MARS: a DRL-based Multi-task Resource Scheduling Framework for UAV with IRS-assisted Mobile Edge Computing System

Feibo Jiang, *Member, IEEE*, Yubo Peng, Kezhi Wang, *Senior Member, IEEE*, Li Dong, Kun Yang, *Fellow, IEEE*

***Abstract*—This paper investigates a dynamic Mobile Edge Computing (MEC) system enhanced by Unmanned Aerial Vehicles (UAVs) and Intelligent Reflective Surfaces (IRSs). We introduce a scalable resource scheduling algorithm aimed at reducing energy consumption for all User Equipments (UEs) and UAVs within the MEC system, accommodating a varying number of UAVs. To address this challenge, we present a Multi-tAsk Resource Scheduling (MARS) framework that employs Deep Reinforcement Learning (DRL). Firstly, we present a novel Advantage Actor-Critic (A2C) structure with the state-value critic and entropy-enhanced actor to reduce variance and enhance the policy search of DRL. Then, we present a multi-head agent with three different heads in which a classification head is applied to make offloading decisions and a regression head is presented to allocate computational resource, and a critic head is introduced to estimate the state value of the selected action. Next, we introduce a multi-task controller to adjust the agent to adapt to the varying number of UAVs by loading or unloading a part of the weights in the agent. Finally, a Light Wolf Search (LWS) is introduced as the action refinement to enhance the exploration in the dynamic action space. Numerical results demonstrate the feasibility and efficiency of the MARS framework.**

***Index Terms*—Mobile edge computing (MEC), intelligent reflecting surface (IRS), unmanned aerial vehicle (UAV), deep reinforcement learning (DRL), resource scheduling.**

I. Introduction

Over the past few years, computationally intensive and low-latency applications have experienced significant growth. These applications demand short response time and substantial computational resource, and encompass areas such as autonomous driving, mixed reality, and the metaverse. This poses a significant challenge for User Equipment (UE) to process these tasks locally. Mobile Edge Computing (MEC) [1] has revolutionized the way that computing tasks are handled on mobile devices, allowing UEs to offload computationally intensive tasks to nearby servers. However, deploying stationary MEC servers on the ground may not be cost-effective, especially in temporary, unexpected, or emergency situations [2]. Hence, researchers have proposed an innovative system: Unmanned Aerial Vehicle (UAV)-assisted MEC. This approach enables the delivery of flexible and efficient computation and communication services to the ground UEs [3]. Despite these benefits, physical signal obstructions and interference from tall buildings and other obstacles can cause frequent disruptions in wireless communication channels between UAVs and UEs.

Intelligent Reflecting Surface (IRS) technology has emerged as a leading research and development focus in wireless communication. By modifying the amplitude and phase of incident signals through a number of affordable passive reflecting elements, IRS can enhance network performance within the propagation environment [4]. Notably, IRS consumes considerably less energy than alternative approaches like active relays. In UAV-assisted MEC systems, IRS can be deployed on building facades, enabling efficient signal reflection between UAVs and UEs even when faced with connection obstructions.

Although UAV and IRS-assisted MEC systems present numerous benefits, they still face challenges that should be addressed: (1) Resource allocation generally involves continuous variables, while offloading decisions comprise integer variables, leading to a Mixed-Integer Nonlinear Programming (MINLP) problem [5]. Such problems are difficult to manage using traditional techniques. (2) The wireless environment becomes increasingly complex as the number of UEs and UAVs varies, particularly when incorporating IRS elements in UAV-assisted MEC systems.

As a result, we propose a dynamic UAV with IRS-assisted MEC system. We introduce a Multi-tAsk Resource Scheduling (MARS) framework based on deep reinforcement learning (DRL) to minimize energy consumption for both UEs and UAVs, considering a variable number of UAVs in the system. The MARS framework achieves online resource scheduling by jointly optimizing UAV location, offloading decisions, resource allocation, and a phase-shifted diagonal matrix. The key contributions of the study are summarized as follows:

- *A2C structure*: We introduce a novel Advantage Actor-Critic (A2C) structure with the state-value critic and entropy-enhanced actor. In the critic network, we use advantage value to reduce variance in the learning process of DRL. In the actor network, we apply policy entropy to enhance the policy learning of DRL.

This work was supported in part by the National Natural Science Foundation of China under Grant nos. 41604117, 41904127.

Feibo Jiang (jiangfb@hunnu.edu.cn) is with Hunan Provincial Key Laboratory of Intelligent Computing and Language Information Processing, Hunan Normal University, Changsha, China.

Yubo Peng (pengyubo@hunnu.edu.cn) is with School of Information Science and Engineering, Hunan Normal University, Changsha, China.

Kezhi Wang (Kezhi.Wang@brunel.ac.uk) is with the Department of Computer Science, Brunel University London, UK.

Li Dong (Dlj2017@hunnu.edu.cn) is with Changsha Social Laboratory of Artificial Intelligence, Hunan University of Technology and Business, Changsha, China.

Kun Yang (kunyang@essex.ac.uk) is with the School of Computer Science and Electronic Engineering, University of Essex, Colchester, CO4 3SQ, U.K., also with Changchun Institute of Technology.

- *Multi-head agent*: We design a multi-head agent with three output heads, in which a classification head is applied to make offloading decisions and a regression head is presented to allocate computational resource in the actor network, and a critic head is introduced to estimate the state value of the selected action in the critic network.
- *Multi-task controller*: We present a multi-task controller to adjust the agent for adapting to the varying number of UAVs. In the multi-task controller, we perform pruning and network retraining to sequentially pack the resource scheduling knowledge of multiple UAVs into the multi-head agent. The agent can reuse the stored resource scheduling knowledge by loading or unloading a part of weights when the number of UAVs is changing without retraining the agent.
- *LWS refinement*: We incorporate the action refinement into the proposed A2C structure to improve the exploration and accelerate the search for the best policy in the dynamic action space. A Light Wolf Search (LWS) is introduced as the action refinement module and a light wolf driven by the channel gains is applied to guide the wolf pack for enhancing the global policy search.

The remainder parts are as follows: Some related works are reviewed in Section II. Section III describes the system model and problem formulation. Section IV captures the architecture of the MARS framework. Numerical results are conducted in Section V, and finally, Section VI draws conclusions.

II. Related Works

A. Resource Allocation in IRS Aided Communication Systems

In [6], researchers explored the fundamental capacity constraints of IRSs, optimizing both the IRS reflection matrix and resource allocation to boost system performance while taking into account factors such as maximum reconfiguration times. [7] employed three Reinforcement Learning (RL)-driven techniques to tackle issues in three distinct IRS scenarios, resulting in dynamic resource allocation for each grant-free user. [8] investigated an IRS-supported wireless-powered communication network and presented three beamforming schemes to enhance overall system efficiency. [9] introduced an innovative IRS-assisted coordinated multi-point system, aiming to optimize energy efficiency. [10] focused on devising optimal resource allocation algorithms for large-scale IRS-assisted systems that improve concurrent wireless information exchange and energy transfer. [11] studied the deployment of the IRS approach, successfully increasing efficiency in wireless energy transmission and task offloading while addressing challenges in wireless connections between hybrid access points and IoT devices.

B. Resource Allocation in UAV Aided Communication Systems

In [12], a resource allocation strategy for UAV networks was proposed, employing multi-agent cooperative environment learning to enhance system performance. The study of [13] integrated UAVs' advantages with Ultra-Reliable Low-Latency Communication (URLLC) systems, primarily aiming to boost the transmission rate of the backward link. [14] presented a tiered network architecture based on a UAV swarm, skillfully managing sensing, computing, and communication resources simultaneously, thereby increasing computing effectiveness. Considering the limited energy resource of UAVs, [15] explored techniques for reducing overall power consumption during training by jointly optimizing factors like local training parameters and resource allocation. To address the different Quality-of-Service (QoS) requirements for tasks and maximize the number of completed tasks, [16] developed a multi-agent proximal policy optimization-based method that simultaneously optimized UAV resource allocation and task offloading to achieve these objectives.

C. Resource Allocation in MEC Systems

Also, reference [17] suggested two resource allocation strategies aimed at optimizing the total computational bits for IoT devices, enhancing the system's computational energy efficiency. To reduce the combined computational and communication overhead associated with joint offloading and resource allocation strategies in vehicular systems, [18] applied a decentralized RL approach based on value iteration. [19] studied dynamic resource management in MEC-assisted railway IoT networks, highlighting the integration of subcarrier allocation, offloading ratios, power distribution, and computational resource assignment. Additionally, [20] analyzed a data processing scenario within IoT architecture, aiming to improve long-term average system utility through the joint optimization of communication resource allocation, data generation and discard strategies, and computational resource allocation. In [21], resource allocation challenges in a proposed millimeter-wave (mmWave) MEC system with dynamic offloading processes were explored. To address these issues, the authors introduced a matching-aided-learning resource allocation framework. Authors in [22] presented a harvest-and-offload protocol that jointly considered wireless energy transfer and computation offloading to minimize total energy consumption.

Despite the advances in the aforementioned studies, few papers explored the UAV and IRS-assisted MEC systems with a variable number of UAVs operating in fast-fading channel conditions. Multi-task learning demonstrates effectiveness in dynamic environments. Therefore, we introduce a MARS framework applied to UAV and IRS-assisted MEC systems that can achieve real-time resource scheduling in dynamic environments.

III. System Model and Problem Formulation

The system model for a UAV with IRS-assisted MEC system is depicted in Fig. 1. In this system, we assume there are N UEs, represented by the set $\mathcal{N} = \{1, 2, \ldots, N\}$, with each UE having a computational task to execute. The UEs are randomly distributed. We also consider L IRSs, represented by the set $\mathcal{L} = \{1, 2, \ldots, L\}$, deployed on buildings. In addition, there are several UAVs, denoted by the set $\mathcal{M} = \{1, 2, \ldots, M\}$, equipped with edge servers. A control center or central cloud collects environmental information (e.g., the number of UEs, IRSs, and UAVs, channel state information, etc.) and task information (e.g., the task's data size and

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TCC.2023.3307582, IEEE Transactions on Cloud Computing

required computing resource). The central cloud is responsible for resource scheduling in the system. Since the amount of collected information is relatively small, the communication overhead of the central cloud is not considered in this study.

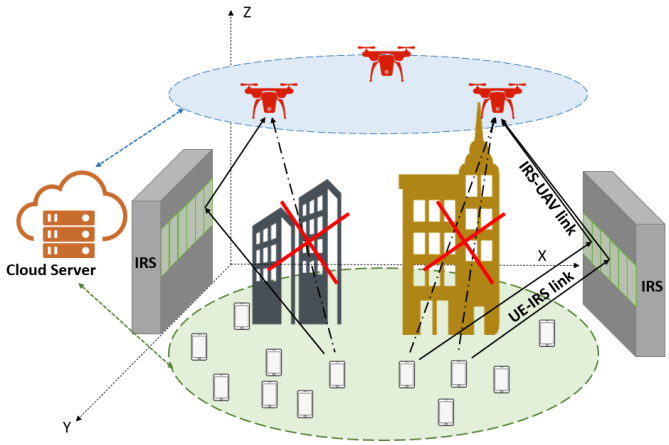


Fig. 1: UAV with IRS-assisted MEC network.

A. MEC Model

We assume that there is a computationally intensive task A_i on each UE, which can be expressed as:

$$A_i = (D_i, F_i), \quad \forall i \in \mathcal{N} \tag{1}$$

where D_i denotes the data size needing transmission to the UAV for execution, and F_i represents the total number of required CPU cycles for the task.

Each UE can either offload the entire task to a single UAV or execute it locally. We denote local execution with a_i^{loc}, and offloading the task to the j-th UAV with a_{ij}^{uav}. We have the following constraints:

$$C1: a_i^{loc} \in \{0,1\}, \quad \forall i \in \mathcal{N} \tag{2}$$

$$C2: a_{ij}^{uav} \in \{0,1\}, \quad \forall i \in \mathcal{N}, \ \forall j \in \mathcal{M} \tag{3}$$

where $a_i^{loc} = 0$ implies that the i-th UE does not execute the task locally, whereas $a_i^{loc} = 1$ indicates that the i-th UE chooses local execution. Similarly, $a_{ij}^{uav} = 0$ represents that the i-th UE does not offload the task to the j-th UAV, and $a_{ij}^{uav} = 1$ otherwise. We assume that each UE can select only one place to execute the task, thus we have the following constraint:

$$C3: a_i^{loc} + \sum_{j \in \mathcal{M}} a_{ij}^{uav} = 1, \quad \forall i \in \mathcal{N}. \tag{4}$$

1) Local Computing: The execution time for local computing at the i-th UE is given by:

$$T_i^{loc} = \frac{F_i}{f_i^{loc}}, \quad \forall i \in \mathcal{N} \tag{5}$$

where f_i^{loc} denotes the local computation capacity of the i-th UE, measured in CPU cycles per second. The computational capacity of the i-th UE is limited by:

$$C4: a_i^{loc} f_i^{loc} \le F_{i,max}^{loc}, \quad \forall i \in \mathcal{N} \tag{6}$$

where $F_{i,max}^{loc}$ represents the maximum local computation capacity of the i-th UE.

Energy consumption during the local computing phase is expressed as:

$$E_i^{loc} = \gamma_1 \left(f_i^{loc}\right)^{v_1 - 1} F_i, \quad \forall i \in \mathcal{N} \tag{7}$$

where $\gamma_1 \ge 0$ represents the effective-switched capacitance, and $v_1 \ge 1$ denotes a constant. To maintain consistency with practical measurements, we assign the values $\gamma_1 = 10^{-27}$ and $v_1 = 3$ [23].

2) Remote Computing: The time of task execution for the i-th UE at the j-th UAV is expressed as:

$$T_{ij}^{uav} = \frac{F_i}{f_{ij}}, \quad \forall i \in \mathcal{N}, \ \forall j \in \mathcal{M} \tag{8}$$

where f_{ij} represents the computation capacity provided by the j-th UAV to the i-th UE. The energy consumption during the remote computing phase is:

$$E_{ij}^{uav} = \gamma_2 \left(f_{ij}\right)^{v_2 - 1} F_i \tag{9}$$

where $\gamma_2 \ge 0$ is the effective-switched capacitance, and $v_2 \ge 1$ is a constant. We set $\gamma_2 = 10^{-28}$ and $v_2 = 3$ [23].

Due to the fact that the computation capacity provided by each UAV is limited, the constrained computational resource of the j-th UAV can be expressed as:

$$C5: \sum_{i \in \mathcal{N}} a_{ij}^{uav} f_{ij} \le F_{j,max}^{uav} \quad \forall j \in \mathcal{M} \tag{10}$$

where $F_{j,max}^{uav}$ indicates the total computation capacity of the j-th UAV.

B. IRS Model

Assume that UEs are located in a prosperous environment with many tall buildings, and the direct links of UAVs are blocked, which suffer from severe path loss. Deploying IRSs on the building surface can improve the communication quality between the UEs and the UAVs. Assume each IRS has several reflecting elements denoted as $\mathcal{K} = \{1, 2, \ldots, K\}$, and all of the reflecting elements can improve the communication quality by adjusting the phase shift of the incident signal. We assume that only one IRS is applied to help one UE with communication. The coordinate of the l-th IRS can be denoted as $\left(X_l^R, Y_l^R, Z_l^R\right)$, the coordinate of the i-th UE can be represented as $\left(X_i^U, Y_i^U, Z_i^U\right)$, the coordinate of the j-th UAV can be denoted as $\left(X_j^M, Y_j^M, Z_j^M\right)$. Thus, the distance from the i-th UE to the l-th IRS is expressed as:

$$d_{i,l}^{U,R} = \sqrt{\left(X_i^U - X_l^R\right)^2 + \left(Y_i^U - Y_l^R\right)^2 + \left(Z_i^U - Z_l^R\right)^2}. \tag{11}$$

Likewise, the distance from the l-th IRS to the j-th UAV can be calculated as:

$$d_{l,j}^{R,M} = \sqrt{\left(X_j^M - X_l^R\right)^2 + \left(Y_j^M - Y_l^R\right)^2 + \left(Z_j^M - Z_l^R\right)^2}. \tag{12}$$

We assume that the communication link between the i-th UE and the j-th UAV is divided into two parts: the initial UE-IRS link, followed by the IRS-UAV link. Assuming that each UE

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TCC.2023.3307582, IEEE Transactions on Cloud Computing

exclusively uses a single IRS for signal transmission. Hence, the channel gain corresponding to the UE-IRS link for the i-th UE is denoted as $h_{i,l}^{U,R}$. The channel gain of the UE-IRS link, $h_{i,l}^{U,R} \in \mathbb{C}^{K\times 1}$, can be expressed as:

$$h_{i,l}^{U,R} = \sqrt{\frac{\beta}{\left(d_{i,l}^{U,R}\right)^{\alpha}}}\left[1, e^{-j\frac{2\pi}{\lambda}d\phi_{i,l}^{U,R}}, \ldots, e^{-j\frac{2\pi}{\lambda}[K-1]d\phi_{i,l}^{U,R}}\right]^T \tag{13}$$

where α represents the path loss exponent of the channel used from the i-th UE to the l-th IRS, while β is the path loss at the reference distance of one meter. Additionally, $\phi_{i,l}^{U,R} = \frac{|X_i^U - X_l^R|}{d_{i,l}^{U,R}}$ indicates the cosine of the Angle Of Arrival (AOA) of the signal transmitted between the i-th UE and the l-th IRS.

We assume that each UE can offload its tasks to a single UAV, hence the channel gain for the IRS-UAV link for the i-th UE is represented as $h_{l,j}^{R,M} \in \mathbb{C}^{K\times 1}$. This can be calculated by:

$$h_{l,j}^{R,M} = \sqrt{\frac{\beta}{\left(d_{l,j}^{R,M}\right)^{2}}}\left[1, e^{-j\frac{2\pi}{\lambda}d\phi_{l,j}^{R,M}}, \ldots, e^{-j\frac{2\pi}{\lambda}[K-1]d\phi_{l,j}^{R,M}}\right]^T \tag{14}$$

where the right term in Eq. (14), represents the array response for the l-th IRS, which contains K reflective components. d is the antenna separation distance between two elements, while $\phi_{l,j}^{R,M} = \frac{|X_l^R - X_j^M|}{d_{l,j}^{R,M}}$ denotes the cosine of the Angle Of Departure (AOD) for the signal transmitted between the j-th UAV and the l-th IRS. λ represents the carrier wavelength.

We assume that there are K elements in each IRS, and the phase shift of the k-th reflecting element of the l-th IRS is denoted by $\theta_{k,i,l,j} \in [0, 2\pi)$. Therefore, we have a phase-shifted diagonal matrix for the IRS-assisted signal transmission, which can be denoted as $\Theta_{i,l,j} = \text{diag}\left\{e^{j\theta_{k,i,l,j}}, \forall k \in \mathcal{K}\right\}$.

When the i-th UE decides to offload the task to the j-th UAV by the l-th IRS, the whole channel gain of the UE-UAV link can be expressed as:

$$H_{ij} = (h_{i,l}^{U,R})^T \Theta_{i,l,j} h_{l,j}^{R,M}, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M}. \tag{15}$$

For any pair of UE and UAV, the data transmission rate can be formulated as:

$$r_{ij} = B\log_2\left(1 + \frac{P_{ij}^{tra}|H_{ij}|^2}{\sigma^2}\right) \tag{16}$$

where B denotes the channel's bandwidth; P_{ij}^{tra} is the transmission power utilized between the i-th UE and j-th UAV, as derived from [24]; σ^2 serves as an indicator of noise spectral density. Importantly, we operate under the assumption that UEs allocate tasks to the UAV employing Orthogonal Frequency Division Multiplexing (OFDM) channels, thus effectively mitigating any interference between the devices.

Then, the transmission time can be expressed as:

$$T_{ij}^{tra} = \frac{D_i}{r_{ij}}, \quad \forall i \in \mathcal{N}, \forall j \in \mathcal{M}. \tag{17}$$

Hence, the energy consumption during transmission can be calculated as:

$$E_{ij}^{tra} = P_{ij}^{tra} T_{ij}^{tra}. \tag{18}$$

C. UAV Model

When the j-th UAV hovers at a fixed position, the consumed energy can be approximated as:

$$E_j^{hov} = P_j^{hov} T_j^{hov} \tag{19}$$

where P_j^{hov} means the hover power of the j-th UAV, and T_j^{hov} means the hover time of the j-th UAV.

The energy consumption during the UAV's flight can be denoted as

$$E_j^{fly} = P_j^{fly}\frac{L_j}{v_j} \tag{20}$$

where P_j^{fly} denotes the power required for the j-th UAV's flight, L_j is the flight distance of the j-th UAV, and v_j represents the constant flight velocity of the j-th UAV.

D. Problem Formulation

Our objective is to minimize the cumulative energy consumption for the UAV with IRS-assisted MEC system. Therefore, the associated optimization issue can be formulated as follows:

$$\begin{aligned} P0: \min_{\mathbf{p},\mathbf{a},\mathbf{f},\mathbf{\Theta}} & \sum_{i\in\mathcal{N}}\left(a_i^{loc}E_i^{loc} + \sum_{j\in\mathcal{M}} a_{ij}^{uav}E_{ij}^{tra}\right) \\ & + \sum_{j\in\mathcal{M}}\left(\eta_1 E_j^{hov} + \eta_2 E_j^{fly} + \sum_{i\in\mathcal{N}} a_{ij}^{uav}E_{ij}^{uav}\right) \\ & \text{s.t. } C1 - C5 \end{aligned} \tag{21}$$

where $\mathbf{p} = \left\{X_j^M, Y_j^M, Z_j^M \middle| j \in \mathcal{M}\right\}$ are the locations of UAVs, $\mathbf{a} = \left\{a_i^{loc}, a_{ij}^{uav} \middle| i \in \mathcal{N}, j \in \mathcal{M}\right\}$ is the offloading decision, $\mathbf{f} = \left\{f_i^{loc}, f_{ij} \middle| i \in \mathcal{N}, j \in \mathcal{M}\right\}$ is the computing resource allocation, $\mathbf{\Theta} = \{\Theta_{i,l,j} | i \in \mathcal{N}, l \in \mathcal{L}, j \in \mathcal{M}\}$ represents the phase-shifted diagonal matrix for the IRS-assisted signal transmission, and η_1 and η_2 are weight coefficients.

IV. The MARS Algorithm

We introduce a DRL-based MARS architecture designed to minimize energy consumption for all UEs and dynamic UAVs in an IRS-assisted MEC system. The MARS framework operates as follows: The central cloud gathers information about the environment and tasks, runs the MARS algorithm, adjusts the UAV positions and IRS phase-shift matrices, and determines real-time task offloading and resource allocation for each UE. Under the guidance of the central cloud, every UE offloads tasks to the appropriate UAV and receives the results afterward.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TCC.2023.3307582, IEEE Transactions on Cloud Computing

Algorithm 1 MARS framework

Input: $D_{i,t}, F_{i,t}$.
Output: $(X_{j,t}^M, Y_{j,t}^M, Z_{j,t}^M), \Theta_{i,l,j,t}, a_{i,j,t}, f_{i,j,t}$.

```
Initialize the parameters θ_m of the multi-head agent for m UAVs randomly.
Initialize a replay buffer R.
Set the iteration number T_AC.
while t < T_AC do
    if the number of UAVs has changed then
        Adjust the parameters θ_{m,t} of the multi-head agent by multi-task controller.
        Optimize the locations (X_{j,t}^M, Y_{j,t}^M, Z_{j,t}^M) of UAVs by LS-FCM algorithm.
    end if
    for i = 1, ..., N do
        Calculate the phase-shifted diagonal matrix Θ_{i,l,j,t} by Eq. (24).
        Take the action A_{i,t} ~ π_{θ_{m,t}}(·|S_{i,t}) by the forward propagation process of the multi-head agent.
        Add A_{i,t} to the epoch register.
    end for
    Execute action A_t, receive reward R_t and the next state S_{t+1}.
    Evaluate the current solution A_t, and execute the offline learning stage when the current solution is abominable.
    Search the optimal action A_t^* from the initial action A_t by Algorithm 2.
    Append the transition {S_{i,t}, A_{i,t}^*, R_{i,t}, S_{i,t+1}} of all UEs to the replay buffer R.
    Sample a batch of transitions by prioritization experience replay strategy.
    Feed the sampled transitions to the multi-head agent.
    Train the multi-head agent and update the parameters θ_{m,t} by the backpropagation process of the multi-head agent.
end while
```

A. Algorithm Overview

The MARS framework workflow, as described in **Algorithm 1**, assumes UAVs are redeployed only when their number changes in order to save the flight energy. By using the large-scale path-loss fuzzy c-means clustering algorithm (LS-FCM) to optimize UAV locations, we obtain $\mathbf{p}$ in $P0$ [25], based on large-scale path-loss factors. Following that, we adopt a quantitative passive beamforming approach to determine the phase-shifted diagonal matrix $\mathbf{\Theta}$, considering the positions of UAVs and UEs. Finally, we develop a novel A2C algorithm to generate offloading decisions $\mathbf{a}$ and computational resource allocations $\mathbf{f}$ for all UEs.

The structure of the DRL part of the MARS framework is illustrated in Fig. 2. The key elements of the DRL in the MARS framework are described as follows:

- State: $\mathcal{S}_t = \{\mathcal{S}_{i,t} | i \in \mathcal{N}\}$ denotes the environment information of the i-th UE at time t, where $\mathcal{S}_{i,t} = \{H_{i,t}, D_{i,t}, F_{i,t}\}$. $H_{i,t} = \{h_{i,j,t} | j \in \mathcal{M}\}$ represents the channel gain between the i-th UE and all UAVs, while $D_{i,t}$ and $F_{i,t}$ signify the task attributes of the i-th UE.
- Action: $\mathcal{A}_t = \{\mathcal{A}_{i,t} | i \in \mathcal{N}\}$ defines the resource scheduling decision of the i-th UE at time t, where $\mathcal{A}_{i,t} = \{a_{i,t}, f_{i,t}\}$, $a_{i,t} \in \mathbb{N}$ indicates user association, and $f_{i,t} \in \mathbb{R}$ denotes the allocated resource for the i-th UE.
- Reward: $\mathcal{R}_{i,t}$ is the reciprocal of the i-th UE's task energy consumption, and $\mathcal{R}_t$ denotes the objective function's reciprocal at time t.
- Transition: $\{\mathcal{S}_{i,t}, \mathcal{A}_{i,t}, \mathcal{R}_{i,t}, \mathcal{S}_{i,t+1}\}$ is stored in the replay buffer and employed for updating the agent's policy. The First In First Out (FIFO) strategy manages transition updates in the replay buffer.

In the following, more details of four main parts in the MARS framework are introduced: (1) A quantitative passive beamforming method (detailed in Section IV-B) is introduced to solve the phase-shifted diagonal matrix according to the positions of UAVs and UEs. (2) A multi-head agent (detailed in Section IV-C) is designed to solve the formulated MINLP by A2C learning. (3) A multi-task controller (detailed in Section IV-D) is presented to adjust the structure of the multi-head agent when the number of UAVs is changed without retraining the whole neural network. (4) A novel LWS (detailed in Section IV-E) is applied to enhance the action exploration of the DRL and accelerate the learning process in dynamic environments.

B. Quantitative Passive Beamforming

We apply a quantitative passive beamforming method to optimize the phase shift matrix of IRSs. Specifically, Eq. (14) can be transformed into the following equation:

$$h_{l,j}^{R,M} = \left[\left|h_{l,j}^{R,M}\right| e^{j\omega_{l,j,1}^{R,M}}, \left|h_{l,j}^{R,M}\right| e^{j\omega_{l,j,2}^{R,M}}, \ldots, \left|h_{l,j}^{R,M}\right| e^{j\omega_{l,j,k}^{R,M}}\right]^T \tag{22}$$

where $\left|h_{l,j}^{R,M}\right|$ is the magnitude and $\omega_{l,j,k}^{R,M} \in [0, 2\pi)$ denotes the phase shift of the k-th reflecting element from the j-th UAV to the l-th IRS. Also, Eq. (13) can be transformed into the following equation:

$$h_{i,l}^{U,R} = \left[\left|h_{i,l}^{U,R}\right| e^{j\omega_{i,l,1}^{U,R}}, \left|h_{i,l}^{U,R}\right| e^{j\omega_{i,l,2}^{U,R}}, \ldots, \left|h_{i,l}^{U,R}\right| e^{j\omega_{i,l,k}^{U,R}}\right]^T \tag{23}$$

where $\left|h_{i,l}^{U,R}\right|$ is the magnitude and $\omega_{i,l,k}^{U,R} \in [0, 2\pi)$ denotes the phase shift of the k-th reflecting element from the l-th IRS to the i-th UE.

To simplify our approach, we consider discrete phase shift angles in the study. The IRS phase shift, $\theta_{k,i,l,j}$, can be selected from the set $\Psi \triangleq \{\frac{2\pi}{N_p} i | i = 0, 1, \ldots, N_p - 1\}$, where N_p represents the number of phase shift values available for each element. By coherently combining signals from different paths at the UAV, the resulting signal maximizes the received power and achievable rate. Therefore, we optimize the phase shift $\theta_{k,l,i,j}$ for the k-th reflecting element of the l-th IRS

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TCC.2023.3307582, IEEE Transactions on Cloud Computing

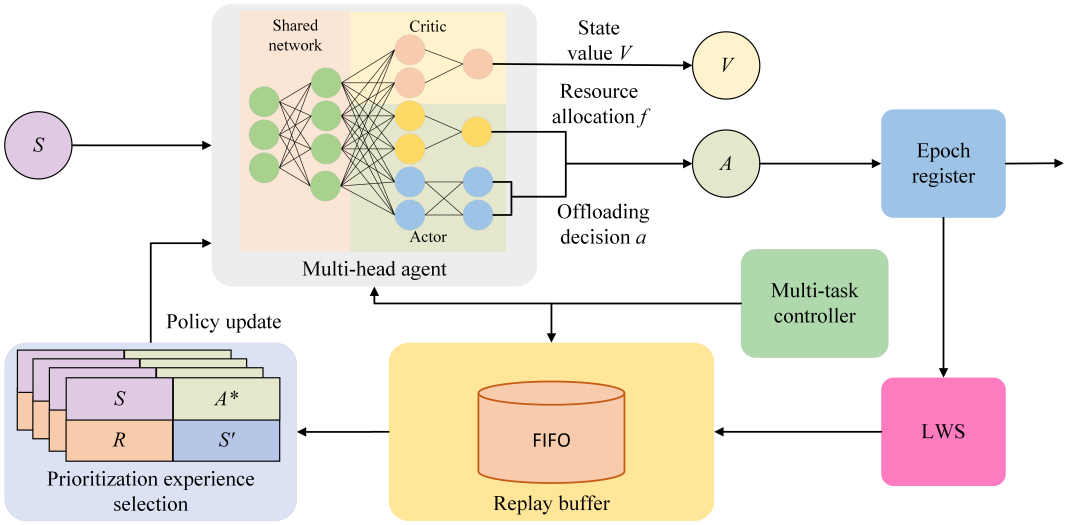


Fig. 2: The DRL part of the MARS framework.

between the i-th UE and the j-th UAV using the following equation [26]:

$$\theta_{k,i,l,j} = \underset{\theta'_{k,i,l,j} \in \Psi}{\operatorname{argmin}} \left| \theta'_{k,i,l,j} - \left(\omega^{R,M}_{l,j,k} + \omega^{U,R}_{i,l,k} \right) \right|. \tag{24}$$

C. Multi-head Agent

We propose an A2C structure with a multi-head agent to address the proposed MINLP. The multi-head agent features two networks, each with three heads. The critic head calculates the state-value function for actions, while the actor head predicts resource allocation and offloading decisions. During the learning process, the actor and critic networks share parameters in their shallow parts to extract common features. They then independently adjust subsequent parameters to learn the unique features of each head, respectively. Fig. 3 illustrates the structure of the multi-head agent.

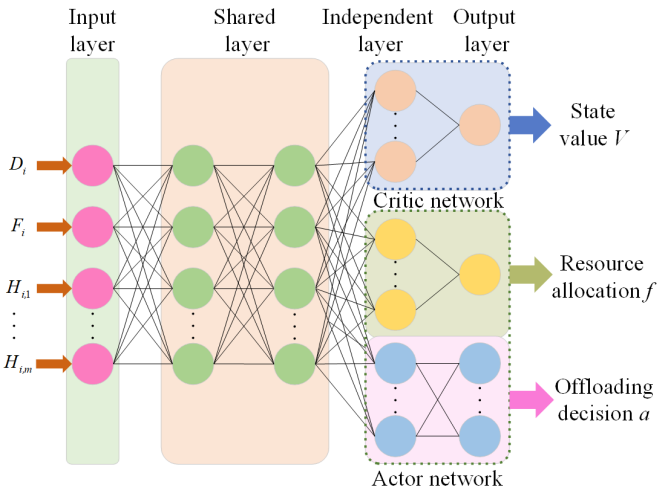


Fig. 3: The multi-head agent.

1) Forward propagation process: In Fig. 3, the multi-head agent has L shared layers and each task has K independent layers. The output of the ℓth shared layer can be described as follows:

$$\mathbf{O}_\ell = \mathrm{ReLU}\left(\mathbf{W}_\ell \mathbf{O}_{\ell-1} + \mathbf{b}_\ell\right) \tag{25}$$

where $\mathbf{W}_\ell$ means the weights of the ℓ-th shared layer and the $\mathbf{b}_\ell$ means the biases of the ℓ-th shared layer, ReLU is the activation function.

The output of the k-th independent layer of the j-th head can be represented as:

$$\mathbf{O}_{j,L+k} = \mathrm{ReLU}\left(\mathbf{W}_{j,L+k} \mathbf{O}_{j,L+k-1} + \mathbf{b}_{j,L+k}\right) \tag{26}$$

where $\mathbf{W}_{j,L+k}$ means the weights of the k-th independent layer for the j-th head and the $\mathbf{b}_{j,L+k}$ means the biases of the k-th independent layer for the j-th head.

The output layer of the j-th head can be represented as:

$$\mathbf{O}_{j,L+K} = \begin{cases} \mathrm{Sigmoid}\left(\mathbf{W}_{j,L+K} \mathbf{O}_{j,L+K-1} + \mathbf{b}_{j,L+K}\right), \\ \text{For the regression head in the actor.} \\ \mathrm{Softmax}\left(\mathbf{W}_{j,L+K} \mathbf{O}_{j,L+K-1} + \mathbf{b}_{j,L+K}\right), \\ \text{For the classification head in the actor.} \\ \mathbf{W}_{j,L+K} \mathbf{O}_{j,L+K-1} + \mathbf{b}_{j,L+K}, \\ \text{For the critic head.} \end{cases} \tag{27}$$

where $\mathbf{W}_{j,L+K}$ means the weights of the output layer for the j-th head and $\mathbf{b}_{j,L+K}$ means the bias of the output layer for the j-th head. Sigmoid and Softmax are the activation functions.

2) Back propagation process: In our study, the critic head is presented to calculate the state-value function of the action. Hence, the advantage value can be calculated as:

$$\delta_t = \mathcal{R}_t + \left(V_{\boldsymbol{\omega}_t}(\mathcal{S}_{t+1}) - \eta_t\right) - V_{\boldsymbol{\omega}_t}(\mathcal{S}_t) \tag{28}$$

where η_t is the discount factor for the state-value function $V_{\boldsymbol{\omega}_t}(\mathcal{S}_{t+1})$. The advantage value highlights the difference between the action-value and state-value functions. Thus, actions with smaller advantage values are less likely to occur, effectively lowering the overall variance [27].

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TCC.2023.3307582, IEEE Transactions on Cloud Computing

The loss function of the critic head can be expressed as:

$$L_c = \frac{1}{Z}\sum_{k}^{Z}\delta_k^2 \tag{29}$$

where Z is the number of selected transitions.

The actor network consists of a regression head for the resource allocation and a classification head for the offloading decision. Hence, we can solve the original MINLP problem with two different heads efficiently.

In the classification head, the loss function of the user association is cross-entropy loss which can be expressed as:

$$L_a = \frac{1}{Z}\sum_{k}^{Z}\left(-\sum_{j=1}^{M} y_{kj}\log p_{kj}\right) \tag{30}$$

where y_{kj} is an indicator variable, $y_{kj} = 1$ means that the true label is same as the predicted label, and p_{kj} denotes the probability that the k-th transition belongs to the j-th class.

In the regression head, the loss function of the resource allocation is Mean Square Error (MSE) loss which can be expressed as:

$$L_f = \frac{1}{Z}\sum_{k}^{Z}\left(\hat{f}_k - f_k\right)^2 \tag{31}$$

where $\hat{f}_k$ means the predicted computational resource and f_k means the true computational resource.

The shared layers of the multi-head agent can be updated by:

$$\begin{aligned}\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t\frac{1}{Z}\sum_{k}^{Z}\nabla_{\boldsymbol{\theta}_t}\log\pi_{\boldsymbol{\theta}_t}\left(\mathcal{A}_k|\mathcal{S}_k\right)(\delta_k) + \\ \beta_t\frac{1}{Z}\sum_{k}^{Z}(\nabla_{\boldsymbol{\omega_t}}(\delta_k)^2) + \gamma_t\frac{1}{Z}\sum_{k}^{Z}\nabla_{\boldsymbol{\theta}_t}H(\pi_{\boldsymbol{\theta}_t}(\mathcal{S}_k))\end{aligned} \tag{32}$$

where α_t is the learning rate of the actor network, β_t is the learning rate of the critic network, $H(\cdot)$ is the entropy of the policy $\pi_{\boldsymbol{\theta}_t}$, and its learning rate is γ_t. Policy entropy assigns the exploration probability according to the advantage value of the action, thus making the agent could explore various actions as much as possible, implying that various state will also be explored [28].

D. Multi-task Controller

Multi-task learning is a machine learning method that puts multiple related tasks together to learn [29]. We consider a dynamic MEC system in which the number and positions of UAVs are variable, and we should memorize the offloading knowledge of different UAVs in the agent. Therefore, we design a novel multi-task controller to meet this challenge. The basic principle of the multi-task controller is to design different learning tasks for different numbers of UAVs, then adjust the structure of the network for different learning tasks. For enhancing the learning efficiency, we free up redundant parameters for each task in the network, so that we can sequentially "pack" multiple tasks into a single network while ensuring minimal performance degradation. This structure can naturally accumulate experiences for varying numbers of UAVs and is immune to catastrophic forgetting. The detailed process of the multi-task controller can be described as follows:

1) Network training: We initialize the agent and train it for the scenario with the minimum number of UAVs (Task 1). Taking into account network redundancy, after the network training is finished, we delete a certain number of weights. At this point, the performance of the network degrades due to the pruning in the network structure, and then we continue to fine-tune the network until the performance of the network is optimal on Task 1. Then, we increase the number of UAVs in the scenario for learning as a new Task 2. On the one hand, we freeze the weights of Task 1 and train all the remaining parameters as the initial weights of Task 2. After the network training is completed, a part of the weights is deleted, and then we fine-tune the network again until the network performance is optimal on Task 2. We then repeat the process until all tasks are trained or the network with no extra free weights.

2) Network pruning: We delete the weights of the agent by network pruning and get a high-performance multi-head agent. In the pruning process of each task, we arrange the weights of each layer by absolute value and remove the smallest weights in a fixed proportion, and then we add a small number of random weights for exploration. It is important to note that we only prune the weights of the current task, not the weights of the previous tasks. This allows the weights of the previous tasks to be used in the later task, but the weights of the later task do not interfere with the previous tasks. This mechanism ensures that when training a new task, the knowledge of old tasks is retained and the performance of old tasks does not change, thus avoiding catastrophic forgetting.

3) Network inference and adjustment: After the training of all tasks in the agent is completed, we will select the corresponding task weights according to the number of UAVs in the current scenario, so that the network structure of the current task corresponds to the number of UAVs. Then the offloading decision and resource allocation are generated according to the current network structure.

The adjustment process of the multi-task controller is described in Fig. 4. The load rule of the ℓ-th layer for the k-th task can be represented as:

$$\mathbf{O}_{\ell,k} = \mathrm{ReLU}(\mathbf{W}_{\ell,c}\mathbf{O}_{\ell-1,c}+\mathbf{b}_{\ell,c}+\sum_{i=c+\mathbf{1}}^{k}(\mathbf{W}_{\ell,i}\mathbf{O}_{\ell-1,c}+\mathbf{b}_{\ell,i})) \tag{33}$$

where c means the weights of the current task, and the unload rule of the ℓ-th layer for the k-th task can be represented as:

$$\mathbf{W}_{\ell,i} = 0,\ \mathbf{b}_{\ell,i} = 0,\ \forall i = k+1,..,c. \tag{34}$$

E. Light Wolf Search

Action refinement is an effective exploration strategy for large-scale action space presented by Google DeepMind [30], and we propose a novel light wolf search algorithm to realize the action refinement in the study. The Gray Wolf Optimizer (GWO) is inspired by the hunting behaviors of wolves [31], and combined with the channel quality information, we design the LWS as follows:

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TCC.2023.3307582, IEEE Transactions on Cloud Computing

1) Solution representation: The solution of LWS is represented as a vector $\mathbf{x} = (\mathbf{a}, \mathbf{f})$, where the $\mathbf{a}$ is the offloading decision, and $\mathbf{f}$ denotes the resource allocation. The initial solution $\mathbf{x_0} = (\mathbf{a_0}, \mathbf{f_0})$ is obtained from the epoch register.

2) Population initialization: We initialize four wolves randomly form a population $\mathbf{P} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ in the d-dimensional search space. For simulating the social hierarchy of wolves, the optimal wolf is selected as the α wolf $\mathbf{x}_\alpha$, the second and the third-best wolves are selected as the β wolf $\mathbf{x}_\beta$ and δ wolf $\mathbf{x}_\delta$, respectively. The initial solution $\mathbf{x_0}$ is marked as the light wolf $\mathbf{x}_\omega$.

3) Parameter updating: We update the hunt parameters of the population $\mathbf{P}$. $A(t)$ and $C(t)$ are coefficient vectors which can be defined as follows:

$$A\left(t\right) = \left(2r_1 - 1\right) \cdot a\left(t\right) \tag{35}$$

$$C\left(t\right) = 2 \cdot r_2 \tag{36}$$

where t is the current iteration number, r_1 and r_2 are independent random numbers in the range of [0,1]. $a(t)$ is the control parameter, whose formula is defined as follows:

$$a\left(t\right) = 2 - \frac{2t}{t_{\max}} \tag{37}$$

where $t_{\max}$ is the maximum iteration number. The control variable $a(t)$ decreases linearly from 2 to 0.

4) Wolf hunt: During the hunt process, wolves encircle their prey, and the update formula of the light wolf is defined as follows:

$$D_e\left(t\right) = \left|C\left(t\right) \cdot \mathbf{x}_e\left(t\right) - \mathbf{x}_\omega\left(t\right)\right| \tag{38}$$

$$\mathbf{x}_\omega\left(t+1\right) = \frac{1}{3} \sum_{e \in \{\alpha, \beta, \delta\}} \left(\mathbf{x}_e\left(t\right) - A(t) \cdot D_e\left(t\right)\right) \tag{39}$$

where $\mathbf{x}_\omega(t)$ is the position vector of the light wolf. $\mathbf{x}_e\left(t\right)$ represents the positions of elite wolves (i.e., α, β and δ). D_e represents the distances between the light wolf and elite wolves. The offloading part of all solutions needs to be rounded to the feasible solution space.

5) Wolf evaluation: We define $P0$ as the fitness function of the wolf pack. We calculate the fitnesses of all wolves, and the optimal wolf is selected as the wolf $\mathbf{x}_\alpha$, the second and the third-best wolves are selected as the wolf $\mathbf{x}_\beta$ and wolf $\mathbf{x}_\delta$, respectively. The remaining individual is marked as the light wolf ω. Finally, we update the offloading part of the light wolf whose offloading decision for each UE is set to the UAV with the highest channel gain.

6) Constraint check: When a new solution is generated, the constraint check will be performed to ensure that the resource allocated by the UAV will be no more than the maximum computational resource of the UAV. When the allocated resource of the UAV is overflowing, we will reduce the allocated resource of the UAV for each UE proportionally until the total computational resource is within the maximum computational resource constraint. The overall algorithm for LWS is summarized in **Algorithm 2**.

Algorithm 2 LWS

Input: $\mathbf{a_0}, \mathbf{f_0}$.
Output: $\mathbf{x}_\alpha$.
1: Initialize the wolf population.
2: Initialize a, A and C.
3: Select elite wolves $\mathbf{x}_\alpha$, $\mathbf{x}_\beta$ and $\mathbf{x}_\delta$.
4: Define initial light wolf as $\mathbf{x}_\omega = (\mathbf{a_0}, \mathbf{f_0})$.
5: **while** $t < t_{\max}$ **do**
6: **for** each wolf **do**
7: Update the position of the current wolf by Eqs. (38)-(39).
8: **end for**
9: Calculate the finesses of all wolves.
10: Update elite wolves $\mathbf{x}_\alpha$, $\mathbf{x}_\beta$ and $\mathbf{x}_\delta$.
11: Update the light wolf $\mathbf{x}_\omega$ according to channel gains.
12: Update a by Eq. (37).
13: Update A and C by Eqs. (35) - (36).
14: $t = t + 1$.
15: Carry out the constraint check and obtain valid solutions.
16: **end while**

F. Convergence Analysis of the MARS Framework

1) Propositions: In the proposed framework, the critic network utilizes linear function approximation for gauging the state-value function, represented as $V(\cdot; \boldsymbol{\omega}) = \phi^\top(\cdot)\boldsymbol{\omega}$ [32]. We depict $\mathbf{A}$ and $\mathbf{b}$ in the following formulation:

$$\mathbf{A} := \mathbb{E}_{\mathcal{S},\mathcal{A},\mathcal{S}'}\left[\phi(\mathcal{S})\left(\phi\left(\mathcal{S}'\right) - \phi(\mathcal{S})\right)^\top\right] \tag{40}$$

$$\mathbf{b} := \mathbb{E}_{\mathcal{S},\mathcal{A},\mathcal{S}'}[(\mathcal{R}(\mathcal{S},\mathcal{A}) - \mathcal{R}(\boldsymbol{\theta}))\phi(\mathcal{S})] \tag{41}$$

where $\mathcal{A}$ is the action and $\mathcal{A} \sim \pi_{\boldsymbol{\theta}}\left(\cdot|\mathcal{S}\right)$; $\mathcal{S}'$ is the next state and $\mathcal{S}' \sim \mathcal{P}\left(\cdot|\mathcal{S},\mathcal{A}\right)$, where $\mathcal{P}(\cdot|\mathcal{S},\mathcal{A})$ is the transition probability measure. $\mathcal{R}$ is the reward function and $\mathcal{R}(\boldsymbol{\theta})$ is the predicted reward. The limiting point $\boldsymbol{\omega}^*(\boldsymbol{\theta})$ satisfies [33]:

$$\mathbf{A}\boldsymbol{\omega}^*(\boldsymbol{\theta}) + \mathbf{b} = 0. \tag{42}$$

The error associated with the linear function approximation is defined as follows:

$$\epsilon_{\text{app}}(\boldsymbol{\theta}) := \sqrt{\mathbb{E}_{\mathcal{S}}\left(\phi(\mathcal{S})^\top\boldsymbol{\omega}^*(\boldsymbol{\theta}) - V(\mathcal{S})\right)^2} \tag{43}$$

where $V(\cdot)$ represents the state-value function.

We assume throughout the study that the approximation error is uniformly bounded for all potential policies:

$$\epsilon_{\text{app}}(\boldsymbol{\theta}) \leq \epsilon_{\text{app}} \tag{44}$$

where ϵ_{app} is a constant, with $\epsilon_{\text{app}} \geq 0$.

2) Assumptions:

Assumption 1. *Regarding every possible policy parameter $\boldsymbol{\theta}$, the previously defined matrix $\mathbf{A}$ exhibits negative definiteness, with the maximum eigenvalue being $-\lambda$ [34].*

Assumption 2. *Given a constant $\boldsymbol{\theta}$, let $\mu_{\boldsymbol{\theta}}(\cdot)$ represent the stationary distribution induced by policy $\pi_{\boldsymbol{\theta}}(\cdot|\mathcal{S})$ and transition probability measure $\mathcal{P}(\cdot|\mathcal{S},\mathcal{A})$. Envision a Markov chain generated through the process $\mathcal{A}_t \sim \pi_{\boldsymbol{\theta}}\left(\cdot|\mathcal{S}_t\right), \mathcal{S}_{t+1} \sim$*

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TCC.2023.3307582, IEEE Transactions on Cloud Computing

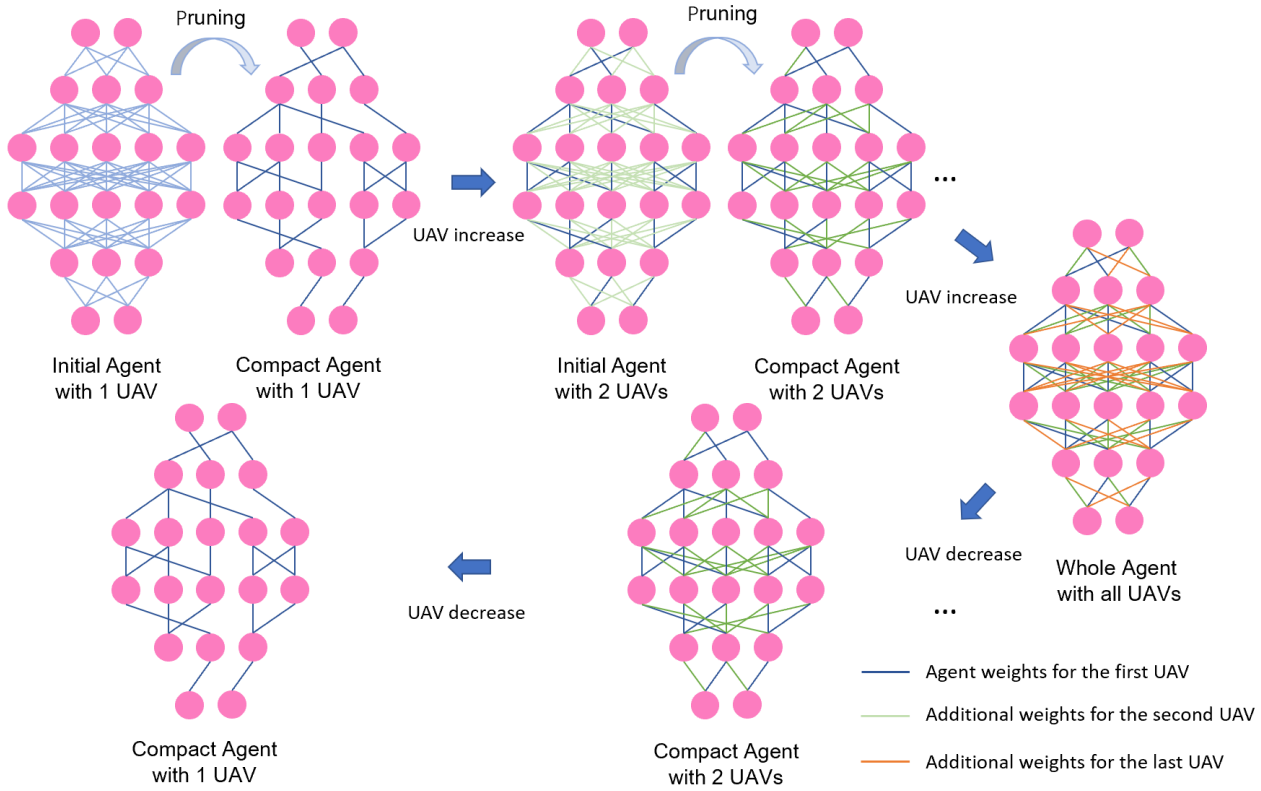


Fig. 4: The multi-task controller with varying number of UAVs.

$\mathcal{P}(\cdot|\mathcal{S}_t, \mathcal{A}_t)$. *Consequently, an* $m > 0$ *and* $\rho \in (0,1)$ *exist, fulfilling the condition [35]:*

$$d_{TV}\left(\mathbb{P}\left(\mathcal{S}_\tau \in \cdot | \mathcal{S}_0 = \mathcal{S}\right), \mu_{\boldsymbol{\theta}}(\cdot)\right) \leq m\rho^\tau, \forall \tau \geq 0, \tag{45}$$

where $d_{TV}(\cdot)$ *represents the total variation norm between two probability measures.*

Assumption 3. *Consider* $\pi_{\boldsymbol{\theta}}(\mathcal{A}|\mathcal{S})$ *as a policy characterized by* $\boldsymbol{\theta}$. *Constants* $L, B, L_l > 0$ *are present, and for any specific state* $\mathcal{S}$ *and action* $\mathcal{A}$, *the subsequent association is maintained [33]:*

$$\|\nabla \log \pi_{\boldsymbol{\theta}}(\mathcal{A} \mid \mathcal{S})\| \leq B, \forall \boldsymbol{\theta} \in \mathbb{R}^d \tag{46}$$

$$\|\nabla \log \pi_{\boldsymbol{\theta}_1}(\mathcal{A} \mid \mathcal{S}) - \nabla \log \pi_{\boldsymbol{\theta}_2}(\mathcal{A} \mid \mathcal{S})\| \leq L_l \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|, \forall \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d \tag{47}$$

$$|\pi_{\boldsymbol{\theta}_1}(\mathcal{A} \mid \mathcal{S}) - \pi_{\boldsymbol{\theta}_2}(\mathcal{A} \mid \mathcal{S})| \leq L \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|, \forall \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d \tag{48}$$

Assumption 4. *Under Assumptions 1 and 2, there exists a constant* $L_* > 0$ *such that [36]*

$$\|\boldsymbol{\omega}^*(\boldsymbol{\theta}_1) - \boldsymbol{\omega}^*(\boldsymbol{\theta}_2)\| \leq L_* \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|, \forall \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^d. \tag{49}$$

3) Convergence Analysis:

Theorem 1. *Assuming that Assumptions 1-3 are valid and setting* $\alpha_t = c_\alpha/(1+t)^\sigma$, *with constants* $\sigma \in (0,1)$ *and* $c_\alpha > 0$, *the critic satisfies the following formulation at the* t*-th iteration [36]:*

$$\frac{8}{t}\sum_{k=1}^{t}\mathbb{E}\|\boldsymbol{\omega}_k - \boldsymbol{\omega}_k^*\|^2 + \frac{2}{t}\sum_{k=1}^{t}\mathbb{E}\left(\eta_k - \mathcal{R}(\boldsymbol{\theta}_k)\right)^2 = \mathcal{E}(t) \tag{50}$$

where $\boldsymbol{\omega}_k$ *signifies the critic's parameter, while* η_k *corresponds to the discount factor at the actor's* k*-th update. Assuming that* $\mathcal{E}(t)$ *constitutes a limited series, the following is established:*

$$\min_{0\leq k\leq t}\mathbb{E}\|\nabla J(\boldsymbol{\theta}_k)\|^2 = \mathcal{O}(\epsilon_{\text{app}}) + \mathcal{O}\left(\frac{1}{t^{1-\sigma}}\right) + \mathcal{O}\left(\frac{\log^2 t}{t^\sigma}\right) + \mathcal{O}(\mathcal{E}(t)) \tag{51}$$

where $\mathcal{O}(\cdot)$ *is used to further hide constants [36].*

Theorem 2. *Presuming that Assumptions 1-3 are valid, and we select* $\alpha_t = c_\alpha/(1+t)^\sigma$ *and* $\beta_t = c_\beta/(1+t)^\nu$, *with* $0 < \sigma < \nu < 1$ *and positive constants* c_α *and* $c_\beta \leq \lambda^{-1}$, *the following is asserted [36]:*

$$\frac{1}{1+t-\tau_t}\sum_{k=\tau_t}^{t}\mathbb{E}\|\boldsymbol{\omega}_k - \boldsymbol{\omega}_k^*\|^2 = \mathcal{O}\left(\frac{1}{t^{1-\nu}}\right) + \mathcal{O}\left(\frac{\log t}{t^\nu}\right) + \mathcal{O}\left(\frac{1}{t^{2(\sigma-\nu)}}\right) \tag{52}$$

$$\frac{1}{1+t-\tau_t}\sum_{k=\tau_t}^{t}\mathbb{E}\left(\eta_k - \mathcal{R}(\boldsymbol{\theta}_k)\right)^2 = \mathcal{O}\left(\frac{1}{t^{1-\nu}}\right) + \mathcal{O}\left(\frac{\log t}{t^\nu}\right) + \mathcal{O}\left(\frac{1}{t^{2(\sigma-\nu)}}\right). \tag{53}$$

Combining **Theorem** 1 and **Theorem** 2, we can deduce the

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TCC.2023.3307582, IEEE Transactions on Cloud Computing

TABLE I: Simulation parameters.

Parameters	Assumptions
Number of UEs N	50
Number of IRSs L	50
Transmitting data size D_i	20 MB
Transmitting power P_{ij}^{tra}	1 W
UAV executing power P_{ij}^{uav}	1 W
UAV hover power P_j^{hov}	1 W
Max computation capacity of the local $F_{i,max}^{loc}$	10^9 cycles/s
Required number of CPU cycles F_i	10^9 cycles/s
Total computation capacity of the UAV $F_{j,max}^{uav}$	3×10^{10} cycles/s
Bandwidth B	1 MHz
Noise spectral density σ^2	10^{-12} W/Hz

convergence rate of the proposed A2C structure as follows:

$$\min_{0\le k\le t} \mathbb{E}\left\|\nabla J\left(\boldsymbol{\theta}_k\right)\right\|^2 = \mathcal{O}\left(\epsilon_{\text{app}}\right) + \mathcal{O}\left(\frac{1}{t^{1-\sigma}}\right) + \mathcal{O}\left(\frac{\log t}{t^{\nu}}\right) + \mathcal{O}\left(\frac{1}{t^{2(\sigma-\nu)}}\right). \tag{54}$$

Finally, the MARS framework can find an ϵ-approximate stationary point of $J(\cdot)$ within T steps, namely

$$\min_{0\le k\le T} \mathbb{E}\left\|\nabla J\left(\boldsymbol{\theta}_k\right)\right\|^2 \le \mathcal{O}\left(\epsilon_{\text{app}}\right) + \epsilon \tag{55}$$

where T is the total iteration number and $\nabla J\left(\boldsymbol{\theta}_k\right)$ is the policy gradient.

V. Numerical Results

A. Simulation Settings

In this section, we present simulation results evaluating the performance of the MARS framework. The simulation environment was established using Python 3.7 and TensorFlow 2.2.0, running on an Intel Xeon CPU with 32 GB RAM and a Tesla T4 GPU with 15 GB SGRAM. The initial multi-head agent consists of two shared layers with 64 and 128 neurons, respectively. Each independent layer contains 32 neurons. The replay buffer size is set at 8000, and the minibatch size is 50. Dropout is utilized during the training process to prevent overfitting. For the multi-task controller, a compression ratio of 75% is assigned for each UAV. In the LWS, the wolf pack size is fixed at 4, and the maximum iteration number is 10. All other simulation parameters are detailed in Table I, unless specified otherwise.

B. Performance Evaluation of the Multi-Head Agent

We compare the performances of agents in the MARS framework with different heads. These agents are introduced as follows:

- Single-head agent: The agent in the A2C structure outputs the state value of the critic network, and the offloading decision and resource allocation of the actor network by a single head.
- Two-head agent: The agent in the A2C structure outputs the state value of the critic network by one head, and the offloading decision and resource allocation of the actor network by the other hand.
- Three-head agent: The agent in the A2C structure outputs the state value of the critic network and the offloading decision and resource allocation of the actor network by three heads.

For fairness, the number of weights in all agents is set to the same. In the single-head agent, MSE loss is used to optimize all network parameters. In the two-head agent, MSE loss is introduced to optimize the critic head and the regression head, and the cross-entropy loss is applied to optimize the classification head. The rewards of all agents are shown in Fig. 5. We can see that all agents can obtain relatively good rewards and achieve convergence at last. However, the three-head agent can obtain the highest reward and this phenomenon can be explained by the reason that different heads have different feature representations, and the MSE loss is good at optimizing the regression head but hard to optimize the critic head and classification head.

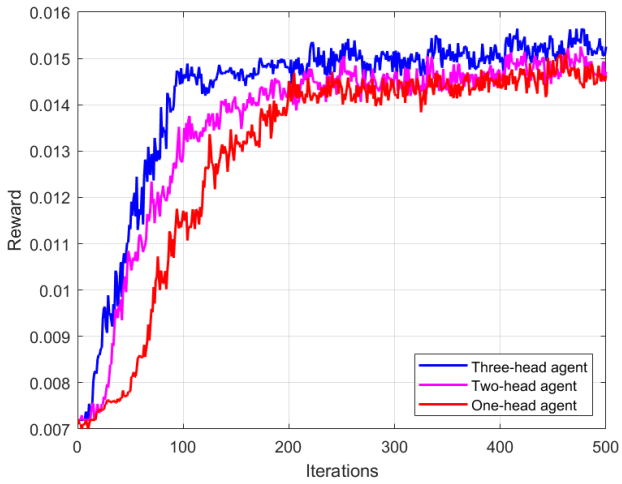


Fig. 5: Rewards of different agents.

C. Performance Evaluation of the Multi-Task Controller

In this section, we simulate a scenario where the number of UAVs is varying and compare the energy consumption of the MEC system when the agent for Task 1 (One UAV) is well-trained. Fig. 6 summarizes the multi-task performances for different multi-task methods, in which we add three UAVs as new tasks one by one to the MEC system. The contender in Fig. 6 is Dynamic-Expansion Net (DEN) [37], which reduces the weights of the previous tasks via sparse-regularization and does not ensure non-forgetting. As shown in Fig. 6, while training for a new task (adding a new UAV), the energy consumption of DEN on Task 1 increases continuously, which means the catastrophic forgetting has already happened, and the knowledge of the old task in DEN has been forgotten.

This article has been accepted for publication in a future proceedings of this conference, but has not been fully edited. Content may change prior to final publication. Citation information: DOI10.1109/TCC.2023.3307582, IEEE Transactions on Cloud Computing

TABLE II: The performance comparison of different tasks.

Different Tasks	Scratch	Finetune	Progressive network	Pruning (75%)
One UAV	78.37	-	78.37	79.02
Two UAVs	80.42	75.86	75.35	75.47
Three UAVs	70.71	66.28	65.04	64.92
Four UAVs	61.55	59.56	58.21	57.69

Our multi-task controller is superior to DEN, and the energy consumption on all tasks remains fixed.

Then, we evaluate the performance of the multi-task controller in different tasks. These contenders are introduced as follows:

- Scratch: All new tasks are trained from scratch without any old task knowledge.
- Finetune: All new tasks are fine-tuned from Task 1 with the previous knowledge.
- Progressive network [38]: The agent is grown by adding nodes or weights for training new tasks.
- Pruning (75%): The agent is pruned by 75% weights, and the new tasks are adjusted by the multi-task controller.

Table II presents the energy consumption comparisons for various multi-task learning methods. For the first task (One UAV), Pruning (75%) exhibits marginally inferior performance compared to other methods, as it needs to compress the agent through pruning. However, for tasks 2 to 4, Pruning (75%) consistently surpasses other methods in nearly every situation. These results illustrate the effectiveness of the multi-task controller in constructing a compact and robust foundation suitable for multi-task learning.

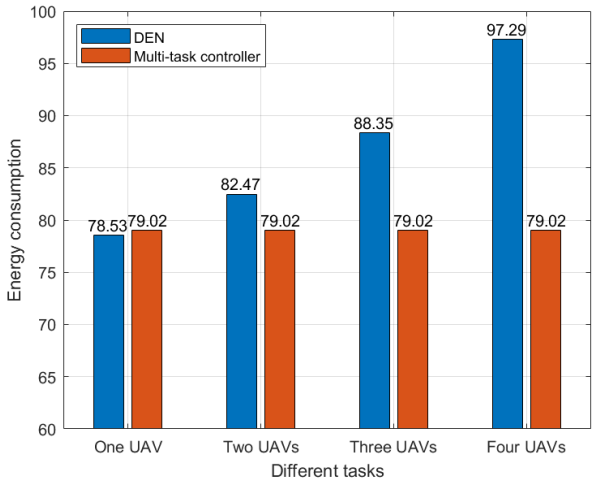


Fig. 6: Task 1 performances of different multi-task methods for different numbers of UAVs.

D. Performance Evaluation of the Action Refinement

In this section, we evaluate the performance of the action refinement on three benchmarks: DRL with LWS (LWS), DRL with Taboo Search (TS) [39], and DRL without action refinement (None). In the experiment, the iteration number of LWS and TS is set to 10, the length of the taboo list is set to 5, and the search neighborhood size is set to 10.

The performances of different action refinements are illustrated in Fig. 7. The results in Fig. 7 show that LWS and TS achieve lower loss than None, and LWS has faster convergence than TS. This is because the LWS adopts the light wolf with the highest channel gain to accelerate the search and the LWS also executes the constraint check to obtain validly high-quality solutions.

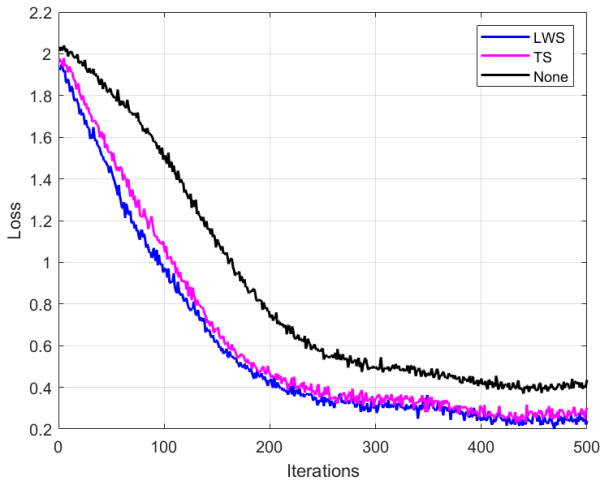


Fig. 7: Training performances of DRL with different action refinements.

E. Performance Evaluation of the MARS Framework

In this section, we evaluate the MARS framework's performance against three well-known DRL algorithms: TD3 [40], PPO [27], and SAC [28]. Table III displays the training time for the Initial task (one UAV), New task (adding a UAV), Old task (removing a UAV), and Average energy consumption across all DRLs. The MARS framework demonstrates minimal training time for the Old task and the lowest average energy consumption. The MARS framework's superior performance is attributed to two factors: (1) The multi-task controller stores parameters from old tasks, allowing agents to reuse policy knowledge without retraining when the number of UAVs changes. This results in reduced training time for Old tasks. (2) The LWS refines actions and enhances exploration, enabling a jump out of local extrema during the search process, yielding the lowest average energy consumption.

Then, five offloading schemes are selected as benchmarks to compare the offloading performance. These benchmarks are introduced as follows:

- Random Offloading (Random): The offloading decision for each UE is determined randomly.
- Local Execution (Local): All UEs execute tasks locally.
- Nearby Offloading (Remote): Each UE decides to offload its task to the nearest UAV.

TABLE III: The performance comparison of different DRLs.

Metric	Initial task	New task	Old task	Average energy consumption
MARS	234.23	126.57	0.45	64.95
TD3	258.13	192.46	130.58	68.37
PPO	263.41	196.41	131.82	67.45
SAC	248.76	185.33	122.63	65.69

- LWS: Light wolf search is applied to determine the optimal offloading decision for all UEs.
- Deep Reinforcement learning-based Online Offloading (DROO): This well-established DRL offloading scheme is designed for MEC systems and referenced in [41].

Fig. 8 presents the energy consumption for five offloading methods. MARS significantly reduces energy consumption compared to Local, Random, Remote, and DROO while delivering performance comparable to LWS. This is attributed to the MARS framework's ability to update its offloading policy using high-quality solutions generated by LWS and its capacity to construct a nonlinear mapping between state information and resource allocation, enabling faster high-quality decisions than traditional heuristic search methods.

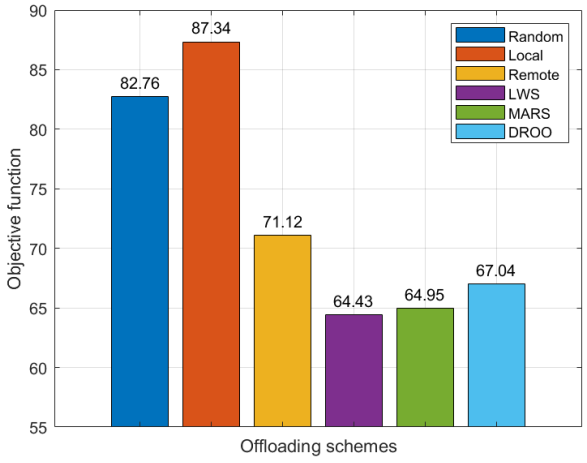


Fig. 8: Energy consumption of different offloading schemes.

VI. Conclusions

In this paper, we present a novel MARS framework for jointly optimizing UAV positions, phase-shifted diagonal matrix, computation offloading, and resource allocation in dynamic UAVs with IRS-assisted MEC systems, aiming to minimize the total energy consumption for all UAVs and UEs. The proposed MARS framework offers several advantages.

(1) The A2C structure reduces variance and enhances DRL policy learning. (2) The multi-head agent efficiently solves the MINLP problem with three output heads. (3) The multi-task controller adapts the agent for a varying number of UAVs. (4) LWS improves DRL exploration and expedites policy search. Simulation results show MARS outperforms existing benchmarks, highlighting its potential in dynamic MEC systems.

References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. PP, no. 99, pp. 1–1, 2017.
[2] Y. Du, K. Wang, K. Yang, and G. Zhang, "Energy-efficient resource allocation in uav based mec system for iot devices," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2019.
[3] F. Jiang, K. Wang, L. Dong, C. Pan, W. Xu, and K. Yang, "Ai driven heterogeneous mec system with uav assistance for dynamic environment: Challenges and solutions," *IEEE Network*, vol. 35, no. 1, pp. 400–408, 2020.
[4] L. Li, T. Jun Cui, W. Ji, S. Liu, J. Ding, X. Wan, Y. Bo Li, M. Jiang, C. W. Qiu, and S. Zhang, "Electromagnetic reprogrammable coding-metasurface holograms," *Nature Communications*, 2017.
[5] F. Jiang, L. Dong, K. Wang, K. Yang, and C. Pan, "Distributed resource scheduling for large-scale mec systems: A multiagent ensemble deep reinforcement learning with imitation acceleration," *IEEE Internet of Things Journal*, vol. 9, no. 9, pp. 6597–6610, 2021.
[6] X. Mu, Y. Liu, L. Guo, J. Lin, and N. Al-Dhahir, "Capacity and optimal resource allocation for irs-assisted multi-user communication systems," *IEEE Transactions on Communications*, vol. 69, no. 6, pp. 3771–3786, 2021.
[7] J. Chen, L. Guo, J. Jia, J. Shang, and X. Wang, "Resource allocation for irs assisted sgf noma transmission: A madrl approach," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 4, pp. 1302–1316, 2022.
[8] M. Hua, Q. Wu, and H. V. Poor, "Power-efficient passive beamforming and resource allocation for irs-aided wpcns," *IEEE Transactions on Communications*, vol. 70, no. 5, pp. 3250–3265, 2022.
[9] J. Chen, Y. Xie, X. Mu, J. Jia, Y. Liu, and X. Wang, "Energy efficient resource allocation for irs assisted comp systems," *IEEE Transactions on Wireless Communications*, vol. 21, no. 7, pp. 5688–5702, 2022.
[10] D. Xu, V. Jamali, X. Yu, D. W. K. Ng, and R. Schober, "Optimal resource allocation design for large irs-assisted swipt systems: A scalable optimization framework," *IEEE Transactions on Communications*, vol. 70, no. 2, pp. 1423–1441, 2022.
[11] S. Mao, N. Zhang, L. Liu, J. Wu, M. Dong, K. Ota, T. Liu, and D. Wu, "Computation rate maximization for intelligent reflecting surface enhanced wireless powered mobile edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 10, pp. 10 820–10 831, 2021.
[12] Z. Dai, Y. Zhang, W. Zhang, X. Luo, and Z. He, "A multi-agent collaborative environment learning method for uav deployment and resource allocation," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 8, pp. 120–130, 2022.
[13] Y. Cai, X. Jiang, M. Liu, N. Zhao, Y. Chen, and X. Wang, "Resource allocation for urllc-oriented two-way uav relaying," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 3, pp. 3344–3349, 2022.
[14] T. Li, S. Leng, Z. Wang, K. Zhang, and L. Zhou, "Intelligent resource allocation schemes for uav-swarm-based cooperative sensing," *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 21 570–21 582, 2022.
[15] Y. Shen, Y. Qu, C. Dong, F. Zhou, and Q. Wu, "Joint training and resource allocation optimization for federated learning in uav swarm," *IEEE Internet of Things Journal*, 2022.
[16] H. Kang, X. Chang, J. Mišić, V. B. Mišić, J. Fan, and Y. Liu, "Cooperative uav resource allocation and task offloading in hierarchical aerial computing systems: A mappo based approach," *IEEE Internet of Things Journal*, 2023.
[17] Y. Ye, L. Shi, X. Chu, R. Q. Hu, and G. Lu, "Resource allocation in backscatter-assisted wireless powered mec networks with limited mec computation capacity," *IEEE Transactions on Wireless Communications*, vol. 21, no. 12, pp. 10 678–10 694, 2022.
[18] N. Waqar, S. A. Hassan, A. Mahmood, K. Dev, D.-T. Do, and M. Gidlund, "Computation offloading and resource allocation in mec-enabled integrated aerial-terrestrial vehicular networks: A reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 21 478–21 491, 2022.
[19] J. Xu, B. Ai, L. Chen, Y. Cui, and N. Wang, "Deep reinforcement learning for computation and communication resource allocation in multiaccess mec assisted railway iot networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 23 797–23 808, 2022.

[20] C. Zhao, S. Xu, and J. Ren, "Aoi aware wireless resource allocation of energy harvesting powered mec systems," *IEEE Internet of Things Journal*, 2022.
[21] Z. Zhao, J. Shi, Z. Li, J. Si, P. Xiao, and R. Tafazolli, "Matching-aided-learning resource allocation for dynamic offloading in mmwave mec system," *IEEE transactions on wireless communications*, 2023.
[22] S. Mao, J. Wu, L. Liu, D. Lan, and A. Taherkordi, "Energy-efficient cooperative communication and computation for wireless powered mobile-edge computing," *IEEE Systems Journal*, vol. 16, no. 1, pp. 287–298, 2022.
[23] H. He, S. Zhang, Y. Zeng, and R. Zhang, "Joint altitude and beamwidth optimization for uav-enabled multiuser communications," *IEEE COMMUNICATIONS LETTERS*, vol. PP, no. 99, pp. 1–1, 2017.
[24] J. Zhang, X. Hu, Z. Ning, C. H. Ngai, L. Zhou, J. Wei, J. Cheng, and B. Hu, "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet of Things Journal*, 2017.
[25] F. Jiang, K. Wang, L. Dong, C. Pan, W. Xu, and K. Yang, "Deep-learning-based joint resource scheduling algorithms for hybrid mec networks," *IEEE Internet of Things Journal*, 2019.
[26] S. Li, B. Duo, X. Yuan, Y.-C. Liang, and M. Di Renzo, "Reconfigurable intelligent surface assisted uav communication: Joint trajectory design and passive beamforming," *IEEE Wireless Communications Letters*, vol. 9, no. 5, pp. 716–720, 2020.
[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[28] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.
[29] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997.
[30] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, "Deep reinforcement learning in large discrete action spaces," *arXiv preprint arXiv:1512.07679*, 2015.
[31] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in engineering software*, vol. 69, pp. 46–61, 2014.
[32] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[33] M. Papini, D. Binaghi, G. Canonaco, M. Pirotta, and M. Restelli, "Stochastic variance-reduced policy gradient," in *International conference on machine learning*. PMLR, 2018, pp. 4026–4035.
[34] S. Zou, T. Xu, and Y. Liang, "Finite-sample analysis for sarsa with linear function approximation," *Advances in neural information processing systems*, vol. 32, 2019.
[35] J. Bhandari, D. Russo, and R. Singal, "A finite time analysis of temporal difference learning with linear function approximation," in *Conference on learning theory*. PMLR, 2018, pp. 1691–1692.
[36] Y. F. Wu, W. Zhang, P. Xu, and Q. Gu, "A finite-time analysis of two time-scale actor-critic methods," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 617–17 628, 2020.
[37] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," *arXiv preprint arXiv:1708.01547*, 2017.
[38] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.
[39] D. Cvijović and J. Klinowski, "Taboo search: an approach to the multiple minima problem," *Science*, vol. 267, no. 5198, pp. 664–666, 1995.
[40] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.
[41] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, 2019.

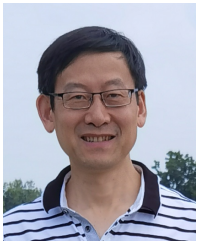
BIOGRAPHIES

Feibo Jiang received his B.S. and M.S. degrees in School of Physics and Electronics from Hunan Normal University, China, in 2004 and 2007, respectively. He received his Ph.D. degree in School of Geosciences and Info-physics from the Central South University, China, in 2014. He is currently an associate professor at the Hunan Provincial Key Laboratory of Intelligent Computing and Language Information Processing, Hunan Normal University, China. His research interests include artificial intelligence, fuzzy computation, Internet of Things, and mobile edge computing.

Yubo Peng received the B.S. degree from Hunan Normal University, Changsha, China, in 2019, where he is currently pursuing the master's degree with the College of Information Science and Engineering. His main research interests include federated learning and semantic communication.

Kezhi Wang received the Ph.D. degree in Engineering from the University of Warwick, U.K. He was with the University of Essex and Northumbria University, U.K. Currently, he is a Senior Lecturer with the Department of Computer Science, Brunel University London, U.K. His research interests include wireless communications, mobile edge computing, and machine learning.

Li Dong received the B.S. and M.S. degrees in School of Physics and Electronics from Hunan Normal University, China, in 2004 and 2007, respectively. She received her Ph.D. degree in School of Geosciences and Info-physics from the Central South University, China, in 2018. She is currently an associate professor at Hunan University of Technology and Business, China. Her research interests include machine learning, Internet of Things, and mobile edge computing.



Kun Yang received his PhD from the Department of Electronic & Electrical Engineering of University College London (UCL), UK. He is currently a Chair Professor in the School of Computer Science & Electronic Engineering, University of Essex, leading the Network Convergence Laboratory (NCL), UK. He is also an affiliated professor at UESTC, China. Before joining in the University of Essex at 2003, he worked at UCL on several European Union (EU) research projects for several years. His main research interests include wireless networks and communications, IoT networking, data and energy integrated networks and mobile computing. He manages research projects funded by various sources such as UK EPSRC, EU FP7/H2020 and industries. He has published 400+ papers and filed 30 patents. He serves on the editorial boards of both IEEE (e.g., IEEE TNSE, IEEE ComMag, IEEE WCL) and non-IEEE journals (e.g., Deputy EiC of IET Smart Cities). He was an IEEE ComSoc Distinguished Lecturer (2020-2021). He is a Member of Academia Europaea (MAE), a Fellow of IEEE, a Fellow of IET and a Distinguished Member of ACM.