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

Feibo Jiang, Yubo Peng, Kezhi Wang, Li Dong and Kun Yang

***Abstract*—This paper studies a dynamic mobile edge computing (MEC) system assisted by unmanned aerial vehicles (UAVs) and intelligent reflective surfaces (IRSs). We propose a scaleable resource scheduling algorithm to minimize the energy consumption of all UEs and UAVs in the MEC system with a variable number of UAVs. We propose a Multi-tAsk Resource Scheduling (MARS) framework based on deep reinforcement learning (DRL) to solve the problem. 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 resources, 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 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. The 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

In recent years, the number of computation-intensive and latency-sensitive applications such as automated driving, mixed reality, and metaverse has grown fast, making it challenging for user equipments (UEs) to handle these tasks locally. Because of recent advances in mobile edge computing (MEC) [1], UEs may offload computationally heavy tasks to neighboring MEC servers. However, the deployment of stationary MEC servers in the terrestrial region may not be cost-effective, particularly in temporary, unexpected, or emergency scenarios [2]. Unmanned aerial vehicle (UAV)-assisted MEC has been suggested to provide flexible and efficient computation and communication services to terrestrial UEs [3]. Nonetheless, physical signal blocking and interference of tall buildings and other obstacles will lead to frequent obstructions of wireless communication channels between UAVs and UEs.

The intelligent reflecting surface (IRS) technology can adjust the amplitude and phase of the incident signals by massive low-cost passive reflecting elements, and it is a promising technology to reorganize the propagation environments and enhance network performance [4]. The IRS uses significantly less energy than existing methods like active relay and backscattering communication. In UAV-assisted MEC systems, the IRS can be deployed easily on building facades, ceilings, and walls to reflect the signals between UAVs and users when the connections are blocked.

Despite the advantages of UAV with IRS-assisted MEC systems, there are some issues that should be resolved: (1) The resource allocation is typically a continuous variable while the offloading decision is typically an integer variable, creating a mixed-integer nonlinear programming (MINLP) problem [5]. The MINLP is challenging to tackle using conventional techniques. (2) The wireless environment becomes more complicated as the number of UEs and UAVs changes, particularly when the IRS is taken into account in the UAV-assisted MEC system.

For overcoming these issues, we design a dynamic UAV with IRS-assisted MEC system. A Multi-tAsk Resource Scheduling (MARS) framework based on deep reinforcement learning (DRL) is presented to minimize the energy consumption of all UEs and UAVs for the variable number of UAVs in the system. By jointly optimizing the UAV location, offloading decision, resource allocation, and phase-shifted diagonal matrix, the MARS framework is achieved for online resource scheduling. The core innovations of the study can be 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.
- *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 resources in the actor network, and a critic head is introduced to estimate the state value of the selected action in the critic network.

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 Hunan Provincial Key Laboratory of Intelligent Computing and Language Information Processing, 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 Commerce, Changsha, China, Kun Yang (kunyang@essex.ac.uk) is with the School of Computer Sciences and Electrical Engineering, University of Essex, CO4 3SQ, Colchester, UK.

- *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 hunt 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 rest of the paper can be organized into the following sections. In Section II, some related works are reviewed. Section III describes the system model and problem formulation. Section IV introduces the structure of the MARS framework. Section V presents the simulation results, while Section VI discusses the conclusions.

II. Related Work

A. Resource Allocation in IRS Aided Communication Systems

In [6], authors studied the fundamental capacity limits of IRS-assisted multi-user wireless communication systems and jointly optimized the IRS reflection matrix and wireless resource allocation under the constraints of a maximum number of IRS reconfiguration times. In [7], three multi-agent deep reinforcement learning-based frameworks were proposed to solve the problem under three different IRS cases, and thus dynamically assigned network resources for each grant-free user. In [8], an IRS-assisted wireless-powered communication network was investigated, and three types of IRS beamforming configurations were presented to strike a balance between the system performance and signaling overhead as well as implementation complexity. A novel IRS-assisted coordinated multi-point system was proposed in [9], which aimed to maximize the energy efficiency of this system. The authors in [10] studied the optimal resource allocation algorithm design for large IRS-assisted simultaneous wireless information and power transfer (SWIPT) systems to minimize the total transmit power of the base station. In [11], the authors aimed to apply the IRS technique to improve the efficiency of wireless energy transfer (WET) and task offloading when wireless links between the hybrid access point and Internet of Things devices are hostile.

B. Resource Allocation in UAV Aided Communication Systems

A resource allocation algorithm for the UAV networks based on multi-agent collaborative environment learning was proposed in [12], which improved the utility of the UAV networks. The authors in [13] combined the advantages of both UAV and ultra-reliable low-latency communication (URLLC) to investigate the resource allocation for a URLLC-enabled two-way UAV relaying system, which maximized the transmission rate of the backward link. In [14], a UAV-swarm-based hierarchical network architecture was proposed to jointly schedule sensing, computing, and communication resources, thus improving computing resource utilization. Considering the limited energy supply of UAVs, the authors in [15] explored how to minimize UAVs' overall training energy consumption by jointly optimizing the local convergence threshold, local iterations, computation resource allocation, and bandwidth allocation. To maximize the amount of computed tasks while satisfying heterogeneous quality-of-service (QoS) requirements of tasks through the joint optimization of UAV resource allocation and task offloading, a multi-agent proximal policy optimization (MAPPO)-based algorithm was designed in [16].

C. Resource Allocation in MEC Systems

Considering the limited computation capacity of the MEC server and the QoS and energy-causality constraints per IoT node, the authors [17] proposed two resource allocation schemes to maximize the total computation bits of all IoT nodes and the system computation energy efficiency, respectively. To minimize the total computation and communication overhead of the joint computation offloading and resource allocation strategies for the vehicles system, a decentralized value-iteration-based reinforcement learning (RL) approach was developed in [18] as the feasible solution. The authors in [19] investigated the dynamic resource management issue of joint subcarrier assignment, offloading ratio, power allocation, and computation resource allocation in MEC-assisted railway IoT networks. In [20], the data analysis scenario in IoT architecture was investigated, and the authors aimed to maximize the long-term average system utility by jointly optimizing the communication resource allocation, the data generating and discarding strategies, and the computing resource allocation. The authors in [21] studied the resource allocation problem for the conceived mmWave MEC system with a dynamic offloading process, and a matching-aided-learning (MaL) resource allocation scheme was proposed to tackle this problem. A harvest-and-offload protocol was proposed in [22], which jointly scheduled wireless energy transfer and cooperative computation offloading and aimed to minimize the total energy consumption.

However, none of the above works studied the UAV and IRS-assisted MEC system with a variable number of UAVs under fast-fading channels. Multi-task learning is a workable method in dynamic environments. Hence, in this paper, we propose a MARS framework for the UAV with IRS-assisted MEC system, which can achieve real-time resource scheduling in dynamic environments.

III. System Model and Problem Formulation

The system model of the UAV with IRS-assisted MEC system is shown in Fig. 1. In the system, we consider there are N UEs, denoted as a set of $\mathcal{N} = \{1, 2, \ldots, N\}$, and each UE has a computational task to be executed and all UEs are randomly distributed. Then, we consider there are L IRSs, denoted as a set of $\mathcal{L} = \{1, 2, \ldots, L\}$, and all IRSs are mounted on buildings. Also, there are some UAVs,

denoted as a set of $\mathcal{M} = \{1, 2, \ldots, M\}$, which have edge server enhanced. A control center (or central cloud) can collect the environment information (e.g., the number of UEs, IRSs, and UAVs, and the channel state information) as well as the task information (e.g., the data size of the task and required computing resources of the task) from the UEs, and carry out the resource scheduling for the system. Since the small amount of collected information, the communication overhead of the central cloud is ignored in the study.

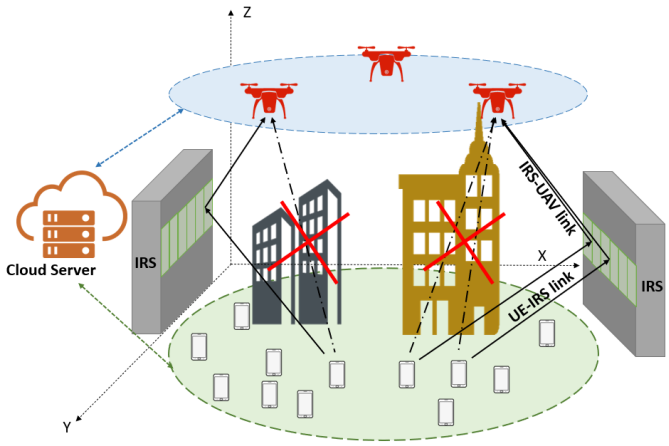


Fig. 1: UAV with IRS-assisted MEC network.

A. MEC Model

We assume that each UE has a computationally intensive task A_i which can be expressed as follows:

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

where D_i represents the data size that needs to be transmitted to the UAV for execution, and F_i represents the total number of required CPU cycles of the task.

Each UE can offload the whole task to one UAV or stay locally for execution. We use a_i^{loc} to denote the task is executed locally, and a_{ij}^{uav} to denote the task is offloaded to the j-th UAV. Then, we have the following constraints:

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

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

where $a_i^{loc} = 0$ means that the i-th UE does not execute the task locally, while $a_i^{loc} = 1$ means that the i-th UE decides to execute the task locally. Also, $a_{ij}^{uav} = 0$ means 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 of the local computing for 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} means that the local computation capacity of the i-th UE, which is measured by CPU cycles per second. The computation capacity of the i-th UE is limited by

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

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

The energy consumption in the local computing phase is given by

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

where $\gamma_1 \geq 0$ is the effective-switched capacitance and $v_1 \geq 1$ is the positive constant. To match the realistic measurements, we set $\gamma_1 = 10^{-27}$ and $v_1 = 3$ [23].

2) Remote computing: The execution time of the task for the i-th UE at the j-th UAV can be 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} means that the computation capacity provided by the j-th UAV to the i-th UE. The energy consumption in the remote computing phase is

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

where γ_2 is the effective-switched capacitance and $v_2 \geq 1$ is the positive 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

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

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

B. IRS Model

Assume that UEs are located in the 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 for 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 between the i-th UE and the l-th IRS is expressed as

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

Similarly, the distance between the l-th IRS and the j-th UAV is given by

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

In this paper, we assume that the communication path between the i-th UE and the j-th UAV is divided into two sections: the UE-IRS link and the IRS-UAV link. We assume that each UE only uses one IRS for signal transmission so that the channel gain of the UE-IRS link for the i-th UE can be expressed as $h_{i,l}^{U,R}$. The channel gain of the UE-IRS link $h_{i,l}^{U,R} \in \mathbb{C}^{K\times 1}$ is given by

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

where α is the path loss exponent of the channel between the i-th UE and the l-th IRS, β is the path loss at the reference distance of one meter, $\phi_{i,l}^{U,R} = \frac{|X_i^U - X_l^R|}{d_{i,l}^{U,R}}$ is the cosine value of the angle of arrival (AOA) of the signal between the i-th UE and the l-th IRS.

We assume that each UE can offload the task to only one UAV, hence the channel gain of the IRS-UAV link for the i-th UE can be expressed as $h_{l,j}^{R,M} \in \mathbb{C}^{K\times 1}$ can be expressed as

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

where the right term in Eq. (14) above is the array response of the l-th IRS which have K reflecting elements, d is the antenna separation distance between two elements, $\phi_{l,j}^{R,M} = \frac{|X_l^R - X_j^M|}{d_{l,j}^{R,M}}$ is the cosine value of the angle of departure (AOD) of the signal between the j-th UAV and the l-th IRS, λ is the carrier wavelength.

We assume 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

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

The achieved data transmission rate between the i-th UE and the j-th UAV can be expressed as

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

where B denotes the channel bandwidth, σ^2 denotes the noise spectral density, P_{ij}^{tra} represents the transmitting power between the i-th UE and the j-th UAV, and we can obtain the P_{ij}^{tra} from [24]. Note that we assume UEs offload their tasks to the UAV via orthogonal frequency division multiplexing (OFDM) channels, which means that there is no interference between each other. Then, the transmission time can be expressed as

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

The energy consumption of transmission can be calculated as follows:

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

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 the T_j^{hov} means the hover time of the j-th UAV.

The flight energy of the UAV can be expressed as

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

where P_j^{fly} means the flight power of the j-th UAV, L_j is the flight distance of the j-th UAV, and v_j is the flight velocity of the j-th UAV, which is a constant value.

D. Problem Formulation

In this paper, we aim to minimize the total energy consumption of the UAV with IRS-assisted MEC system, and the optimization problem can be formulated as follows:

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

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 propose a MARS framework based on DRL to minimize the energy consumption of all UEs and UAVs for the variable number of UAVs in the UAV with IRS-assisted MEC system. The data flow of the MARS framework can be described as follows: first, the central cloud collects the global environment information as well as task information from the system. Then, the central cloud executes the MARS algorithm, updates the locations of UAVs and the phase-shifted diagonal matrix of IRSs, and performs the online decisions of the user association and resource allocation for each UE. Finally, based on the decision received from the central cloud, each UE can offload the task to the suitable UAV, and then receive the results accordingly.

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^M_{j,t}, Y^M_{j,t}, Z^M_{j,t}) 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 workflow of the MARS framework is described in **Algorithm 1**. For saving flight energy, we assume that UAVs need to be redeployed only when the number of UAVs is changed. We use large-scale path-loss fuzzy c-means clustering algorithm (LS-FCM) to optimize the locations of UAVs and obtain the $\mathbf{p}$ in $P0$ [25], which is based on the large-scale path-loss factors. Then, we introduce a quantitative passive beamforming method to solve the phase-shifted diagonal matrix $\mathbf{\Theta}$ according to the positions of UAVs and UEs. Finally, we design a novel A2C to generate the offloading decision $\mathbf{a}$ and computational resource allocation $\mathbf{f}$ of 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: $\mathcal{S}_t = \{\mathcal{S}_{i,t} | i \in \mathcal{N}\}$ where $\mathcal{S}_{i,t} = \{H_{i,t}, D_{i,t}, F_{i,t}\}$ is the environment information of the i-th UE at the t-th timeslot, in which $H_{i,t} = \{h_{i,j,t} | j \in \mathcal{M}\}$ is the channel gain between the i-th UE and all UAVs, $D_{i,t}$ and $F_{i,t}$ are the task attributions of the i-th UE.
- Action: $\mathcal{A}_t = \{\mathcal{A}_{i,t} | i \in \mathcal{N}\}$ where $\mathcal{A}_{i,t} = \{a_{i,t}, f_{i,t}\}$ is the resource scheduling decision of the i-th UE at the t-th timeslot. $a_{i,t} \in \mathbb{N}$ is the user association and $f_{i,t} \in \mathbb{R}$ is the allocated resource of the i-th UE.
- Reward: $\mathcal{R}_{i,t}$ is defined as the reciprocal of the energy consumption of the i-th UE'task, and $\mathcal{R}_t$ is defined the reciprocal of the objective function at the t-th timeslot.
- 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 applied to update the policy of the agent. The first in first out (FIFO) strategy is introduced to update transitions 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 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.

For simplicity, we consider discrete phase shift angles in this paper, and the phase shift $\theta_{k,i,l,j}$ of the IRS is chosen from the following set of $\Psi \triangleq \left\{\frac{2\pi}{N_p} i, i = 0, 1, \ldots, N_p - 1\right\}$, where N_p denotes the number of the phase shift values that can be selected for every element. When the signals from different paths are combined coherently at the UAV, the coherent signal construction can maximize the received signal power, thereby maximizing the achievable rate. Hence, we optimize the phase shift $\theta_{k,l,i,j}$ of the k-th reflecting element of the l-th IRS between the i-th UE and the j-th UAV with the following equation [26]:

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

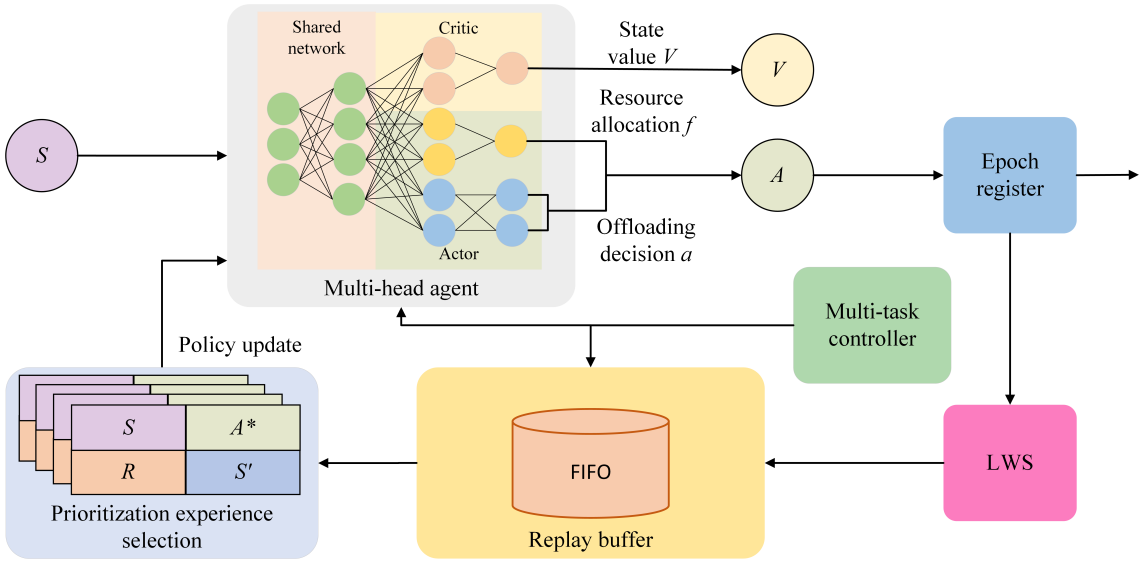


Fig. 2: The DRL part of the MARS framework.

C. Multi-head Agent

We propose an A2C structure with the multi-head agent to solve the MINLP. The multi-head agent has two networks with three heads. The critic head is presented to calculate the state-value function of the action, and the actor head is applied to predict the resource allocation and offloading decision. In the learning process, the actor network and critic network share parameters of the shallow part of the network to extract the common features and then adjust the independent parameters of the subsequent part of the networks to learn the unique features of each head respectively. The structure of the multi-head agent is shown in Fig. 3.

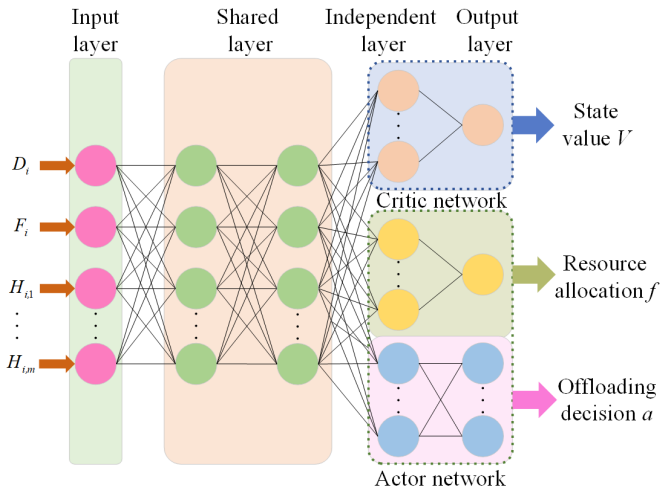


Fig. 3: The multi-head agent.

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].

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

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

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 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:

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: $\mathbf{x}_\alpha$.

```
Initialize the wolf population.
Initialize a, A and C.
Select elite wolves x_α, x_β and x_δ.
Define initial light wolf as x_ω = (a_0, f_0).
while t < t_max do
    for each wolf do
        Update the position of the current wolf by Eqs. (38)-(39).
    end for
    Calculate the finesses of all wolves.
    Update elite wolves x_α, x_β and x_δ.
    Update the light wolf x_ω according to channel gains.
    Update a by Eq. (37).
    Update A and C by Eqs. (35) - (36).
    t = t + 1.
    Carry out the constraint check and obtain valid solutions.
end while
```

F. Convergence Analysis of the MARS Framework

1) Propositions: We consider that the critic network uses linear function approximation to estimate the state-value function, which can be represented as $V(\cdot;\boldsymbol{\omega}) = \phi^\top(\cdot)\boldsymbol{\omega}$ [32]. We define $\mathbf{A}$ and $\mathbf{b}$ as:

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

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

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 approximation error of the linear function 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 $V(\cdot)$ is the state-value function.

Throughout this paper, we assume the approximation error for all potential policies is uniformly bounded, namely

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

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

2) Assumptions:

Assumption 1. *For all potential policy parameters $\boldsymbol{\theta}$, the matrix $\mathbf{A}$ defined above is negative definite and has the maximum eigenvalues as $-\lambda$ [34].*

Assumption 2. *For a fixed $\boldsymbol{\theta}$, denote $\mu_{\boldsymbol{\theta}}(\cdot)$ as the stationary distribution induced by the policy $\pi_{\boldsymbol{\theta}}(\cdot|\mathcal{S})$ and the transition probability measure $\mathcal{P}(\cdot|\mathcal{S},\mathcal{A})$. Consider a Markov chain*

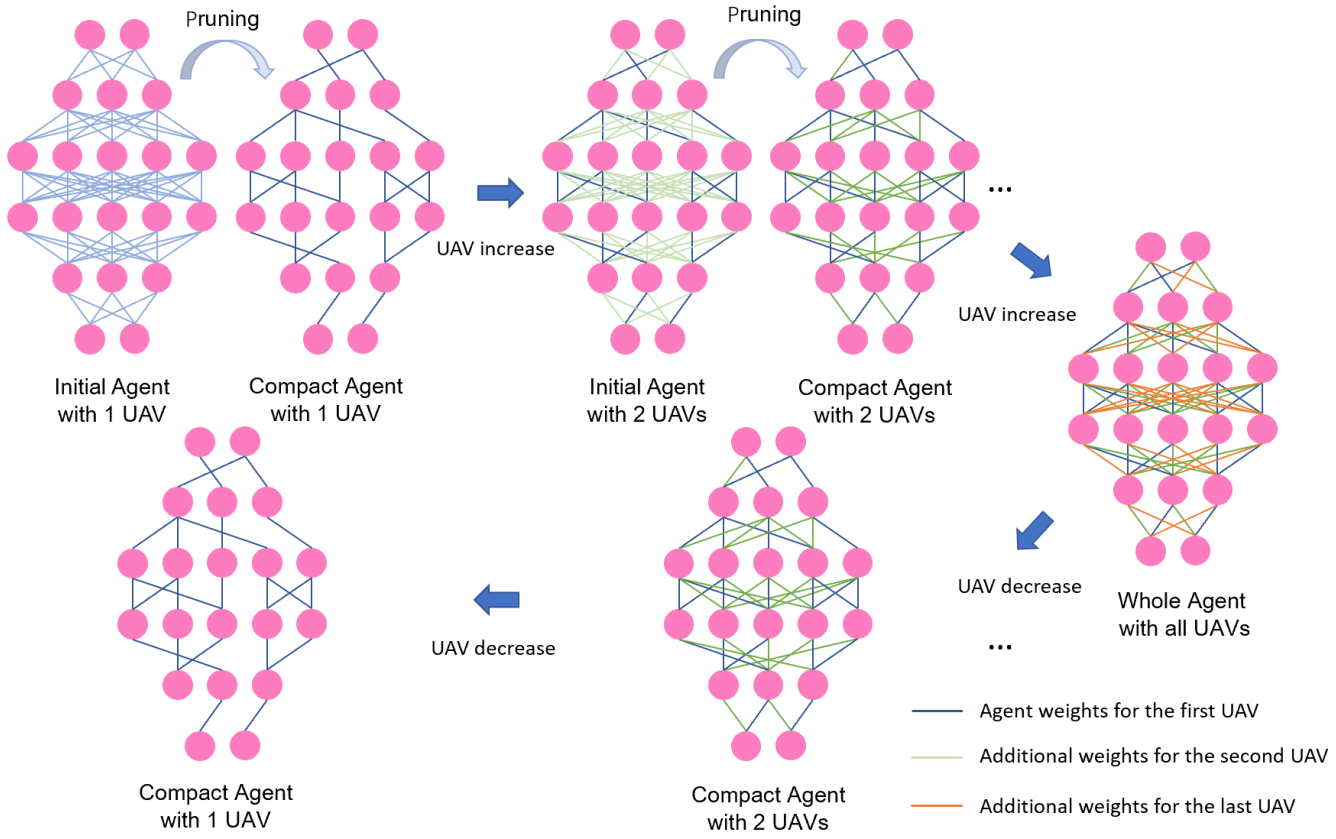


Fig. 4: The multi-task controller with varying number of UAVs.

generated by the rule $\mathcal{A}_t \sim \pi_{\boldsymbol{\theta}}(\cdot|\mathcal{S}_t), \mathcal{S}_{t+1} \sim \mathcal{P}(\cdot|\mathcal{S}_t, \mathcal{A}_t)$. *Then, there exists* $m > 0$ *and* $\rho \in (0,1)$ *such that [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 $d_{TV}(\cdot)$ *is the total variation norm between two probability measure.*

Assumption 3. *Let* $\pi_{\boldsymbol{\theta}}(\mathcal{A}|\mathcal{S})$ *be a policy parameterized by* $\boldsymbol{\theta}$. *There exist constants* $L, B, L_l > 0$, *such that for all given state* $\mathcal{S}$ *and action* $\mathcal{A}$ *it holds [33]:*

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

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

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

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. *Suppose Assumptions 1-3 hold and we choose* $\alpha_t = c_\alpha/(1+t)^\sigma$, *where* $\sigma \in (0,1)$ *and* $c_\alpha > 0$ *are constants. At the* t*-th iteration, the critic satisfies [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 $\boldsymbol{\omega}_k$ *is the critic parameter and* η_k *is the discount factor at the* k*-th iteration of the actor's update,* $\mathcal{E}(t)$ *is a bounded sequence, then we have*

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

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

Theorem 2. *Suppose Assumptions 1-3 hold and we choose* $\alpha_t = c_\alpha/(1+t)^\sigma$ *and* $\beta_t = c_\beta/(1+t)^\nu$, *where* $0 < \sigma < \nu < 1$ *and* c_α *and* $c_\beta \leq \lambda^{-1}$ *are positive constants, we have [36]*

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

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

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

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:

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

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

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

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

V. Numerical Results

A. Simulation Settings

In this section, we provide some simulation results to evaluate the performance of the MARS framework. The simulation is conducted in Python 3.7 and TensorFlow 2.2.0 environment running on an Intel Xeon CPU with 32GB RAM and a Tesla T4 GPU with 15 GB SGRAM. The initial multi-head agent has two shared layers with 64 and 128 neurons, respectively. The number of neurons in each independent layer is set to 32. The replay buffer size is set to 8000, and the minibatch size is set to 50. Dropout is applied to the training process to avoid overfitting. In the multi-task controller, the compression ratio is set to 75% for each UAV. In the LWS, the size of the wolf pack is set to 4 and the maximum iteration number is set to 10. Other parameters used in the simulations are summarized in Table I unless otherwise specified.

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.

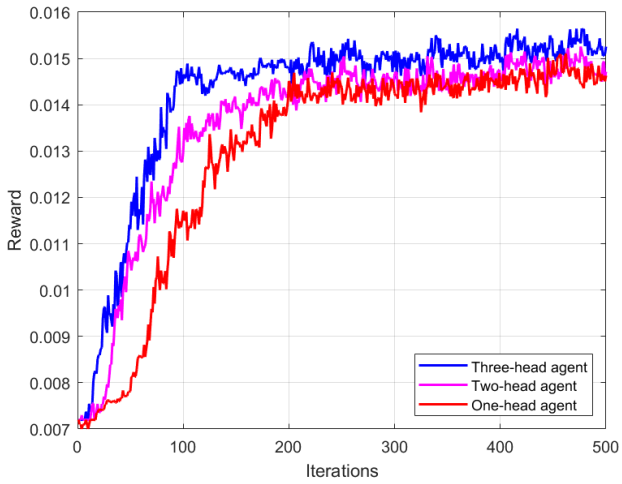


Fig. 5: Rewards of different agents.

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.

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 characterizes the energy consumption of different multi-task learning methods. It can be seen that for the first task (One UAV), Pruning (75%) performs slightly worse than the others since it has to compress the agent via pruning. Then, for tasks 2 to 4, Pruning (75%) outperforms the others in almost all cases, which shows the superiority of the multi-task controller on building a compact and unforgettable base for multi-task learning.

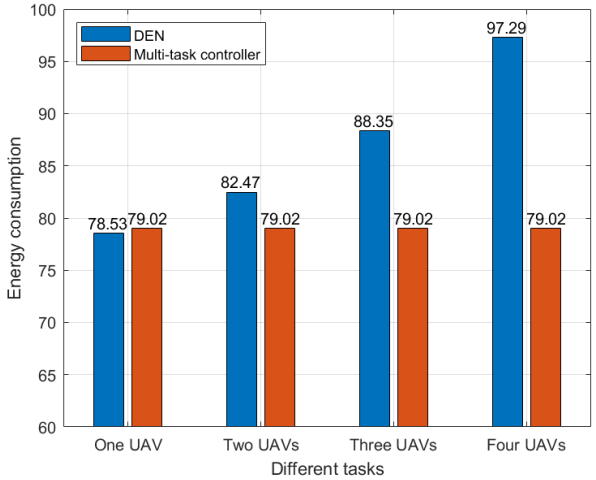


Fig. 6: Task 1 performances of different multi-task methods for different numbers of UAVs.

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.

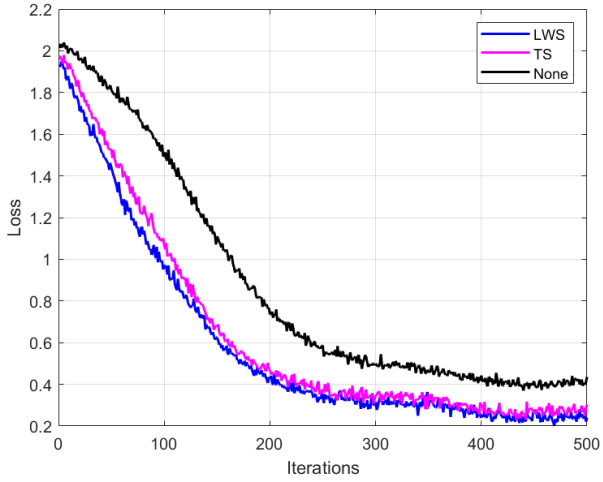


Fig. 7: Training performances of DRL with different action refinements.

E. Performance Evaluation of the MARS Framework

In this section, we evaluate the performance of the whole MARS framework. We first compare the MARS framework with three well-known DRL algorithms: TD3 [40], PPO [27] and SAC [28]. TABLE III characterizes the training time of the Initial task (only one UAV at the beginning), New task (adding a new UAV), Old task (removing a UAV), and Average energy consumption in all DRLs. It can be seen that the MARS framework achieves the least training time for the old task and the lowest average energy consumption. The superiority of the MARS framework can be explained as follows: (1) The multi-task controller can store the parameters of the old tasks, and the agent can reuse the stored policy knowledge when the number of UAVs is changing without retraining the agent again. Hence, the training time for the old task in the MARS framework is the least. (2) The LWS refines the action and enhances the exploration, which leads to jump out of the local extremum in the search process. Hence, the average energy consumption of the MARS framework is the lowest.

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

- Random offloading (Random) denotes that the offloading decision of each UE is randomly determined.
- Local executed (Local) denotes that all UEs execute tasks locally.

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

- Offloading nearby (Remote) denotes that each UE decides to offload the task to the closest UAV.
- LWS denotes light wolf search applied to find the best offloading decision for all UEs.
- Deep reinforcement learning-based online offloading (DROO) is a celebrated DRL offloading scheme for the MEC system [41].

The energy consumption of these five offloading methods is shown in Fig. 8, it can be seen that the energy consumption of the proposed MARS framework is much lower than the energy consumption of Local, Random, Remote and DROO. Meanwhile, the energy consumption of the proposed MARS framework is close to the LWS. This is because the MARS method can update the offloading policy from the high-quality offloading solutions generated by LWS. Moreover, the MARS method can construct a nonlinear mapping from the state information to the offloading decision and resource allocation, which can make high-quality decisions much faster than traditional heuristic search methods.

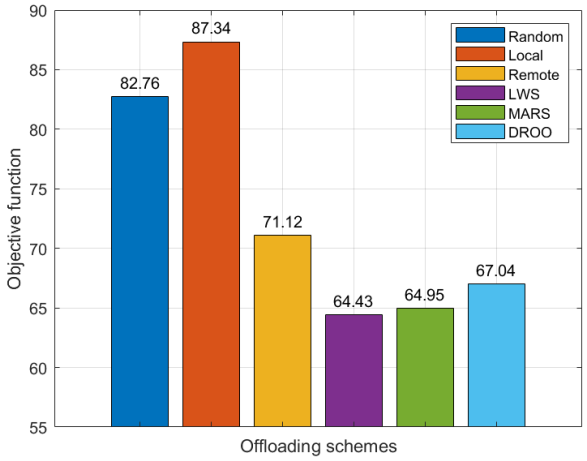


Fig. 8: Energy consumption of different offloading schemes.

VI. Conclusion

In this article, a novel MARS framework has been proposed to jointly optimize the positions of UAVs, phase-shifted diagonal matrix, computation offloading, and recourse allocation for dynamic UAV with IRS-assisted MEC systems. The objective of the MARS framework is to minimize the sum of energy consumption for all UAVs and UEs. Overall, the proposed MARS framework has the following advantages.

(1) The A2C structure is introduced to reduce variance and enhance the policy learning of the DRL. (2) The multi-head agent with three output heads is presented to solve the MINLP efficiently. (3) The multi-task controller is used to adjust the agent for adapting to the varying number of UAVs. (4) The LWS is applied to enhance the exploration of the DRL and accelerate the hunt for the best policy.

The simulation results demonstrate that the MARS framework has better performance than the existing benchmarks, and it exhibits enormous 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.

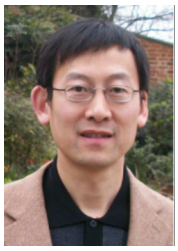
BIOGRAPHIES

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 federated learning, 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, mobile edge computing. He manages research projects funded by various sources such as UK EPSRC, EU FP7/H2020 and industries. He has published 150+ journal papers and filed 10 patents. He serves on the editorial boards of both IEEE and non-IEEE journals. He is a Senior Member of IEEE (since 2008) and a Fellow of IET (since 2009).