^{u,m} \le F^{\text{UAV}}, \qquad u \in \mathcal{U},\ n \in \mathcal{N}, \tag{4}$$

where all UAVs are assumed to have the same maximal computational capacity F^{UAV}.

3) Remote BS: The location of BS is denoted as

$$\boldsymbol{\ell}^{\text{BS}} \triangleq \left(x^{\text{BS}}, y^{\text{BS}}, H^{\text{BS}}\right),$$

where x^{BS} and y^{BS} are the x coordinate and y coordinate, respectively. The BS could not be linked directly by MDs, whereas it could be linked by UAVs via line-of-sight communication links due to their high altitude. In this case, the UAVs act as relays by forwarding the tasks offloaded from MDs to the BS for further computation.

Similar to prior research [22], [24], [29], we assume that the BS is endowed with powerful computing servers and energy supply, hence the execution time is negligible and the energy consumption is not considered for all tasks performed on the BS. Under the assumption of neglected execution time in the BS, MDs are more preferable to offload their tasks to the BS to obtain low task latency and energy consumption. However, in the case of BS execution, the task delay (consisting of the transmission time from the MD to the UAV and from the UAV to the BS) could be significantly high or even higher than the maximum tolerable delay T^{req} when the wireless channel condition between the MDs and UAVs or between the UAVs and the BS is poor. In addition, in the case of BS execution, there should exist one available UAV that plays the role of relay. In general, the number of MDs are much higher than that of UAVs. If all adjacent UAVs have been occupied by other tasks for computation or relay, the task could not be offloaded.

4) Task Association: All tasks are assumed to be offloaded with binary policies, which means that each task $\boldsymbol{j}_n^m$ could only be executed in one of the following three types of platforms: MD^m where it arrives, UAV^u which it is offloaded to by MD^m, and BS where it is further offloaded by one of the UAVs. Therefore, a new variable $a_n^m(k)$ is defined to indicate the offloading target of $\boldsymbol{j}_n^m$ as follows,

$$a_n^m(k) = \{0, 1\} \qquad m \in \mathcal{M},\ n \in \mathcal{N},\ k \in \mathcal{K}, \tag{C3}$$

where $\mathcal{K} = \{0, 1, 2, \ldots, U, U+1\}$. $a_n^m(k) = 1$ indicates that $\boldsymbol{j}_n^m$ is associated with the platform $k \in \mathcal{K}$. $\mathcal{K}$ is the set that denotes the $U+2$ possible subjects of task association. Since the task is assumed to be indivisible, there could be three types of locations for executing each task: the MD (i.e., $\{0\} \in \mathcal{K}$) where it arrives, one of the UAVs (i.e., $\{1, 2, \ldots, U\} \in \mathcal{K}$) via direct task offloading, or the BS (i.e., $\{U+1\} \in \mathcal{K}$) via indirect offloading through a UAV. It is worth noting that, in the case of the BS execution via indirect offloading through a UAV, both one element of $\{1, 2, \ldots, U\}$ and $\{U+1\}$ should be designated for task association. An example of possible values of $a_n^m(k), k \in \mathcal{K}$ is shown in Table II.

As shown in Table II, with respect to different task execution destinations, the task association variable $a_n^m(k)$ could be designated with the following three types of values:

TABLE II
POSSIBLE VALUES OF TASK ASSOCIATION VARIABLES OF MD^m AT DIFFERENT TIME SLOTS

TS^n	$a_n^m(k)$						Platform
	$k=0$	$k=1$	$k=2$	.....	$k=U$	$k=U+1$	
1	0	0	1	0	0	0	UAV^2
2	1	0	0	0	0	0	MD^m
3	0	1	0	0	0	1	BS
4	0	0	0	0	1	1	BS
.....							
N	0	0	1	0	0	1	BS

- MD execution (e.g., Line 2 of Table II): The task $\boldsymbol{j}_n^m$ is executed locally in the MD where it arrives, without any offloading. Hence, $a_n^m(0)=1$ and $a_n^m(k)=0, \forall k \in \{1,2,\ldots,U+1\}$. The MD^m will allocate computing resource for task execution.
- UAV execution (e.g., Line 1 of Table II): The task $\boldsymbol{j}_n^m$ will be offloaded to a UAV (supposed to be the UAV^2 as shown in Line 2 of Table II) for execution, without any further offloading to the BS. Hence, $a_n^m(2)=1$ and $a_n^m(k)=0, \forall k \in \{0,1,3,\ldots,U+1\}$. The MD^m will allocate transmitting resource for task offloading and the UAV^2 will allocate computing resource for task execution.
- BS execution (e.g., Line 3, 4 and N of Table II): The task $\boldsymbol{j}_n^m$ will be first offloaded to a UAV (supposed to be the UAV^1 as shown in Line 2 of Table II) for relay and further offloaded to the BS for execution. Hence, $a_n^m(1)=1$, $a_n^m(U+1)=1$ and $a_n^m(k)=0, \forall k \in \{0,2,3,\ldots,U\}$. The MD^m and UAV^1 will allocate transmitting resource for task offloading and the BS will allocate computing resource for task execution.

Therefore, the following constraints should be satisfied:

$$\begin{cases} \sum_{k=0}^{U} a_n^m(k)=1 & \text{if} \quad a_n^m(U+1)=0 \\ \sum_{k=1}^{U} a_n^m(k)=1 & \text{if} \quad a_n^m(U+1)=1 \\ \sum_{k=0}^{U+1} a_n^m(k) \le 2 & \end{cases}. \tag{C4}$$

Besides, each UAV is assumed to be able to offload at most one task to BS for further execution at each time-slot, hence $a_n^m(k)$ should also satisfy:

$$\begin{cases} \sum_{m=1}^{M} a_n^m(U+1) \le U \\ \sum_{k=1}^{U} a_n^m(k) a_n^{m'}(k)=0 \quad \text{if} \quad a_n^m(U+1)=a_n^{m'}(U+1)=1 \end{cases}, \tag{C5}$$

for $\forall m, m' \in \mathcal{M}$ and $\forall n \in \mathcal{N}$.

In addition, due to the introduction of the task association variable $a_n^m(k)$, Equations (2) and (4) should be adjusted as

$$a_n^m(0) f_n^m \le F^{\text{MD}}, \qquad m \in \mathcal{M},\ n \in \mathcal{N} \tag{C6}$$

$$\sum_{m=1}^{M} a_n^m(u) f_n^{u,m} \le F^{\text{UAV}}, \qquad u \in \mathcal{U},\ n \in \mathcal{N}, \tag{C7}$$

respectively.

C. Computation Model

Task computation could be performed in the MDs, UAVs, or BS, which are named as Local Computing, UAV Computing, and BS Computing, respectively.

1) Local Computing: If $\boldsymbol{j}_n^m$ is executed locally (i.e., $a_n^m(0)=1$), according to the state-of-the-art research on MEC [18], [30], [31], the required computation time of $\boldsymbol{j}_n^m$ could be calculated as

$$t_n^{m,\text{local}} = a_n^m(0)\frac{I_n^m C_n^m}{f_n^m}, \qquad m \in \mathcal{M},\ n \in \mathcal{N}, \tag{5}$$

and the corresponding energy consumption could be calculated as

$$e_n^{m,\text{local}} = a_n^m(0)\kappa^m (f_n^m)^{\nu_m}\frac{I_n^m C_n^m}{f_n^m},\ m \in \mathcal{M}, n \in \mathcal{N}, \tag{6}$$

where I_n^m and C_n^m represent the task data size (measured in bits) and required number of CPU cycles to compute one bit of $\boldsymbol{j}_n^m$, respectively, κ^m and ν^m are positive coefficients depending on the CPU model of MD^m [32].

2) UAV Computing: If $\boldsymbol{j}_n^m$ is executed on UAV^u (i.e., $a_n^m(u)=1$), then the computation time could be calculated as

$$t_n^{m,\text{UAV}} = \sum_{u=1}^{U} t_n^m(u), \qquad m \in \mathcal{M},\ n \in \mathcal{N}, \tag{7}$$

where

$$t_n^m(u) = a_n^m(u)\frac{I_n^m C_n^m}{f_n^{u,m}}, \qquad m \in \mathcal{M},\ n \in \mathcal{N},\ u \in \mathcal{U}, \tag{8}$$

and the energy consumption could be calculated as

$$e_n^{m,\text{UAV}} = \sum_{u=1}^{U} e_n^m(u),\ m \in \mathcal{M},\ n \in \mathcal{N}, \tag{9}$$

where

$$e_n^m(u) = a_n^m(u)\kappa^u (f_n^{u,m})^{\nu_u}\frac{I_n^m C_n^m}{f_n^{u,m}}, \tag{10}$$

where κ^u and ν^u are positive coefficients depending on the CPU model of UAV^u. It should be noticed that there could only be one item of the summation in Equation (7) or (9) being non-zero, since each task $\boldsymbol{j}_n^m$ could only be offloaded to one UAV.

3) BS Computing: If $\boldsymbol{j}_n^m$ is executed on the BS (i.e., $a_n^m(U+1)=1$), then the task execution time is considered approximately as zero and the task energy consumption is not taken into account, due to the assumption of BS's powerful computing servers and energy supply as stated in section II-B3.

D. Communication Model

There are two types of communication links in the mUB-MEC: the communication link between MDs and UAVs, and the communication link between UAVs and BS. In addition, similar to prior work, UAVs are allocated with orthogonal frequency resources so as to avoid potential communication interference among UAVs [33], and the wireless communication channels between UAVs and MDs/BS are primarily dominated by line-of-sight links owing to their high altitude [34], [35].

The distances between MD^m and UAV^u, and UAV^u and BS at TS^n can be calculated as

$$d_n^{m,u} = \sqrt{(x_n^u - x_n^m)^2 + (y_n^u - y_n^m)^2 + H^2}, \tag{11}$$

$$d_n^{u,U+1} = \sqrt{(x_n^u - x^{\mathrm{BS}})^2 + (y_n^u - y^{\mathrm{BS}})^2 + H^2}, \tag{12}$$

respectively. Therefore, the wireless channel power gains between MD^m and UAV^u, and UAV^u and BS at TS^n can be expressed as

$$h_n^{m,u} = \frac{g_0}{(d_n^{u,m})^2} = \frac{g_0}{(x_n^u - x_n^m)^2 + (y_n^u - y_n^m)^2 + H^2}, \tag{13}$$

$$h_n^{u,U+1} = \frac{g_0}{(d_n^{u,U+1})^2} = \frac{g_0}{(x_n^u - x^{\mathrm{BS}})^2 + (y_n^u - y^{\mathrm{BS}})^2 + H^2}, \tag{14}$$

respectively, where g_0 is the received power gain at the reference distance of one meter.

Hence, if $\boldsymbol{j}_n^m$ is scheduled to be offloaded from MD^m to UAV^u or from UAV^u to BS at TS^n, the offloading transmission rate can be given by

$$r_n^{m,u} = Blog_2\left(1 + \frac{h_n^{m,u} p_n^m}{\sigma^2}\right), \tag{15}$$

$$r_n^{u,U+1} = Blog_2\left(1 + \frac{h_n^{u,U+1} p_n^u}{\sigma^2}\right), \tag{16}$$

respectively, where B is the system bandwidth, p_n^m and p_n^u represent the wireless transmitting powers of MD^m and UAV^u at TS^n for task offloading, and σ^2 stands for the noise power. p_n^m and p_n^u should be constrained by

$$\begin{cases} p_n^m \le P^m \\ p_n^u \le P^u \end{cases}, \tag{C8}$$

respectively, where P^m and P^u are the maximum available transmitting powers of MD^m and UAV^u.

According to Equations (1), (15), and (16), the task offloading times from MD^m to UAV^u and from UAV^u to BS can be given by

$$t_n^{m,u} = \frac{I_n^m}{r_n^{m,u}} = \frac{I_n^m}{Blog_2\left(1 + \frac{h_n^{m,u} p_n^m}{\sigma^2}\right)}, \tag{17}$$

$$t_n^{u,U+1} = \frac{I_n^m}{r_n^{u,U+1}} = \frac{I_n^m}{Blog_2\left(1 + \frac{h_n^{u,U+1} p_n^u}{\sigma^2}\right)}, \tag{18}$$

respectively, and the energy consumption of MD^m for offloading $\boldsymbol{j}_n^m$ to UAV^u and UAV^u for offloading $\boldsymbol{j}_n^m$ to BS can be given by

$$e_n^{m,u} = t_n^{m,u} p_n^m = \frac{I_n^m p_n^m}{Blog_2\left(1 + \frac{h_n^{m,u} p_n^m}{\sigma^2}\right)}, \tag{19}$$

$$e_n^{u,U+1} = t_n^{u,U+1} p_n^u = \frac{I_n^m p_n^u}{Blog_2\left(1 + \frac{h_n^{u,U+1} p_n^u}{\sigma^2}\right)}, \tag{20}$$

respectively.

E. Problem Formulation

In the considered mUB-MEC, this paper attempts to minimize the total consumption of both MDs' and UAVs' energy, under task-delay constraints and system constraints (e.g., UAVs' maximum speed, UAVs' minimum distance, and maximal computational capacity).

1) Task-Delay Constraint: When the task $\boldsymbol{j}_n^m$ is executed in the MD (where it arrived), the UAV (where it is directly offloaded), or the BS (where it is finally offloaded), its delay can be given by

$$\begin{cases} t^{\mathrm{MD}} = t_n^{m,\mathrm{local}} \\ t^{\mathrm{UAV}} = t_n^{m,u} + t_n^{m,\mathrm{UAV}} \\ t^{\mathrm{BS}} = t_n^{m,u} + t_n^{u,U+1} \end{cases}, \tag{21}$$

respectively. Taking the task-association variable $\alpha_n^m(k)$ into account, the task delay of $\boldsymbol{j}_n^m$ can be uniformly expressed as

$$\begin{aligned} \mathcal{T}_n^m = \; & a_n^m(0)\frac{I_n^m C_n^m}{f_n^m} + \\ & (1 - a_n^m(U+1)) \sum_{u=1}^{U} a_n^m(u) \left(\frac{I_n^m}{r_n^{m,u}} + \frac{I_n^m C_n^m}{f_n^{u,m}}\right) \\ & + a_n^m(U+1) \sum_{u=1}^{U} a_n^m(u) \left(\frac{I_n^m}{r_n^{m,u}} + \frac{I_n^m}{r_n^{u,U+1}}\right), \end{aligned} \tag{22}$$

for $\forall m \in \mathcal{M}$ and $\forall n \in \mathcal{N}$. Therefore, the constraint w.r.t. task delay could be given by

$$\mathcal{T}_n^m \le T^{\mathrm{req}} \qquad \forall m \in \mathcal{M},\ \forall n \in \mathcal{N}. \tag{C9}$$

2) Energy-Consumption Constraints: Letting e^{MD} and e^{UAV} represent the energy budget of each MD and UAV, respectively, the following energy-consumption constraints should be satisfied:

$$\sum_{n=1}^{N} \left(e_n^{m,Local} + \sum_{u=1}^{U} a_n^m(u) e_n^{m,u}\right) \le e^{\mathrm{MD}}, \tag{C10}$$

$$\sum_{n=1}^{N} \left(e_n^{u,\mathrm{fly}} + \sum_{m=1}^{M} \left(e_n^m(u) + a_n^m(U+1) e_n^{u,U+1}\right)\right) \le e^{\mathrm{UAV}}, \tag{C11}$$

for $\forall m \in \mathcal{M}$ and $\forall u \in \mathcal{U}$.

3) Total Energy Consumption of MDs: At TS^n, the energy consumption of MD^m for executing $\boldsymbol{j}_n^m$ includes two possibilities:

- If $\boldsymbol{j}_n^m$ is executed locally by the MD, i.e., $a_n^m(0) = 1$, then the energy consumption of MD^m is $e_n^{m,\mathrm{local}}$ (for local computing).
- If $\boldsymbol{j}_n^m$ is executed remotely by a UAV or the BS, i.e., $a_n^m(0) = 0$, then the energy consumption of MD^m is $e_n^{m,u}$ (for task offloading).

Hence, the energy consumption of MD^m for executing $\boldsymbol{j}_n^m$ could be uniformly represented as

$$e_n^{m,\mathrm{MD}} = a_n^m(0) e_n^{m,\mathrm{local}} + (1 - a_n^m(0)) e_n^{m,u}. \tag{23}$$

The energy consumption of all MDs during task executing time could be given by

$$\mathcal{E} = \sum_{n\in\mathcal{N}} \sum_{m\in\mathcal{M}} e_n^{m,\mathrm{MD}}. \tag{24}$$

4) Problem Formulation: To minimize the total energy consumption of MDs for all tasks during the run-time of mUB-MEC, the problem $\mathcal{P}$ is formulated:

$$\mathcal{P}: \min_{\{\boldsymbol{L},\boldsymbol{A},\boldsymbol{P},\boldsymbol{F}\}} \mathcal{E},$$
$$\text{s.t. } (C1)-(C11), \tag{25}$$

where $\boldsymbol{L}=\{\boldsymbol{\ell}_n^u | u\in\mathcal{U}, n\in\mathcal{N}\}$, $\boldsymbol{A}=\{a_n^m(k) | m\in\mathcal{M}, n\in\mathcal{N}, k\in\mathcal{K}\}$, $\boldsymbol{F}=\{f_n^m, f_n^{u,m} | m\in\mathcal{M}, u\in\mathcal{U}, n\in\mathcal{N}\}$, and $\boldsymbol{P}=\{p_n^m, p_n^u | m\in\mathcal{M}, u\in\mathcal{U}, n\in\mathcal{N}\}$ are the variables to be optimized.

In the problem $\mathcal{P}$, $C1$ and $C2$ represent that the maximum speed of UAVs and minimum distance between UAVs should not be violated. $C3-C5$ guarantee that each task could only be binarily offloaded and executed in the arrived MD, the offloaded UAV, or the BS (relayed by one UAV), and each UAV could only relay at most one task to the BS at each time-slot. $C6$ and $C7$ ensure that the allocated computation resources for local computing and UAV computing at each time-slot should not exceed the maximal computation capacity of MDs and UAVs, respectively. $C8$ indicates the allocated transmitting powers of MDs and UAVs should not exceed the maximal allowable values. $C9$ guarantees the delay requirements of each task. $C10$ and $C11$ denote that the energy budgets of MDs and UAVs should not be exhausted during the run-time. These constraints are highly coupled with each other, which makes $\mathcal{P}$ difficult to solve. For instance, if some task need to be offloaded to the UAVs or BS (under the constraints $C3-C5$) for edge services due to the insufficient computing resource of itself (under the constraints $C6$ and $C7$), the offloaded UAV is hoped to fly to the MD as close as possible to save transmitting energy consumption for not violating the constraint $C11$ and reduce transmission delay for not violating the constraint $C9$, meanwhile the UAV's flying speed should not violate the constraints $C1$ and $C2$ and the allocated transmitting power and computing resource should not exceed the maximal allowable values (corresponding to the constraints $C8$ and $C6$, $C7$).

In general, the problem $\mathcal{P}$ is hard to solve due to the following aspects: (1) $\mathcal{P}$ is a mixed integer non-linear programming (MINLP) problem due to the reason that $\boldsymbol{A}$ is binary while $\boldsymbol{L}$, $\boldsymbol{P}$, and $\boldsymbol{F}$ are continuous. (2) Real-time scheduling decisions should be made at each time-slot. (3) $\mathcal{P}$ needs to be solved in dynamic environment due to the variation of $\boldsymbol{\ell}_n^m$. This paper decomposes $\mathcal{P}$ into three sub-problems, which includes UAV trajectory planning ($\mathcal{P}1$), task-association scheduling ($\mathcal{P}2$), and computing-and-transmitting resource allocation ($\mathcal{P}3$), so that efficient scheduling policies of the mUB-MEC network can be obtained with significantly reduced solving complexity of the optimization problem.

III. The Proposed Scheduling Algorithm

This section first introduces the decomposition of the whole problem $\mathcal{P}$ and the correspondent three sub-problems, and then describes the proposed hybrid heuristic and learning based scheduling algorithm and sub-algorithms in detail.

A. Problem Decomposition

To reduce the computation complexity, $\mathcal{P}$ is decomposed into the following three sub-problems:

(1) UAV trajectory planning: Among the scheduling variables ($\boldsymbol{L}$, $\boldsymbol{A}$, $\boldsymbol{P}$, and $\boldsymbol{F}$), UAV-trajectory $\boldsymbol{L}$ are weakly dependent of other three variables and should be optimized primarily based on the observation of MDs' locations with the object of being close to both MDs and the BS as much as possible. Therefore, UAV trajectory planning is formulated as the sub-problem $\mathcal{P}1$:

$$\mathcal{P}1: \min_{\boldsymbol{L}} \sum_{u=1}^{U}\left(\sum_{m\in\mathcal{M}^u} d_n^{m,u} + d_n^{u,U+1}\right), \quad \forall n\in\mathcal{N}$$
$$\text{s.t. } (C1)-(C2), \tag{26}$$

where $\mathcal{M}^u$ is the possible cluster of MDs that are designated to be served by UAV^u, and satisfies $\sum_{u=1}^{U}\mathcal{M}^u = M$.

(2) Task-association scheduling: Once $\boldsymbol{L}$ at TS^n is determined, $\boldsymbol{A}$ could be optimized before $\boldsymbol{P}$ and $\boldsymbol{F}$. Based on the designated clusters ($\mathcal{M}^u, u\in\mathcal{U}$), $\boldsymbol{A}$ is optimized at the aim of minimizing the maximal $\mathcal{T}_n^m$ of all tasks at TS^n, so that C11 of $\mathcal{P}$ could be more easily satisfied. Therefore, the task-association scheduling sub-problem $\mathcal{P}2$ is formulated as follows,

$$\mathcal{P}2: \min_{\boldsymbol{A}} \max_{\forall u\in\mathcal{U}} \left\{\max_{\forall m\in\mathcal{M}^u} \mathcal{T}_n^m\right\}, \qquad \forall n\in\mathcal{N}$$
$$\text{s.t. } (C3)-(C5). \tag{27}$$

(3) Computing-and-transmitting resource allocation: After solving $\mathcal{P}1$ and $\mathcal{P}2$, the rest variables $\boldsymbol{P}$ and $\boldsymbol{F}$ could be optimized at the aim of minimizing $\mathcal{E}$ under the constraints C6 - C11, which is formulated as the problem $\mathcal{P}3$:

$$\mathcal{P}3: \min_{\{\boldsymbol{P},\boldsymbol{F}\}} \mathcal{E},$$
$$\text{s.t. } (C6)-(C11). \tag{28}$$

B. H2LS Algorithm Overview

According to the above decomposition, the proposed optimization algorithm H2LS is described in this part.

As shown in Fig. 2, H2LS is composed of three components (UAV Trajectory Planning, Task Association Scheduling, and Resource Allocation), which are corresponding to the three sub-problems ($\mathcal{P}1$, $\mathcal{P}2$, and $\mathcal{P}3$). In the beginning of each time-slot, the Environment (mUB-MEC network) generates two state variables ($\boldsymbol{\ell}_n^m$ and $\boldsymbol{j}_n^m$). $\boldsymbol{\ell}_n^m$ is input to UAV Trajectory Planning (UTP), and $\boldsymbol{j}_n^m$ is input to both Task Association Scheduling (TAS) and Resource Allocation (RA).

(1) First, $\boldsymbol{\ell}_n^m$ is processed by UTP. Since the locations of MDs vary during time slots, UTP will attempt to direct UAVs

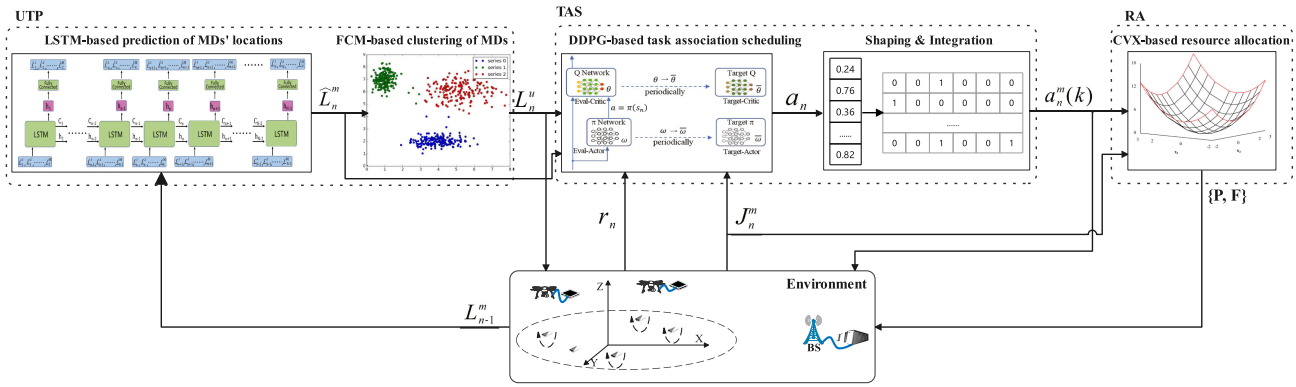

Fig. 2. Schematic of the H2LS algorithm for the mUB-MEC network.

to appropriate locations according to its prediction on MDs' movements. As the movements of MDs may be neither Gaussian distributed nor linear, this paper adopts Long Short-Term Memory (LSTM) [36] to model the distribution of MDs' movements. After the prediction, MDs needs to be divided appropriately into U clusters corresponding to the number of UAVs, so that each UAV could serve a desirable cluster of MDs. To perform soft clustering where each MD could be served by more than one UAV during different time slots (but served by not more than one UAV during each time slot), fuzzy c-means (FCM) [37] is employed in UTP to cluster MDs based on the similarity measure of channel power gains. In addition, we dynamically change the execution frequency of the clustering algorithm according to the moving trend of cluster centroids, so as to achieve a trade-off between the system overhead (due to the frequent clustering) and the system performance. Concretely speaking, the cluster frequency increases if the average sum of the movement distances of all centroids of the current time slot is larger than that of the previous time slot, and decreases otherwise. The clustering frequency is set to 1 initially. After clustering, the centroid of each cluster is output by UTP as the movement actions $\boldsymbol{\ell}_n^m$ of UAVs.

(2) Second, TAS receives $\boldsymbol{\ell}_n^m$ and $\boldsymbol{j}_n^m$ from UTP and Environment, respectively. TAS attempts to generate desirable values of task-association scheduling variables $a_n^m(k)$ based on the time-varying channel conditions and task requirements. The state-of-the-art deep reinforcement learning (DRL) method — Deep Deterministic Policy Gradient (DDPG) [38] is leveraged to output the optimized action $\boldsymbol{a}_n$ according to its interaction experience with the Environment. In each time slot, the output $\boldsymbol{a}_n$ is a one-dimensional vector and composed of $N' = |\mathcal{M}| \times |\mathcal{K}|$ items, each of which is relaxed to a continuous variable valued between 0 and 1. Each item of $\boldsymbol{a}_n$ could be considered as the execution probability (which is the reason for each item being continuous values between 0 and 1) of $\boldsymbol{j}_n^m$ on the platform k. Since the task-association variables should be two-dimensional and binary values, all items of $\boldsymbol{a}_n$ are shaped and integrated to 1 or 0 based on task-association constraints, and taken as the output $a_n^m(k)$ of TAS.

(3) Third, $a_n^m(k)$ and $\boldsymbol{j}_n^m$ are input to RA for final processing. According to equation (28), $\boldsymbol{P}$ and $\boldsymbol{F}$ need to be optimized to minimize $\mathcal{E}$ under the constraints of $C6 - C11$, which could be solved directly via popular convex optimization methods (e.g., CVX [39], CVXPY [40]). After the optimization, $\boldsymbol{P}$ and $\boldsymbol{F}$ are output by RA and sent to the Environment.

To this end, the environment receives the entire action, which is composed of $\boldsymbol{\ell}_n^u$, $a_n^m(k)$, and $\{\boldsymbol{P}, \boldsymbol{F}\}$ sent by UTP, TAS, and RA, respectively. The environment carries out the action and generates a reward r_n (input to DDPG) and a new state (the element of which is sent to the corresponding component of H2LS). Thereafter, the algorithm goes into the new time slot and repeats the above three steps.

C. UAV Trajectory Planning

This subsection introduces the UAV Trajectory Planning component of H2LS in detail, including two parts: the prediction of MDs' movements based on PF and the clustering of MDs based on FCM.

1) LSTM-Based Prediction of MDs' Movements: In the mUB-MEC network, the distance between UAVs and MDs is a primary factor that will impact other scheduling variables, therefore the UAVs are expected to move in advance near to MDs as closely as possible. To this end, H2LS attempts to predict the locations of MDs $\boldsymbol{\ell}_n^m$ in advance to guide the movements of UAVs. Since the prediction of $\boldsymbol{\ell}_n^m$ is primarily based on its previous locations, the state-of-the-art recurrent neural network — LSTM is leveraged to model the temporal distribution of $\boldsymbol{\ell}_n^m$.

As shown in Fig. 3 a, LSTM is one type of recurrent neural network that accepts both external inputs ($\{\boldsymbol{\ell}_n^m | m \in \mathcal{M}\}$) and feedback inputs ($\boldsymbol{C}_{n-1}$ and $\boldsymbol{h}_{n-1}$). The output of LSTM includes two items ($\boldsymbol{C}_n$ and $\boldsymbol{h}_n$), which are input to the LSTM itself for procession at next time slot. Among the two output items, $\boldsymbol{C}_n$ is obtained via the following operation:

$$\boldsymbol{C}_n = f_n * \boldsymbol{C}_{n-1} + i_n * \tilde{\boldsymbol{C}}_n, \tag{29}$$

where

$$\begin{aligned} f_n &= \sigma(\mathbf{W}_f \cdot [\boldsymbol{h}_{n-1}, \boldsymbol{\ell}_n^m] + \boldsymbol{b}_f), \\ i_n &= \sigma(\mathbf{W}_i \cdot [\boldsymbol{h}_{n-1}, \boldsymbol{\ell}_n^m] + \boldsymbol{b}_i), \\ \tilde{\boldsymbol{C}}_n &= \tanh(\mathbf{W}_C \cdot [\boldsymbol{h}_{n-1}, \boldsymbol{\ell}_n^m] + \boldsymbol{b}_C), \end{aligned}$$

σ and tanh are the sigmoid and hyperbolic tangent activation functions, $\mathbf{W}_f$ and $\mathbf{W}_i$ are the weight matrix of the σ function,

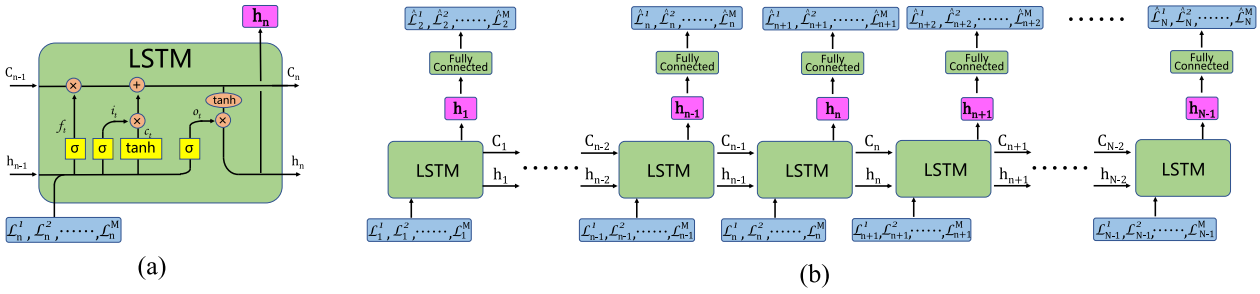

Fig. 3. The prediction model (shown in 3(a)) of MDs' locations based on LSTM (whose architecture is shown in 3(b).

$\mathbf{W}_C$ is the weight matrix of the tanh function, $\boldsymbol{b}_f$ and $\boldsymbol{b}_i$ are the bias vector of the σ function, $\boldsymbol{b}_C$ is the weight vector of the tanh function. $\mathbf{W}_f$, $\mathbf{W}_i$, $\mathbf{W}_C$, $\boldsymbol{b}_f$, $\boldsymbol{b}_i$ and $\boldsymbol{b}_C$ need to be trained via deep learning. Based on $\boldsymbol{C}_n$, $\boldsymbol{h}_n$ is obtained as follows,

$$\boldsymbol{h}_n = \boldsymbol{o}_n * \tanh(\boldsymbol{C}_n), \tag{30}$$

where

$$\boldsymbol{o}_n = \sigma(\mathbf{W}_o \cdot [\boldsymbol{h}_{n-1}, \boldsymbol{\ell}_n^m] + \boldsymbol{b}_o),$$

$\mathbf{W}_o$ and $\boldsymbol{b}_o$ are the weight matrix and weight vector of the σ function, respectively. Similarly, $\mathbf{W}_o$ and $\boldsymbol{b}_o$ need to be trained via deep learning.

Based on Eqs. (29) and (30), we develop the prediction model of MDs' locations, whose unfold schematic over time series is shown in Fig. 3 b. At each time slot, current MDs' locations are input to the LSTM, and the LSTM's output h_n is obtained based on Eq. (30). To predict the MDs' locations at the next time slot, a fully connected layer is additionally added to the output to fine-tune h_n as follows

$$\hat{\boldsymbol{\ell}}_{n+1}^m = \text{relu}\left(\mathbf{W}_{\hat{\ell}} \cdot \boldsymbol{h}_n + \boldsymbol{b}_{\hat{\ell}}\right), \tag{31}$$

where relu is the rectified linear unit activation function, $\mathbf{W}_{\hat{\ell}}$ and $\boldsymbol{b}_{\hat{\ell}}$ are the weight matrix and bias vector of the relu function, respectively (trained via deep learning).

2) FCM-Based Clustering of MDs: Based on the prediction of MDs' next locations $\hat{\boldsymbol{\ell}}_{n+1}^m$, it is favorable to cluster MDs into U groups which could be served by UAVs in a load-balanced manner. To perform the clustering, FCM is adopted to develop the model. From fuzzy theory, each MD^m is assigned a degree $d_{m,u}$ of the membership of each cluster $u \in \mathcal{U}$ at TS^{n+1} as follows,

$$d_{m,u} = \left(\sum_{k=1}^{U}\left(\frac{\|\hat{\boldsymbol{\ell}}_{n+1}^m - \boldsymbol{c}_u\|}{\|\hat{\boldsymbol{\ell}}_{n+1}^m - \boldsymbol{c}_k\|}\right)^2\right)^{-1}, \tag{32}$$

where $\boldsymbol{c}_k$ is denoted as the centroid of the k-th cluster as follows,

$$\boldsymbol{c}_k = \frac{\sum_{i=1}^{M}\left(d_{i,k}^2 \cdot \hat{\boldsymbol{\ell}}_{n+1}^i\right)}{\sum_{i=1}^{M} d_{i,k}^2}, \quad k \in \mathcal{U}. \tag{33}$$

$d_{m,u}$ and $\boldsymbol{c}_k$ are calculated iteratively under the objective of minimizing

$$O = \sum_{m=1}^{M}\sum_{u=1}^{U}\left(d_{m,u}^2 \|\hat{\boldsymbol{\ell}}_{n+1}^m - \boldsymbol{c}_u\|^2\right) \tag{34}$$

until the difference between two degrees of successive calculations is less than a specified threshold ϵ_d for any membership. Before iteration, all $\boldsymbol{c}_u$ should be initialized. Instead of random initialization, each $\boldsymbol{c}_u$ is initialized with the value of $\boldsymbol{\ell}_n^u$, since the MDs could only move within a small scale and their new centroids are likely to be close to the previous centroids (which are planed as the location $\boldsymbol{\ell}_n^u$ of UAVs' movements).

At the end of the iteration, each MD^m is assigned with a degree $d_{m,u}$ representing its membership of the u-th cluster. We further tune $d_{m,u}$ into binary clustering decisions via an exploration strategy, which could alleviate the tendency of falling into local minimum of O. Letting ϵ_c denotes the exploration threshold, MD^m is clustered into the group with the largest degree with probability $1-\epsilon_c$, and clustered into other groups with probability ϵ_c. For the groups with the probability ϵ_c, each group is selected with the probability of its normalized degree.

In summary, the FCM-based clustering of MDs at the n-th time slot is described in detail as shown in Algorithm 1, whose output $\boldsymbol{c}_u$ is taken to guide UAVs' movements $\boldsymbol{\ell}_n^u$.

D. Task-Association Scheduling

This subsection introduces the Task-Association Scheduling component of H2LS in detail, including two parts: the scheduling of task-association variables based on DDPG and the integration of scheduled variables.

1) DDPG-Based Task Association Scheduling: After the planning of the UAVs' movements, H2LS first adopts DDPG to learn relaxed task-association scheduling policies

$$\pi\left(\boldsymbol{j}_n^m, \hat{\boldsymbol{\ell}}_n^m, \boldsymbol{\ell}_n^u, p_{n-1}^m, p_{n-1}^u, f_{n-1}^m, f_{n-1}^{u,m}\right) = a_n^m(k), \tag{35}$$

where $u \in \mathcal{U}$, $m \in \mathcal{M}$, $n \in \mathcal{N}$, $k \in \mathcal{K}$, $a_n^m(k) \in [0,1]$; $\boldsymbol{j}_n^m, \hat{\boldsymbol{\ell}}_n^m, \boldsymbol{\ell}_n^u$ are the task, MD's location, UAV's location at current time slot n while $p_{n-1}^m, p_{n-1}^u, f_{n-1}^m, f_{n-1}^{u,m}$ are the allocated transmitting power and computing resource at previous time slot $n-1$. π can be considered as a mapping function that accepts the mUB-MEC (namely the environment) state

$$\boldsymbol{s}_n \triangleq \left\{\boldsymbol{j}_n^m, \hat{\boldsymbol{\ell}}_n^m, \boldsymbol{\ell}_n^u, p_{n-1}^m, p_{n-1}^u, f_{n-1}^m, f_{n-1}^{u,m} \,|\, m \in \mathcal{M}, u \in \mathcal{U}, n \in \mathcal{N}\right\}$$

as input and generates the continuous task-association variable

$$\boldsymbol{a}_n \triangleq \{a_n^m(k) \,|\, m \in \mathcal{M}, n \in \mathcal{N}, k \in \mathcal{K}\}$$

Algorithm 1: FCM-Based Clustering of MDs at TS^n.

Initialize the centroid $\boldsymbol{c}_u^0$ of each cluster with ℓ_{n-1}^u;
Initialize $d_{m,u}^0$ based on Eq. (32);
for $iter = 1$ to I_{Max} **do**
  For each $u \in \mathcal{U}$, calculate the centroid $\boldsymbol{c}_u^{iter}$ based on Eq. (33);
  For each $m \in \mathcal{M}$ and $u \in \mathcal{U}$, update the degree $d_{m,u}^{iter}$ based on Eq. (32);
  For each $m \in \mathcal{M}$ and $u \in \mathcal{U}$, calculate the difference of successive degrees $\delta_{m,u} = \|d_{m,u}^{iter} - d_{m,u}^{iter-1}\|$;
  $\delta_{\text{Max}} = \max_{m\in\mathcal{M},u\in\mathcal{U}} \delta_{m,u}$;
  if $\delta_{\text{Max}} < \epsilon_d$ **then**
    break;
  end if
end for
for $m = 1$ to M **do**
  $i_{\text{Max}} = \arg\max_u d_{m,u}$, $I_{-\text{Max}} = \mathcal{U} - i_{\text{Max}}$;
  $r = random()$ within [0,1];
  if $r > \epsilon_c$ **then**
    $d_{m,i_{\text{Max}}} = 1$;
    $d_{m,j} = 0, \quad \forall j \in I_{-\text{Max}}$;
  else
    $d_{m,i_{\text{Max}}} = 0$;
    Randomly select i' from $I_{-\text{Max}}$ according to the normalized degrees of the items in $I_{-\text{Max}}$;
    $d_{m,i'} = 1$;
    $d_{m,j} = 0, \quad \forall j \in I_{-\text{Max}}$ & $j \neq i'$;
  end if
end for
$\begin{cases} \text{calculate } \boldsymbol{c}_u \text{ via Eq. (33)} & \text{if } \sum_{i\in\mathcal{M}} d_{i,u}^2 \neq 0 \\ \boldsymbol{c}_u = \ell_{n-1}^u & \text{else} \end{cases}, \forall u \in \mathcal{U}$;

as output. Each item of $\boldsymbol{a}_n$ is a continuous value within [0,1], and the number of the outputs of π is $|\boldsymbol{a}_n| = |\mathcal{M}| \times |\mathcal{K}|$.

Via reinforcement learning, near optimal solutions for π could be obtained by maximizing the total utility (also named as Q value)

$$Q(\boldsymbol{s}_n, \boldsymbol{a}_n) = r_n(\boldsymbol{s}_n, \boldsymbol{a}_n) + \gamma \max_{\boldsymbol{a}'} Q(\boldsymbol{s}_{n+1}, \boldsymbol{a}'), \tag{36}$$

where $\boldsymbol{s}_{n+1}$ is the new state of the environment after taking $\boldsymbol{a}_n$ under $\boldsymbol{s}_n$,

$$r_n(\boldsymbol{s}_n, \boldsymbol{a}_n) = (\mathcal{E}_n)^{-1} \tag{37}$$

is the immediate reward at TS^n, and γ is the discount factor for future rewards. Since the state and action space of the environment could be of high dimension, two neural networks are employed to approximate the eval-actor π (parameterized by ω) and the eval-critic Q (parameterized by θ), respectively, as shown in Fig. 4. Two target networks (target-actor and target-critic parameterized by $\bar{\omega}$ and $\bar{\theta}$, respectively) are adopted to make the learning process more stable and updated periodically by the parameters of the eval-actor (ω) and eval-critic (θ).

At TS^n, the environment transits from $\boldsymbol{s}_n$ to $\boldsymbol{s}_{n+1}$ after accepting $\boldsymbol{a}_n = \pi(\boldsymbol{s}_n)$ from the eval-actor and generates a reward r_n.

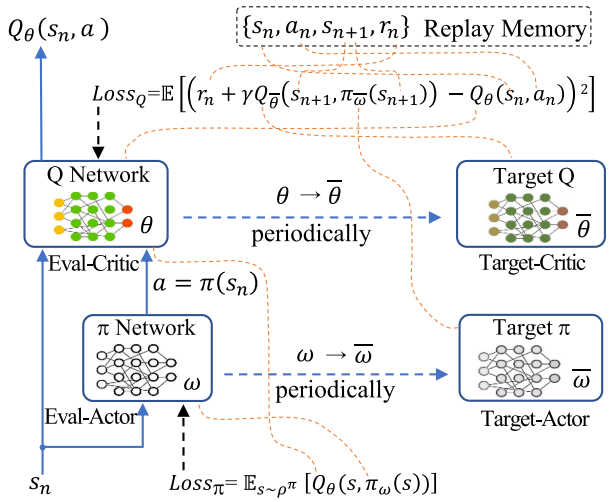

Fig. 4. Schematic of the DDPG-based task-association scheduling.

These four items will be packaged as a tuple $(\boldsymbol{s}_n, \boldsymbol{a}_n, \boldsymbol{s}_{n+1}, r_n)$ and stored in a replay memory. Mini-batches of the stored tuples will be sampled from the replay memory to train the eval-critic network (namely the parameter θ) based on the following loss function:

$$Loss_Q = \mathbb{E}\left[\left(r_n + \gamma Q_{\bar{\theta}}(\boldsymbol{s}_{n+1}, \pi_{\bar{\omega}}(\boldsymbol{s}_{n+1})) - Q_\theta(\boldsymbol{s}_n, \boldsymbol{a}_n)\right)^2\right], \tag{38}$$

where $\boldsymbol{s}_n$, $\boldsymbol{a}_n$, $\boldsymbol{s}_{n+1}$, and r_n are sampled from the replay memory, $Q_{\bar{\theta}}$ and $\pi_{\bar{\omega}}$ are the target-critic and target-actor network, Q_θ is the eval-critic network that needs to be trained using current $Loss_Q$. The eval-actor network is trained directly by sampled actions from the eval-actor network based on the following loss function:

$$Loss_\pi = \mathbb{E}_{s\sim\rho^\pi}\left[Q_\theta\left(s, \pi_\omega(s)\right)\right] \tag{39}$$

Here, it should be noticed that the training of the eval-actor network should be assisted by the eval-critic network, since the output of the eval-actor is directly sent to the eval-critic (as shown in Fig. 4). Therefore, the back-propagation of updating the eval-actor's parameters is performed based on the gradient of $Loss_\pi$ as follows,

$$\begin{aligned} \nabla_\omega Loss_\pi &\approx \mathbb{E}_{\dot{s}\sim\rho^\pi}\left[\nabla_a Q_\theta(s,a)|_{s=\dot{s},a=\pi(\dot{s})}\ \nabla_\omega \pi_\omega(s)|_{s=\dot{s}}\right] \\ &\approx \frac{1}{\dot{N}}\sum_{i=1}^{\dot{N}}\left[\nabla_a Q_\theta(s,a)|_{s=\dot{s}_i,a=\pi(\dot{s}_i)}\ \nabla_\omega \pi_\omega(s)|_{s=\dot{s}_i}\right] \end{aligned} \tag{40}$$

where $\dot{s}$ is a state sampled from the state distribution under current eval-actor's policy π^ω, $\dot{N}$ is the number of samples used for the back-propagation of eval-actor network. The training algorithm of task-association scheduling based on DDPG is detailed in Algorithm 2.

2) Scheduling Variable Shaping and Integration: Since the output $\boldsymbol{a}_n$ of eval-actor network is a one-dimensional vector and each item of $\boldsymbol{a}_n$ is a continuous value within [0,1], $\boldsymbol{a}_n$ should be shaped in two-dimensional manner and integrated into 0 or 1 for further task association.

Algorithm 2: Training Process of DDPG-Based Task Association Scheduling.

Initialize θ and ω with random parameters;
Initialize target networks by $\bar{\theta} \leftarrow \theta$ and $\bar{\omega} \leftarrow \omega$;
for $epi = 1$ to I_{Max} **do**
  Initialize the mUB-MEC for current episode;
  for $n = 1$ to N_{Max} **do**
    Receive tasks $\boldsymbol{j}_n^m$ ($m \in \mathcal{M}$) from mUB-MEC;
    Receive predicted $\hat{\boldsymbol{\ell}}_n^m$ ($m \in \mathcal{M}$) and planned $\boldsymbol{\ell}_n^u$ ($u \in \mathcal{U}$) from UTP;
    Package $\boldsymbol{s}_n$ with $(\boldsymbol{j}_n^m, \hat{\boldsymbol{\ell}}_n^m, \boldsymbol{\ell}_n^u)$;
    $\boldsymbol{a}_n = \pi_\omega(\boldsymbol{s}_n) + \sigma_n$; ($\sigma_n \sim$ random noise)
    Generate $\dot{a}_n$ via variable-integration (next subsection);
    Send $\dot{a}_n$ to RA and mUB-MEC;
    mUB-MEC generates the immediate reward $r(\boldsymbol{s}_n, \boldsymbol{a}_n)$ and next state $\boldsymbol{s}_{n+1}$ after the execution of RA;
    Store $(\boldsymbol{s}_n, \boldsymbol{a}_n, r(\boldsymbol{s}_n, \boldsymbol{a}_n), \boldsymbol{s}_{n+1})$ in Replay Memory;
    Sample N experiences randomly from Replay Memory;
    Update θ and ω based on (38) and (40);
    if $n \% T_{\text{period}} = 0$ **then**
      $\bar{\theta} \leftarrow \theta$, $\bar{\omega} \leftarrow \omega$;
    end if
  end for
end for

Algorithm 3: Shaping and Integration of the One Dimensional Continuous Task-Association Variables at TS^n.

1: Initialize a two-dimensional array $a[m][k]$ ($m \in \mathcal{M}, k \in \mathcal{K}$) with zeros;
2: **for** $m = 1$ to $\mathcal{M}$**do**
3: $a_{\max} = 0$
4: **for** $k = 1$ to $\mathcal{K}$**do**
5: $i = (m-1) \times \mathcal{K} + k$;
6: $a[m][k] = \boldsymbol{a}_n[i]$;
7: **if** $a[m][k] > a_{\max}$ **then**
8: $j = k$, $a_{\max} = a[m][k]$;
9: **end if**
10: **end for**
11: **if** $j = \mathcal{K}$ **then**
12: $j' = \arg\min_k a[m][k],\ k \in \{2, 3, \mathcal{K}-1\}$;
13: $a[m][j] = 1,\ a[m][j'] = 1$;
14: $\forall k \neq j \& j,'\ a[m][k] = 0$;
15: **else**
16: $a[m][j] = 1,\ a[m][k] = 0, \forall k \neq j$;
17: **end if**
18: **end for**

As shown in Algorithm 3, the shaping and integration of $\boldsymbol{a}_n$ are performed via two iterations, whose time complexity is $|\mathcal{M}| \times |\mathcal{K}|$. The outer iteration corresponds to the shaping and integration for the association variables of $\boldsymbol{j}_n^m$ at TS^n, while the inner iteration corresponds to those for each variable of $\boldsymbol{j}_n^m$. First, in the inner iteration, the index for the largest variable (i.e., the platform where $\boldsymbol{j}_n^m$ would be executed with the highest probability) of $\boldsymbol{j}_n^m$ is recorded in j; Then, after the inner iteration, association variables of $\boldsymbol{j}_n^m$ are integrated according to the value of j based on (C4):

- If $j = \mathcal{K}$, which means the BS is the most probable platform for task execution, UAV $^{j'}$ whose association variable is the largest among all UAVs for $\boldsymbol{j}_n^m$ is selected as the transmitting relay for $\boldsymbol{j}_n^m$. Therefore, $a[m][j]$ and $a[m][j']$ are both integrated with 1 and other $a[m][\cdot]$ s are integrated with 0.
- If $j \neq \mathcal{K}, a[m][j]$ is directly integrated with 1 whatever $j = 1$ (corresponding to local execution) or $j \in \{2, 3, \ldots, \| - 1\}$ (corresponding to remote execution on UAVs), and other $a[m][\cdot]$s are integrated with 0.

After the above shaping and integration of task-association variables, the output $a[m][k]$ of Algorithm 3 is sent to RA for the optimization of resource allocation (which can be performed using convex optimization toolkits and thus is omitted here).

E. Scheduling Complexity Analysis of H2LS

After the training of H2LS, the scheduling overhead lies primarily on the three components (FCM-based clustering, shaping and integration, and CVX-based resource allocation), since the other two components (LSTM-based prediction and DDPG-based task association) only need to perform forward inference of learned neural networks within negligible time.

- For the FCM-based clustering component, the scheduling complexity comes from two parts corresponding to the two iterations (Lines 3-11 and Lines 12-24). (1) Within each iteration of the first part, each line of Lines 4-6 can be calculated in parallel within O(1) time, and the calculation of Line 7 could also be implemented with O(1) complexity by remembering and updating the variable δ_{Max} when each parallelly calculated $\delta_{m,u}$ is larger than δ_{Max}. Hence, the scheduling complexity of the first part is O(I_{Max}). (2) Within each iteration of the second part, the calculation of i_{Max} in Line 13 could also be implemented by remembering and updating i_{Max} during the parallel calculation of $d_{m,u}$ in Line 5, hence the scheduling complexity of the second part is $O(M)$. Combining the analysis of the above two parts, the scheduling complexity of the FCM-based clustering component is $O(\max\{I_{\text{Max}}, M\})$.
- For the shaping and integration component, the scheduling complexity mainly comes from the nested loops of Lines 2 and 4, which gives the complexity of $O(MK)$.
- For the CVX-based resource allocation, the problem $\mathcal{P}3$ is solved directly using the CVX toolkit. According to the framework of H2LS, the variables $\boldsymbol{L}$ and $\boldsymbol{A}$ have been optimized and input as constants to the CVX toolkit. Hence, the problem $\mathcal{P}3$ could be transformed into a linear programming problem and optimized using the interior point

TABLE III
NETWORK PARAMETERS OF NUMERICAL SIMULATIONS

Parameters	Values
Radius of the mUB-MEC network	100 m
Location of the BS ($x^{\mathrm{BS}}, y^{\mathrm{BS}}, H^{\mathrm{BS}}$)	$(1000, 0, 10)$
UAVs' flying altitude (H)	100 m
The number of UAVs (U)	8
The number of MDs (M)	64
The length of time slot (τ)	1 s
The number of time slots (N)	100
Maximal allowable execution time (T^{req})	1 s
Maximal speed of UAVs (V^{U})	20 m/s
The weight of UAVs (M_{g})	9.65 kg
Channel power gain (g_0)	-30 dB
Noise power	10^{-9} W
Task size (I_n^m)	$[500, 1000]$ kb
CPU cycles for one bit of task (C_n^m)	10^3
Transmission bandwidth (B)	10 MHz
Local computational capability of MDs (F^{MD})	1 GHz
Edge computational capability of UAVs (F^{UAV})	10 GHz
Coefficients of CPU energy consumption (κ, ν)	10^{-27}, 3
Threshold ϵ_d and ϵ_c	$0.001, 0.01$
Discount factor for future rewards (γ)	0.92
Learning rate of neural networks	0.0008
Mini-batch size of samples	32

method [41], which leads to the complexity of $O(n^3/\log n)$ (where $n = |\boldsymbol{P}| + |\boldsymbol{F}|$).

Given the above analysis, the scheduling complexity of H2LS could be expressed as $O(\max\{I_{\mathrm{Max}}, MK, n^3/\log n\})$.

IV. SIMULATION RESULTS

This section evaluates the performance of H2LS against those of baseline approaches via numerical simulations, which are conducted on a workstation equipped with 2.2 GHz Core i7-10700 K CPU, 16 GB RAM and RTX 2080Ti GPU and installed with Python 3.6. Primary parameters are summarized in Table III, unless specified otherwise. At each TS^n, one task arrives at each MD^m with the task size I_n^m following the uniform distribution between 500 kb and 1000 kb and the workload per bit C_n^m equaling 10^3 cycles. Hence, each task could be guaranteed to be locally completed within one time slot, since the local computation capability of each MD is 1 GHz. In addition, the maximal allowable execution time T^{req} is set to the value 1 s, similar to the definition of the maximum latency of the object detection task considered in the work [42], which investigates the QoS-guaranteed optimization problem in MEC environments and proposes a novel orchestration scheme to reduce the number of service level agreement violations.

At each time slot n, the random movement (composed of Δx_n^m and Δy_n^m) decision of each MD m is generated following a Gaussian distribution with mean 0 and variances σ_x^2, σ_y^2, which means that the probability for the movement decision to take the values x and y is

$$p(\Delta x_n^m = x) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{x^2}{2\sigma_x^2}},$$

$$p(\Delta y_n^m = y) = \frac{1}{\sqrt{2\pi}\sigma_y} e^{-\frac{y^2}{2\sigma_y^2}}.$$

Based on the generated Δx_n^m and Δy_n^m (The random sampling of which could be performed by calling the $normal$ function of the "numpy" package in Python), the new location $\{x_n^m, y_n^m, 0\}$ of the MD m at time slot n is calculated by $x_n^m = x_{n-1}^m + \Delta x_n^m$ and $y_n^m = y_{n-1}^m + \Delta y_n^m$.

The training data for LSTM, FCM-based clustering and reinforcement learning (DDPG) are collected in the following ways:

- For LSTM, the current locations of MDs are leveraged as the label of the previous training data, i.e., the current locations of MDs are packaged with the previous locations as one piece of training data. Under the assumption that MDs move obeying an implicit distribution, LSTM could be trained to model and approximate this distribution.
- For FCM-based clustering, the output of the LSTM model is taken as the input data since FCM-based clustering is an unsupervised machine learning method without the need of labeling data.
- For reinforcement learning (DDPG), the training data of the inner neural networks are generated in the form of the designed four-tuple $(\boldsymbol{s}_n, \boldsymbol{a}_n, r_n, \boldsymbol{s}_{\mathrm{n+1}})$ (defined in Section 3.4.1) via the interaction of the reinforcement learning agent with the environment (MEC network).

The baseline approaches for performance evaluation include:

- Local computing (LC): all tasks are assumed to be executed with the maximal computational capability on the MD where it arrives, without any offloading. The UAVs' locations keep unchanged during the experiment.
- Random offloading (RO): $U + 1$ tasks are randomly selected to be offloaded to the BS and UAVs who move randomly during the experiment. In addition, the MDs and UAVs are assumed to compute and transmit at its maximal capability.
- Greedy offloading (GO): The nearest MD is selected to offload its task to the UAV. One additional sub-nearest MD is selected to offload its task to the UAV who is closest to the BS (where the task is finally offloaded via the UAV). In addition, the MDs and UAVs are assumed to compute and transmit at its maximal capability.
- Alternating optimization (AO): The whole problem is solved by alternately addressing one of the sub-problems while fixing the other two [22], and each alternating solution is performed via the Lagrange duality method or SCA-based CVX.
- Branch-and-Bound (BB): The MINLP problem $\mathcal{P}$ is solved using the branch-and-bound algorithm [43] to obtain the global optimal scheduling policy at each time slot without considering the computation complexity.

H2LS is evaluated from the following aspects: (1) Efficiency (improvements of energy consumption over baseline approaches); (2) Robustness (energy consumption under sudden changes of task size I_n^m during running); (3) Scalability (energy consumption when M increases); (4) Reliability (success rates of task execution when M increases).

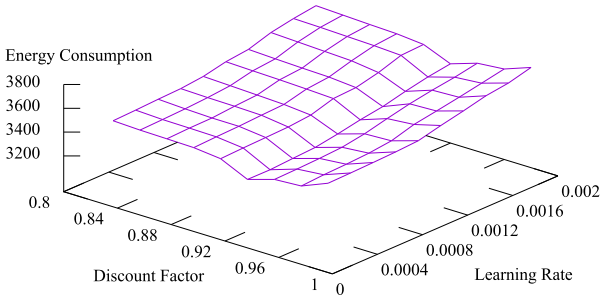

Fig. 5. Energy consumption of H2LS with different parameters.

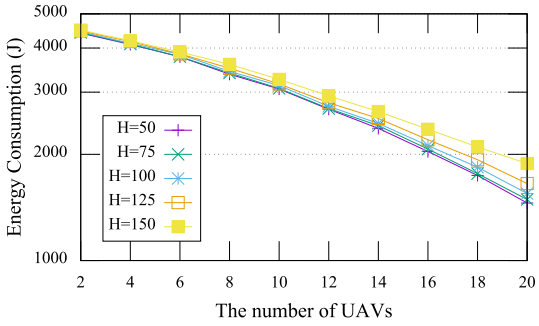

Fig. 6. Energy consumption of H2LS with different heights and numbers of UAVs.

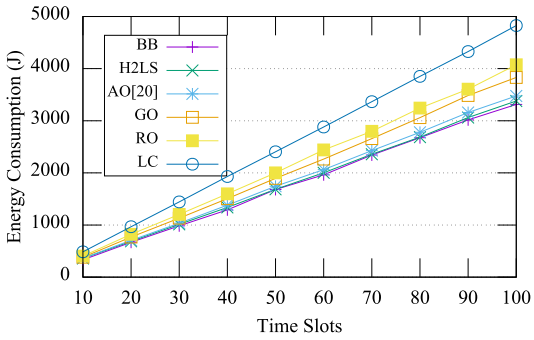

Fig. 7. Energy consumption of different approaches during task executing time.

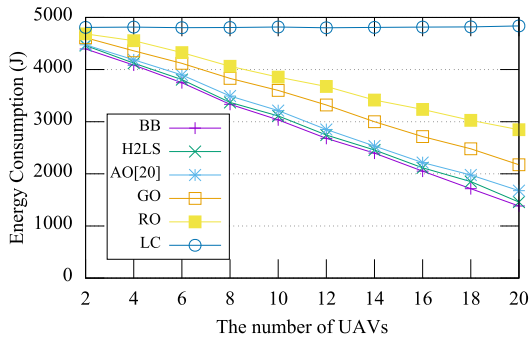

Fig. 8. Energy consumption of different approaches with different number of UAVs.

A. *Parameter Evaluation of H2LS*

We first investigate the impact of the hyper-parameters (the discount factor and learning rate) on the scheduling performance. As shown in Fig. 5, the energy consumption of H2LS is evaluated when the discount factor increases from 0.8 to 0.98 and the learning rate increases from 0.0002 to 0.002. It can be noticed that the energy consumption reduces slowly when the discount factor increases from 0.8 to 0.92 and grows dramatically when the discount factor increases from 0.92 to 0.98. In addition, the energy consumption reduces from 3800 to 3200 when the learning rate decreases from 0.002 to 0.0002. However, when the learning rate decreases from 0.0008 to 0.0002, the energy consumption declines very slowly while the training of H2LS also becomes very slow. Hence, the discount factor and learning rate are designated as 0.92 and 0.0008, respectively, for the sake of reaching the trade-off between scheduling performance and training difficulty.

Then, we evaluate the performance of H2LS w.r.t. energy consumption with different numbers and heights of UAVs, as shown in Fig. 6. It can be seen that the energy consumption of H2LS declines obviously when the number of UAVs increase from 2 to 20, which could be resulted from the more opportunities for MDs to offloading their tasks to UAVs or the BS. Meanwhile, when the height of UAVs increases, obvious increments of energy consumption could be noticed, which suggests that UAVs should not fly too high when satisfactory line-of-sight communication have been achieved. In addition, it could also be found that the higher are the UAVs the larger are the increments of energy consumption with the same number of UAVs, which could be due to the heavy impact of transmission distance on scheduling decisions and energy consumption.

B. *Efficiency Evaluation of H2LS*

In this part, we evaluate the efficiency of H2LS by comparing its energy consumption with those of baseline approaches. As shown in Fig. 7, the energy consumption of all approaches increases over time slots during the execution of mUB-MEC network. It can be noticed that BB achieves the lowest energy consumption among all approaches which benefits from BB's exhaustive search of the global optimal scheduling policy. However, the high computation complexity of BB prevents its widespread application in MEC. In comparison, the energy consumption of LC is the highest, since all tasks of LC are performed in MDs locally. In addition, GO outperforms RO in terms of MDs' energy consumption, which could be due to the reason that the MDs selected by GO to offload tasks are nearest to UAVs and consumes lower energy for wireless transmission than RO does. However, the energy consumption of H2LS is lower than that of AO, close to that of BB, and much lower than those of GO, RO, and LC, while the computation complexity of H2LS is much lower than that of BB and more feasible for practical application in large-scale mUB-MEC. In Fig. 8, the energy consumption of all approaches is evaluated when the number of UAVs increases from 2 to 20. It can be seen that the energy consumption of LC remains constant when the number of UAVs changes while the energy consumption of RO, GO, AO, H2LS and BB reduces gradually when the

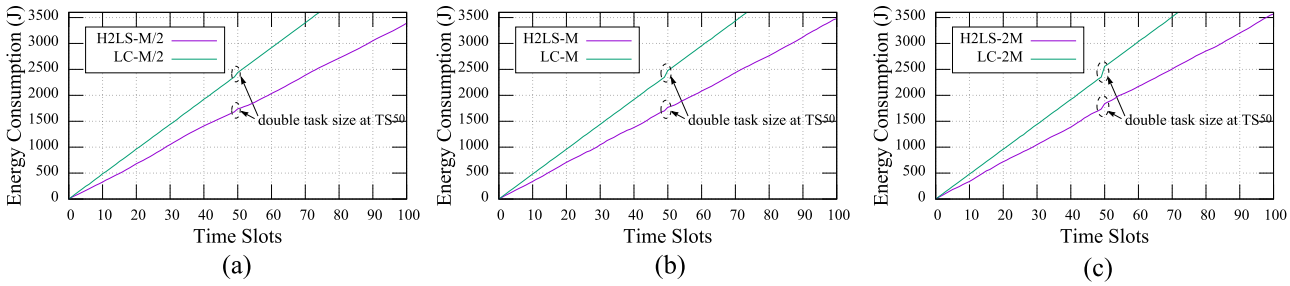

Fig. 9. Energy consumption of H2LS under sudden change of task size (double size from 50 kb to 100 kb) at TS^{50}.

number of UAVs increases. Whereas, the MDs' energy saved by BB, H2LS and AO is much more than that by RO and GO when the number of UAVs grows, which could be resulted from the reason that the scheduling policies obtained by BB, H2LS and AO are more desirable than those by RO and GO and can make better use of the increased UAVs' computation resources to save MD energy consumption. Furthermore, it can be found that the energy consumption of H2LS is lower than that of AO and close to that of BB, which indicates the promising scheduling performance of H2LS in terms of energy consumption.

C. Robustness Evaluation of H2LS

In Fig. 9, we evaluate the robustness of H2LS in terms of energy consumption when the task size I_n^m of MDs changes suddenly.

As shown in Fig. 9(a), when $U/2$ MDs change double task size at TS^{50}, the energy-consumption increments of H2LS and LC are both small and close to each other. The increment of H2LS could mainly result from the transmission of size-doubled tasks, while the increment of LC results from the local computation of size-doubled tasks. When the number of MDs doubling task size is small, i.e., $U/2$, the energy consumption for local computation and task transmission are both small and close to each other.

As shown in Fig. 9(b), when U MDs change double task size at TS^{50}, the energy-consumption increment of H2LS remains small (less than 50 J), while the increment of LC is distinctly larger than that in Fig. 9(a). It indicates that H2LS is more robust than LC w.r.t. energy consumption when the MEC network encounters a burst of task size change (U MDs double task size). The reason could be that H2LS can adapt to the sudden task-size change of U MDs by selecting and offloading the most suitable tasks from double-sized tasks for edge computing based on its learned near-optimal scheduling policies, which could lead to a low increment of energy consumption mainly resulting from task transmission. Whereas, LC could only choose to execute the double-sized tasks locally, which give rise to a large increment of energy consumption proportion to the number of MDs doubling task sizes.

A similar result could be found in Fig. 9(c), where the energy-consumption increment of LC is much higher than that of H2LS.

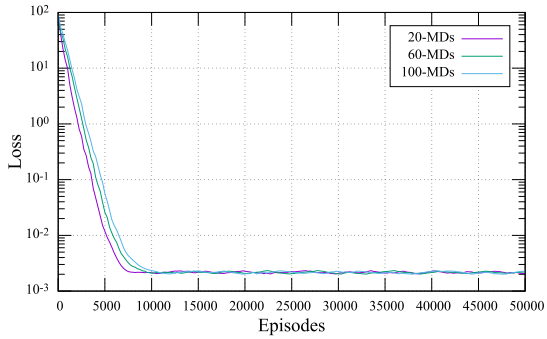

Fig. 10. The loss of H2LS with different numbers of MDs.

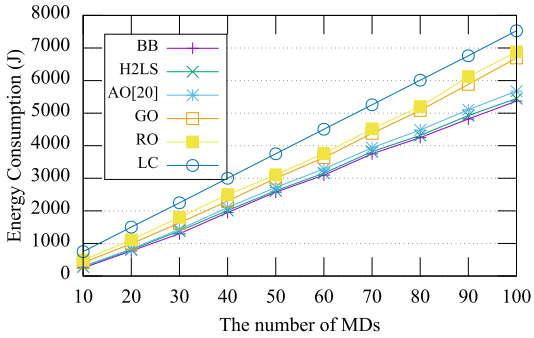

Fig. 11. Energy consumption of different approaches with different numbers of MDs.

D. Scalability Evaluation of H2LS

In Fig. 10 and Fig. 11, we evaluate the scalability (w.r.t. different numbers of MDs) of H2LS in terms of the loss value and energy consumption, respectively.

As shown in Fig. 10, the train losses of H2LS with different numbers (20, 60 and 100) of MDs are evaluated to demonstrate its convergence scalability. The evaluation sees all convergences for H2LS with 20, 60 and 100 MDs (labeled as H2LS-20, H2LS-60 and H2LS-100, respectively). It indicates that H2LS remains scalable when the mUB-MEC network scale increases. Meanwhile, it can be found that the convergence time grows with the number of MDs: H2LS-20 converges at about 7000 episodes, while H2LS-60 and H2LS-100 converge at 9000 and 12000, respectively. This could mainly result from the fact that the increase of MDs gives rise to the expansion of state and action spaces, which raises the difficulty of H2LS for exploration and learning.

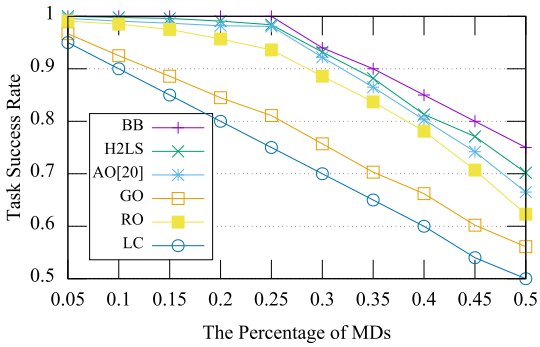

Fig. 12. Task success rate w.r.t. different percentages of MDs increasing task computation intensity C_n^m.

As shown in Fig. 11, the energy consumption of H2LS with different numbers (from 10 to 100) of MDs are compared with those of BB, AO, GO, RO and LC to demonstrate its performance scalability. When the number of MDs rises, the energy consumption of all approaches grows gradually. The energy-consumption increment of LC is the largest, which is close to 7000 J, while H2LS can obtain a much lower increment (5000 J) which is rather close to that of the optimal approach (BB) and lower than that of AO. The possible reason could be that LC can only choose local computing while H2LS can offload high energy-consuming tasks for edge computing. It is worth noting that the energy-consumption increment of H2LS could be much lower if the number of UAVs increases correspondingly with the number of MDs. This demonstrates that the energy-consumption performance of H2LS remains acceptable and is scalable when the network scale increases.

E. Reliability Evaluation of H2LS

In Fig. 12, the task success rates of H2LS w.r.t. different settings of computation intensity C_n^m are compared with those of other approaches to evaluate its reliability. When the percentage of MDs (whose arrived tasks' computation intensity C_n^m doubles) increases from 5% to 50%, all approaches witness the decline of task success rate. This is caused by the reason that the intensity-doubled tasks could not finish the computation in the MDs if they are not scheduled for computation offloading, since their required CPU cycles exceed the computation supply of local MDs within one time slot τ or the maximal allowable execution time T^{req}. Meanwhile, it can be noticed that the task success rate of LC declines most dramatically and is almost inversely linear with the percentage of MDs. However, the success-rate decline of H2LS is only half of LC and close to that of BB. The results also show that the success rate of H2LS is close to 1 when the percentage is less than 25%, which indicates the substantial reliability of H2LS over AO, GO, RO and LC under a medium percentage of intensity-doubled MDs. In addition, further improvements of H2LS's task success rate against the increase of computation intensity could be anticipated, if more UAVs are endowed with the mUB-MEC network.

V. Conclusions and Future Work

This paper investigated a hybrid enabled MEC system (mUB-MEC) where multiple UAVs and one terrestrial BS are deployed to provide MEC services for ground MDs. A joint optimization problem is formulated to minimize the MDs' energy consumption of mUB-MEC, which is achieved by decomposing the whole problem into three sub-problems and solving them via the proposed H2LS algorithm. The experimental results show that H2LS obtains substantial performance improvements over baseline approaches in various aspects, including low energy consumption with different numbers of UAVs, robust to the outbreak of task-size change, scalable to hundreds of MDs, promising reliability for computation-intensive tasks, etc. In the future work, we will extend the study to consider improving MDs' quality of experience w.r.t. other metrics, e.g., average task latency, maximum task latency, weighted cost of task latency and energy consumption. In addition, the condition of all UAVs working on the same transmission frequency will be considered, the scenario where the BS is endowed with limited computing and energy resources will be studied, and the adoption of some promising t(wireless power transfer, massive multiple-input multiple-output, etc.) in mUB-MEC will be investigated to further improve the overall MEC performance.

References

[1] G. Forecast, "Cisco visual networking index: Global mobile data traffic forecast update, 2017- 2022," *Update*, vol. 1, no. 1, pp. 6–29, Feb. 2019.
[2] I. Update, "Ericsson Mobility Report," Ericsson: Stockholm, Sweden, pp. 3–4, Jun. 2021.
[3] M. Patel *et al.*, "Mobile-edge computing introductory technical white paper," *White Paper, Mobile-Edge Computing (MEC) Industry Initiative*, vol. 29, pp. 854–864, 2014.
[4] M. Satyanarayanan, "The emergence of edge computing," *Comput.*, vol. 50, no. 1, pp. 30–39, Jan. 2017.
[5] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surv. Tut.*, vol. 19, no. 3, pp. 1628–1656, Jul-Sep. 2017.
[6] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of MEC in the Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 5, no. 4, pp. 84–91, Oct. 2016.
[7] W. Yu *et al.*, "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.
[8] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.
[9] C. Chen, J. Hu, T. Qiu, M. Atiquzzaman, and Z. Ren, "CVCG: Cooperative V2V-aided transmission scheme based on coalitional game for popular content distribution in vehicular ad-hoc networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 12, pp. 2811–2828, Dec. 2018.
[10] T. Qiu, X. Wang, C. Chen, M. Atiquzzaman, and L. Liu, "TMED: A spider-web-like transmission mechanism for emergency data in vehicular ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8682–8694, Sep. 2018.
[11] Y. Zeng, R. Zhang, and T. J. Lim, "Throughput maximization for UAV-enabled mobile relaying systems," *IEEE Trans. Commun.*, vol. 64, no. 12, pp. 4983–4996, Dec. 2016.
[12] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-UAV enabled wireless networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 3, pp. 2109–2121, Mar. 2018.
[13] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with unmanned aerial vehicles: Opportunities and challenges," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 36–42, May. 2016.
[14] M. M. Azari, F. Rosas, K.-C. Chen, and S. Pollin, "Ultra reliable UAV communication using altitude and cooperation diversity," *IEEE Trans. Commun.*, vol. 66, no. 1, pp. 330–344, Jan. 2018.

[15] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient uav control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2059–2070, Sep. 2018.
[16] 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 Things J.*, vol. 6, no. 2, pp. 1879–1892, Apr. 2019.
[17] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 1927–1941, Sep. 2018.
[18] H. Guo and J. Liu, "UAV-enhanced intelligent offloading for Internet of Things at the edge," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2737–2746, Apr. 2020.
[19] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, "Energy-efficient UAV-assisted mobile edge computing: Resource allocation and trajectory optimization," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3424–3438, Mar. 2020.
[20] 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 *Proc. IEEE Conf. Comput. Commun. Workshops*, 2020, pp. 150–156.
[21] Z. Yu, Y. Gong, S. Gong, and Y. Guo, "Joint task offloading and resource allocation in UAV-enabled mobile edge computing," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3147–3159, Apr. 2020.
[22] X. Hu, K. Wong, K. Yang, and Z. Zheng, "UAV-assisted relaying and edge computing: Scheduling and trajectory optimization," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4738–4752, Oct. 2019.
[23] B. Shang and L. Liu, "Mobile-edge computing in the sky: Energy optimization for air-ground integrated networks," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7443–7456, Aug. 2020.
[24] G. Zhang, Y. Chen, Z. Shen, and L. Wang, "Distributed energy management for multiuser mobile-edge computing systems with energy harvesting devices and QoS constraints," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4035–4048, Jun. 2018.
[25] D.-H. Tran, T. X. Vu, S. Chatzinotas, S. ShahbazPanahi, and B. Ottersten, "Coarse trajectory design for energy minimization in UAV-enabled," *IEEE Trans. Veh. Technol.*, vol. 69, no. 9, pp. 9483–9496, Sep. 2020.
[26] P. X. Nguyen *et al.*, "Backscatter-assisted data offloading in OFDMA-based wireless-powered mobile edge computing for iot networks," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 9233–9243, Jun. 2021.
[27] E. L. Sueur and G. Heiser, "Dynamic voltage and frequency scaling: The laws of diminishing returns," in *Proc. Int. Conf. Power Aware Comput. Syst.*, 2010, pp. 1–8.
[28] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2329–2345, Apr. 2019.
[29] X. Chen *et al.*, "Information freshness-aware task offloading in air-ground integrated edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 243–258, Jan. 2022.
[30] Y. Wang, Z.-Y. Ru, K. Wang, and P.-Q. Huang, "Joint deployment and task scheduling optimization for large-scale mobile users in multi-UAV-enabled mobile edge computing," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3984–3997, Sep. 2020.
[31] L. Wang *et al.*, "Rl-based user association and resource allocation for multi-UAV enabled MEC," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf.*, 2019, pp. 741–746.
[32] J. Kwak, Y. Kim, J. Lee, and S. Chong, "DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2510–2523, Dec. 2015.
[33] Z. Yang, C. Pan, K. Wang, and M. Shikh-Bahaei, "Energy efficient resource allocation in UAV-enabled mobile edge computing networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 9, pp. 4576–4589, Sep. 2019.
[34] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2049–2063, Mar. 2018.
[35] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3747–3760, Jun. 2017.
[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[37] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy c-means clustering algorithm," *Comput. Geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.
[38] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2016.
[39] M. Grant and S. Boyd, "CVX: Matlab Software for Disciplined Convex Programming, Version 2.1," [Online]. Available: http://cvxr.com/cvx
[40] S. Diamond and S. Boyd, "CVXPY: A python-embedded modeling language for convex optimization," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2909–2913, 2016.
[41] F. A. Potra and S. J. Wright, "Interior-point methods," *J. Comput. Appl. Math.*, vol. 124, no. 1-2, pp. 281–302, 2000.
[42] J. Ahn, J. Lee, D. Niyato, and H.-S. Park, "Novel QoS-guaranteed orchestration scheme for energy-efficient mobile augmented reality applications in multi-access edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13 631–13 645, Nov. 2020.
[43] P. M. Narendra and K. Fukunaga, "A branch and bound algorithm for feature subset selection," *IEEE Comput. Archit. Lett.*, vol. 26, no. 09, pp. 917–922, Sep. 1977.

Bin Dai (Member, IEEE) received the B.Eng. and M.S. degrees in 2010 and 2013, respectively, in computer science and engineering from Beihang University, Beijing, China, where he is currently working toward the Ph.D. degree in computer science and engineering, where he works on mobile edge computing and federated learning.

Jianwei Niu (Senior Member, IEEE) received the M.S. and Ph.D. degrees in computer science from Beihang University, Beijing, China, in 1998 and 2002, respectively. From 2010 to 2011, he was a Visiting Scholar with the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA. 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.

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

Zheyuan Hu (Graduate Student Member, IEEE) received the B.S. degree in computer science and engineering from Northeastern University, Shenyang, China, in 2017. He is currently working toward 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.

Mohammed Atiquzzaman (Senior Member, IEEE) is currently the Edith Kinney Gaylord Presidential Professor of computer science with the University of Oklahoma, Norman, OK, USA. He teaches courses in Data Networks and Computer Architecture. His research interests and publications include next generation computer networks, wireless and mobile networks, satellite networks, switching and routing, optical communications, and multimedia over networks. Many of his research activities are supported by the National Science Foundation, National Aeronautics and Space Administration, the U.S. Air Force, Honeywell, and Cisco. He is the Editor-in-Chief of the *Journal of Network and Computer Applications* and *Vehicular Communications* journal and an Associate Editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON MOBILE COMPUTING, *International Journal of Communication Systems*, *International Journal of Sensor Networks*, *International Journal of Communication Networks and Distributed Systems*, and *Journal of Real-Time Image Processing*.