Learning-Based Computation Offloading Approaches in UAVs-Assisted Edge Computing

Shichao Zhu, Lin Gui, *Member, IEEE*, Dongmei Zhao, *Member, IEEE*, Nan Cheng, *Member, IEEE*, Qi Zhang, and Xiupu Lang

***Abstract*—Technological evolutions in unmanned aerial vehicle (UAV) industry have granted UAVs more computing and storage resources, leading to the vision of UAVs-assisted edge computing, in which the computing missions can be offloaded from a cellular network to a UAV cloudlet. In this paper, we propose a UAVs-assisted computation offloading paradigm, where a group of UAVs fly around, while providing value-added edge computing services. The complex computing missions are decomposed as some typical task-flows with inter-dependencies. By taking into consideration the inter-dependencies of the tasks, dynamic network states, and energy constraints of the UAVs, we formulate the average mission response time minimization problem and then model it as a Markov decision process. Specifically, each time a mission arrives or a task execution finishes, we should decide the target helper for the next task execution and the fraction of the bandwidth allocated to the communication. To separate the evaluation of the integrated decision, we propose multi-agent reinforcement learning (MARL) algorithms, where the target helper and the bandwidth allocation are determined by two agents. We design respective advantage evaluation functions for the agents to solve the multi-agent credit assignment challenge, and further extend the on-policy algorithm to off-policy. Simulation results show that the proposed MARL-based approaches have desirable convergence property, and can adapt to the dynamic environment. The proposed approaches can significantly reduce the average mission response time compared with other benchmark approaches.**

***Index Terms*—Bandwidth allocation, computation offloading, inter-dependencies, multi-agent reinforcement learning, UAV.**

I. Introduction

THE rapid development of 5G technology has motivated the evolution of a wide spectrum of computation-intensive services, such as virtual reality, autonomous driving, and video navigation [1], which pose severe challenges on the portable devices with limited computing resources. One probable solution is edge computing (EC), which allows computing missions to be offloaded to and executed at edge servers. An edge server can be placed at an LTE Evolved Node B (eNB) or Wi-Fi access point (AP). However, considering the construction cost and the gradual updating process of the infrastructures, not all of the eNBs or APs can be integrated with powerful computing resources. Thus, mobile platforms such as cars and unmanned aerial vehicles (UAVs), are envisioned as optional candidates for edge servers [2]–[4]. Recently, research and applications of UAVs are becoming increasingly prosperous due to the maturity of the aircraft technology and regulations. UAVs are being utilized mainly in package delivery, public surveillance, environment monitoring, emergency assistance, and construction [5], which are termed as the UAVs' major services. Meanwhile, with the development of chip industry and miniaturization of smart components, more computing and storage resources can be placed onboard. This enables the UAVs the potential to simultaneously provide value-added computing services (VACS) [6]–[8], where the computing missions are offloaded from the cellular network to the UAVs. The UAVs fly around mainly for their major services, and meanwhile they are further exploited as edge servers, providing considerable computing resources without affecting the major services, which is envisioned to play a prominent role in edge computing implementation.

There are extensive research works on the UAVs-assisted edge computing (UEC), including energy efficiency [8], trajectory design [9], and joint optimization of communication and computation [10]. However, the existing works are with several shortcomings. Firstly, few of them consider the scenario of multiple UAVs. A single UAV possesses limited computing resources because of the limited payload, and the UEC need a group of UAVs to share the computation workload. Secondly, the air-to-ground channel pathloss is usually assumed as known. But in practice, the channel pathloss is highly related to the obstacles' density [11], and the accurate parameters in the channel model expression are difficult to obtain, which means the relationship between the pathloss value and the air-to-ground distance is ambiguous. Thirdly, further research is required to model complex computing missions as task-flows by considering the inter-dependencies of the tasks. For example, a video navigation mission involves graphics, face detection, camera preview, and video processing, and can be expressed as a dependency graph of 14 tasks [12]. Since the UAVs have high dynamics, leading to varying channel states, changing the target UAV after a task finishes help to achieve shorter communication time. Letting

Manuscript received May 8, 2020; revised August 29, 2020 and November 15, 2020; accepted December 27, 2020. Date of publication January 5, 2021; date of current version February 12, 2021. This work was supported in part by the Natural Science Foundation of China under Grant 61671295, and in part by the National Fundamental Research Key Project of China under Grant JCKY2017203B082. The review of this article was coordinated by Prof. P. Lorenz. *(Corresponding author: Lin Gui.)*

Shichao Zhu, Lin Gui, Qi Zhang, and Xiupu Lang are with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: zhushichao@sjtu.edu.cn; guilin@sjtu.edu.cn; qizhang_sjtu@sjtu.edu.cn; langxp2018@sjtu.edu.cn).

Dongmei Zhao is with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON L8S 4L8, Canada (e-mail: dzhao@mail.ece.mcmaster.ca).

Nan Cheng is with the School of Telecommunication, Xidian University, Xi'an 710071, Shaanxi, China (e-mail: dr.nan.cheng@ieee.org).

Digital Object Identifier 10.1109/TVT.2020.3048938

multiple UAVs execute multiple tasks in parallel also shortens the processing time. Therefore, the offloading decisions should be made with the granularity of a task rather than a whole mission. The authors in [2] and [13] studied complex mission division in vehicle-based cloudlet, and we believe the same idea also applies in UEC. To this end, we aim to study the scenario where complex computing missions are executed by multiple UAVs. There are two challenging issues in this scenario. Firstly, we consider multiple UAVs with varying channel states to address the first and the second shortcomings described above. This results in a more complicated scenario: The different computing capabilities and the varying channel states of UAVs lead to different processing time and communication time, respectively; and the different energy conditions of UAVs also affect the offloading results. The offloading policy should be carefully designed considering the complicated and dynamic network conditions and resources. Secondly, the third shortcoming described above motivates us to study the effect of task inter-dependencies on the offloading policy. Such dependence determines whether tasks should be processed sequentially or in parallel. How to effectively describe task features changing with time is a problem to be solved.

In this paper, we present a learning-based method to find a near-optimal offloading policy of task allocation and bandwidth allocation in the UEC for minimizing the average response time of computing missions. This work takes into consideration the inter-dependencies of tasks, the dynamic channel states, and the energy constraints of UAVs. Each time a new mission arrives or a task finishes, we should pick a proper helper to execute the incoming task, and decide the bandwidth for the possible communication. Based on the observations in [2], the complex inter-dependencies can be divided into three basic logic topologies, i.e., linear, mesh, and tree. We aim to design an expression of the mission status to cover these basic topologies and propose an algorithm to accurately estimate the functional relationship between the environment and the offloading policy. To address this issue, firstly, the problem is formulated as a Markov decision process (MDP), with well-designed expressions of state, action, and reward. The environment information is considered as the state; the offloading policy is considered as the action; and the reward is relevant to the response time performance. Secondly, we propose a multi-agent reinforcement learning (MARL) framework to learn the effect of the complicated environment on the offloading policy. We let the task allocation and bandwidth allocation respectively handled by two agents, and design respective advantage evaluation functions for them to solve the credit assignment challenge. The original algorithm is on-policy, which means only the latest experience is collected for training. To improve the efficiency of the interactions and reduce the training cost, we discuss the rationality that the original algorithm can be extended to off-policy and present the off-policy algorithm. Extensive simulations are performed by considering the dynamics in the number of missions, task characteristics, and mobility model. We give the performance comparisons under different parameters, and analyze the parameters stability region. To the best of our knowledge, our work is the first to concurrently consider the task allocation, bandwidth allocation, and energy constraint for complex computing missions in UEC. This work provides valuable insights for the UEC framework design and implementation. The main contributions of this paper are summarized as follows.

- We formulate the computation offloading problem for complex computing missions in the UEC, considering the task allocation, bandwidth allocation, and energy constraint. The problem is further defined as an MDP, with well-designed expressions of state, action, and reward to represent the environment features.
- We propose framework to learn the near-optimal offloading policy by interactions with the environment. In MARL, we separate the evaluation of the integrated offloading decision, and design respective advantage evaluation functions for the agents. We first provide the original on-policy algorithm and then extend it to off-policy to reduce the training cost.
- The performances of the proposed algorithms are evaluated through extensive simulations concerning different design parameters. We compare the performance of the solutions acquired by with those of the single-agent reinforcement learning (SARL) and the greedy-based method in various task topologies to validate the superiority of the proposed algorithms.

The remainder of this paper is organized as follows. Section II summarizes the backgrounds and related work. In Section III, the system model of UEC is described in detail, followed by the problem formulation. The MARL-based solutions are proposed in Section IV, including the on-policy and off-policy versions. Simulation results are presented in Section V. Section VI draws the conclusions of the paper and directs our future work. Useful notations used throughout the paper are listed in Table I.

II. Related Work

A. UAVs-Assisted Edge Computing

Since UEC is envisioned as a promising approach to enlarge the service scope of EC, a considerable amount of research has been conducted on the UEC architecture [4], [14], [15]. In [14] and [15], the researchers did comprehensive surveys of the future applications of UAVs, and both prospected great potentials of UEC. In [8], Motlagh *et al.* claimed that the UAVs are dispatched mainly for the major services with pre-planned moving dynamics, and the EC-related services are value-added, which should be designed to fit the pre-planned path and not affect the major services. In [9], Jeong *et al.* solved the optimization problem about bit allocation and path planning by means of successive convex approximation, and achieved a global minimum of energy consumption. Zhang *et al.* [10] studied the bits allocation, time allocation, power allocation, and UAV trajectory design, aiming to minimize the total energy consumption of the UAV, including communication energy, computing energy, and flight energy. In [16], Zhang *et al.* considered to minimize the average weighted energy consumption of users and the UAV by a Lynapunov-based approach, in which the joint optimization problem is decomposed into three subproblems. Liu *et al.* [17]

TABLE I
NOTATIONS USED IN THIS PAPER

Notation	Description
K	Number of missions in an episode
N	Max number of UAVs in the UC
d_k, b_k	Number of tasks, and number of tasks which need to transmit back the result in the k-th mission
ϕ_i^k, α_i^k, β_i^k, ω_i^k	The tuple expression, input data size, output data size, and computation to data ratio of the i-th task in mission k
x_i^k	The helper executing task ϕ_i^k
ρ_i^k	Fraction of the bandwidth allocated for the transmission of the input data of task ϕ_i^k
$\mathcal{C}(x_i^k)$	Computing capability of x_i^k
B	Total bandwidth for eNB-UAV, UAV-UAV, and UAV-eNB communication
N_a	Number of channels
$I_{i,j}^k$	Indicator about whether task j needs the output data of task i in mission k
P_u, P_e	Transmit power of UAV and eNB
ε, p_h	Energy radiation efficiency and transmit power of the stratosphere balloon
λ	Arrival interval of missions
t_k^G	Arrival time of mission k
r_i^k	The starting time of task ϕ_i^k
$q_i^k, c_i^k, p_i^k, t_i^k$	Queueing time, communication time, processing time, and total response time of task ϕ_i^k

minimized the energy consumption of UAV by jointly optimizing the CPU frequencies, offloading amount, transmit power, and UAV's trajectory. A successive convex approximation-based algorithm is proposed to tackle the non-convex problem. In [18], Zhou *et al.* considered a scenario where the UAV can provide the service of edge computing and wireless power transfer to the users at the same time. The researchers proposed a two-stage algorithm and a three-stage alternative algorithm to handle the non-convex problem of computation rate maximization.

B. Reinforcement Learning for Network Resource Allocations

Reinforcement learning (RL) is a powerful tool solving model-free MDP problems, and is widely studied in the field of video games [19] and continuous action control [20]. Recently, researchers attempted to make use of RL algorithms in network resource allocations, since the optimal resource allocation problem is also an optimal decision making problem. If there exist certain regularities in the network dynamics, the RL agent can understand it and learn a proper policy. In [21], Chen *et al.* designed computation offloading policies in EC based on the channel state, energy queue state, and task queue state, with the help of deep Q-networks. In [22], Ye *et al.* studied the deep reinforcement learning (DRL)-based resource allocation mechanism in vehicle-to-vehicle (V2V) communications. The authors considered a V2V link as an agent, which makes decisions to find the optimal sub-band and power level for transmissions. In [23], Sun *et al.* studied the computation migration problem in the vehicular cloud scenario, considering the complex missions with linear inter-dependency topology. The authors used a polynomial to estimate the value function and adopted the SARSA algorithm for training. Zhu *et al.* [24] considered the UAVs-assisted edge computing for complex missions with three basic inter-dependency topologies. Single-agent actor-critic (AC) algorithm was adopted to train an agent for choosing a proper target UAV for each task. In [25], Min *et al.* proposed an RL-based offloading scheme for an IoT device with energy harvesting. The current battery level, the previous radio transmission rate to each edge device, and the predicted amount of the harvested energy are considered to generate the optimal edge device selection and offloading rate.

Different from the previous work, in this work we simultaneously consider the task allocation and bandwidth allocation in complex computing missions offloading. This can be considered as an extension to reference [24], which also used an RL method to solve the task offloading problem in UAVs-assisted edge computing. Since parallel transmission was not considered in [24], bandwidth allocation was not studied. In addition, [24] did not consider the energy constraint of UAVs and the dynamic task and mobility model. The simultaneous consideration of the task allocation and bandwidth allocation greatly complicates the task execution model and poses severe challenges to the RL framework design. In Section V, we will use extensive simulations to prove that the proper bandwidth allocation is of great importance, and the approaches proposed in this paper can lead to significant performance improvement compared with the single-agent learning method in [24].

III. SYSTEM MODEL AND PROBLEM FORMULATION

A. System Overview

Fig. 1 shows the system of UEC. We assume the UAVs in this situation are dispatched for their major services, such as surveillance or monitoring. In the meanwhile, the UAVs can also be equipped with some computing resources to act as edge servers, providing VACS as a UAV cloudlet (UC) [7], [8]. The UAVs should always collectively fly around along with a pre-planned path according to the requirements of the major services, but cannot hover at a fixed point for edge computing. Since a UAV is an energy-limited platform, the energy consumed by the computation and communication in a UAV should be no more than the energy supply. A UAV's energy can be supplied by the wireless power transferred from another node [25]–[27]. An eNB collects the computing requests from the portable devices within its coverage, processes parts of the computing missions by itself, and offloads the remainder to the UC. The eNB and UAVs are defined as helpers. Each time a task needs to be scheduled, a target helper should be assigned for the task execution,

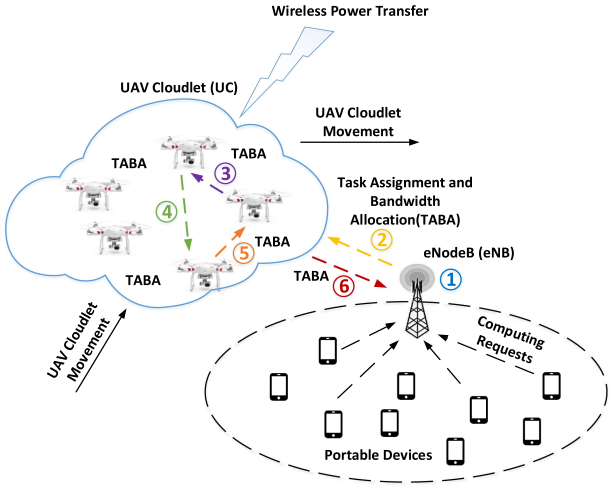


Fig. 1. Scenario of UAVs-assisted edge computing.

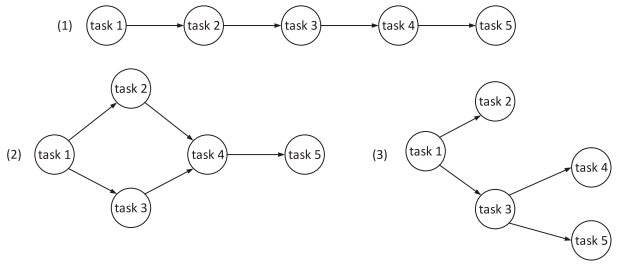


Fig. 2. Task-flow models.

and some bandwidth should be allocated for the computing input data transmission. The computing capability of the portable devices is not comparable with that of the eNB and UC [28], thus we do not consider the local execution at the portable devices. We denote a set of UAVs in the UC as $\mathcal{U} = \{1, \ldots, N\}$, where N is the maximum number of UAVs in the UC. We consider a dynamic scenario where UAVs leave and join the UC, thus the practical number of UAVs is time-varying. The UAV set in the UC at time t is defined as $\mathcal{U}_t$, and $\mathcal{U}_t \subseteq \mathcal{U}$. We define a period of time when computing missions arrive with some regularities as an episode, and the set of computing missions in the episode is denoted as $\mathcal{M} = \{m_1, \ldots, m_K\}$, where K is a random variable representing the number of missions in an episode. We assume the computing missions are time-driven [3], and use λ to denote the arrival interval of the computing missions in an episode.

B. Task Model

We consider complex computing missions which can be modeled as some typical task-flows with the inter-dependencies of tasks [2]. The complex inter-dependencies can be divided into three basic logic topologies, i.e., linear, mesh and tree, as shown in Fig. 2. For a given mission, a task can be processed when all the upstream tasks in the mission have been processed.

To describe the parametric context of each task, we define a 3-tuple as $\phi_i^k = (\alpha_i^k, \beta_i^k, \omega_i^k)$, where α_i^k, β_i^k and ω_i^k, respectively, represent the input data size, the output data size, and the computation to data ratio [28] of the i-th task of mission m_k. Denoting the number of tasks in m_k as d_k, the inter-dependencies in m_k can be described as a square matrix $\mathbf{I}^k = (I_{i,j}^k)_{d_k \times d_k}$, where $I_{i,j}^k = \{0, 1\}$ is the indicator about whether task j needs the output data of task i. Therefore, the relationship between the data sizes of the tasks in m_k can be described as $\alpha_i^k = \sum_j \beta_j^k I_{j,i}^k$. In addition, b_k is defined as the number of tasks whose results should be sent back to the eNB. For missions (1) and (2) shown in Fig. 2, $b_k = 1$; and for mission (3) shown in Fig. 2, $b_k = 3$. The index set of the K missions is denoted as $\mathcal{K} = \{1, \ldots, K\}$, and the index set of the d_k computing tasks plus the b_k results feedback in m_k is denoted as $\mathcal{D}_k = \{1, \ldots, d_k + b_k\}$.

C. Communication Model

We consider the pathloss-based channel gains. Small scale fading is not considered because the time scale of decision making is much larger. Decisions on offloading targets and resource allocations are not changed when tasks are being executed. Referring to [29], for the air-to-ground (A2G) communications, the pathloss is given by

$$PL_{A2G}(r,h) = 20\log\left(\frac{4\pi f_c\sqrt{h^2+r^2}}{c}\right) + P_{LoS}(r,h)\eta_{LoS} + \left(1 - P_{LoS}(r,h)\right)\eta_{NLoS}, \quad (1)$$

where h and r (in m) denote the UAV's altitude and the horizontal distance between the UAV and the ground node, respectively, f_c (in Hz) is the carrier frequency, and c (in m/s) is the speed of light. P_{LoS} denotes the line-of-sight transmission probability of the A2G link, and is given by

$$P_{LoS}(r,h) = \frac{1}{1 + a\exp\left(-b(\arctan\left(\frac{h}{r}\right) - a)\right)}. \quad (2)$$

In addition, parameters (a, b, η_{LoS}, η_{NLoS}) are related with the obstacles' density and heights, and are hard to be acquired a priori. Thus, the relationship between (r, h) and pathloss is not clear unless a communication link is actually set up.

Since the missions be generated intensively, more than one communication link be needed at a time. We adopt LTE transmission scheme, where an eNB allocates orthogonal frequency division multiplexing (OFDM) resource blocks (RBs) to users. We assume the communications within the UC use dedicated frequencies that do not cause mutual interference with the ground users, thus the bandwidth for communication is set as half of the total available bandwidth in LTE. According to the LTE standard [30], each resource block occupies about 180 kHz bandwidth and 0.5 ms time interval, thus the total bandwidth can be divided into 56 RBs in the frequency domain, which is much more than the number of UAVs analyzed in this work [31]. Therefore, we do not consider further allocating the communication resource in the time domain, i.e., when a link is allocated within some bandwidth, it will occupy the bandwidth in the following successive slots. Besides, since the number of UAVs usually cannot reach the maximum allowed number of users, allocating the bandwidth with the units of RBs will increase unnecessary complexity. Thus, we divide the total bandwidth into N_a logical channels, and each channel equals to a multiple of 180 kHz. The proper value of N_a will be obtained by simulations.

Based on the analysis above, for a given UAV u_i, the eNB-UAV data rate and the UAV-eNB data rate are given by

$$r_{eNB,i}(\rho) = \rho B \log_2\left(1 + \frac{P_e 10^{\frac{-PL_{A2G}(r_i,h_i)}{10}}}{\sigma^2 \rho B}\right), \quad (3)$$

and

$$r_{i,eNB}(\rho) = \rho B \log_2\left(1 + \frac{P_u 10^{\frac{-PL_{A2G}(r_i,h_i)}{10}}}{\sigma^2 \rho B}\right), \quad (4)$$

respectively, where B denotes the total available bandwidth, ρ denotes the fraction of the bandwidth allocated to the currently scheduled communication link and is a multiple of $\frac{1}{N_a}$, P_e and P_u denote the transmit power of the eNB and the UAV, respectively, and σ^2 denotes the power spectrum density of the white Gaussian noise.

The air-to-air (A2A) communication is dominated by line-of-sight transmissions [32], thus the free space pathloss model is adopted, which is given by

$$PL_{A2A}(d) = 32.45 + 20\log f_c + 20\log d, \quad (5)$$

where f_c is the carrier frequency (in MHz), and d is the distance between the UAVs (in Km). Thus, the UAV-UAV data rate is given by

$$r_{i,j}(\rho) = \rho B \log_2\left(1 + \frac{P_u 10^{\frac{-PL_{A2A}(d_{i,j})}{10}}}{\sigma^2 \rho B}\right). \quad (6)$$

We consider unicast for all the communications. If a task has multiple downstream tasks processed in different UAVs, the output data should be transmitted through separated links.

D. Energy Consumption and Harvesting Model

Endurance is always the performance bottleneck of UAV-based applications. In this work, we consider that the UAVs have major services, which should be already well-scheduled with the adaption to the battery condition, and the VACS should not affect the battery planning of the major services. Therefore, we refer to [27] and introduce a multi-tier UAV network architecture, where low altitude UAVs can be charged by stratosphere balloons continuously through laser beams, and the balloons can harvest renewable energy, such as solar or wind energy. We adopt this model since it is with high transmission efficiency and has been studied in many references [27], [33]–[37]. The communication and computing energy consumed by each UAV must be no more than the energy obtained from the stratosphere balloons. We do not consider the UAV propulsion energy consumption, which should be already considered by the planning of major services. The energy consumed in the eNB is also not considered since the eNB can be powered by the grid.

We use the communication energy consumption model in [38], where the uplink power mainly depends on the signal transmitting power, and the downlink power is relevant to the downloading data rate. The uplink and downlink power consumption are given by

$$P_{UL} = k_{tx,1} + k_{tx,2} P_{tx}, \quad (7)$$

and

$$P_{DL} = k_{rx,1} + k_{rx,2} r_{DL}, \quad (8)$$

respectively, where $k_{tx,1}$ denotes the uplink circuit baseline power, $k_{tx,2}$ denotes the efficiency of the amplifier, P_{tx} denotes the signal transmitting power and is equivalence of P_u, $k_{rx,1}$ denotes the downlink circuit baseline power, $k_{rx,2}$ denotes the data rate-to-power coefficient, and r_{DL} denotes the downloading data rate. Based on the real measurements on an LTE mobile phone [38], $k_{tx,1}$, $k_{tx,2}$, $k_{rx,1}$, and $k_{rx,2}$ are 0.4 W, 18, 0.4 W, and 2.86×10^{-3} W/Mbps, respectively.

Denoting the energy consumption of 1 CPU cycle at the UAV edge server as η J/cycle, the helper of task ϕ_i^k as x_i^k, the upstream and downstream task set of task ϕ_i^k as $\mathcal{I}_0$ and $\mathcal{I}_1$, respectively, i.e., $I_{i_0,i}^k = 1, \forall i_0 \in \mathcal{I}_0$, and $I_{i,i_1}^k = 1, \forall i_1 \in \mathcal{I}_1$, the energy consumption for the upstream helpers, current helper, and downstream helpers caused by ϕ_i^k are given by

$$E_{x_{i_0}^k} = P_{UL} \frac{\beta_{i_0}^k}{r_{x_{i_0}^k, x_i^k}}, \forall i_0 \in \mathcal{I}_0, \quad (9)$$

$$E_{x_i^k} = P_{DL} \sum_{i_0 \in \mathcal{I}_0} \frac{\beta_{i_0}^k}{r_{x_{i_0}^k, x_i^k}} + \eta \alpha_i^k \omega_i^k + P_{UL} \sum_{i_1 \in \mathcal{I}_1} \frac{\beta_{i_1}^k}{r_{x_i^k, x_{i_1}^k}}, \quad (10)$$

and

$$E_{x_{i_1}^k} = P_{DL} \frac{\beta_i^k}{r_{x_i^k, x_{i_1}^k}}, \forall i_1 \in \mathcal{I}_1, \quad (11)$$

respectively.

The balloon floats at the fixed altitude of 20 km, and uses the 810 nm laser to radiate power. The radiated power is described as p_h, and referring to [27], the power transmission efficiency ε is 0.02. All UAVs are in the coverage of the beam, thus each one will be supplied with the power as εp_h.

E. Task Execution Model

Each time a computing mission arrives or a task execution is finished, the system will assign a target helper for the incoming task and decide the communication bandwidth. We aim at minimizing the average response time of missions under the energy constraint of the UAVs. The response time of task ϕ_i^k consists of 3 parts: queueing time q_i^k, communication time c_i^k, and processing time p_i^k, which reflect the 3 stages of task execution.

Denoting the fraction of the bandwidth allocated for the input data transmission of task ϕ_i^k as ρ_i^k, the communication time is given by

$$c_i^k = \sum_{i_0 \in \mathcal{I}_0} \frac{\beta_{i_0}^k}{r_{x_{i_0}^k, x_i^k}\left(\rho_i^k\right)}. \quad (12)$$

If the target helper is the same as the task's current location, the communication time is 0. The processing time depends on the computing workload of the task and the computing capability of

the selected helper. Denoting the computing capability of helper x_i^k as $\mathcal{C}(x_i^k)$, the processing time is given by

$$p_i^k = \frac{\alpha_i^k \omega_i^k}{\mathcal{C}\left(x_i^k\right)}. \tag{13}$$

The queueing time consists of two parts: local queueing time and remote queueing time, which are represented as $\delta_1(x_i^k, \rho_i^k)$ and $\delta_2(x_i^k, \rho_i^k)$, respectively. The local queueing time is the time waiting for adequate bandwidth; and the remote queueing time occurs after data transmission for task ϕ_i^k has finished while the target help is still occupied by other tasks.

Before expressing $\delta_1(x_i^k, \rho_i^k)$, we temporarily assume ν as possible local queueing time. We further define an indicator about whether task ϕ_i^k and task $\phi_{i'}^{k'}$ share the total bandwidth, i.e., whether the communication time of task ϕ_i^k and $\phi_{i'}^{k'}$ have any overlapping, by

$$\begin{aligned}\vartheta_{i'}^{k'} = \mathbb{I}(&\left[r_i^k + \nu, r_i^k + \nu + c_i^k\right] \\ &\cap \left[r_{i'}^{k'} + \delta_1(x_{i'}^{k'}, \rho_{i'}^{k'}), r_{i'}^{k'} + \delta_1(x_{i'}^{k'}, \rho_{i'}^{k'}) + c_{i'}^{k'}\right] \neq \emptyset),\end{aligned} \tag{14}$$

where $\mathbb{I}(\cdot)$ is an indicator which is one if the condition between the brackets is true and zero otherwise; r_i^k is the start time of task ϕ_i^k which is also the time when the upstream tasks are finished. Then, we define a function f with respect to ν to denote the sum of bandwidth allocation fraction if there are multiple communication links during the communication time of ϕ_i^k, which is given by

$$f(\nu) = \rho_i^k + \sum_{k' \neq k} \sum_{i'=1}^{d_{k'}+b_{k'}} \rho_{i'}^{k'} \cdot \vartheta_{i'}^{k'}. \tag{15}$$

If the task is to be transmitted to another helper, the local queueing time is the minimum time waiting for the sum of bandwidth allocation fraction no more than 1 during the communication process; otherwise, the local queueing time will be 0 since there is no need for communication. Thus, the local queueing time is given by

$$\delta_1\left(x_i^k, \rho_i^k\right) = \min\left\{\arg_\nu f(\nu) \leq 1\right\} \cdot \mathbb{I}\left(x_i^k \neq x_{i_0}^k\right). \tag{16}$$

We define $q_{1,i}^k$ as the time when task ϕ_i^k finishes transmitting its input data, then

$$q_{1,i}^k = r_i^k + \delta_1\left(x_i^k, \rho_i^k\right) + c_i^k, \tag{17}$$

and use the union set of the other tasks' processing time in the helper x_i^k to represent the helper busy time caused by the other tasks, which is given by

$$\mathcal{O}_{i^-}^{k^-} = \bigcup_{\phi_{i'}^{k'} \neq \phi_i^k, x_{i'}^{k'} = x_i^k} \left[r_{i'}^{k'} + q_{i'}^{k'} + c_{i'}^{k'}, r_{i'}^{k'} + q_{i'}^{k'} + c_{i'}^{k'} + p_{i'}^{k'}\right]. \tag{18}$$

Then, the remote queueing time is given by

$$\delta_2(x_i^k, \rho_i^k) = \min\left\{\arg_\upsilon [q_{1,i}^k + \upsilon, q_{1,i}^k + \upsilon + p_i^k] \cap \mathcal{O}_{i^-}^{k^-} = \emptyset\right\}, \tag{19}$$

which is the minimum time waiting for the target helper to become vacant. Thus, the total queueing time of ϕ_i^k is given by

$$q_i^k = \delta_1\left(x_i^k, \rho_i^k\right) + \delta_2\left(x_i^k, \rho_i^k\right). \tag{20}$$

The response time of task ϕ_i^k is defined as the sum of queueing time, communication time, and processing time, and is given by

$$t_i^k = q_i^k + c_i^k + p_i^k, k \in \mathcal{K}, i \in \{1, \ldots, d_k\}. \tag{21}$$

Particularly, the last b_k transmission processes are only composed of the local queueing time and the communication time, and is given by

$$t_i^k = \delta_1\left(x_i^k, \rho_i^k\right) + c_i^k, k \in \mathcal{K}, i \in \{d_k + 1, \ldots, d_k + b_k\}. \tag{22}$$

F. Problem Formulation

Based on the response time of a single task, we need to calculate the total response time of a complex mission. Since some tasks be executed in parallel, we cannot directly add the response time of all tasks together. For example, in Fig. 2, task 2 and task 3 of mission (2) are executed in parallel; task 2, task 4, and task 5 of mission (3) are executed in parallel. We define the response time set of the tasks in parallel with ϕ_i^k as $\mathcal{T}_{i^-}^k$. If task ϕ_i^k does not have any tasks executed in parallel, $\mathcal{T}_{i^-}^k = \emptyset$. Denoting the offloading decisions of all the tasks as $\mathbf{x} = (x_1^1, \ldots, x_{d_1+b_1}^1, \ldots, x_1^K, \ldots, x_{d_K+b_K}^K)$ and $\mathbf{p} = (\rho_1^1, \ldots, \rho_{d_1+b_1}^1, \ldots, \rho_1^K, \ldots, \rho_{d_K+b_K}^K)$, the optimization problem of minimizing the average response time of all missions is given by $\mathbf{P1}$. The objective function (23) is to minimize the average response time of the missions in an episode, where each mission's response time equals to the latest completion time of its tasks. Constraint (24) indicates that the target helper can be any UAV in the current UC or the eNB, and Constraint (25) gives the possible bandwidth fractions. Constraint (26) gives the start time of each task based on the task inter-dependency, where t_k^G is the arrival time of mission m_k. Constraint (27) limits the total fraction of allocated bandwidth to be no more than 1. Constraint (28) gives the energy constraint, where the left hand side is the energy consumption of a UAV and the right hand side is the energy that the same UAV harvests in the whole episode.

$$\mathbf{P1}: \min_{\mathbf{x},\mathbf{p}} \quad \frac{1}{K}\sum_{k=1}^{K} \max\left\{\left(r_i^k + t_i^k - t_G^k\right) | i \in \mathcal{D}_k\right\} \tag{23}$$

$$\text{s.t.} \quad x_i^k = \mathcal{U}_t \cup \{N+1\}, \forall k \in \mathcal{K}, i \in \{1, \ldots, d_k\} \tag{24}$$

$$\rho_i^k = \{1, 2, \ldots, N_a\}, \forall k \in \mathcal{K}, i \in \mathcal{D}_k \tag{25}$$

$$r_1^k = t_k^G, r_i^k = r_{i_0}^k + \max\left\{\left(t_{i_0}^k, t_{i_0^-}^k\right) \middle| t_{i_0^-}^k \in \mathcal{T}_{i_0^-}^k\right\}, \quad \forall k \in \mathcal{K}, i \in \mathcal{D}_k, I_{i_0,i}^k = 1 \tag{26}$$

$$\rho_i^k + \sum_{k' \neq k} \sum_{i'=1}^{d_{k'}+b_{k'}} \rho_{i'}^{k'} \cdot \vartheta_{i'}^{k'} \leq 1, , \forall k \in \mathcal{K}, \forall i \in \mathcal{D}_k \tag{27}$$

$$\sum_{k=1}^{K}\sum_{i=1}^{d_k} E_u\left(x_i^k\right) \le \{\max\left(r_i^k + t_i^k - t_1^G\right) \mid k \in \mathcal{K}, i \in \mathcal{D}_k\} \cdot \varepsilon p_h, \forall u \in \mathcal{U} \tag{28}$$

There exist three main challenges to solve this problem. Firstly, there are unknown variables from the environment, such as the actual parameters in (2), which influence the communication time and queueing time. Secondly, the objective function represents the long term profit, which is highly related to the task-flow topologies. Thirdly, since the decision contains both the target helper selection and the bandwidth allocation, the solution space is large, and it is hard to separate the evaluation of each component in the integrated decision. To address these challenges, we propose approaches to learn the near-optimal policy for achieving the maximum long term return without prior knowledge of the environment.

IV. MARL-Based Computation Offloading Scheme

Since we consider that the complex computing missions arrive consecutively [3], the changing characteristics of computation and communication workload in a period have regularities. In this section, a proper offloading policy will be designed to adapt to the regularities.

A. Markov Decision Process Formulation

The problem of minimizing the average mission response time is essentially an MDP, since it has the non-aftereffect property [39]. An MDP can be defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R})$, where $\mathcal{S}$ is the set of possible system states, $\mathcal{A}$ is the set of possible actions, and $\mathcal{R}$ is the set of reward fed back by the environment. The entity observing the state and making the decision is defined as an agent, which is located at the eNB. We assume there exists an independent control channel between the eNB and the UC, thus some of the environment states, such as the mission status, channel condition, and the UAVs condition, can be acquired by the eNB through the UC's report. The MDP of the average mission response time minimization problem is formulated as follows.

1) States: The state at time t is defined as

$$s_t = (t, y, i, x_{i_0,1}^k, x_{i_0,2}^k, x_{i^-,1}^k, x_{i^-,2}^k, \alpha_i^k, \omega_i^k, h_1, \ldots, h_{N+1}, l_1, \ldots, l_N, o_{1,1}, o_{1,2} \ldots, o_{W,1}, o_{W,2}, e_1, \ldots, e_N, \mathcal{C}_1, \ldots, \mathcal{C}_{N+1}) \tag{29}$$

where

- t is the current time, and is used to help the agent learn a policy with adaption to the dynamic patterns of the UAVs' movement.
- y denotes the kind of task-flow model of the currently scheduled mission m_k, which consists of 3 binary numbers: (1,0,0) represents linear; (0,1,0) represents mesh; and (0,0,1) represents tree.
- i denotes the index of the currently scheduled task in m_k, and is one-hot encoded [40] with the dimension as the number of tasks in a mission.
- $x_{i_0,1}^k$ and $x_{i_0,2}^k$ denote the helpers executing the upstream tasks of ϕ_i^k, and if there is only one upstream task, these items are the same; $x_{i^-,1}^k$ and $x_{i^-,2}^k$ are reserved for the helpers executing the task in parallel with ϕ_i^k, and if there is no such task, they are set as zeros. All these items are one-hot encoded with the dimension as the maximum number of helpers.
- α_i^k and ω_i^k are with the same meaning as described in Section III.
- $(h_1, \ldots, h_{N+1})$ denote the remaining time for each helper to finish its current execution, which help to find a helper with short queueing time.
- $(l_1, \ldots, l_N)$ are the spatial distances between the eNB and the UAVs, which help to find UAVs with favorable channel states. When a UAV leaves the UC, the corresponding component will be set as a large value.
- $(o_{1,1}, o_{1,2} \ldots, o_{W,1}, o_{W,2})$ are the following W times bandwidth occupation condition's change after this step caused by the scheduled tasks, where $o_{w,1}, w \in [1, W]$ is the fraction of the occupied bandwidth and $o_{w,2}, w \in [1, W]$ is the corresponding duration.
- $(e_1, \ldots, e_N)$ denote the difference to the energy bound for each UAV at the current time, which are used to handle the energy constraint.
- $(\mathcal{C}_1, \ldots, \mathcal{C}_{N+1})$ are the computing capabilities of the available helpers. When a UAV leaves the UC, the corresponding component will be set as 0.

2) Actions: The action is the decision on the target helper selection and the bandwidth allocation for the currently scheduled task, and is given by

$$a_t = (a_{1,t}, a_{2,t}), a_{1,t} \in \{1, \ldots, N+1\}, a_{2,t} \in \{1, \ldots, N_a\}, \tag{30}$$

where N_a is the total number of channels. Notice that if the target helper is the task's current location, there is no communication, and the bandwidth decision will not be fed back to the environment in this situation. In addition, since in the last b_k steps, the computation results should be transmitted back to the eNB by default, there are only bandwidth decisions but no valid target helpers in these steps.

3) Reward: We set an indicator φ_i^k about whether there is a reward for task ϕ_i^k. Specifically, in Fig. 2, for all the tasks in mission (1), tasks 1, 2, 4, and 5 in mission (2), and tasks 1, 3, and 5 in mission (3), $\varphi_i^k = 1$; otherwise, $\varphi_i^k = 0$. If at time t, the scheduled task is ϕ_i^k, we define a pre-reward at time t as

$$\tilde{r}_t = \varphi_i^k \cdot \left\{\max\left[\left(r_i^k + t_i^k, r_{i^-}^k + t_{i^-}^k\right) \middle| t_{i^-}^k \in \mathcal{T}_{i^-}^k\right] - r_i^k\right\}. \tag{31}$$

Although there is no reward for some tasks, the agent can still evaluate the actions taken in these steps with the help of the next state value. This setting can guarantee the accumulated reward is relevant to the average response time of the missions, without recording the results from the tasks executed in parallel repeatedly. Thus, the optimization of (23) is just the optimization of the MDP. We define an action is infeasible if at least one of the three following conditions holds: 1. the energy constraint (28) is not satisfied; 2. constraint (24) for the target helper is not

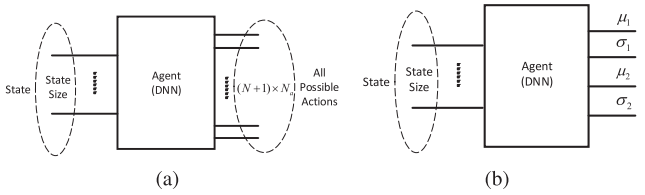


Fig. 3. Two kinds of output.

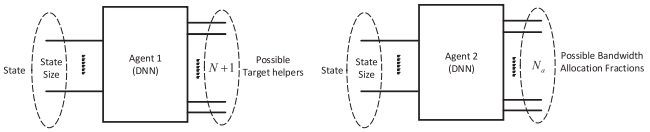


Fig. 4. Multi-agent(actors).

satisfied; or 3. the target helper leaves the UC when the task has not finished. Considering the scaling for training, the reward is given by

$$r_t = \begin{cases} (H_1 - \tilde{r}_t)/H_2, & \varphi_i^k = 1, \text{and } a_t \text{ is feasible} \\ -H_3, & a_t \text{ is infeasible} \\ 0, & \text{otherwise} \end{cases} \tag{32}$$

where H_1 is the baseline factor to control the reward with mean value about 0; H_2 is the regularization factor to make the reward approximately between -1 and 1; and H_3 is a large penalty if a_t is infeasible.

Since the environment is not totally known, the reward and the state transition cannot be acquired before the action is actually taken, and the MDP must be model-free [39].

B. Multi-Actors-Critic Algorithm With Credit Assignment

Deep reinforcement learning is a typical method to solve the model-free MDP problem, where the relationships between states and optimal actions and the evaluations of state and action pairs are estimated by deep neural networks (DNNs). However, there exist obstacles that prevent adopting common SARL in this system. If we use the SARL, there are two options of the agent's output: probabilities of all possible actions, or mean and variance of the action distribution, as shown in Fig. 3. Since the action is integrated, the number of possible actions is large, which will make the DNN too large and hard to train. On the other hand, the decision of the target helper is not a continuous variable. The helper indexes are nominal numbers representing different helpers, and thus using the Gaussian distribution as the distribution of action is also unreasonable. Therefore, these two options are both unsatisfactory.

In this paper, we consider using two independent agent DNNs to separate the integrated action, as shown in Fig. 4. The mainstream algorithms are based on the actor-critic (AC) framework [41], because the execution and evaluation in AC are already separated, and it is convenient to extend one actor to multiple actors. In the AC-based MARL, the actors are the agents, and there is a centralized critic to evaluate the $Q(s,a)$ as shown in Fig. 5. It should be noted that all the input items of DNNs must be normalized between 0 and 1. In the critic DNN, the input about action 1 should be one-hot encoded, thus the dimension is $N+1$; while action 2 is about the continuous

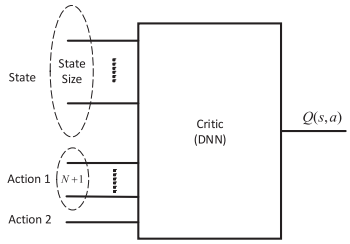


Fig. 5. Centralized critic.

bandwidth occupation, and there is no need to do the one-hot encoding for that.

When adopting MARL, there exists a challenge named credit assignment [41]–[43], concerning the reward assignment proportion to each dimension's action. The integrated action contains the decision of the target helper and the fraction of the bandwidth allocation, which cannot be taken in each dimension independently, thus the reward must be based on the integrated action, which cannot be divided to evaluate each dimension's action independently. For example, a task is allocated to u_4 and occupies $0.8B$ bandwidth, and the reward is poor. In this situation, we cannot figure out which dimension in the action mainly causes the poor reward: Maybe the target helper is proper but the occupied bandwidth is too much, resulting in a long local queueing time, or maybe the target helper has a long task queue, resulting in a long remote queueing time. This dilemma is the credit assignment problem.

The principle of solving the credit assignment problem can be simply explained as follows: When we are training actor 1, the action of actor 2 is fixed, thus we can estimate the credit of actor 1 by calculating the expectation of the critic's output which is only with respect to actor 1's action [41]. Next, we will introduce the mathematical derivation and give the pseudo-codes of the algorithm.

We parameterize the actor 1 and actor 2 DNNs as θ_1 and θ_2, respectively. The objective of RL is to learn a policy with the maximum expected accumulated reward in an episode, which is given by

$$\max_{\theta_1,\theta_2} \bar{R}_{\theta_1,\theta_2} = \sum_{\tau} R(\tau) p_{\theta_1,\theta_2}(\tau), \tag{33}$$

where τ denotes a state and action transformation trajectory as $\tau = \{s_1, a_1, s_2, a_2, \ldots, s_{t_{\max}}, a_{t_{\max}}\}$; $R(\tau)$ is the accumulated reward of the trajectory τ; $p_{\theta_1,\theta_2}(\tau)$ denotes the probability of the trajectory τ occurs under the actors parameterized by θ_1 and θ_2. The expected accumulated reward $\bar{R}_{\theta_1,\theta_2}$ is the expectation of the accumulated reward in all possible trajectories. The problem (33) is solved by the gradient ascent method, where the parameters θ_1 and θ_2 are adjusted gradually along with the direction of gradient ascent for achieving the local maximum. The gradient of $\bar{R}_{\theta_1,\theta_2}$ with respect to θ_1 and θ_2 is given by

$$\begin{aligned} &\nabla_{\theta_i} \bar{R}_{\theta_1,\theta_2} \\ &\quad = \nabla_{\theta_i} \left[\sum_{\tau} R(\tau) \cdot p_{\theta_1,\theta_2}(\tau) \right] \\ &\quad = \sum_{\tau} \left[R(\tau) \cdot \nabla_{\theta_i} p_{\theta_1,\theta_2}(\tau) \right] \end{aligned}$$

$$= \sum_{\tau} \left[R(\tau) \cdot p_{\theta_1,\theta_2}(\tau) \cdot \nabla_{\theta_i} \log(p_{\theta_1,\theta_2}(\tau)) \right]$$

$$= \frac{1}{N_0} \sum_{n=1}^{N_0} \left[R(\tau^n) \cdot \nabla_{\theta_i} \log\left(p_{\theta_1,\theta_2}(\tau^n)\right) \right]$$

$$= \frac{1}{N_0} \sum_{n=1}^{N_0} \sum_{t=1}^{T_n} \left[R(\tau_t^n) \nabla_{\theta_i} \log\left(p_{\theta_i}(a_{i,t}^n | s_t^n)\right) \right], i \in \{1,2\}. \tag{34}$$

Since it is hard to exhaust all possible trajectories, N_0 trajectories are sampled to estimate the expectation. To make decisions at time t, the accumulated reward of the whole episode will be replaced by the accumulated reward from this moment, i.e.,

$$R(\tau_t^n) = \sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n - b_t^n, \tag{35}$$

where the discount factor γ denotes the future fading of the influence about the action taken in this step, and the baseline b_t^n is used to differentiate the effect of the actions by setting the mean value as 0.

The summation on the right hand side of (35) is calculated with the help of the critic. The critic DNN parameterized by ω outputs the Q-value $Q^{\pi_{\theta_1,\theta_2}}(s_t^n, a_t^n)$, which means the accumulated reward after time t, i.e., $\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n$. To solve the credit assignment problem, we design different expressions of baseline b_t^n for actor 1 and actor 2 as $b_{i,t}^n, i \in \{1,2\}$, which is given by

$$b_{i,t}^n = \sum_{a_{i,t}^n} \pi_i\left(a_{i,t}^n | s_t^n\right) Q^{\pi_{\theta_1,\theta_2}}\left(s_t^n, \left(a_{i,t}^n, a_{i^-,t}^n\right)\right), i \in \{1,2\}, \tag{36}$$

where $\pi_i(a_{i,t}^n | s_t^n)$ denotes the probability of taking $a_{i,t}^n$ according to the current policy of actor i, and $a_{i^-,t}^n$ denotes the action of the other agent. This expression gives the expectation of the Q-value with respect to the action of actor i, while letting the other agent's action unchanged.

Thus, we can rewrite (35) as

$$R^i(\tau_t^n) = A^i(s_t^n, a_t^n) = Q^{\pi_{\theta_1,\theta_2}}(s_t^n, a_t^n) - \sum_{a_{i,t}^n} \pi_i\left(a_{i,t}^n | s_t^n\right) Q^{\pi_{\theta_1,\theta_2}}\left(s_t^n, \left(a_{i,t}^n, a_{i^-,t}^n\right)\right), i \in \{1,2\}, \tag{37}$$

which is called the advantage evaluation function. In this transformation, the agents have different baseline values. For actor i, (37) can evaluate the quality of $a_{i,t}^n$ objectively when the other one is already fixed, rather than evaluating the integrated action a_t^n.

According to the Bellman Equation [39], if s_t is not the terminal state, the critic output's target value is given by

$$y_t^n = r_t^n + E^{\pi}\left(Q^{\pi_{\theta_1,\theta_2}}\left(s_{t+1}^n, a_{t+1}^n\right)\right) = r_t^n + \sum_{a_{1,t+1}^n} \sum_{a_{2,t+1}^n} \pi_1\left(a_{1,t+1}^n | s_{t+1}^n\right) \pi_2\left(a_{2,t+1}^n | s_{t+1}^n\right) \cdot Q^{\pi_{\theta_1,\theta_2}}\left(s_{t+1}^n, \left(a_{1,t+1}^n, a_{2,t+1}^n\right)\right), \text{ if } s_t^n \text{is not the end state.} \tag{38}$$

Algorithm 1: Training of Multi-Actors-Critic Based Computation Offloading Decision Making.

Input: Missions information: α_i^k, β_i^k, ω_i^k, $\mathbf{I}^k$, t_k^G, K; Helpers information: eNB and UAVs positions in each slot, $\mathcal{C}(x_i^k)$, εp_h.

Output: Actor networks and critic network parameters θ_1, θ_2, and ω.

```
Randomly initialize the actors and critic networks
for episode = 1, ..., T do
    while the missions have not been completed do
        Observes the state from the environment and input it to its actor networks
        Selects action a_{i,t} ~ π(·|s_t, θ_i), i ∈ {1,2}
        Execute a_t, and observe the reward r_t and next state s_{t+1}
        Calculates the TD-error ΔQ^{π_{θ1,θ2}}(s_t^n, a_t^n) as (38), (39) and (40)
        Update the critic network as
        ω ← ω − α_2 ∇_ω (ΔQ^{π_{θ1,θ2}}(s_t, a_t))^2
        Calculates the advantage evaluation function A^i(s_t^n, a_t^n) as (37)
        Updates the actor networks as
        θ_i ← θ_i +
        α_1 ∇ log(p_{θ_i}(a_{i,t}|s_t)) A^i(s_t, a_t), i ∈ {1,2}
```

Otherwise, the target value equals to the reward directly, i.e.,

$$y_t^n = r_t^n, \text{if} s_t^n \text{ is the end state.} \tag{39}$$

The estimated value is $Q^{\pi_{\theta_1,\theta_2}}(s_t^n, a_t^n)$, and the temporal difference error (TD-error) is given by

$$\Delta Q^{\pi_{\theta_1,\theta_2}}(s_t^n, a_t^n) = y_t^n - Q^{\pi_{\theta_1,\theta_2}}(s_t^n, a_t^n). \tag{40}$$

The critic should let the TD-error minimized, thus the critic's parameters are updated based on the gradient descent method with the gradient as the TD-error.

The training process of the multi-actors-critic algorithm is as follows: Firstly, the eNB has the initial actors' policies π_{θ_1} and π_{θ_2}. The actors interact with the environment based on the policies, so there is a piece of empirical data. Secondly, the eNB uses this empirical data and the actors and critic networks to calculate the TD-error $\Delta Q^{\pi_{\theta_1,\theta_2}}$. The critic network updates the parameters by the gradient descent method with the gradient as the TD-error. Thirdly, the eNB calculates the advantage evaluation function $A^i(s_t^n, a_t^n)$ with the help of the critic network. The actor networks update parameters using the gradient ascend method, based on the advantage evaluation function. These three steps iterate until the actors' policies converge. The pseudo-code of our proposed multi-actors-critic (MAC) algorithm is given in Algorithm 1, where α_1 and α_2 are learning rates for the actors and the critic, respectively.

C. Off-Policy Multi-Actors-Critic Algorithm

The multi-actors-critic algorithm proposed in the previous subsection maintains the on-policy property of the classic actor-critic algorithm, which means the (s, a) pairs in the empirical data for training must have the same distribution of the current policies [20]. Therefore, in practice, if we want to update the

DNNs in each step (as shown in Algorithm 1), we can only use the empirical data of the previous one step, and then this data will be discarded. Otherwise, if we want to use more empirical data for training, we can collect N_0 trajectories as (34), and update the networks' parameters each N_0 episodes, which is extremely inefficient. Therefore, the training process of MAC has a low sampling efficiency. Each piece of empirical data can be used only once, and the agent must interact with the environment many times to collect data. In this paper's scenario, the training process is online, and the real computing missions are used for training as the cost. We hope the agents could acquire the proper policies by using as few episodes as possible to reduce the cost. In this subsection, we will extend MAC to the off-policy version to improve the sampling efficiency, which means the experience can be stored in a buffer and repeatedly used. We will first introduce why the classic actor-critic algorithm must be on-policy, and explain why MAC can be extended to off-policy. Then, the details of the off-policy multi-actors-critic algorithm will be given.

In the classic actor-critic algorithm with a single actor, the advantage evaluation function is given by

$$A^{\pi}\left(s_t, a_t\right) = r_t + \gamma V^{\pi}\left(s_{t+1}\right) - V^{\pi}\left(s_t\right). \tag{41}$$

The critic learns $V^{\pi}(s_t)$ directly, rather than $Q^{\pi}(s_t, a_t)$. In RL, the definition of $V^{\pi}(s_t)$ is the expected Q-value of the actions a_t in state s_t when a_t is distributed referring to the policy π, i.e., $E(Q^{\pi}(s_t, a_t))$, where $a_t \sim \pi(\cdot|s_t, \theta)$. However, if we store the empirical data in the experience buffer, and extract a batch of data for training, the condition $a_t \sim \pi(\cdot|s_t, \theta)$ does not hold, because the state and action is from the history, and the action is not generated based on the current policy π. Using the action and reward recorded in the history to learn $V^{\pi}(s_t)$ is unreasonable. Therefore, the algorithm must be on-policy, i.e., the action and reward information must be generated by the current policy. However, in order to solve the credit assignment problem, we have already changed the objective of learning $V^{\pi}(s_t)$ to learning $Q^{\pi}(s_t, a_t)$, as shown in the previous subsection. Using the arbitrarily distributed (s_t, a_t) pairs to learn $Q^{\pi}(s_t, a_t)$ is unbiased, as long as the next state and action pairs (s_{t+1}, a_{t+1}) is according to the policy π. The form of the empirical data is (s_t, a_t, r_t, s_{t+1}), which has no restrict to a_{t+1}. Therefore, the proposed algorithm can be extended to the off-policy version unbiasedly.

Given the batch size as M, the advantage evaluation functions for the actors and the TD-error for the critic are given by

$$A^{i}(s_t, a_t) = \frac{1}{M} \cdot \sum_{m=1}^{M} [Q^{\pi_{\theta_1}, \theta_2}(s_t^m, a_t^m) - \sum_{a_{i,t}^m} \pi_i(a_{i,t}^m | s_t^m) Q^{\pi_{\theta_1}, \theta_2}(s_t^m, (a_{i,t}^m, a_{i^-,t}^m))], i \in \{1, 2\}, \tag{42}$$

and

$$\Delta Q^{\pi_{\theta_1}, \theta_2}(s_t, a_t) = \frac{1}{M} \sum_{m=1}^{M} \left[y_t^m - Q^{\pi_{\theta_1}, \theta_2}\left(s_t^m, a_t^m\right)\right], \tag{43}$$

Algorithm 2: Training of Off-Policy Multi-Actors-Critic Based Computation Offloading Decision Making.

Input: Missions information: α_i^k, β_i^k, ω_i^k, $\mathbf{I}^k$, t_k^G, K; Helpers information: eNB and UAVs positions in each slot, $\mathcal{C}(x_i^k)$, εp_h.

Output: Actor networks and critic network parameters θ_1, θ_2, and ω.

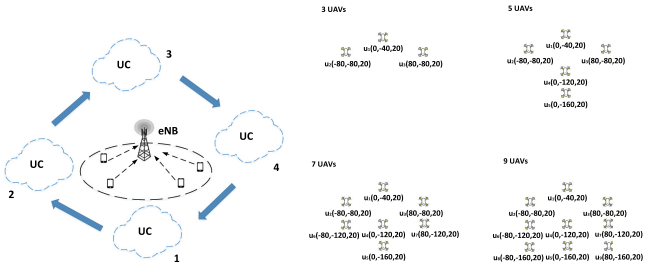
```
Randomly initialize the actors and critic networks
Initialize the experience pool
for episode = 1, ..., T do
    while the missions have not been completed do
        observe the state from the environment and input it to the actor networks
        select action a_{i,t} ~ π(·|s_t, θ_i), i ∈ {1, 2}
        execute a_t, and observe the reward r_t and next state s_{t+1}
        if the experience pool is full then
            discard the oldest experience
        Store (s_t, a_t, r_t, s_{t+1}) as an item in the experience pool
        if the number of items in the experience pool is no less than M then
            Extract M items from the experience pool
            calculate the TD-error ΔQ^{π_{θ1},θ2}(s_t^n, a_t^n) as (38), (39) and (43)
            update the critic network
            ω ← ω − α_2 ∇_ω (ΔQ^{π_{θ1},θ2}(s_t, a_t))^2
            calculate the advantage evaluation function A^i(s_t, a_t) as (42)
            update the actor networks
              θ_i ← θ_i +
              α_1 ∇ log(p_{θ_i}(a_{i,t}|s_t)) A^i(s_t, a_t), i ∈ {1, 2}
```

respectively. The pseudo-code of our proposed off-policy multi-actors-critic (OP-MAC) algorithm is given in Algorithm 2. The actors interact with the environment according to the current policies, and the empirical data (s_t, a_t, r_t, s_{t+1}) will be stored in an experience pool. In each step, a batch of data is extracted for training, and the DNNs are trained to fit the data in the selected batch.

Theoretically, OP-MAC needs fewer episodes for training compared with MAC, but needs more actual time in each training step. In OP-MAC, each training step makes the DNNs more suitable for a batch of data, thus OP-MAC takes effect earlier compared with MAC. On the other hand, the batch gradient descent method in OP-MAC needs more time for calculation, and it also consumes more storage to maintain the experience pool. Therefore, if the eNB can provide considerable computing power for the networks training and the time of each step's training is comparable to the time scale of tasks execution, we recommend OP-MAC; otherwise, MAC will run faster. For each training step, OP-MAC needs $(N + 1 + N_a)M$ times forward-propagation and 1 time back-propagation, while MAC needs $(N + 1 + N_a)$ times forward-propagation and 1 time back-propagation, thus denoting the number of neurons as N_l, the time complexities of OP-MAC and MAC are $\mathcal{O}(M(N + 1 + N_a)N_l)$ and $\mathcal{O}((N + 1 + N_a)N_l)$, respectively.

TABLE II
SYSTEM PARAMETERS

Parameter	Value
t_u	0.1s
$(a, b, \eta_{LoS}, \eta_{NLoS})$	$(9.61, 0.16, 2.3, 34)$
N	$[3, 5, 7, 9]$
$\mathcal{C}\left(x_i^k\right)$	$4 \sim 6$ GC/s
d_k	5
K	$[9, 10, 11, 12, 13, 14, 15]$
α_i^k	$1.8 \sim 7.8$MB
ω_i^k	$1000 \sim 2000$cycles/byte
P_u, P_e	200mW
B	10MHz
N_a	$[1, 3, 5, 7, 9]$
λ	$[4, 6, 8, 10, 12]\ t_u$
p_h	$[40, 60, 80, 100]$W
ε	0.02
η	10^{-10}J/cycle
W	8



(a) (b)

Fig. 6. Simulation Scenario: (a) UC trajectory; (b) Formation of UC.

V. SIMULATION RESULTS

A. System Parameters Setting

The system parameters used in our simulation are provided in Table II. For the convenience of simulation, we define a unit time slot t_u to approximately represent the time as an integer multiple of t_u. The value of t_u is set as 0.1 s. The environment-related parameters in the channel model are set referring to the high-rise urban scenario [11]. The trajectory of the UC and its formation is shown in Fig. 6. The UC movement trajectory is macroscopically pre-planned as a square, and the moving range is set according to the surveillance service referring to [44] and [45], as several hundred meters. We test the number of UAVs as 3, 5, 7, and 9, and the formation of UC is set as a cluster referring to [31]. Fig. 6(b) depicts the center points of the UAVs, and a UAV be randomly located in a range around the corresponding center point, which is caused by the collision avoidance requirement on the fly. In the simulation, the range around the centers is set as 5 meters. The UC moves with the velocity of 28 m/s. The height of UC is set as 20 m, considering that the eNB has the height about 50 m and the flying altitude of UAVs is about 70m [3]. When considering the UAVs' leaving, we adopt time-driven and position-driven leaving patterns in simulation. For time-driven leaving, the UAVs' leaving is subject to a Poisson process within a certain period of time; and for position-driven leaving, when a UAV is near its leaving point, it can leave the UC. The UAVs' leaving times and leaving points are unknown a priori to the eNB. Furthermore, to avoid that the number of UAVs in the UC is always decreasing,

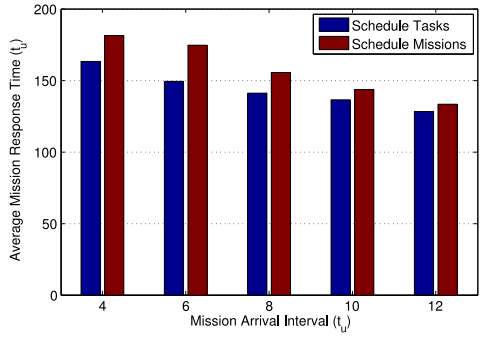


Fig. 7. Effect of task allocation. Tree topology missions. $N = 5$, and $p_h = 100\ W$. eNB is overloaded.

we assume that another UAV outside the UC will fly to the position of the leaving UAV after a random time. Referring to [28], the UAV edge server's computing capability is set from 4GC/s to 6GC/s (GC $= 10^9$ cycles). In the basic linear, mesh, and tree task-flow models, each mission consists of 5 tasks, as shown in Fig. 2. The number of missions in each episode is randomly generated between 9 and 15. We consider video-related tasks, and the input data size values are set as several Megabytes. Different tasks have different computation-to-data ratios, and ω_i^k is set between 1000 cycles/byte and 2000 cycles/byte [28]. Since the frequency band of the UC is different from that of the ground users, we set the bandwidth for communication as half of the total available bandwidth in LTE, i.e., 10 MHz. The numbers of channels are tested as $[1, 3, 5, 7, 9]$. The missions arrival interval λ is chosen between $4t_u$ and $12t_u$. The stratosphere balloon can radiate $40 \sim 100$ W power and the transmission efficiency is 0.02 [27]. As for the energy consumption, the parameters in the communication energy have been already introduced in Section III-D, and the computing energy consumption rate is 10^{-10} J/cycle [28].

B. Numerical Results

In this subsection, we first verify the rationality of some assumptions and considerations; then we compare the algorithms performance; and the stability regions of the system parameters are analyzed at last [46].

1) Why Task Allocation, Bandwidth Allocation and Energy Constraint are Meaningful?: Fig. 7 validates the advantage that we change the target helper in the granularity of a task rather than a whole mission. In the former case, there exist parallel execution between tasks in the same mission, and selecting the UAV nearer the eNB for executing the tasks with eNB-UAV or UAV-eNB communication requirements also leads to shorter communication time. Thus the average mission response time of scheduling tasks is shorter than that of scheduling missions.

Fig. 8 proves the advantage of bandwidth allocation and indicates the proper number of channels. We use the missions with a tree topology and $\lambda = 4$ as the highest requirement for parallelism. Notice that we use the sum of queueing time in a mission to analyze the buffer capacity cost, which be more than the average mission response time. It is because there exist parallel executions in the average mission response time calculation,

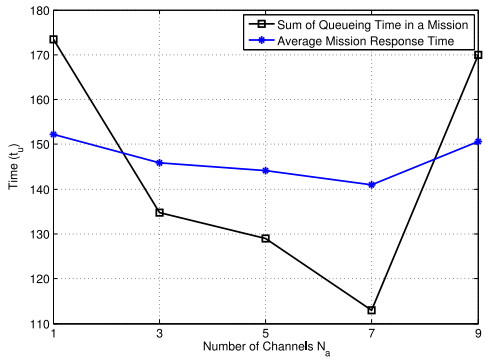


Fig. 8. Effect of bandwidth allocation. Tree topology missions. $N = 5$, $\lambda = 4t_u$, and $p_h = 100\ W$.

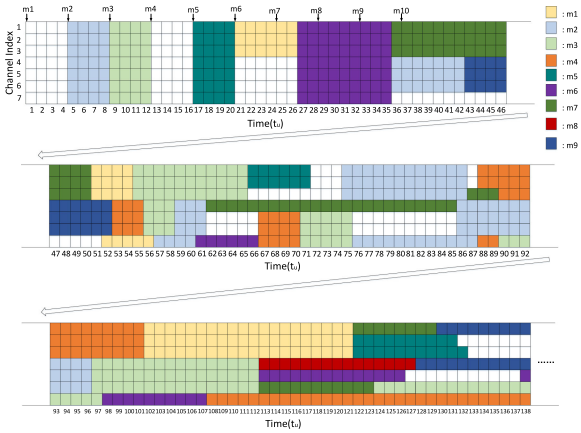


Fig. 9. Bandwidth allocation results in an episode with $K = 9$. Tree topology missions. $N = 5$, $\lambda = 4t_u$, and $p_h = 100\ W$.

but when analyzing the buffer cost, the queueing time of tasks are directly added. According to Fig. 8, when considering the inseparable bandwidth [24], i.e., $N_a = 1$, the tasks will suffer long queueing time since there is no parallel transmission, thus although the communication time is short due to the sufficient bandwidth, the overall mission response time is still longer. Meanwhile, the lack of parallel transmission causes long queues in the helpers, indicating high buffer cost in practice. When the number of channels is small, the granularity is not enough for the parallel transmission. On the other hand, the performance does not always improve as N_a increases, and a large value of N_a also implies high complexity and difficulty in training. When $N_a = 7$, the average mission response time is at the minimum, and the average queue length in the helpers is also the shortest. Thus we set $N_a = 7$ in the following analysis, and each channel represents 8 RBs in the frequency domain, i.e., 1.44 MHz. The result of $N_a = 7$ is based on the system parameters as described in the previous subsection, which provides a reference for the practical use.

Fig. 9 shows the bandwidth allocation results in an episode. Since the eNB is idle at first, some tasks can be directly processed at the eNB, and the bandwidth is vacant at that time. When the eNB is increasingly busy, more tasks will be offloaded to the UC. At first, the transmission workload is not heavy, thus all

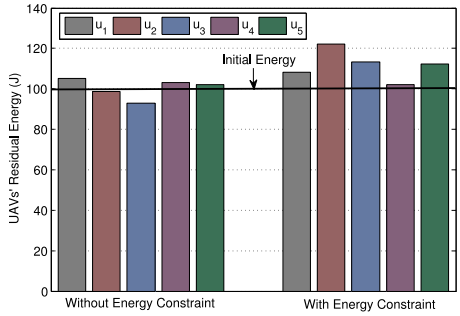


Fig. 10. Effect of energy constraint. Linear topology missions, $N = 5$, $\lambda = 10t_u$, $K = 10$, and $p_h = 70\ W$.

the transmissions occupy the total bandwidth. Afterwards, with the increasing demand for task offloading, the bandwidth is divided for parallel transmission.

Fig. 10 shows the effect of energy constraint. If we set p_h to be relatively small and do not consider the energy constraint, some UAVs' residual energy will be less than the initial energy. We set the 5 UAVs' computing capability as $[4, 6, 4, 5, 5]$ GC/s. For convenient observations, in this part, we let the missions only arrive in the time when the UC moves from location 1 to location 3 in Fig. 6(a). During the movement, u_3 is selected as the entrance UAV most of the time, and u_2 has the highest computing capability. Therefore, u_3 and u_2 are used more, ending with the energy level lower than the initial energy. When the energy constraint is considered, some delay performance is sacrificed for the energy benefit: the workload of u_3 and u_2 is not that heavy, and all the values of the UAVs' residual energy are more than the initial energy. Meanwhile, since the offloading decisions are not as good as those without energy constraint, the total response time of the whole episode is longer, which means the time of energy supply is longer. Therefore, the average level of the UAVs' residual energy is higher than that without energy constraint.

2) Performance Comparisons and Analysis: In this part, we will evaluate the performance of the proposed MAC and OP-MAC algorithms in different settings with the comparisons of the single-agent actor-critic (AC) [24] and the greedy-based method. The actor DNN in AC is designed as Fig. 3(b), since there are too many possible combinations of actions to implement Fig. 3(a)'s structure when the number of helpers is large. The greedy-based method can be described as follows: The eNB first checks the task queue of each UAV and itself, and will choose the helper with the shortest task queue; if there are more than one helper having the shortest task queue, the eNB will choose the helper that has both the shortest task queue and highest computing capability.

Fig. 11 shows the convergence processes of the proposed OP-MAC and MAC for handling the missions with linear topology. Fig. 11 shows that the basic single-agent AC cannot handle the integrated action, and it fails to acquire a satisfactory policy, while the proposed MARL-based approaches perform better than the basic AC algorithm. Comparing OP-MAC and MAC, OP-MAC takes effect much earlier than MAC, and can acquire a result better than MAC.

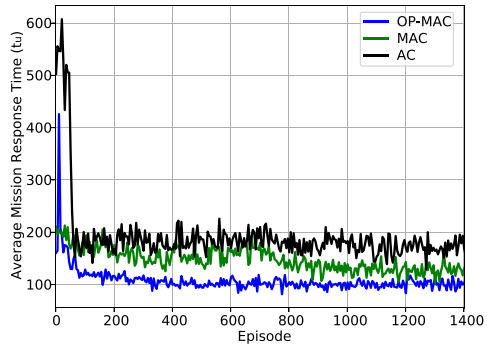


Fig. 11. Convergence performance of our proposed algorithms for handling the missions with linear topology. $N = 5$, $\lambda = 10t_u$, and $p_h = 100\ W$.

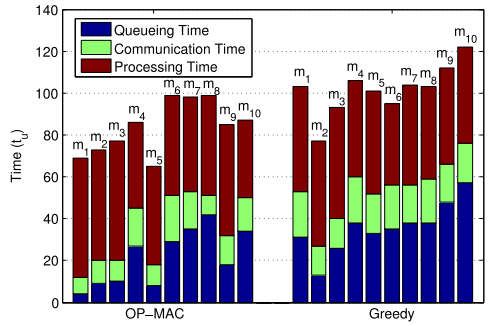


Fig. 12. Overall queueing, communication, and processing time of the missions with linear topology. $N = 5$, $\lambda = 10t_u$, and $p_h = 100\ W$.

Fig. 12 shows the overall queueing, communication, and processing time of missions with linear topology. This figure demonstrates that the proposed algorithm can choose the proper UAVs as the entrance and exit to achieve shorter communication time. As for the queueing time, the OP-MAC's results are also shorter than those of the greedy-based method. The greedy-based method only makes the short-sighted choices based on the current status, but does not consider the possible upcoming tasks. Specifically, a task be offloaded to a helper with a temporary shorter queue for its own shorter queueing time, but it cause more communication demands, which will occupy the bandwidth to make the other tasks suffer long queueing time waiting for the channels available. Therefore, although the original idea of the greedy-based method is to achieve a shorter queueing time, it actually increases the overall queueing time.

Fig. 13 shows the performance of the missions with basic task-flow topologies when changing the mission arrival interval λ, energy supply power p_h, the number of UAVs N, and the number of leaving UAVs.

The results in Fig. 13(a)–(c) show that the average mission response time is negatively correlated with λ. When the missions are generated more intensively, the response time is longer, which is logical. In all λ values, the results of OP-MAC and MAC are better than the results of the greedy-based method. Comparing the results of Fig. 13(a)–(c), it can be observed that the missions with tree topology experience the longest response time. This is because in each tree topology mission, there have 3 tasks that need to transmit back the output data, and there also exist multiple transmissions of the input data at the same time (task 2 and task 3; task 4 and task 5), which lead to bandwidth shortage and long queueing time.

In Fig. 13(d)–(f), for the missions with a linear topology, the response time increases when the energy constraint becomes tight. It is because when the energy is sufficient, only the UAVs with good channel states are used; and when the energy supply power is small, some UAVs with poor channel states or low computing capabilities have to be requisitioned for balancing the energy consumption. On the other hand, for the tree topology missions with high parallelism, the energy constraint does not influence the performance a lot, since a relatively large number of UAVs are already used to support the parallel execution. There exist some points where the results of MAC are worse than those of greedy-based method, while the results of OP-MAC are much better. It is because in MAC, the DNNs parameters are easy to be rapidly changed when facing the large penalty, while OP-MAC can comprehensively analyze the situations with and without the penalty in each training step for making better decisions thanks to the experience replay. Therefore, OP-MAC should be used when the energy constraint is tight.

Fig. 13(g)–(i) show that when the number of UAVs is not large, more UAVs can bring more computing resources, which is beneficial. The average mission response time values of OP-MAC and MAC decrease when the number of UAVs is from 3 to 7. When the number of UAVs is up to 9, according to Fig. 6(b), the newly added UAVs are far from the eNB, which have unsatisfactory channel states, acting as a temptation of other local optimum solutions, so the results rise a little. This phenomenon reflects the local-optimality of the solution. The approaches cannot guarantee to converge to the global optimum. Even though, the results of the greedy-based method rise more, which means it achieves a worse local optimum compared with our proposed approaches. Generally, in almost all cases, OP-MAC and MAC outperform the single-agent AC and greedy-based method.

Fig. 13(j)–(l) show the influence of the number of leaving UAVs. According to these figures, the UAVs' leaving can degrade the performance, and larger numbers of leaving UAVs lead to more degradation. Since the leaving patterns are unknown a priori, in the greedy-based method, no task can be offloaded to the UAVs which are about to leave; while the agent can gradually acquire the UAVs' leaving regularities by performing the proposed approaches, and tasks can still be offloaded to the UAVs when the current time is much earlier than the leaving time or the current position is far from the leaving point. OP-MAC always performs best, and MAC outperforms the greedy-based method most of the time. However, when considering the tree topology missions and the number of leaving UAVs is large, MAC does not perform well. It is because the training of the adaption to dynamic patterns is based on the penalty when actions are infeasible as shown in (32), and MAC is not good at handling the sudden penalty. This phenomenon is consistent with the observations in Fig. 13(d)–(f), thus OP-MAC should be used in this situation. Furthermore, notice that although we set the leaving and joining of UAVs subject to some specific patterns in simulation, our proposed approaches can also handle the situation where UAVs leave the UC with no regularity. The

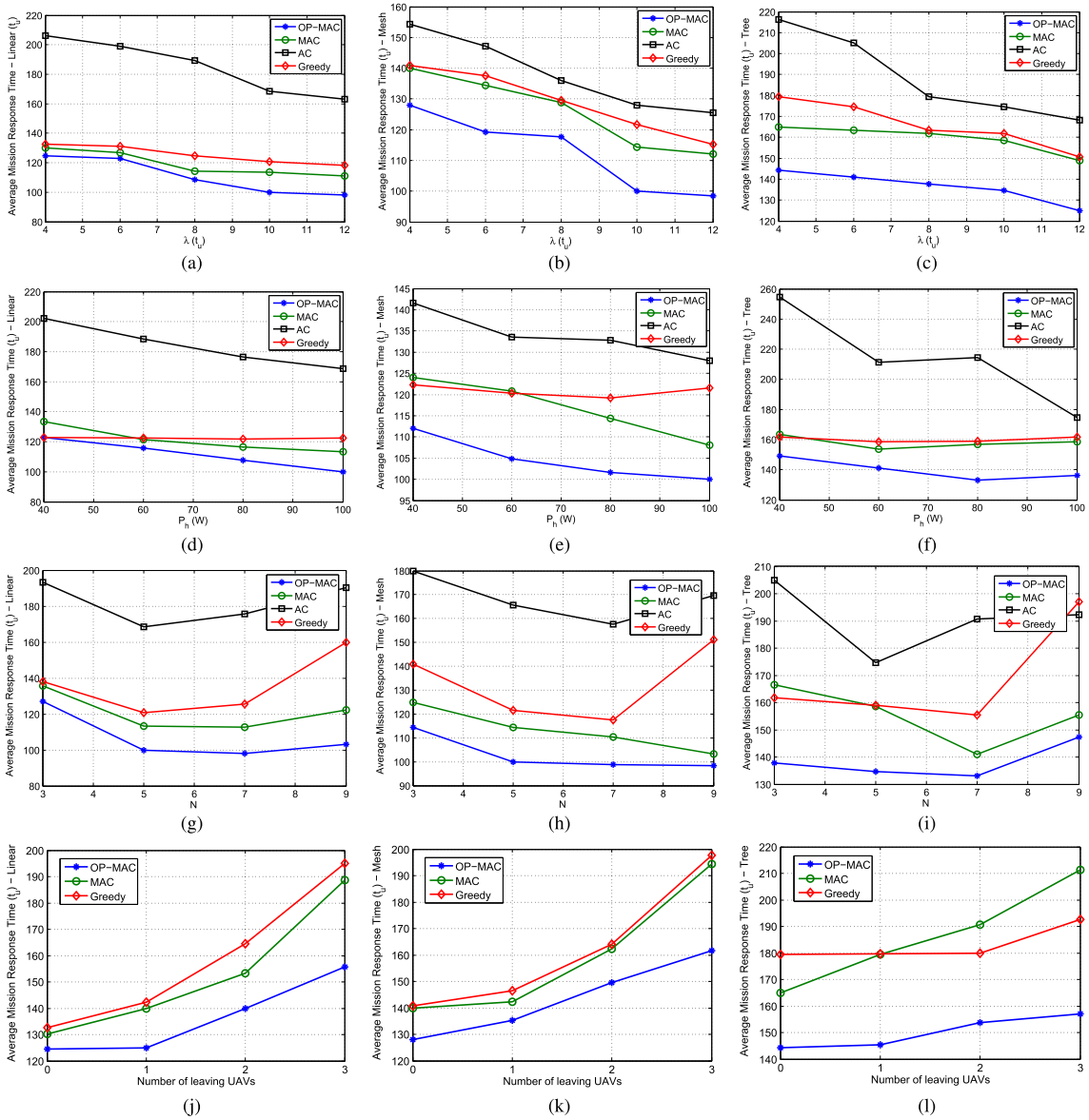


Fig. 13. MARL-based approaches evaluation under different environment settings: (a)–(c) With different missions arrival intervals; (d)–(f) With different power transmitted by the balloon; (g)–(i) With different numbers of UAVs; (j)–(l) With different numbers of leaving UAVs (The results of AC are not shown in (j)–(l) since they are too bad).

situation is even simpler for training the agent, since the agent will just do not offload any tasks to these UAVs, and there is no accurate information such as the leaving time should be learned.

Fig. 14 shows that our proposed approaches perform well in all possible combinations of the basic task-flows considered in this paper. No matter what kinds of task-flows the missions have, the agents can learn proper policies to adapt to it, as long as the missions arrive repeatedly with a specific pattern.

3) Parameters Stability Region Analysis: The methods proposed in this paper are based on the scenario where the changes of task features with time in each episode follow some certain regularities, and there should not be any unfinished tasks at the beginning of each episode. We consider an on-off model for mission arrivals: the missions arrive with the fixed interval $\lambda = 10t_u$ at the "on" stage, and no missions arrive at the "off"

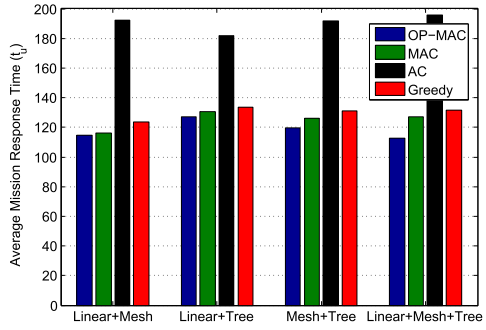


Fig. 14. Average mission response time for missions with different task-flow models combinations. $\lambda = 10t_u$, $p_h = 100\ W$. In combination of 2 kinds of models, each model's missions occupy 1/2 of the missions in an episode; in the combination of 3 kinds of models, each model's missions occupy 1/3 of the missions in an episode.

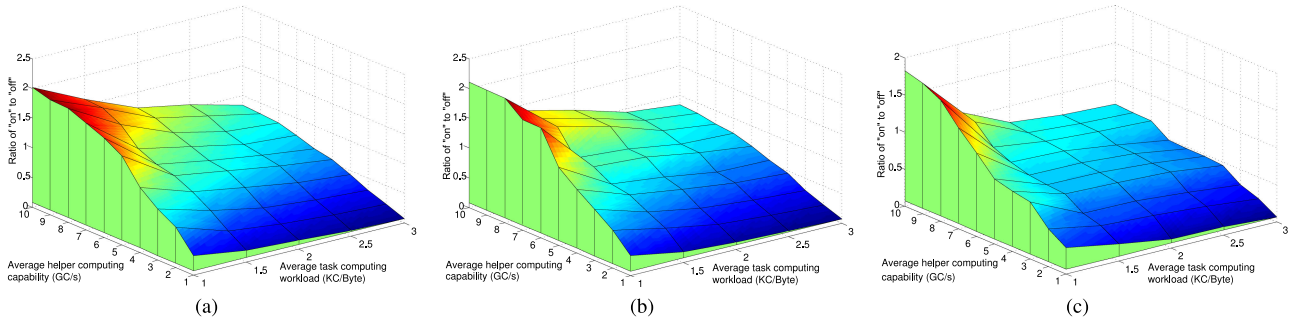


Fig. 15. Parameters stability regions: (a) For linear topology missions; (b) For mesh topology missions; (c) For tree topology missions.

stage. The allowable range of the "on" to "off" ratio is related to the tasks' computing workloads and the helpers' computing capabilities.

Fig. 15 shows that the maximum allowable ratio of "on" to "off" increases with the average helper computing capability and decreases with the average task computing workload; and the slopes of both increasing and decreasing processes decrease with the parameters, which means when the parameters' values are small, the changes will have more obvious impacts on the performance. The results of the tree topology missions are generally worse than those of the linear topology missions, since the tree topology missions have more communication requirements, leading to bandwidth shortage and long queueing time. This phenomenon is consistent with the observations in Fig. 13(a)–(c).

The results in this part can provide guidance for the selection of the on-board CPUs/GPUs and the upper layer's flow control to guarantee the mission response time to be finite.

VI. Conclusion and Future Work

In this paper, we have investigated the problem of providing the shortest average mission response time in the UAVs-assisted edge computing scenario. We have proposed a computation offloading scheme for determining the proper target helper and bandwidth allocation. The response time minimization problem has been formulated as an MDP, with well-designed expressions of state, action, and reward. framework and the modified advantage evaluation functions have been employed to improve the learning performance. Simulation results have validated the convergence and efficiency of the proposed approaches. Our proposed approaches can significantly reduce the response time of complex missions, and provide useful guidelines for the design of UAVs-assisted edge computing. In the future, we will consider the scenario of multiple BSs, and study partial cooperation mechanisms.

References

[1] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the edge: A scalable IoT architecture based on transparent computing," *IEEE Netw.*, vol. 31, no. 5, pp. 96–105, 2017.

[2] F. Sun *et al.*, "Cooperative task scheduling for computation offloading in vehicular cloud," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11049–110 61, Nov. 2018.

[3] N. H. Motlagh, T. Taleb, and O. Arouk, "Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 899–922, Dec. 2016.

[4] N. Sharma, M. Magarini, D. N. K. Jayakody, V. Sharma, and J. Li, "On-demand ultra-dense cloud drone networks: Opportunities, challenges and benefits," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 85–91, Aug. 2018.

[5] S. Hayat, E. Yanmaz, and R. Muzaffar, "Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2624–2661, Oct.–Dec. 2016.

[6] S. W. Loke, "The internet of flying-things: Opportunities and challenges with airborne fog computing and mobile cloud in the clouds," *Dept. Comput. Sci. Inf. Technol.*, La Trobe Univ., Melbourne VIC, Australia, Jul. 2015, Accessed: Jun. 4, 2016. [Online]. Available: https://arxiv.org/abs/1507.04492.

[7] N. H. Motlagh, M. Bagaa, and T. Taleb, "UAV selection for a UAV-based integrative IoT platform," in *Proc. IEEE Global Commun. Conf.*, Dec. 2016, pp. 1–6.

[8] N. H. Motlagh, M. Bagaa, and T. Taleb, "Energy and delay aware task assignment mechanism for UAV-based IoT platform," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6523–6536, Aug. 2019.

[9] 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.

[10] T. Zhang, Y. Xu, J. Loo, D. Yang, and L. Xiao, "Joint computation and communication design for UAV-assisted mobile edge computing in IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5505–5516, Aug. 2020.

[11] R. Amorim, H. Nguyen, P. Mogensen, I. Z. Kovács, J. Wigard, and T. B. Sørensen, "Radio channel modeling for UAV communication over cellular networks," *IEEE Wirel. Commun. Lett.*, vol. 6, no. 4, pp. 514–517, Aug. 2017.

[12] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 301–313, Apr. 2019.

[13] Z. Wang, Z. Zhong, D. Zhao, and M. Ni, "Vehicle-based cloudlet relaying for mobile computation offloading," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11181–11191, Nov. 2018.

[14] M. Mozaffari, W. Saad, M. Bennis, Y. Nam, and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2334–2360, Jul.–Sept. 2019.

[15] B. Li, Z. Fei, and Y. Zhang, "UAV communications for 5G and beyond: Recent advances and future trends," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2241–2263, Apr. 2019.

[16] J. Zhang *et al.*, "Stochastic computation offloading and trajectory scheduling for UAV-assisted mobile edge computing," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3688–3699, Apr. 2019.

[17] Y. Liu, K. Xiong, Q. Ni, P. Fan, and K. B. Letaief, "UAV-assisted wireless powered cooperative mobile edge computing: Joint offloading, CPU control and trajectory optimization," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2777–2790, Apr. 2020.

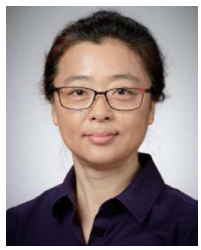
[18] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 1927–1941, Sep. 2018.

[19] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, Jan. 2016.

[20] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2016, pp. 1928–1937.
[21] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Performance optimization in mobile-edge computing via deep reinforcement learning," in *Proc. IEEE VTC-Fall*, Aug. 2018, pp. 1–6.
[22] H. Ye, G. Y. Li, and B. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3163–3173, Apr. 2019.
[23] F. Sun, N. Cheng, S. Zhang, H. Zhou, L. Gui, and X. Shen, "Reinforcement learning based computation migration for vehicular cloud computing," in *Proc. IEEE GLOBECOM*, Dec. 2018, pp. 1–6.
[24] S. Zhu, L. Gui, N. Cheng, Q. Zhang, F. Sun, and X. Lang, "UAV-enabled computation migration for complex missions: A reinforcement learning approach," *IET Commun.*, vol. 14, no. 15, pp. 2472–2480, 2020.
[25] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.
[26] J. Huang, Y. Zhou, Z. Ning, and H. Gharavi, "Wireless power transfer and energy harvesting: Current status and future prospects," *IEEE Wirel. Commun.*, vol. 26, no. 4, pp. 163–169, Aug. 2019.
[27] Y. Huo, X. Dong, T. Lu, W. Xu, and M. Yuen, "Distributed and multilayer UAV networks for next-generation wireless communication and power transfer: A feasibility study," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 7103–7115, Aug. 2019.
[28] N. Cheng *et al.*, "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.
[29] W. Shi *et al.*, "Multiple drone-cell deployment analyses and optimization in drone assisted radio access networks," *IEEE Access*, vol. 6, pp. 12518–12529, 2018.
[30] "Evolved Universal Terrestrial Radio Access (E-UTRA), Physical Layer Procedures, 3GPP TS 36.213 V10.6.0, 2012."
[31] Y. Ben-Asher, S. Feldman, P. Gurfil, and M. Feldman, "Distributed decision and control for cooperative UAVs using ad hoc communication," *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 3, pp. 511–516, May 2008.
[32] A. A. Khuwaja, Y. Chen, N. Zhao, M. Alouini, and P. Dobbins, "A survey of channel modeling for UAV communications," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 2804–2821, Oct.–Dec. 2018.
[33] I. Bor-Yaliniz and H. Yanikomeroglu, "The new frontier in ran heterogeneity: Multi-tier drone-cells," *IEEE Commun. Mag.*, vol. 54, no. 11, pp. 48–55, 2016.
[34] S. Sekander, H. Tabassum, and E. Hossain, "Multi-tier drone architecture for 5g/b5g cellular networks: Challenges, trends, and prospects," *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 96–103, Mar. 2018.
[35] M. Mozaffari, W. Saad, M. Bennis, Y. Nam, and M. Debbah, "A tutorial on uavs for wireless networks: Applications, challenges, and open problems," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2334–2360, Jul.–Sept. 2019.
[36] J. Huang, Y. Zhou, Z. Ning, and H. Gharavi, "Wireless power transfer and energy harvesting: Current status and future prospects," *IEEE Wirel. Commun.*, vol. 26, no. 4, pp. 163–169, Aug. 2019.
[37] B. Galkin, J. Kibilda, and L. A. DaSilva, "UAVs as mobile infrastructure: Addressing battery lifetime," *IEEE Commun. Mag.*, vol. 57, no. 6, pp. 132–137, Jun. 2019.
[38] O. Muñoz, A. Pascual-Iserte, and J. Vidal, "Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading," *IEEE Trans. Veh. Technol.*, vol. 64, no. 10, pp. 4738–4755, Oct. 2015.
[39] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, Sept. 1998.
[40] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean Field Multi-Agent Reinforcement Learning," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2018, pp. 1–10.
[41] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson., "Counterfactual multi-agent policy gradients," in *Proc. Int. Conf. Assoc. Adv. Artif. Intell.*, May 2017, pp. 2974–2983.
[42] Y. H. Chang, T. Ho, and L. P. Kaelbling, "All learning is local: Multi-agent learning in global reward games," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Dec. 2003, pp. 807–814.
[43] K. Tumer and A. K. Agogino, "Distributed agent-based air traffic flow management," in *Proc. Int. Joint Conf. Auton. Agents Multiagent Syst.*, May 2007, pp. 1–8.
[44] D. Kingston, R. W. Beard, and R. S. Holt, "Decentralized perimeter surveillance using a team of UAVs," *IEEE Trans. Rob.*, vol. 24, no. 6, pp. 1394–1404, Dec. 2008.
[45] H. Kim and J. Ben-Othman, "A collision-free surveillance system using smart UAVs in multi domain IoT," *IEEE Commun. Lett.*, vol. 22, no. 12, pp. 2587–2590, Dec. 2018.
[46] L. Tassiulas and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Trans. Inf. Theory*, vol. 39, no. 2, pp. 466–478, Mar. 1993.

Shichao Zhu received the B.Eng. degree from the School of Aeronautics, Northwestern Polytechnical University, Xi'an, China, in 2016. He is currently working toward the Ph.D. degree with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China. His current research interests include data and computation offloading in UAV-enabled heterogeneous networks, and the application of AI for wireless networks.

Lin Gui (Member, IEEE) received the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2002. Since 2002, she has been with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China, where she is currently a Professor. She is also the Vice Dean of Graduate School, Shanghai Jiao Tong University. Her current research interests include wireless communications, space–air integrated networks, and network resource management.



Dongmei Zhao (Member, IEEE) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. She joined the Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON, Canada, where she is currently a Full Professor. From April 2004 to March 2009, she was an Adjunct Assistant Professor with the Department of Electrical and Computer Engineering, University of Waterloo. She is an Associate Editor for the IEEE INTERNET OF THINGS JOURNAL. Between 2007 and 2017, she was an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. She was also an Editor of EURASIP *Journal on Wireless Communications and Networking* and *Journal of Communications and Networks*. She was the Co-Chair of the Wireless Networking Symposium in the IEEE GLOBECOM Conference 2007, the Technical Program Committee for the IEEE International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD) 2016, the General Symposium of the International Wireless Communications and Mobile Computing (IWCMC) Conference 2007, and the Vehicular Networks Symposium of the IWCMC from 2012 to 2018. She has been on the Technical Program Committee of many international conferences in her fields. She is a Professional Engineer of Ontario. Her current research interests include mobile computation offloading, energy-efficient wireless networking, and vehicular communication networks.

Nan Cheng (Member, IEEE) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, in 2016, and the B.E. and M.S. degrees from the Department of Electronics and Information Engineering, Tongji University, Shanghai, China, in 2009 and 2012, respectively. He is currently a Professor with the School of Telecommunication Engineering, Xidian University, Shaanxi, China. From 2017 to 2018, he was a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada. His current research interests include space–air–ground integrated system, big data in vehicular networks, and self-driving system, performance analysis, MAC, opportunistic communication, application of AI for vehicular networks, and space–air–ground integrated networks.

Qi Zhang received the B.Eng. degree from the School of Electronics and Information, Northwestern Polytechnical University, Xi'an, China, in 2015. He is currently working toward the Ph.D. degree with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests include mobile edge computing, resource management, and optimization.

Xiupu Lang received the B.Eng. degree from the School of Electronics and Information, Harbin Institute of Technology, Harbin, China, in 2018. He is currently working toward the Ph.D. degree with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests include satellite networks and SDN protocol design.