AoI Minimization in Heterogeneous MEC Networks: A Federated Learning-Assisted Hybrid DRL and Convex Approach

Xiaoying Liu, *Senior Member, IEEE*, Junhao Zheng, Kechen Zheng, *Senior Member, IEEE*, Jia Liu, *Senior Member, IEEE*, Tarik Taleb, *Senior Member, IEEE*, and Norio Shiratori, *Life Fellow, IEEE*

***Abstract*—This paper investigates a dynamic heterogeneous mobile edge computing network (HMECN), where mobile devices (MDs) could offload their full tasks to a small base station (SBS) directly or the macro base station (MBS) in direct or relay mode. As age of information (AoI) is a comprehensive and accurate metric to capture the freshness of computation results, we formulate a long-term weighted sum AoI (LWSA) minimization problem in the HMECN by jointly optimizing the offloading decisions of MDs as well as the bandwidth and computation resource allocation of all base stations, subject to energy, delay and peak AoI constraints. To address the formulated non-convex mixed integer nonlinear programming problem, we decompose it into the offloading decision optimization (ODO) top-problem and the resource allocation optimization (RAO) sub-problem. Based on the decomposition, we propose a federated learning (FL)-assisted hybrid DRL and convex approach that is comprised of a safe multi-agent DRL algorithm, convex optimization and FL. The ODO top-problem is solved by the safe multi-agent DRL algorithm, which strictly ensures that the actions of each agent do not exceed its energy constraint and then paves the way for using convex optimization to solve the RAO sub-problem. FL is used to alleviate the training instability problem aggravated by multi-agent settings via breaking the limitation of partial knowledge for each individual agent. Simulation results demonstrate the superiority of the proposed approach in terms of the LWSA, convergence, scalability and robustness in dynamic environments.**

***Index Terms*—Age of information, mobile edge computing, heterogeneous network, multi-agent deep reinforcement learning, federated learning.**

This work was supported in part by the National Natural Science Foundation of China under Grant 62372412 and Grant 62372413; in part by Zhejiang Provincial Natural Science Foundation of China under Grant No. LZ26F010008; in part by the Fundamental Research Funds for the Provincial Universities of Zhejiang under Grant RF-B2024002; in part by JSPS KAKENHI Grant Number JP25K15087; in part by the Project of Cyber Security Establishment with Inter-University Cooperation; in part by the Federal Ministry of Research, Technology, and Space (BMFTR), Germany, through the Project 6GEM+ under Grant 16KIS2411; and in part by the European Union through the 6G-Path project under Grant 101139172. (*Corresponding author: Kechen Zheng*)

X. Liu, J. Zheng, and K. Zheng are with the School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China. E-mail: {xiaoyingliu, 211123120116, kechenzheng}@zjut.edu.cn.

J. Liu is with the Center for Strategic Cyber Resilience Research and Development, National Institute of Informatics, Tokyo 101-8430, Japan. E-mail: jliu@nii.ac.jp.

T. Taleb is with the Faculty of Electrical Engineering and Information Technology, Ruhr University Bochum, Bochum 44801, Germany. E-mail: tarik.taleb@rub.de.

N. Shiratori is with the Research and Development Initiative, Chuo University, Tokyo 112-8551, Japan. E-mail: norio.shiratori.e8@tohoku.ac.jp.

I. Introduction

A. Background

With the rapid development of mobile communication technologies and wireless networks, computing-intensive and delay-sensitive network applications, such as autonomous driving and virtual reality, are emerging at an unprecedented speed in recent years [1]. However, the tasks of these applications are difficult to be processed timely in mobile devices (MDs) due to the limited battery capacity and computing capabilities. Although cloud computing can relieve the computation tasks of MDs, the influx of massive tasks into the cloud may impose a heavy burden on backhaul links, thereby causing unacceptable end-to-end service delay [2]. Therefore, mobile edge computing (MEC), which migrates cloud computing capability to the edge of networks via integrated MEC servers at base stations (BSs), has emerged as a promising paradigm [3]. Compared with cloud computing, MEC removes long backhaul delay and enjoys lower energy consumption of MDs, by enabling MDs to offload their tasks to nearby BSs via wireless links [4].

Due to the contradiction between the increasing number of MDs and the limited resources for computation offloading, a critical challenge, MEC networks are confronted with, is how to make appropriate offloading decisions and reasonably allocate resources to MDs, so as to optimize various network performance. Many works [5]–[7] have been devoted to addressing this challenge. Specifically, Chen *et al.* [5] adopted the deep deterministic policy gradient algorithm to minimize the long-term average delay of all tasks by jointly optimizing the transmission power of devices and the computation resource allocation of the BS. Tan *et al.* [6] proposed an alternative method to minimize the energy consumption of all MDs in a multi-user collaborative scenario by optimizing the task offloading and resource allocation. With the help of intelligent reflecting surface (IRS), Chen *et al.* [7] proposed two iterative algorithms to maximize the sum computation rate by optimizing offloading decisions, time allocation and IRS beamforming. Nevertheless, the above works [5]–[7] focused on the MEC networks with a single BS, which makes it difficult to timely process massive tasks from the increasing number of MDs.

Consequently, there have been many works [8]–[11] on the MEC networks with multiple BSs represented by the heterogeneous MEC networks (HMECNs), which are in accordance with the real-world cellular networks. HMECN is formed by the deployment of multiple low-power small BSs (SBSs) in the underlying area of a traditional cellular BS, i.e.,

TABLE I
DIFFERENCES BETWEEN THIS PAPER AND THE RELATED WORKS

Ref.	Optimization objective	Approach	Multiple servers	Indivisible task	Decision making	Computation resource allocation	Communication resource allocation	Server cooperation
[14]	AoI, latency, and energy consumption	Single-agent DRL	✓		✓			
[17]	AoI and energy consumption	Single-agent DRL	✓		✓			✓
[18]	AoI, rental price, and energy consumption	Single-agent DRL	✓		✓			
[19]	AoI	Heuristic algorithm	✓		✓			
[20]	Energy consumption	Alternative algorithm	✓		✓	✓	✓	
[21]	AoI	FL-assisted multi-agent DRL	✓	✓	✓		✓	
[22]	AoI	Alternative algorithm	✓	✓	✓	✓		
[23]	AoI and energy consumption	Multi-agent DRL	✓		✓	✓		
Ours	AoI	FL-assisted hybrid DRL and convex approach	✓	✓	✓	✓	✓	✓

Note: ✓ denotes the existence of the feature.

macro BS (MBS) [8], and can offload MDs' tasks from the MBS to SBSs, thereby alleviating the burden on the MBS. Undoubtedly, the issues of offloading decision making and resource allocation in HMECNs are more complicated than those in MEC networks with a single BS. Under the offloading model where MDs directly offload tasks to the MBS or a SBS, Dai *et al.* [9] aimed to minimize the network energy consumption by jointly optimizing the offloading decisions and computation resource allocation with the task completion delay constraint. Xu *et al.* [10] applied non-orthogonal multiple access technology in HMECNs, and minimized the energy consumption of all MDs by jointly optimizing offloading decisions, local CPU frequency, transmit power, as well as subchannel and computation resource allocation. Different from the previous works [9], [10] where every BS is integrated with a server, Liu *et al.* [11] focused on the scenario where the MBS and SBSs connect to a server room, and minimized the overall task completion delay of the HMECN by jointly optimizing the BS selection, bandwidth allocation, offloading decision, and computation resource allocation.

So far, the research on MEC networks mainly focuses on the analysis and optimization of delay [5], [11], energy consumption [6], [9], [10], and computation rate [7]. These traditional metrics, however, cannot capture the freshness of computation results which is critical for computing-intensive and delay-sensitive applications [12], [13]. For example, in the internet of vehicle applications, the real-time task processing is essential for instant navigation and driving decisions. Outdated computation results potentially have a negative impact on vehicle performance and driving safety [14]. To quantify the freshness of computation results, age of information (AoI) proposed in [15] is employed to evaluate the real-time performance of MEC networks. Different from the delay characterizing the time elapsed from the start of transmission to the completion of computation, AoI takes the whole lifetime of tasks into account and is defined as the time elapsed since the generation of the latest received computation result [16]. Therefore, it is essential to study the AoI issue in MEC networks with multiple servers.

B. Related Works

Recently, some efforts have been made to study the AoI-aware MEC networks with multiple servers [14], [17]–[23]. For the dependent task offloading scenario where tasks can be computed locally, or partly offloaded to edge servers or a cloud server, Peng *et al.* [14] proposed a single-agent DRL algorithm, named the improved dueling double deep Q-network (D3QN), to minimize the weighted sum of AoI, latency and energy consumption of the dependent tasks by optimizing task offloading decisions. Considering a aerial-ground MEC network where a unmanned aerial vehicle (UAV) first collects computation tasks from ground devices according to the planned trajectory, and then the tasks are computed on the local UAV or partly offloaded to a high-altitude platform, Song *et al.* [17] proposed a multi-objective learning algorithm based on proximal policy optimization to minimize the total AoI and total energy consumption of the UAV by optimizing the the UAV's trajectory and task offloading ratios. Goudarzi *et al.* [18] also investigated a UAV-assisted MEC network, where tasks could be computed on a local vehicle or partly offloaded to an associated UAV. They used a soft actor-critic-based algorithm to minimize AoI while also minimizing energy consumption and rental costs of vehicles by optimizing UAV's trajectory, vehicle association and offloading decisions. Wei *et al.* [19] developed a Karush-Kuhn-Tucker (KKT) condition-based heuristic algorithm to minimize the average AoI by jointly optimizing the server selection and the time of computing and communication phases in multi-server edge computing networks, where tasks can be partially offloaded to multiple servers.

However, the works [14], [17]–[19] mainly focused on the offloading decision making issue without involving the resource allocation issue. Therefore, a few recent works [20]–[23] have attempted to investigate the AoI-aware joint offloading decision making and resource allocation issue in AoI-aware MEC networks with multiple servers. Under the partial offloading policy, Shen *et al.* [20] employed the traditional mathematical method to minimize the energy consumption of a UAV-enabled MEC network where UAVs act as servers, by jointly optimizing offloading decisions, transmit power of devices, hovering locations, as well as the bandwidth and computation resource allocation of UAVs, subject to the average peak AoI constraint. Under the full offloading policy, Zhu *et al.* [21] considered a scenario where mobile edge devices preprocess the collected data before offloading it to a cloud center, and proposed a federated learning (FL)-assisted multi-agent actor-critic learning

algorithm to minimize the average AoI of all data by jointly optimizing mobile edge devices' movements, preprocessing and offloading decisions, and the bandwidth allocation. Works [22] and [23] focused on an air-ground integrated MEC network, where MDs offload tasks to a UAV or a ground server. Qin *et al.* [22] proposed an alternating optimization-based iterative algorithm to minimize the weighted sum AoI by optimizing offloading decisions, computation resource allocation and UAV's trajectory. Chen *et al.* [23] developed a distributed multi-agent deep reinforcement learning (DRL) algorithm to minimize the weighted sum of average AoI and energy consumption by optimizing offloading decisions and bandwidth allocation. The differences between this paper and the related works are summarized in Table I.

C. Motivations and Contributions

Nevertheless, the aforementioned related works [20]–[23] focused on the UAV-enabled MEC networks without involving the cooperation between different servers. Furthermore, only the work [20] jointly optimized the offloading decisions as well as the computation and communication resource allocation, but the AoI is just a constraint. How to jointly optimize the offloading decisions as well as the computation and communication resource allocation to minimize the long-term AoI in the HMECN, i.e., the real-world mobile heterogeneous cellular-enabled MEC network, has seldom been investigated, which is difficult to achieve due to the long-term feature, the mobility of MDs and the cooperation between MBS and SBS.

To fill this gap, we investigate the long-term weighted sum AoI (LWSA) minimization problem in a dynamic HMECN, which consists of one MBS, multiple SBSs and a large number of MDs with random task generation. According to MDs' positions, priorities, task generation, and so on, MDs could choose not to offload tasks, or offload tasks to a SBS, the MBS directly or the MBS through a SBS, i.e., the cooperation with a SBS. Since MDs move across time slots, the task that is determined to be offloaded should be processed within one time slot, which requires meticulous allocation of BSs' bandwidth and computation resources to MDs in each time slot. Besides the task completion delay constraint, the energy and peak AoI constraints should also be satisfied in order to ensure the sustainable and normal operation. In such a context, the LWSA minimization problem involves both continuous variables, i.e., the bandwidth and computation resource allocation of BSs, and discrete variables, i.e., the offloading decisions of MDs. Considering the non-convexity and long-term feature of the optimization problem as well as the dynamic network environment, it is difficult to handle it by conventional optimization methods such as convex optimization and approximation algorithms, which require a great number of iterations and usually cannot capture the variable optimization correlations between different time slots. DRL, where one or more agents continuously learn from interactions with the dynamic network environment to maximize the long-term returns, is suitable to handle the optimization problem [24], [25]. Due to the advantages of the distributed multi-agent DRL in learning efficiency, scalability and robustness [21], [24], [26], we take each MD as an agent, and each agent independently learns the optimal policy with a collaborative reward setting so as to avoid vicious competition. In such a multi-agent setting, the training instability problem would be aggravated, since agents learn by interacting with a shared network environment. To alleviate this problem, we turn to FL which has the capability to break the limitation of partial knowledge on the network for each individual agent. The main contributions of this paper are summarized as follows.

- We formulate the LWSA minimization problem with the energy, delay and peak AoI constraints in a HMECN, where MDs moving across time slots are allowed to offload tasks to the MBS through a SBS, by jointly optimizing the offloading decisions of MDs as well as the bandwidth and computation resource allocation of BSs.
- To efficiently address the formulated non-convex mixed-integer nonlinear programming (MINLP) problem, we decompose it into the offloading decision optimization (ODO) top-problem and the resource allocation optimization (RAO) sub-problem. Based on the decomposition, we propose a FL-assisted hybrid DRL and convex approach that is comprised of a *safe multi-agent DRL algorithm* for solving the ODO top-problem, *convex optimization* for solving the RAO sub-problem, and *FL* for alleviating the training instability problem aggravated by the multi-agent setting. It is worth noting that the safe multi-agent DRL algorithm consists of the multi-agent D3QN (MAD3QN) algorithm and a proposed safe learning mechanism, which strictly ensures that the actions of each MD do not exceed its energy constraint by adjusting the output of the MAD3QN, and paves the way for using convex optimization to solve the RAO sub-problem.
- Simulation results demonstrate the superiority of the proposed approach in terms of the LWSA, convergence, scalability and robustness in dynamic environments by comparing it with comparative algorithms. It is observed that the LWSA first decreases and then increases with the transmit power of MDs due to the tradeoff between the number of MDs that could offload tasks to BSs and the transmission delay of the MDs.

The remainder of this paper is organized as follows. In Section II, we introduce the system model of the HMECN. In Section III, we formulate the LWSA minimization problem, followed by the proposed approach, called FL-assisted safe multi-agent DRL and convex optimization (FL-SMADC) algorithm, in Section IV. In Section V, we present the simulations and discussions. Section VI concludes this paper.

II. System Model

A. Network Model

As shown in Fig. 1, we consider a HMECN consisting of one U_M-antenna MBS, K U_S-antenna SBSs, and I single-antenna MDs. Let $\mathcal{K} = \{1, ..., K+1\}$ denote the set of BSs, and B_k with $k \in \mathcal{K}$ denote the kth BS. $B_{\tilde{k}}$ with $\tilde{k} \in \{1, ..., K\}$ denotes the $\tilde{k}$th SBS, and B_{K+1} denotes the MBS. Let $\mathcal{I} = \{1, ..., I\}$ denote the set of MDs, and D_i denote the ith MD. The BSs that consist of SBSs and MBS provide computing services for

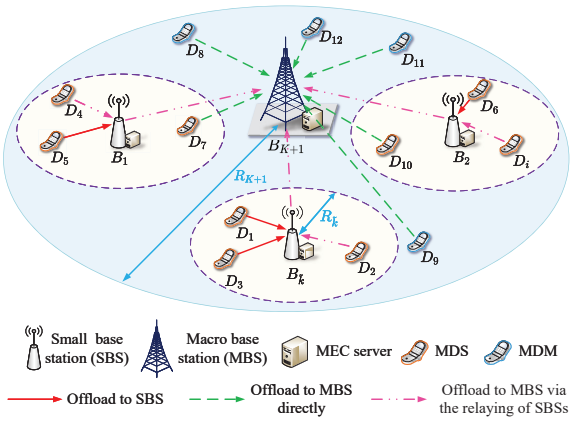


Fig. 1. Structure of the HMECN.

MDs. Due to the fact that the low-frequency radio frequency (RF) signals decay more slowly than the high-frequency RF signals [27], we consider that the MBS operates on low-frequency bands, e.g., 700 MHz, whereas each SBS operates on high-frequency bands, e.g., 3.5 GHz [11]. The coverage area of the MBS is defined as a circle centered at the MBS with radius R_{K+1}, and similarly the coverage area of SBS $B_{\tilde{k}}$ is defined as a circle centered at SBS $B_{\tilde{k}}$ with radius $R_{\tilde{k}}$. To increase the utilization of SBSs and avoid the issue of inter-cell interference, we consider that the coverage areas of SBSs are non-overlapping as those in [9]–[11].

According to the time-varying locations, MDs are divided into non-overlapping sets $\mathcal{I}_k$ with $k \in \mathcal{K}$ at any time, and $\mathcal{I}_k$ at any time satisfies

$$\bigcup_{k\in\mathcal{K}} \mathcal{I}_k = \mathcal{I}, \tag{1}$$

where $\mathcal{I}_a \cap \mathcal{I}_b = \varnothing$ holds for $a, b \in \mathcal{K}, a \neq b$. Namely, D_i with $i \in \mathcal{I}_{\tilde{k}}$ denotes that the ith MD is located within the coverage area of the $\tilde{k}$th SBS and that of the MBS, and is referred to as the MDS. While D_i with $i \in \mathcal{I}_{K+1}$ denotes that the ith MD is located outside the coverage areas of SBSs and only within the coverage area of the MBS, and is referred to as the MDM. The MDs within the coverage area can offload tasks to the corresponding SBS or MBS.

Due to the limited energy and computation capacities of MDs, we consider that the tasks of MDs can hardly be computed locally [28]. Moreover, since the tasks of many real-world applications are indivisible [7], we consider that each task can not be divided [7]–[10]. In other words, each task is either not offloaded or offloaded to one SBS or the MBS. Each MD is equipped with a pre-processing buffer that stores at most one task. The generated task can be stored in the buffer before being offloaded or replaced by the newly generated task.

As shown in Fig. 2, we consider that the system time is divided into $\mathcal{T}$ time slots with equal duration T, and time slot $t \in \{1, ..., \mathcal{T}\}$ is divided into two phases: control phase (CP) with duration τ_0 and offloading phase (OP) with duration $T - \tau_0$. At the beginning of CP, each MD moves following the Gauss-Markov random model (GMRM) within the duration of movement τ_m. After the movements of MDs, each MD generates a task with probability $P_g \in [0, 1]$, and stores the generated task in the buffer. Then MDSs transmit the information about their current AoI and the information of

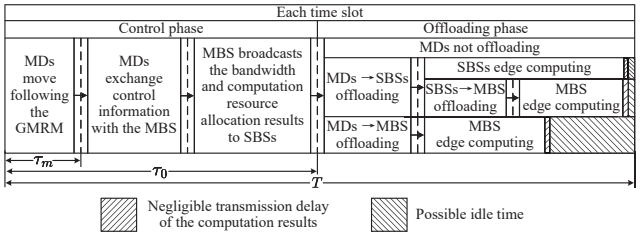


Fig. 2. Time slot structure of the HMECN.

their current tasks to the corresponding SBSs, which forward the received information to the MBS. While MDMs transmit their information to the MBS directly. The MBS broadcasts the collected information to MDs[1]. Then, according to the proposed FL-SMADC algorithm in Section IV, each MD obtains the offloading decision; The MBS obtains the bandwidth and computation resource allocation results, and broadcasts them to SBSs. During OP, each MD respectively executes the feasible offloading decision, i.e., the MD does not offload the task, offloads it to a SBS or the MBS directly, or offloads it to the MBS via the relaying of a SBS. After the tasks are computed by BSs, BSs transmit the computation results to MDs, and the transmission delay of the computation results is neglected due to the small size of the computation results [28]. The time slot structure shows that due to the mobility of MDs, the task determined to be offloaded must be processed within one time slot. Consequently, a great number of pending tasks would compete for bandwidth and computation resources.

B. Mobility Model

Due to the randomness and memory of the real-world device movement, we consider that each MD moves following the GMRM, which is versatile and can cover various mobility scenarios [31]. Specifically, each MD moves within the coverage area of the MBS for duration τ_m at the beginning of each time slot, and keeps static during the remaining time slot. Let $(0, 0)$ denote the position coordinate of the MBS, $(x_{\tilde{k}}, y_{\tilde{k}})$ denote the position coordinate of SBS $B_{\tilde{k}}$, and $(x_i(t), y_i(t))$ denote the position coordinate of D_i at time slot t. Let $\overrightarrow{V_i(t)} = (v_i(t), \varphi_i(t))$ denote the velocity vector of D_i at time slot t, where $v_i(t)$ denotes the velocity of D_i, and $\varphi_i(t)$ denotes the direction of D_i. The velocity vector of D_i at time slot $(t+1)$ is updated as [32]

$$\begin{aligned} v_i(t+1) &= \mu_1 v_i(t) + (1-\mu_1)\overline{v} + \sqrt{1-\mu_1^2}\Lambda_i, \\ \varphi_i(t+1) &= \mu_2 \varphi_i(t) + (1-\mu_2)\overline{\varphi}_i + \sqrt{1-\mu_2^2}\Gamma_i, \end{aligned} \tag{2}$$

where $0 \leq \mu_1, \mu_2 \leq 1$ are tuning parameters used to vary the memory level, $\overline{v}$ denotes the average velocity of MDs, and $\overline{\varphi}_i$ denotes the average direction of D_i. Λ_i and Γ_i denote two Gaussian processes characterized by distinct mean-variance

[1]According to the IEEE 754 standard for floating-point representation [29], a floating-point number occupies 32 bits. According to [30], the total header size of each data packet is 80 bits. Then, combined with the number of states that each MD needs to transmit and receive and the number of MDs in the network, the size of the exchanged state information packets can be calculated. Based on the settings of the duration of CP and the bandwidth of the MBS in the simulation, these packets can be transmitted completely during the CP, thus the corresponding communication overhead is acceptable during the CP.

pairs for D_i, and reflect the randomness in the velocities and directions of MDs, respectively. Based on (2), the position coordinate of D_i at time slot $(t+1)$ is updated as

$$\begin{aligned} x_i(t+1) &= x_i(t) + v_i(t)\cos(\varphi_i(t))\tau_m, \\ y_i(t+1) &= y_i(t) + v_i(t)\sin(\varphi_i(t))\tau_m. \end{aligned} \tag{3}$$

C. Communication Model

We consider a block-fading channel model in the HMECN, i.e., the channel gain keeps unchanged within a time slot, and varies across different time slots. Let $h_{i,k}(t)$ denote the channel gain between D_i and B_k at time slot t as

$$h_{i,k}(t) = d_{i,k}(t)^{-\ell}\left|g_{i,k}(t)\right|^2, \tag{4}$$

where $d_{i,k}(t)$ denotes the Euclidean distance between D_i and B_k at time slot t, ℓ denotes the path loss exponent, and $g_{i,k}(t)$ denotes Rayleigh fading as [33]

$$g_{i,k}(t) = \rho g_{i,k}(t-1) + \sqrt{1-\rho^2}o. \tag{5}$$

In (5), $\rho \in [0,1]$ denotes the correlation factor that accounts for the correlations of channel gains in two successive time slots, and o denotes a Gaussian random variable following the complex Gaussian distribution with mean 0 and variance 1. Similarly, let $h_{\tilde{k},K+1}(t)$ denote the channel gain between SBS $B_{\tilde{k}}$ and MBS B_{K+1} at time slot t as

$$h_{\tilde{k},K+1}(t) = d_{\tilde{k},K+1}{}^{-\ell}\left|g_{\tilde{k},K+1}(t)\right|^2, \tag{6}$$

where $d_{\tilde{k},K+1}$ denotes the Euclidean distance between $B_{\tilde{k}}$ and B_{K+1}, and $g_{\tilde{k},K+1}(t)$ denotes the Rayleigh fading.

To constraint the interference in the coverage area of each BS, we adopt the orthogonal frequency-division multiple access scheme. Specifically, each SBS independently allocates the bandwidth to the MDs that either offload tasks to it directly, or transmit tasks to it for relaying purpose. The MBS allocates the bandwidth to the MDs that offload tasks to it directly and SBSs that relay the received tasks to it. Due to the distinct frequency bands for SBSs and MBS and the non-overlapping coverage areas of SBSs, the transmission between MDs and BSs within different coverage areas does not interfere with each other.

Let $\phi_{i,k}(t) \in [0,1]$ denote the proportion of the allocated bandwidth by the kth BS at time slot t, where $\phi_{i,\tilde{k}}(t)$ denotes the proportion of the bandwidth that SBS $B_{\tilde{k}}$ allocates to D_i at time slot t, and $\phi_{i,K+1}(t)$ denotes the proportion of the bandwidth that MBS B_{K+1} allocates to $B_{\tilde{k}}$ for relaying the task of D_i or directly allocates to D_i at time slot t. As the sum of the allocated bandwidth by each BS does not exceed the total bandwidth of each BS, $\phi_{i,k}(t)$ satisfies $\sum_{i=1}^{I}\phi_{i,k}(t) \le 1$.

The transmission rate from D_i to B_k at time slot t is expressed as

$$R_{i,k}(t) = \phi_{i,k}(t)W_k \log_2\left(1 + \frac{p_i h_{i,k}(t)}{\sigma^2}\right), \tag{7}$$

where W_k denotes the total bandwidth of B_k, p_i denotes the transmit power of D_i, and σ^2 denotes the power of the additive white Gaussian noise. Based on (7), the transmission delay of D_i to transmit the task to B_k at time slot t directly (i.e., by one hop), denoted by $\tau_{i,k}^{\text{o,tran}}(t)$, is expressed as

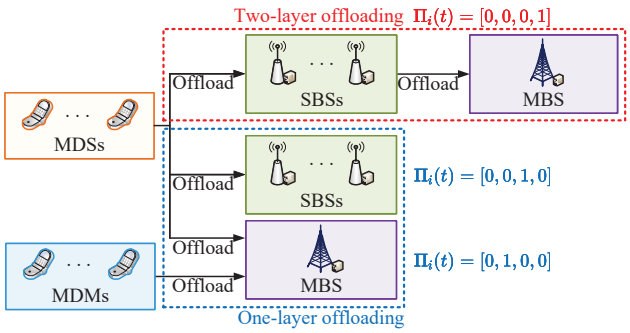


Fig. 3. Offloading process of the HMECN.

$$\tau_{i,k}^{\text{o,tran}}(t) = \frac{M_i(t)}{R_{i,k}(t)}, \tag{8}$$

where $M_i(t)$ denotes the size of the task stored by D_i at time slot t. The amount of the energy consumed by D_i to offload the task to B_k, denoted by $e_{i,k}(t)$, is expressed as

$$e_{i,k}(t) = p_i \tau_{i,k}^{\text{o,tran}}(t). \tag{9}$$

The transmission rate from $B_{\tilde{k}}$ to B_{K+1} for relaying the task of D_i at time slot t is expressed as

$$R_{\tilde{k},K+1}^{i}(t) = \phi_{i,K+1}(t)W_{K+1}\log_2\left(1+\frac{p_{\tilde{k}}^{\text{s}} h_{\tilde{k},K+1}(t)}{\sigma^2}\right), \tag{10}$$

where W_{K+1} denotes the total bandwidth of MBS B_{K+1}, and $p_{\tilde{k}}^{\text{s}}$ denotes the transmit power of SBS $B_{\tilde{k}}$. Based on (8) and (10), the transmission delay of D_i to offload the task to MBS B_{K+1} via the relaying of SBS $B_{\tilde{k}}$ (i.e., by two hop), denoted by $\tau_{i,\tilde{k}}^{\text{t,tran}}(t)$, is expressed as

$$\tau_{i,\tilde{k}}^{\text{t,tran}}(t) = \frac{M_i(t)}{R_{i,\tilde{k}}(t)} + \frac{M_i(t)}{R_{\tilde{k},K+1}^{i}(t)}, \tag{11}$$

where $R_{i,\tilde{k}}(t)$ denotes the transmission rate from D_i to SBS $B_{\tilde{k}}$.

D. Offloading and Computing Model

Although the feasible offloading decision of MDs has been introduced in subsection II-A, in this subsection, we would like to present the offloading process of MDs mathematically. As shown in Fig. 3, the process that the MD offloads the task to the selected SBS or the MBS directly is referred to as one-layer offloading, and the process that the MD offloads the task to the MBS via the relaying of a SBS is referred to as two-layer offloading. Let $\Pi_i(t) = [\alpha_i(t), \beta_i(t), \tilde{\beta}_{i,\tilde{k}}(t), \gamma_{i,\tilde{k}}(t) | \alpha_i(t), \beta_i(t), \tilde{\beta}_{i,\tilde{k}}(t), \gamma_{i,\tilde{k}}(t) \in \{0,1\}]$ represent the offloading decision of D_i at time slot t as

$$\Pi_i(t) = \begin{cases} [1,0,0,0]\text{, not offloading,} \\ [0,1,0,0]\text{,} \\ [0,0,1,0]\text{,} \end{cases} \begin{array}{l} \\ \text{one-layer offloading,} \\ \\ \end{array} \quad \begin{cases} \\ \\ [0,0,0,1]\text{, two-layer offloading.} \end{cases} \tag{12}$$

$\Pi_i(t) = [1,0,0,0]$ represents that D_i does not have the task or does not offload it at time slot t, $\Pi_i(t) = [0,1,0,0]$ represents that D_i offloads the task to MBS B_{K+1} at time slot t, $\Pi_i(t) = [0,0,1,0]$ represents that D_i with $i \in \mathcal{I}_{\tilde{k}}(t)$ offloads the task to SBS $B_{\tilde{k}}$ at time slot t, and $\Pi_i(t) = [0,0,0,1]$ represents

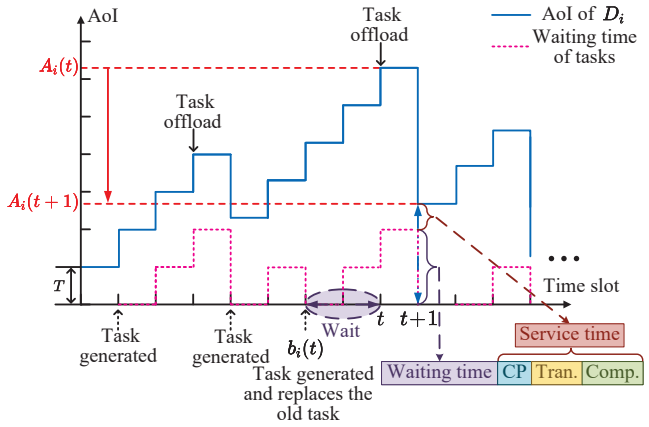


Fig. 4. An illustration of AoI evolution of D_i. 'Tran.' and 'Comp.' stand for transmission delay and computation delay, respectively.

that D_i with $i \in \mathcal{I}_{\tilde{k}}(t)$ offloads the task to MBS B_{K+1} via the relaying of SBS $B_{\tilde{k}}$ at time slot t. As each MD selects only one offloading decision at a time slot, the offloading decision variables satisfy

$$\alpha_i(t) + \beta_i(t) + \sum_{\tilde{k}=1}^{K} \tilde{\beta}_{i,\tilde{k}}(t) + \sum_{\tilde{k}=1}^{K} \gamma_{i,\tilde{k}}(t) = 1. \quad (13)$$

After tasks are offloaded to BSs, BSs provide computing services for MDs. Let F_k denote the computation capacity of the kth BS, where $F_{\tilde{k}}$ and F_{K+1} denote the computation capacity of SBS $B_{\tilde{k}}$ and MBS B_{K+1}, respectively. Let $f_{i,k}(t)$ denote the computation resources that the kth BS allocates to D_i at time slot t, where $f_{i,\tilde{k}}(t)$ denotes the computation resources that SBS $B_{\tilde{k}}$ allocates to D_i at time slot t and $f_{i,K+1}(t)$ denotes the computation resources that MBS B_{K+1} allocates to D_i at time slot t. As the sum of the allocated computation resources by each BS does not exceed the computation capacity, $f_{i,k}(t)$ satisfies

$$\sum_{i=1}^{I} f_{i,k}(t) \le F_k. \quad (14)$$

The computation delay of the kth BS B_k to compute the task of D_i at time slot t is expressed as

$$\tau_{i,k}^{\text{comp}}(t) = \frac{M_i(t)C_i}{f_{i,k}(t)}, \quad (15)$$

where C_i denotes the number of CPU cycles required for computing one bit for D_i.

E. AoI Model

As the definition of AoI in [16], the AoI of D_i is defined as the time difference between the generation time of the computed task most recently received by D_i and the current time. Mathematically, the AoI of D_i is denoted by $A_i(t)$ at time slot t. Then we introduce the AoI model according to three cases in (12) as follows.

1) Not Offloading: For $\Pi_i(t) = [1, 0, 0, 0]$, the AoI of D_i at time slot $(t+1)$ is updated as

$$A_i(t+1) = A_i(t) + T, \quad (16)$$

TABLE II
KEY NOTATIONS

Notations	Definitions
D_i	The ith MD, $i \in \{1, ..., I\}$
B_k	The kth BS, $k \in \{1, ..., K+1\}$
$B_{\tilde{k}}$	The $\tilde{k}$th SBS, $\tilde{k} \in \{1, ..., K\}$
B_{K+1}	The MBS
$\mathcal{I}_{\tilde{k}}(t)$	The set of MDSs within the coverage area of SBS $B_{\tilde{k}}$
$\mathcal{I}_{K+1}(t)$	The set of MDMs outside coverage areas of all SBSs
T	Time slot duration
τ_0	Duration of CP
τ_m	Duration of movement
$\Pi_i(t)$	Offloading decision of D_i
$h_{i,k}(t)$	Channel gain between D_i and B_k
$h_{\tilde{k},K+1}(t)$	Channel gain between SBS $B_{\tilde{k}}$ and MBS B_{K+1}
p_i	Transmit power of D_i
$p_{\tilde{k}}^{\text{s}}$	Transmit power of SBS $B_{\tilde{k}}$
$\phi_{i,k}(t)$	Proportion of B_k's allocated bandwidth to D_i
$\phi_{i,K+1}(t)$	Proportion of MBS B_{K+1}'s allocated bandwidth to D_i
$\tau_{i,k}^{\text{o,tran}}(t)$	Transmission delay of one hop from D_i to B_k
$\tau_{i,\tilde{k}}^{\text{t,tran}}(t)$	Transmission delay of two hop from D_i to B_{K+1} via $B_{\tilde{k}}$
$\alpha_i(t)$	Indicator of whether D_i does not offload the task
$\beta_i(t)$	Indicator of whether D_i offloads to the MBS
$\beta_{i,\tilde{k}}(t)$	Indicator of whether D_i offloads to SBS $\tilde{k}$
$\gamma_{i,\tilde{k}}(t)$	Indicator of whether D_i offloads to the MBS via SBS $\tilde{k}$
$f_{i,\tilde{k}}(t)$	Computation resources that are allocated to D_i by SBS $B_{\tilde{k}}$
$f_{i,K+1}(t)$	Computation resources that are allocated to D_i by the MBS
$\tau_{i,k}^{\text{comp}}(t)$	Computation delay of B_k for D_i
$\tau_i^{\text{wait}}(t)$	Waiting time of D_i's current task
$\tau_i^{\text{srv}}(t)$	The service time of D_i

since D_i does not receive new computation result at time slot t, and the AoI of D_i increases by a time slot with duration T at time slot $(t+1)$.

2) One-layer Offloading: For $\Pi_i(t) = [0, 1, 0, 0]$ or $[0, 0, 1, 0]$, B_k computes the task and transmits the computation result to D_i within time slot t. Then the AoI of D_i decreases at time slot $(t+1)$. As shown in Fig. 4, the updated AoI of D_i at time slot $(t+1)$ consists of the waiting time and service time. The waiting time is expressed as

$$\tau_i^{\text{wait}}(t) = (t - b_i(t))T, \quad (17)$$

where $\tau_i^{\text{wait}}(t)$ denotes the time difference between the time slot when the task is generated and the current time slot, and $b_i(t)$ denotes the generation time slot of the task that is stored in the buffer of D_i at time slot t. The service time consists of three components: the duration of CP, i.e., τ_0, the transmission delay of D_i to offload the task to B_k at time slot t, i.e., $\tau_{i,k}^{\text{o,tran}}(t)$, and the computation delay of B_k to compute the task of D_i at time slot t, i.e., $\tau_{i,k}^{\text{comp}}(t)$. Accordingly, the AoI of D_i at time slot $(t+1)$ is updated as

$$A_i(t+1) = \tau_i^{\text{wait}}(t) + \tau_0 + \tau_{i,k}^{\text{o,tran}}(t) + \tau_{i,k}^{\text{comp}}(t), \quad (18)$$

where $\tau_{i,k}^{\text{o,tran}}(t)$ is shown in (8).

3) Two-layer Offloading: For $\Pi_i(t) = [0, 0, 0, 1]$, B_{K+1} computes the task and transmits the computation result to D_i within time slot t. Accordingly, the AoI of D_i at time slot $(t+1)$ is updated as

$$A_i(t+1) = \tau_i^{\text{wait}}(t) + \tau_0 + \tau_{i,\tilde{k}}^{\text{t,tran}}(t) + \tau_{i,K+1}^{\text{comp}}(t), \quad (19)$$

where $\tau_{i,\tilde{k}}^{\text{t,tran}}(t)$ is shown in (11).

Based on (16), (18), and (19), the AoI of D_i at time slot $(t+1)$ is updated as

$$A_i(t+1)=\begin{cases} A_i(t)+T, & \text{if } \Pi_i(t)=[1,0,0,0], \\ \tau_i^{\text{wait}}(t)+\tau_0+\tau_{i,k}^{\text{o,tran}}(t)+\tau_{i,k}^{\text{comp}}(t), & \\ \qquad \text{if } \Pi_i(t)=[0,1,0,0] \text{ or } [0,0,1,0], & \\ \tau_i^{\text{wait}}(t)+\tau_0+\tau_{i,\tilde{k}}^{\text{t,tran}}(t)+\tau_{i,K+1}^{\text{comp}}(t), & \\ \qquad \text{if } \Pi_i(t)=[0,0,0,1]. \end{cases} \quad (20)$$

In Table II, we summarize the key notations throughout this paper.

III. Problem Formulation

In this section, we formulate the problem of minimizing the long-term weighted sum AoI (LWSA) of MDs by optimizing the offloading decisions, bandwidth allocation, and computation resource allocation at each time slot as

$$\mathbf{P0}: \min_{\mathbf{\Pi},\boldsymbol{\phi},\boldsymbol{f}} \sum_{t=1}^{\mathcal{T}}\sum_{i=1}^{I} w_i A_i(t) \quad (21a)$$

$$s.t. \quad e_{i,k}(t) \le e_{\max}, \forall i \in \mathcal{I}, \forall k \in \mathcal{K}, \quad (21b)$$

$$A_i(t) \le A_{\max}, \forall i \in \mathcal{I}, \forall k \in \mathcal{K}, \quad (21c)$$

$$\sum_{i=1}^{I}(\tilde{\beta}_{i,\tilde{k}}(t)+\gamma_{i,\tilde{k}}(t))\phi_{i,\tilde{k}}(t) \le 1, \forall \tilde{k} \in \{1,...,K\}, \quad (21d)$$

$$\sum_{i=1}^{I}\sum_{\tilde{k}=1}^{K}(\beta_i(t)+\gamma_{i,\tilde{k}}(t))\phi_{i,K+1}(t) \le 1, \quad (21e)$$

$$\sum_{i=1}^{I}\tilde{\beta}_{i,\tilde{k}}(t) f_{i,\tilde{k}}(t) \le F_{\tilde{k}}, \forall \tilde{k} \in \{1,...,K\}, \quad (21f)$$

$$\sum_{i=1}^{I}\sum_{\tilde{k}=1}^{K}(\beta_i(t)+\gamma_{i,\tilde{k}}(t)) f_{i,K+1}(t) \le F_{K+1}, \quad (21g)$$

$$\tau_0+\tau_{i,k}^{\text{o,tran}}(t)+\tau_{i,k}^{\text{comp}}(t) \le T, \forall i \in \mathcal{I}, \forall k \in \mathcal{K}, \quad (21h)$$

$$\tau_0+\tau_{i,\tilde{k}}^{\text{t,tran}}(t)+\tau_{i,K+1}^{\text{comp}}(t) \le T, \quad \forall i \in \mathcal{I}, \forall \tilde{k} \in \{1,...,K\}, \quad (21i)$$

$$\alpha_i(t), \beta_i(t), \tilde{\beta}_{i,\tilde{k}}(t), \gamma_{i,\tilde{k}}(t) \in \{0,1\}, \quad \forall i \in \mathcal{I}, \forall k \in \mathcal{K}, \forall \tilde{k} \in \{1,...,K\}, \quad (21j)$$

$$\alpha_i(t)+\beta_i(t)+\sum_{\tilde{k}=1}^{K}\tilde{\beta}_{i,\tilde{k}}(t)+\sum_{\tilde{k}=1}^{K}\gamma_{i,\tilde{k}}(t)=1, \quad (21k)$$

where $\mathbf{\Pi}=[\mathbf{\Pi}(1),...,\mathbf{\Pi}(\mathcal{T})]$ in (21a) represents the $I \times 4 \times \mathcal{T}$ matrix that includes the binary integer variables for offloading decisions of MDs from time slot 1 to $\mathcal{T}$, and $\mathbf{\Pi}(t)=[\Pi_1(t),...,\Pi_I(t)] \in \mathbb{C}^{I\times 4}$ represents the offloading decisions of MDs at time slot t. $\boldsymbol{\phi}=[\boldsymbol{\phi}(1),...,\boldsymbol{\phi}(\mathcal{T})]$ in (21a) represents the $I \times (K+1) \times \mathcal{T}$ matrix that includes the bandwidth allocation of BSs for MDs from time slot 1 to $\mathcal{T}$, where $\boldsymbol{\phi}(t)=[\phi_{1,1}(t),...,\phi_{i,k}(t),...,\phi_{I,K+1}(t)] \in \mathbb{C}^{I\times(K+1)}$ represents the bandwidth allocation of BSs for MDs at time slot t. $\boldsymbol{f}=[\boldsymbol{f}(1),...,\boldsymbol{f}(\mathcal{T})]$ in (21a) represents the $I \times (K+1) \times \mathcal{T}$ matrix that includes the computation resource allocation of BSs for MDs from time slot 1 to $\mathcal{T}$, where $\boldsymbol{f}(t)=[f_{1,1}(t),...,f_{i,k}(t),...,f_{I,K+1}(t)] \in \mathbb{C}^{I\times(K+1)}$ represents the computation resource allocation of BSs for MDs at time slot t. w_i in (21a) denotes the priority of D_i. The priorities of MDs reflect their different importance in the real scenarios [16], [22]. (21b) ensures that, the amount of the energy consumed by D_i at each time slot does not exceed the maximum energy consumption $e_{\max}$ [34]. (21c) provides an upper bound of AoI for each MD $A_{\max}$, which denotes the maximum tolerant AoI of each MD in the unit of the time slot duration T. (21d) and (21e) represent the bandwidth constraint of SBSs and MBS, respectively. (21f) and (21g) represent the computation resource constraint of SBSs and MBS, respectively. (21h) and (21i) ensure that, the service time of processing a single task does not exceed the time slot duration T. (21j) and (13) represent the value ranges of offloading decision variables.

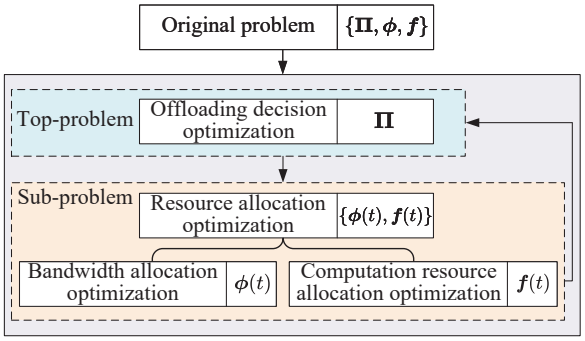


Fig. 5. The decomposition of $\mathbf{P0}$.

Due to the nonlinearity in the objective function and the availability of binary variables which are difficult to handle directly, the formulated optimization problem $\mathbf{P0}$ is a non-convex MINLP and NP-hard problem [35], and there are no standard methods to solve it optimally in general. There are two challenges in addressing the optimization problem. First, due to the long-term feature of $\mathbf{P0}$, the variable optimization across different time slots is mutually correlated. As a result, the offloading decisions and resource (i.e., bandwidth and computation resources) allocation are tightly coupled. Second, due to the dynamic network environment where the position, task generation, and data size of tasks change over time slots, the method should be able to adapt to the dynamic network environment and provide solution in time. All these challenges make it difficult to implement conventional methods and motivate us to design a learning-based method. To overcome these challenges, we decompose $\mathbf{P0}$ into the ODO top-problem and the RAO sub-problem as shown in Fig. 5, and then propose a FL-SMADC algorithm that leverages the DRL, convex optimization and FL to solve it in the next section.

IV. FL-SMADC Algorithm

In this section, we propose a FL-SMADC algorithm, which is inspired by the decomposition of $\mathbf{P0}$: ODO top-problem and RAO sub-problem. The ODO top-problem, in essence a sequential decision-making problem, optimizes the offloading decision $\mathbf{\Pi}$ from time slot 1 to $\mathcal{T}$, subject to the energy constraint (21b), the AoI constraint (21c), the service time constraints (21h)-(21i), and the offloading decision variable constraints (21j)-(13). With the output of the ODO top-problem $\mathbf{\Pi}$, the RAO sub-problem optimizes the bandwidth resource $\boldsymbol{\phi}$

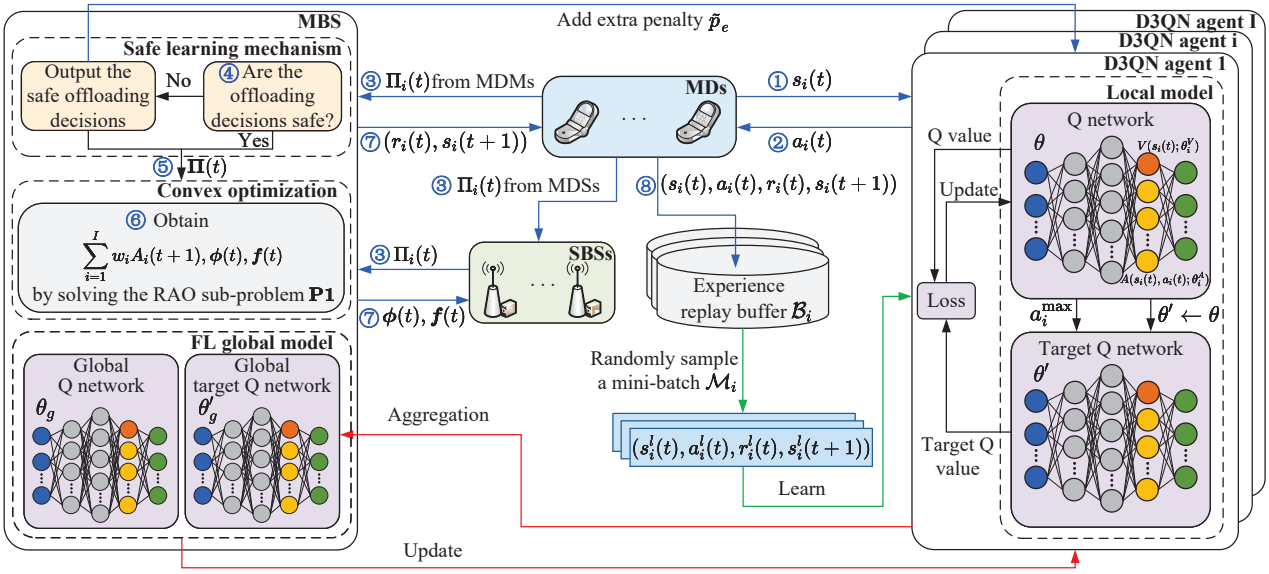


Fig. 6. The framework of the proposed FL-SMADC.

and computation resource $\boldsymbol{f}$ at each time slot t, subject to the energy constraint (21b), the bandwidth constraints (21d)-(21e), and the CPU frequency constraints (21f)-(21g).

As shown in Fig. 6, the proposed FL-SMADC algorithm consists of three components: safe multi-agent DRL algorithm, convex optimization, and FL. In the following, we specify the framework of the proposed FL-SMADC algorithm by elaborating the three components.

- **Safe multi-agent DRL algorithm**, comprised of a local multi-agent DRL algorithm and a safe learning mechanism, is developed to solve the ODO top-problem.
  1) *Local multi-agent DRL algorithm*: Multiple distributed agents at MDs output the discrete actions, i.e., the offloading decisions, by employing the MAD3QN algorithm, which does well in searching for an efficient policy with discrete variables for multiple agents. Then MDs individually transmit their offloading decisions to MBS.
  2) *Safe learning mechanism*: This mechanism evaluates whether the offloading decisions outputted by the MAD3QN algorithm are safe, i.e., whether the output strictly satisfies the energy constraint (21b). If not, the output would be adjusted by a developed elite-preserving genetic algorithm (EGA). This mechanism paves the way for using convex optimization to solve the RAO sub-problem.
- **Convex optimization** is employed to solve the RAO sub-problem. Based on the received offloading decisions of MDs, the MBS optimizes the bandwidth allocation $\boldsymbol{f}(t)$ and computation resource allocation $\boldsymbol{\phi}(t)$ for MDs by proving the convexity of the RAO sub-problem. Then the MBS broadcasts the bandwidth and computation resource allocation results to SBSs.
- **FL** is incorporated to alleviate the training instability problem aggravated by the multi-agent setting, so as to achieve better training performance and accelerate the training process of the multi-agent DRL algorithm. Specifically, every few episodes, each MD transmits its local DRL model parameters to the MBS for aggregation, and updates its local DRL model once receiving the global model parameters from the MBS.

A. Safe multi-agent DRL for the ODO Top-problem

To address the ODO sub-problem, a Markov decision process (MDP) model with state space, action space, and reward function is defined as follows.

- State space ($\mathcal{S}$): At the beginning of CP at time slot t, the MBS collects the current state information and broadcasts it to MDs. Then each MD equipped with a local agent combines the local state information and the received state information as its state. The state of D_i at time slot t is defined as $s_i(t) \in \mathcal{S}$, the components of which are listed as follows.
  1) $(x_i(t), y_i(t))$: The position coordinate of D_i at time slot t.
  2) $\{\Phi_{1,\tilde{k}}(t), ..., \Phi_{I,\tilde{k}}(t)\}$: The ratio of the rate from MDs to the corresponding SBSs to bandwidth at time slot t. $\Phi_{i,\tilde{k}}(t)$ is expressed as
  $$\Phi_{i,\tilde{k}}(t) = \begin{cases} \log_2\left(1+\frac{p_i h_{i,\tilde{k}}(t)}{\sigma^2}\right), & i \in \mathcal{I}_{\tilde{k}}(t), \\ 0, & i \in \mathcal{I}_{K+1}(t). \end{cases} \quad (22)$$
  3) $\{\Phi_{1,K+1}(t), ..., \Phi_{I,K+1}(t)\}$: The ratio of the rate from MDs to the MBS to bandwidth at time slot t. $\Phi_{i,K+1}(t)$ is expressed as
  $$\Phi_{i,K+1}(t) = \log_2\left(1+\frac{p_i h_{i,K+1}(t)}{\sigma^2}\right). \quad (23)$$
  4) $\{s_1^c(t), ..., s_I^c(t)\}$: The sets that MDs belong to at time slot t. $s_i^c(t)$ is expressed as
  $$s_i^c(t) = \begin{cases} \tilde{k}, & i \in \mathcal{I}_{\tilde{k}}(t), \\ K+1, & i \in \mathcal{I}_{K+1}(t). \end{cases} \quad (24)$$

5) $\{A_1(t), ..., A_I(t)\}$: The AoI of MDs at time slot t.
6) $\{\tau_1^{\text{wait}}(t), ..., \tau_I^{\text{wait}}(t)\}$: The waiting time of the tasks stored by MDs at time slot t.
7) $\{M_1(t), ..., M_I(t)\}$: The data size of the tasks stored by MDs at time slot t.
8) $\{w_1, ..., w_I\}$: The priorities of MDs.

- Action space ($\mathcal{A}$): The action space of D_i at time slot t, denoted by $\mathcal{A}_i(t)$, is expressed as

$$\mathcal{A}_i(t) = \{a_i(t)\} \in \mathcal{A}. \tag{25}$$

The discrete action $a_i(t)$ is given by

$$a_i(t) = \begin{cases} a_i^{\text{s}}, & i \in \mathcal{I}_{\tilde{k}}(t), \\ a_i^{\text{m}}, & i \in \mathcal{I}_{K+1}(t). \end{cases} \tag{26}$$

$a_i^{\text{s}} \in \{0, 1, 2, 3\}$ denotes the action of D_i that satisfies $i \in \mathcal{I}_{\tilde{k}}(t)$, where $a_i^{\text{s}} = 0$ represents that D_i does not offload the task, $a_i^{\text{s}} = 1$ represents that D_i offloads it to SBS $B_{\tilde{k}}$, $a_i^{\text{s}} = 2$ represents that D_i offloads it to MBS B_{K+1} directly, and $a_i^{\text{s}} = 3$ represents that D_i offloads it to MBS B_{K+1} via the relaying of SBS $B_{\tilde{k}}$. $a_i^{\text{m}} \in \{0, 1\}$ denotes the action of D_i that satisfies $i \in \mathcal{I}_{K+1}(t)$, where $a_i^{\text{m}} = 0$ represents that D_i does not offload the task, and $a_i^{\text{m}} = 1$ represents that D_i offloads it to MBS B_{K+1}.

- Reward function ($\mathcal{R}$): The reward function should consider both the objective function and constraints. Based on (20), the action at time slot t $a_i(t)$ essentially affects the AoI at time slot $(t+1)$, hence the reward function of D_i at time slot t is defined as

$$r_i(t) = -\chi_i^e(t) - c_A \chi_i^A(t) - c_{\text{srv}} \chi_i^{\text{srv}}(t) - \sum_{i=1}^{I} w_i A_i(t+1), \tag{27}$$

where c_A and c_{srv} are positive constants to adjust the magnitudes of various terms in the reward function. $\chi_i^e(t)$, $\chi_i^A(t)$, and $\chi_i^{\text{srv}}(t)$ are respectively expressed as

$$\chi_i^e(t) = \begin{cases} \tilde{p}_e, & e_{i,k}(t) > e_{\max}, \\ 0, & e_{i,k}(t) \le e_{\max}, \end{cases} \tag{28}$$

$$\chi_i^A(t) = \begin{cases} A_i(t+1) - A_{\max}, & A_i(t+1) > A_{\max}, \\ 0, & A_i(t+1) \le A_{\max}, \end{cases} \tag{29}$$

$$\chi_i^{\text{srv}}(t) = \begin{cases} \tau_i^{\text{srv}}(t) - T, & \tau_i^{\text{srv}}(t) > T, \\ 0, & \tau_i^{\text{srv}}(t) \le T, \end{cases} \tag{30}$$

where $\chi_i^e(t)$ denotes the penalty of violating (21b), $\tilde{p}_e$ is a positive constant, $\chi_i^A(t)$ denotes the penalty of violating (21c), $\chi_i^{\text{srv}}(t)$ denotes the penalty of violating (21h)-(21i), and $\tau_i^{\text{srv}}(t)$, the service time of D_i at time slot t, is expressed as

$$\tau_i^{\text{srv}}(t) = \begin{cases} \tau_0 + \tau_{i,k}^{\text{o,tran}}(t) + \tau_{i,k}^{\text{comp}}(t), \\ \text{if } \Pi_i(t) = [0,1,0,0] \text{ or } [0,0,1,0], \\ \tau_0 + \tau_{i,\tilde{k}}^{\text{t,tran}}(t) + \tau_{i,K+1}^{\text{comp}}(t), \\ \text{if } \Pi_i(t) = [0,0,0,1]. \end{cases} \tag{31}$$

In DRL, by denoting π as a mapping from state to action, each agent i at $D_i (i \in \mathcal{I})$ takes appropriate actions based on the current state and trial-and-error experience to find an optimal policy π_i^* that maximizes the long-term discounted accumulative reward, i.e.,

$$\pi_i^* = \underset{\pi_i}{\text{argmax}}\, \mathbb{E}_{\pi_i} \left[\sum_{t \in \mathcal{T}} \tilde{\gamma}^{t-1} r_i(t) \right], \tag{32}$$

where $\tilde{\gamma} \in (0, 1]$ denotes the discount factor. In general, the agent i takes an action $a_i(t)$ following the policy π_i once observing $s_i(t)$. Based on $a_i(t)$, the agent receives the reward $r_i(t)$ and the next time slot state $s_i(t+1)$. The process continues until it obtains the terminal state. In value-based DRL, the agent continuously executes the process to maximize the action-value function $Q_{\pi_i}(s, a)$, also known as Q value function. That is to say, $Q_{\pi_i}(s, a)$ is an indicator to measure whether the policy π_i is optimal. Based on (32), $Q_{\pi_i}(s, a)$ is expressed as

$$Q_{\pi_i}(s, a) = \mathbb{E}_{\pi_i} \left[\sum_{t=\hat{t}}^{\mathcal{T}} \tilde{\gamma}^{t-\hat{t}} r_i(t) \Big| s = s_i(t), a = a_i(t) \right]. \tag{33}$$

The iterative updating method of $Q_{\pi_i}(s, a)$ is based on the Bellman equation [9], which is expressed as

$$Q_{\pi_i}(s_i(t), a_i(t)) = \mathbb{E}_{\pi_i}[r_i(t) + \tilde{\gamma} Q_{\pi_i}(s_i(t+1), a_i(t+1))]. \tag{34}$$

Accordingly, the optimal Q value function that corresponds to the optimal policy π_i^* is formulated as

$$Q_{\pi_i}^*(s_i(t), a_i(t)) = r_i(t) + \tilde{\gamma} \max_{a_i(t+1) \in \mathcal{A}_i(t+1)} Q_{\pi_i}^*(s_i(t+1), a_i(t+1)). \tag{35}$$

According to the overview of the value-based DRL, we employ the MAD3QN as a function approximator to estimate Q value function, so as to output the offloading decisions.

1) MAD3QN Algorithm: We employ the MAD3QN to address the ODO top-problem due to the following two considerations. First, the action space formulated in the MDP model is discrete. Second, as D3QN takes advantage of a blend of the dueling DQN and double DQN structures, it benefits from the ability to decompose state from action advantage of the dueling DQN and the enhanced action selection accuracy of the double DQN [25].

The process for each episode of the MAD3QN is specified as follows. At each time slot in a episode, each agent i obtains its state $s_i(t)$ and inputs it into the Q network. As the D3QN adopts the ϵ-greedy policy to balance the tradeoff between exploration and exploitation, each agent randomly selects an action with probability ϵ. Otherwise, it selects the action that owns the maximal estimated Q value with probability $1 - \epsilon$. The Q value $Q(s_i(t), a_i(t); \theta_i)$ is estimated in the manner of the dueling DQN as

$$Q(s_i(t), a_i(t); \theta_i) = V(s_i(t); \theta_i^V) + A(s_i(t), a_i(t); \theta_i^A) - \frac{1}{|\mathcal{A}_i(t)|} \sum_{a_i(t) \in \mathcal{A}_i(t)} A(s_i(t), a_i(t); \theta_i^A), \tag{36}$$

where θ_i, θ_i^V and θ_i^A denote the parameters of the deep neural network (DNN), state value network and action advantage network, respectively. $|\mathcal{A}_i(t)|$ represents the dimension of the discrete action space at time slot t. The state value function $V(s_i(t); \theta_i^V)$ estimates the value of state $s_i(t)$, and the action advantage function $A(s_i(t), a_i(t); \theta_i^A)$ measures the advantage of taking a specific action $a_i(t)$ in state $s_i(t)$. By decomposing the Q value function into $V(s_i(t); \theta_i^V)$ and $A(s_i(t), a_i(t); \theta_i^A)$,

the dueling DQN evaluates the value of states and the advantage of actions more effectively than DQN, thus improving the learning efficiency and policy performance [14]. After the selected action $a_i(t)$ is executed, each agent receives a feedback reward $r_i(t)$ and the next time slot state $s_i(t+1)$. Meanwhile, the tuple $(s_i(t), a_i(t), r_i(t), s_i(t+1))$ is stored into the experience replay buffer $\mathcal{B}_i$.

Once the above process runs for a episode, each agent samples a mini-batch of experience $\mathcal{M}_i \in \mathcal{B}_i$ randomly. Let $E_i^l(t) = (s_i^l(t), a_i^l(t), r_i^l(t), s_i^l(t+1))$ represent the lth experience of the mini-batch $\mathcal{M}_i$. Then each agent calculates the loss function in the manner of the double DQN. Specifically, there are two DNNs, i.e., the aforementioned Q network and the target Q network that have the same structure as the Q network. The target Q network with parameters θ_i' uses the state $s_i^l(t+1)$ and reward $r_i^l(t)$ of $E_i^l(t)$ to calculate the target Q value as

$$y_i^l(t) = r_i(t) + \tilde{\gamma} Q'\Big(s_i^l(t+1), \underset{a_i(t+1)\in\mathcal{A}_i(t+1)}{\operatorname{argmax}} Q(s_i^l(t+1), a_i(t+1); \theta_i); \theta_i'\Big). \quad (37)$$

Then the loss function is calculated based on the temporal difference error between the Q value calculated by the Q network in (36) and the target Q value calculated by the target Q network in (37) as

$$L(\theta_i) = \frac{1}{|\mathcal{M}_i|} \sum_{l=1}^{|\mathcal{M}_i|} (y_i^l(t) - Q(s_i^l(t), a_i^l(t); \theta_i))^2, \quad (38)$$

where $|\mathcal{M}_i|$ represents the size of the mini-batch. By using two DNNs to calculate the loss function, the D3QN reduces the fluctuation of the Q value and ensures stability during training. Based on (38), the Q network parameters of agent i are updated through the gradient descent method as

$$\theta_i \leftarrow \theta_i - \tilde{\alpha} \nabla_{\theta_i} L(\theta_i), \quad (39)$$

where $\tilde{\alpha}$ denotes the learning rate of the Q network. The target Q network parameters θ_i' of agent i are periodically synchronized as $\theta_i' \leftarrow \theta_i$ every $\tilde{\tau}$ episodes. The parameters θ_i and θ_i' of each agent in the MAD3QN are updated for every episode until the MAD3QN converges, at which point the optimal policy $\boldsymbol{\pi} = \{\pi_i^* | i \in \mathcal{I}\}$ is obtained.

2) Safe Learning Mechanism: Due to the trial-and-error exploration of DRL, the agent may output certain actions violating some constraints. Compared with the performance constraints such as the AoI and service time constraints, the energy constraint should be strictly ensured, since energy is the necessary condition for the operation of MDs. This mechanism is designed to strictly ensure that the actions of each MD do not exceed its energy constraint, i.e., (21b), which is also a constraint in the RAO sub-problem. Since the tasks of MDs can not be divided, the whole task, i.e., $M_i(t)$, should be transmitted completely within time slot t. To achieve it, the energy and bandwidth used for the transmission should be guaranteed. Accordingly, based on (7)-(9), (21b) can be rewritten as

$$e_{i,k}(t) = p_i \tau_{i,k}^{\text{o,tran}}(t) = \frac{p_i M_i(t)}{\phi_{i,k}(t) W_k \log_2\left(1 + \frac{p_i h_{i,k}(t)}{\sigma^2}\right)} \le e_{\max}$$

$$\Rightarrow \phi_{i,k}(t) \ge \frac{p_i M_i(t)}{e_{\max} W_k \log_2\left(1 + \frac{p_i h_{i,k}(t)}{\sigma^2}\right)}. \quad (40)$$

Consequently, the energy constraint of D_i is equivalent to the minimum bandwidth constraint of D_i. The right term of (40) is the required minimum proportion of the bandwidth allocated by B_k to D_i to complete task offloading.

As each BS independently allocates its bandwidth, we design the safe learning mechanism from the perspective of the bandwidth of each BS. Specifically, for each time slot, after the actions of MDs are executed, the safe learning mechanism operated at the MBS will judge whether the output of MAD3QN satisfies the energy constraint, i.e., whether the sum of the required minimum bandwidth of MDs offloading to the kth BS is not more than the bandwidth W_k with $k \in \{1, ..., K+1\}$. If so, the RAO phase in the next subsection will be executed directly. If not, the safe learning mechanism will adjust the output of MAD3QN, so that the sum of the required minimum bandwidth of MDs ultimately offloading to the BS is not more than its bandwidth and the total reward of the MDs is as large as possible. Meanwhile, inspired by [36], in order to guide MDs to learn a safe policy, an extra penalty $\tilde{p}_e$ is added to the rewards of the MDs, whose offloading decisions are adjusted to be not offloading, as shown in (28).

In the following, we first prove the non-convexity and NP-harness of the MAD3QN output adjustment problem, and then develop a to solve it.

Lemma 1: The problem of adjusting the output of the MAD3QN in the safe learning mechanism is non-convex and NP-hard.

Proof: It is well known that the integer knapsack problem is non-convex and NP-hard. We first formalize the integer knapsack problem as follows: Given a knapsack with maximum capacity $\hat{W}$ and J items, each with a weight $\hat{w}_j$ and a value $\hat{v}_j$, determine which items are included in the knapsack so that the total value is the largest under the premise that the total weight is not more than $\hat{W}$. Mathematically, the integer knapsack problem is formulated as

$$\max_{\boldsymbol{x}} \sum_{j\in J} \hat{v}_j x_j \quad (41a)$$

$$s.t. \sum_{j\in J} \hat{w}_j x_j \le \hat{W}, x_j \in \{0, 1\}, \quad (41b)$$

where $x_j = 1$ represents that the jth item is included in the knapsack. Otherwise, $x_j = 0$ holds.

Then, we prove that the problem of adjusting the output of the MAD3QN in the safe learning mechanism is equivalent to an integer knapsack problem. Specifically, the kth BS, whose bandwidth W_k is smaller than the sum of the required minimum bandwidth of MDs determined to offload tasks to the kth BS by the MAD3QN, corresponds to the knapsack. Its bandwidth W_k corresponds to the maximum capacity of the knapsack $\hat{W}$. The MDs which are determined to offload tasks to the kth BS by the MAD3QN corresponds to the set of items, and x_j represents whether the jth MD ultimately offloads the task to the kth BS. The required minimum bandwidth for the jth MD to complete the transmission at time slot t corresponds to the

weight $\hat{w}_j$ as

$$\hat{w}_j = \frac{p_j M_j(t)}{e_{\max} W_k \log_2\left(1+\frac{p_j h_{j,k}(t)}{\sigma^2}\right)}. \tag{42}$$

As the MD that ultimately does not offload causes an increase of the AoI and then results in a reduction in its reward, maximizing the total reward of the MDs that ultimately offload is equivalent to finding a set of MDs such that the sum of the difference between the reward for offloading and the reward for not offloading is the largest. The updated AoI of the MD that ultimately offloads is presented in (18) and (19), and that of the MD that ultimately does not offload is presented in (16). Considering that the transmission delay and computation delay in (18) and (19) cannot be directly obtained in the safe learning mechanism, we approximate the corresponding AoI as $\tau_j^{\text{wait}}(t)$. Thus, the value $\hat{v}_j$ corresponds to the difference in reward as

$$\hat{v}_j = w_j\left(\left(A_j(t)+T+c_A\chi_j^A(t)\right)-\tau_j^{\text{wait}}(t)\right). \tag{43}$$

This completes the proof. ■

As the EGA, which performs the genetic evolution of populations by using selection, crossover and variation operators, is a fast and efficient global searching method for problems with large search space [37], we employ the EGA to address the problem of adjusting the output of the MAD3QN, which has been formulated as an integer knapsack problem in Lemma 1. Generally, the EGA first initializes a population set P, where each chromosome is encoded as a vector $\psi \in P$, and then evaluates the feasibility $\xi_\psi \in \{0,1\}$ and fitness ζ_ψ of each chromosome. Chromosomes that do not meet the feasibility, i.e., $\xi_\psi = 0$, will be replaced by the next generation. Otherwise, based on the elitist preservation strategy, chromosomes that meet the feasibility, i.e., $\xi_\psi = 1$, are preserved for the next generation with probability $\frac{\zeta_\psi}{\sum_{\psi\in P}\zeta_\psi}$. The elitist preservation strategy allows the fittest chromosomes from the current generation to carry over to the next. That is to say, the solution quality obtained by the EGA would be better from one generation to the next [37]. The generation process repeats until the number of iterations g reaches the predetermined maximum number of iterations G. For the problem of adjusting the output of the MAD3QN, the candidate offloading decision of MDs corresponds to the chromosome as

$$\psi = \begin{bmatrix} x_1 \\ \vdots \\ x_J \end{bmatrix}. \tag{44}$$

The optimization objective of the problem corresponds to the fitness of the chromosome as

$$\zeta_\psi = \sum_{j\in J}\hat{v}_j x_j, \psi = [x_1 ... x_J]^T. \tag{45}$$

Whether the candidate offloading decision of the MDs, i.e., (44), satisfies the constraint (41b), where $\hat{w}_j$ is presented in (42), corresponds to whether the chromosome meets feasibility as

$$\xi_\psi = \begin{cases} 1, & \sum_{j\in J}\hat{w}_j x_j \le \hat{W}, \\ 0, & \text{Otherwise.} \end{cases} \tag{46}$$

By using the defined chromosome in (44), fitness in (45), and feasibility in (46), EGA can be applied straightforwardly.

B. Convex Optimization for the RAO Sub-problem

After the offloading decision $\mathbf{\Pi}(t)$ that strictly satisfies (21b) is given by the safe multi-agent DRL algorithm, $\mathbf{P0}$ degenerates into the RAO sub-problem that optimizes the bandwidth and computation resource allocation at each time slot t as

$$\mathbf{P1}: \min_{\boldsymbol{\phi}(t),\boldsymbol{f}(t)} \sum_{i=1}^{I} w_i A_i(t+1) \tag{47a}$$

$$s.t. \quad (21b), (21d)\text{-}(21g). \tag{47b}$$

The variables $\boldsymbol{\phi}(t)$ and $\boldsymbol{f}(t)$ in $\mathbf{P1}$ are independent of each other, hence the constraints (21b), (21d), (21e) and the constraints (21f), (21g) are separable. Accordingly, we can decompose two time slot-level sub-problems as follows.

1) Bandwidth Allocation Optimization: The first time slot-level sub-problem concerning the variable $\boldsymbol{\phi}(t)$ is to optimize the bandwidth allocation for task offloading as

$$\mathbf{P2}: \min_{\boldsymbol{\phi}(t)} \sum_{i=1}^{I} w_i \Bigg(\Big(\beta_i(t) + \sum_{\tilde{k}=1}^{K} \tilde{\beta}_{i,\tilde{k}}(t)\Big)\tau_{i,k}^{\text{o,tran}}(t) + \sum_{\tilde{k}=1}^{K} \gamma_{i,\tilde{k}}(t)\tau_{i,\tilde{k}}^{\text{t,tran}}(t)\Bigg) \tag{48a}$$

$$s.t. \quad (21b), (21d), (21e), \tag{48b}$$

where $\tau_{i,k}^{\text{o,tran}}(t)$ and $\tau_{i,\tilde{k}}^{\text{t,tran}}(t)$ are functions of $\phi_{i,k}(t)$ according to (7)-(8) and (10)-(11), respectively. The objective function in (48a) can be proven to be a convex function of $\boldsymbol{\phi}(t)$ as the second-order derivative of (48a) with respect to $\boldsymbol{\phi}(t)$ is greater than 0. Based on (40), (21b) is a linear constraint of $\boldsymbol{\phi}(t)$. Therefore, $\mathbf{P2}$ is a convex optimization problem. As the closed-form solution of $\mathbf{P2}$ can not be obtained through KKT conditions, we use the primal-dual interior-point method via a convex optimization tool to solve it.

2) Computation Resource Allocation Optimization: The second time slot-level sub-problem concerning the variable $\boldsymbol{f}(t)$ is to optimize BSs' computation resources, which is expressed as

$$\mathbf{P3}: \min_{\boldsymbol{f}(t)} \sum_{i=1}^{I} w_i \Bigg(\sum_{\tilde{k}=1}^{K} \tilde{\beta}_{i,\tilde{k}}(t)\tau_{i,\tilde{k}}^{\text{comp}}(t) + \Big(\beta_i(t) + \sum_{\tilde{k}=1}^{K}\gamma_{i,\tilde{k}}(t)\Big)\tau_{i,K+1}^{\text{comp}}(t)\Bigg) \tag{49a}$$

$$s.t. \quad (21f), (21g). \tag{49b}$$

As the computation resource allocation among BSs is independent of each other, $\mathbf{P3}$ can degenerate into the computation resource allocation optimization problem for each BS, which is formulated as

$$\mathbf{P4}: \min_{\boldsymbol{f_k}(t)} \sum_{n\in\mathcal{N}_k(t)} \frac{w_n M_n(t) C_n}{f_{n,k}(t)} \tag{50a}$$

$$s.t. \sum_{n\in\mathcal{N}_k(t)} f_{n,k}(t) \le F_k. \tag{50b}$$

In $\mathbf{P4}$, $\mathcal{N}_k(t)$ denotes the set of MDs that offload tasks to B_k at time slot t and $\boldsymbol{f_k}(t)=\big[f_{1,k}(t), ..., f_{|\mathcal{N}_k(t)|,k}(t)\big]$ represents

the computation resource allocation of B_k for MDs at time slot t, where $|\mathcal{N}_k(t)|$ represents the size of $\mathcal{N}_k(t)$. $\mathbf{P4}$ is a typical convex optimization problem with respect to $\boldsymbol{f_k}(t)$, and we adopt the Lagrange multiplier method to solve it. The Lagrangian function is expressed as

$$L(\boldsymbol{f_k}(t),\lambda)=\sum_{n\in\mathcal{N}_k(t)}\frac{w_nM_n(t)C_n}{f_{n,k}(t)}-\lambda\left(F_k-\sum_{n\in\mathcal{N}_k(t)}f_{n,k}(t)\right), \quad (51)$$

where λ denotes the Lagrange multiplier. By respectively calculating the partial derivatives of (51) with respect to $f_{n,k}(t)$ and λ, we obtain the KKT conditions as

$$\begin{cases} \left.\frac{\partial L(\boldsymbol{f_k}(t),\lambda)}{\partial f_{n,k}(t)}\right|_{n\in\mathcal{N}_k(t)}=\frac{-w_nM_n(t)C_n}{(f_{n,k}(t))^2}+\lambda=0, \\ \frac{\partial L(\boldsymbol{f_k}(t),\lambda)}{\partial\lambda}=F_k-\sum_{n\in\mathcal{N}_k(t)}f_{n,k}(t)\geq 0, \\ \lambda(F_k-\sum_{n\in\mathcal{N}_k(t)}f_{n,k}(t))=0, \\ \lambda\geq 0. \end{cases} \quad (52)$$

By solving (52), we obtain the closed-form solution of the kth BS's computation resource allocation for the MDs offloading to it as

$$f^*_{n,k}(t)=\frac{F_k\sqrt{w_nM_n(t)C_n}}{\sum_{n\in\mathcal{N}_k(t)}\sqrt{w_nM_n(t)C_n}},\forall n\in\mathcal{N}_k(t). \quad (53)$$

Accordingly, we obtain the closed-form solution of $\mathbf{P3}$.

C. Federated Learning

When adapting the single-agent DRL algorithms to the multi-agent setting, there exists a problem named the training instability problem [38]. To be specific, during the training process, the policy of each agent is changing, which leads to continuous variations in the environment. However, the variations are not explainable by changes in each agent's own policy, hence the environment becomes non-stationary from the perspective of each individual agent. The training instability problem can destabilize the training process and lead to poor performance. According to [39], FL has positive impacts on the multi-agent DRL, hence we turn to FL to alleviate this problem. FL is integrated with multi-agent DRL to improve the training performance by consolidating and synchronizing the local DRL model parameters of distributed agents [40]. In this way, each agent can implicitly obtain certain knowledge of the network beyond its individual training experience, so as to make adequate offloading decisions.

In FL, local devices use their local data to train DRL models and send the model parameters rather than raw data to a central server. The central server aggregates these model parameters to obtain the global DRL model and then sends the global model parameters back to local devices [24]. Here, MDs are regarded as local devices, and the MBS is regarded as the central server. At the beginning of the training process, all MDs download initial global model parameters from the MBS. After all MDs complete a certain number of episodes d_{agg}, they undergo one round of federated aggregation. Specifically, MDs transmit their local DRL model parameters θ_i and θ'_i to the MBS, and the MBS aggregates all local DRL model parameters to obtain new global DRL model parameters θ_g and θ'_g, which are respectively expressed as

$$\theta_g\leftarrow\frac{1}{I}\sum_{i\in I}\theta_i,\ \theta'_g\leftarrow\frac{1}{I}\sum_{i\in I}\theta'_i. \quad (54)$$

Once the DRL model parameter aggregation is completed, the MBS broadcasts the new global DRL model parameters θ_g and θ'_g to all MDs. All MDs use the new global DRL model parameters to update their local DRL model parameters as

$$\theta_i\leftarrow\theta_g,\ \theta'_i\leftarrow\theta'_g. \quad (55)$$

Then each MD uses the updated local DRL model for decision making and training at the next episode.

Until now, the three components of the proposed FL-SMADC algorithm are finished, and the complete FL-SMADC algorithm is given in the Algorithm 1.

Algorithm 1: Training Algorithm of FL-SMADC

Input: Training episode length EP; training steps $\mathcal{T}$; learning rate $\tilde{\alpha}$; discount factor $\tilde{\gamma}$; experience replay buffer of each agent $\mathcal{B}_i$; size of mini-batch $|\mathcal{M}_i|$; exploitation noise ϵ; FL aggregation interval d_{agg}; population size $|P|$; maximum number of iterations G.
Output: Optimal policy $\boldsymbol{\pi}=\{\pi^*_i|i\in\mathcal{I}\}$.

Initialize: The Q network and the target Q network of each agent, e.g., $Q(s,a;\theta_i)$ with parameters θ_i, and $Q'(s,a;\theta'_i)$ with parameters $\theta'_i\leftarrow\theta_i$ of the agent, the experience replay buffers, and the global model parameters (θ_g,θ'_g).
for $ep=1,2,\ldots,EP$ **do**
  Initialize the distribution and priorities of MDs;
  // Note: The priorities of MDs ω_i with $i\in\{1,2,...,I\}$ keep unchanged within each episode but vary across different episodes. This setting, on the one hand, is consistent with the real-world scenario where the priorities of MDs would not change frequently, and on the other hand, makes each agent learn a robust policy that adapts to different priorities.
  for $t=1,2,\ldots,\mathcal{T}$ **do**
    The MBS collects the current state information, and then broadcasts it and the rewards of agents at time slot $(t-1)$ to MDs;
    for each agent $i=1,2,\ldots,I$ **do**
      Agent i obtains reward $r_i(t-1)$ and current state $s_i(t)$, and stores tuple $(s_i(t-1),a_i(t-1),r_i(t-1),s_i(t))$ into $\mathcal{B}_i$;
      Agent i selects action $a_i(t)$ according to the ϵ-greedy policy;
    end
    MDs transmit offloading decisions to the MBS;
    The MBS receives MDs' offloading decisions and calculates the sum of the required minimum bandwidth of MDs for each BS;
    if the sum of the required minimum bandwidth of MDs offloading to the kth BS $>W_k$ **then**
      while $g<G$ **do**
        The safe learning mechanism initializes a population set of P with random chromosomes, and evaluates ζ_ψ and ξ_ψ according to (45) and (46);
        The safe learning mechanism performs according to the elitist preservation strategy;
        $g\leftarrow g+1$;
      end
      The MBS obtains the adjusted offloading decisions for the kth BS;
    end
  end
  The MBS calculates bandwidth allocation $\boldsymbol{\phi}(t)$ by using the primal-dual interior-point method, and computation resource allocation $\boldsymbol{f}(t)$ by closed-form solution in (53);
  The MBS calculates the rewards of MDs according to (27);
end
for each agent $i=1,2,\ldots,I$ **do**
  Agent i randomly samples a mini-batch $\mathcal{M}_i$ from $\mathcal{B}_i$ and calculates loss function according to (38);
  Agent i updates Q network parameters θ_i according to (39), and updates target Q network parameters θ'_i every 2 episodes;
end
if ep mod $d_{\text{agg}}==0$ **then**
  MDs transmit θ_i and θ'_i to the MBS;
  The MBS aggregates the parameters of MDs to obtain the global DRL model parameters as $\theta_g\leftarrow\frac{1}{I}\sum_{i\in I}\theta_i$, $\theta'_g\leftarrow\frac{1}{I}\sum_{i\in I}\theta'_i$, and broadcasts the global DRL model parameters (θ_g,θ'_g);
  MDs update the local DRL parameters as $\theta_i\leftarrow\theta_g$, $\theta'_i\leftarrow\theta'_g$;
end

D. Computational Complexity Analysis

As the proposed FL-SMADC is a multi-agent DRL algorithm, where multiple agents operate in parallel, the computational complexity analysis in this subsection refers to the computational complexity of each agent. In the following, we describe the computational complexity of FL-SMADC per episode, which consists of three parts: DRL agents' forward propagation in each training step (i.e., each time slot), the calculations of $\boldsymbol{\phi}(t)$ and $\boldsymbol{f}(t)$ by convex optimization in each training step, and DRL agents' backward propagation in each episode. For both the forward and backward propagation, let L denote the number of DNN layers and n_q with $q \in \{1, ..., L\}$ denote the number of neurons at the qth layer. The computational complexity of the forward propagation in each training step is $O(\sum_{q=1}^{L-1} n_q n_{q+1})$, and that of the backward propagation in each episode is $O(|\mathcal{M}_i| \sum_{q=1}^{L-1} n_q n_{q+1})$. As $\boldsymbol{\phi}(t)$ is calculated by the primal-dual interior-point method, the corresponding computational complexity is $O(I^3 m)$ [41], where m is the length of a binary coding of the input data. As the closed-form solution of $\boldsymbol{f}(t)$ is calculated by the KKT conditions, the corresponding computational complexity is $O(I)$. Accordingly, the computational complexity of the calculations of $\boldsymbol{\phi}(t)$ and $\boldsymbol{f}(t)$ by convex optimization in each training step is $O(I^3 m + I)$. Based on the computational complexities of the above three parts, we have the following conclusions: the computational complexity of FL-SMADC algorithm for one time slot is $O(\sum_{q=1}^{L-1} n_q n_{q+1} + I^3 m + I)$; the computational complexity for one episode is $O(\mathcal{T}(\sum_{q=1}^{L-1} n_q n_{q+1} + I^3 m + I) + |\mathcal{M}_i| \sum_{q=1}^{L-1} n_q n_{q+1})$, where $\mathcal{T}$ denotes the number of training steps for one episode and $|\mathcal{M}_i|$ denotes the size of mini-batch; and the computational complexity for the whole training process is $O(N_{EP}(\mathcal{T}(\sum_{q=1}^{L-1} n_q n_{q+1} + I^3 m + I) + |\mathcal{M}_i| \sum_{q=1}^{L-1} n_q n_{q+1}))$, where N_{EP} denotes the number of episodes required for convergence (NERC).

V. Simulation Results

A. Simulation Settings and Comparative Algorithms

We consider the HMECN as a circular area with radius $R_{K+1} = 300\,\text{m}$ [28], where the MBS is located at the center of the area, and SBSs and MDs are distributed following the Poisson point process. The data size of the generated task follows the Gaussian distribution [5] with the range of [100,1000] Kb. The priorities of MDs follow the uniform distribution [1] with the set of values {1,2,3,4,5} [22]. Other simulation parameters are summarized in Table III unless otherwise specified.

To demonstrate the effectiveness and superiority of the proposed FL-SMADC algorithm, we compare it with the following comparative algorithms:

- Random offloading and convex optimization (ROC): Each agent at the MD makes its offloading decision in a random way and is bound to offload the task once its AoI in the next time slot is predicted to exceed $A_{\max}$. The MBS employs the random adjustment scheme, i.e., the MDs that offload tasks are randomly selected to adjust the offloading decisions until the energy constraints of all MDs are satisfied. Then the MBS still calculates the bandwidth and computation resource by convex optimization.
- FL-assisted safe multi-agent DQN and convex optimization (FL-SMADQC): The only difference between this algorithm and the proposed FL-SMADC algorithm is that each MD in this algorithm employs DQN to output its offloading decision instead of D3QN. The values of hyperparameters of the multi-agent DRL, safe learning mechanism and FL in FL-SMADQC are the same as those in FL-SMADC.
- FL-assisted multi-agent DRL and centralized twin delayed deep deterministic policy gradient (FL-MADTD3): The differences between this algorithm and the proposed FL-SMADC algorithm are that, this algorithm only uses MAD3QN to output the offloading decisions without the safe learning mechanism and employs the centralized TD3 to output the bandwidth and computation resource allocation instead of the convex optimization in FL-SMADC. The values of hyperparameters of the multi-agent DRL and FL in FL-MADTD3 are the same as those in FL-SMADC.
- Safe multi-agent DRL and convex optimization (SMADC): Compared with the proposed FL-SMADC algorithm, this algorithm only consists of two components, i.e., the safe multi-agent DRL algorithm and convex optimization, without FL. The values of hyperparameters of the multi-agent DRL and safe learning mechanism in SMADC are the same as those in FL-SMADC.
- FL-assisted safe multi-agent DRL and convex optimization with coverage-based offloading model (FL-SMADC-CO): This algorithm adopts the same solution method as the

TABLE III
Simulation parameters

Parameters	**Values**
Number of MDs I	30
Number of SBSs K	4
Radius of SBSs $R_{\tilde{k}}$	100 (m) [42]
Number of time slots $\mathcal{T}$	2000
Duration of a time slot T	0.5 (s)
Duration of CP τ_0	0.1 (s)
Movement duration of MDs τ_m	0.05 (s)
Task generation probability P_g	0.7
Velocity memory parameter μ_1	0.9 [32]
Direction memory parameter μ_2	0.9 [32]
Average velocity of MDs $\overline{v}$	10 (m/s)
Average direction of MDs $\overline{\varphi}_i$	$[-\pi, \pi]$
Path loss exponent ℓ	3
Channel correlation factor ρ	0.6 [33]
Total bandwidth of each SBS $W_{\tilde{k}}$	20 (MHz) [25]
Total bandwidth of the MBS W_{K+1}	50 (MHz) [1]
Transmit power of MDs p_i	23 (dBm)
Transmit power of SBSs $p_{\tilde{k}}^{\text{s}}$	40 (dBm)
Noise power σ^2	-110 (dBm) [9]
CPU cycles required for computing one bit C_i	100
Computation capacity of each SBS $F_{\tilde{k}}$	1 (GHz) [28]
Computation capacity of the MBS F_{K+1}	20 (GHz) [24]
Maximum energy consumption of MDs $e_{\max}$	1 (mJ)
Maximum tolerant AoI $A_{\max}$	2 (s)
Training episode length EP	1000
Learning rate $\tilde{\alpha}$	0.00005
Discount factor $\tilde{\gamma}$	0.95
FL aggregation interval d_{agg}	10
Mini-batch size $\lvert\mathcal{M}_i\rvert$	256
Experience replay buffer size $\lvert\mathcal{B}_i\rvert$	2^{17}
Population size $\lvert P\rvert$	50
Maximum number of iterations G	100

proposed FL-SMADC and the only difference between them lies in the offloading model, i.e., MDSs in this algorithm can only offload tasks to SBSs as [9] and [10]. The values of hyperparameters of the multi-agent DRL, safe learning mechanism and FL in FL-SMADC-CO are the same as those in FL-SMADC.

- Fully centralized TD3 (FCTD3) [43]: Taking the MBS as a centralized agent, this algorithm employs TD3 to output the offloading decisions, and the bandwidth and computation resource allocation. The values of learning rate, discount factor, mini-batch size and the experience replay buffer size in FCTD3 are the same as those in FL-SMADC.

B. Performance Evaluation

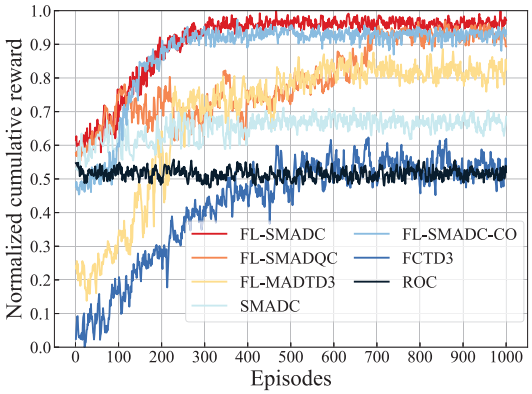


Fig. 7. The normalized cumulative reward versus episodes.

Fig. 7 shows the normalized cumulative reward under seven algorithms versus episodes. We observe that the proposed FL-SMADC converges to the largest reward among all the algorithms and converges fastest except for FL-SMADC-CO. The reason is that, compared with FL-SMADQC, D3QN in FL-SMADC greatly mitigates the overestimation of Q value in DQN by taking advantages of the dueling DQN and double DQN. Compared with FL-MADTD3, with the help of the safe learning mechanism, which provides the guarantee condition for the proof of the convexity of the RAO sub-problem, convex optimization in FL-SMADC outputs the optimal bandwidth and computation resource allocation under the given offloading decision and avoids to train another DRL algorithm, i.e., the TD3 in FL-MADTD3. Compared with SMADC, FL in FL-SMADC breaks the limitation of partial knowledge on the network for each individual agent in MAD3QN. By comparing with the above three algorithms, the superiorities of the three components in FL-SMADC are respectively verified. Compared with FL-SMADC-CO, each agent in FL-SMADC has more offloading decisions (i.e., a higher dimensional action space) to fully use the whole network resource. Consequently, FL-SMADC achieves a larger reward at the cost of worse convergence performance. Compared with FCTD3, where the state and action space dimensions increase significantly with the number of MDs, resulting in long training time and poor performance, FL-SMADC avoids the curse of dimensionality by distributing the offloading decision-making process to multiple MDs and using convex optimization to obtain the optimal resource allocation.

TABLE IV
PERFORMANCE COMPARISON UNDER 95% CONFIDENCE INTERVAL (CI)

Algorithm	Mean value	95% CI	CI width
FL-SMADC	0.95	[0.92, 0.98]	0.06
FL-SMADQC	0.93	[0.88, 0.98]	0.10
FL-MADTD3	0.85	[0.76, 0.94]	0.18
SMADC	0.74	[0.69, 0.79]	0.10
FL-SMADC-CO	0.93	[0.90, 0.96]	0.06
FCTD3	0.65	[0.53, 0.77]	0.24

Note: The mean value represents the normalized cumulative reward obtained by each algorithm after convergence. Results are obtained based on 10 independent training runs.

TABLE V
QUANTITATIVE PERFORMANCE GAINS OF FL-SMADC COMPARED WITH OTHER ALGORITHMS

Algorithm	LWSA Reduction (%)	NERC Reduction (%)
vs FL-SMADQC	3.81	58.59
vs FL-MADTD3	13.94	47.55
vs SMADC	37.03	20.35
vs FL-SMADC-CO	6.10	-24.42
vs FCTD3	49.74	39.09
vs ROC	51.64	–

Note: AoI reduction (%) and NERC reduction (%) represent the relative gain achieved by FL-SMADC compared with other algorithms. The symbol "–" indicates that the metric is not applicable for ROC (a non-DRL algorithm).

To strengthen the convincingness of the superiority of the proposed FL-SMADC in terms of performance and training stability, we further provide two tables that compare FL-SMADC with the comparative algorithms in a quantitative way. Based on 10 independent training runs, we construct the 95% confidence interval (CI) for the normalized cumulative reward of each algorithm after convergence in Table IV. We observe that the proposed FL-SMADC achieves the largest mean value of 0.95 with the shortest CI width of 0.06, strongly validating the superiority of the proposed algorithm in terms of the reward and training stability. Table V shows the LWSA reduction (i.e., LWSA performance gain) and the NERC reduction (i.e., convergence performance gain) of the proposed FL-SMADC compared with the comparative algorithms. We observe that the proposed FL-SMADC achieves the best LWSA performance and the second-best convergence performance, strongly validates the superiority of the proposed algorithm in terms of LWSA and convergence performance. The reason why FL-SMADC achieves better LWSA performance but worse convergence performance than FLSMADC-CO is explained in Fig. 7, and we do not repeat it.

Fig. 8 shows the normalized cumulative reward under FL-SMADC with different values of learning rate $\tilde{\alpha}$ versus episodes. We observe that FL-SMADC converges faster with the increase of $\tilde{\alpha}$. This is due to the reason that, the updated magnitude of parameters in FL-SMADC is positively related to $\tilde{\alpha}$. We also observe that the normalized cumulative reward first increases with $\tilde{\alpha} \in [0.0000005, 0.000005]$ and then decreases with $\tilde{\alpha} \in [0.00005, 0.005]$. The reason is that, for the small learning rates, the agents are difficult to jump out of the region of the local optima, and then they would fall into the local optima rather than the global optima. For the large learning rates, the agents are easy to skip the region of the global optima,

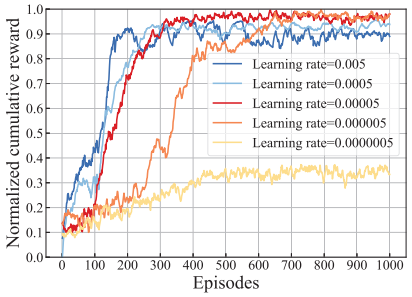


Fig. 8. The normalized cumulative reward under FL-SMADC with different values of learning rate $\tilde{\alpha}$ versus episodes.

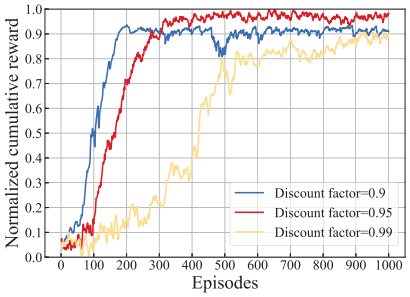


Fig. 9. The normalized cumulative reward under FL-SMADC with different values of discount factor $\tilde{\gamma}$ versus episodes.

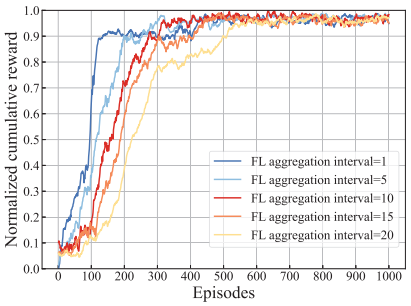


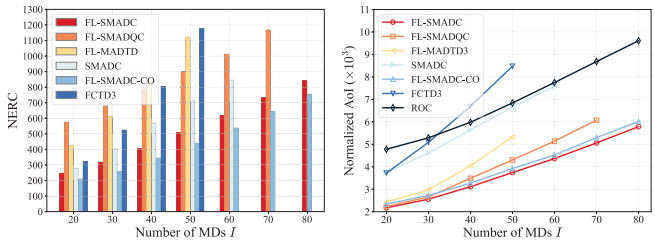
Fig. 10. The normalized cumulative reward under FL-SMADC with different values of FL aggregation interval d_{agg} versus episodes.

and then they are likely to tend towards the local optima. To balance the convergence performance and the normalized cumulative reward, we choose the learning rate as 0.00005.

Fig. 9 shows the normalized cumulative reward under FL-SMADC with different values of discount factor $\tilde{\gamma}$ versus episodes. We observe that the normalized cumulative reward first increases and then decreases with $\tilde{\gamma}$, and the convergence performance of FL-SMADC becomes worse with the increase of $\tilde{\gamma}$. The reason is that, a small value of $\tilde{\gamma}$ means that the DRL agent focuses more on the immediate reward than the future reward, leading to short-sighted decision making and a fast convergence speed. In contrast, a large value of $\tilde{\gamma}$ means that the DRL agent focuses more on the future reward than the immediate reward. However, in the dynamic HMECN, it is difficult for the DRL agent to accurately predict environmental states in the far future, and inaccurate environmental states result in inaccurate and unstable decision making, which further leads to a slow convergence speed. Based on the aforementioned discussion, we adopt $\tilde{\gamma} = 0.95$ in FL-SMADC to balance the immediate and future reward.

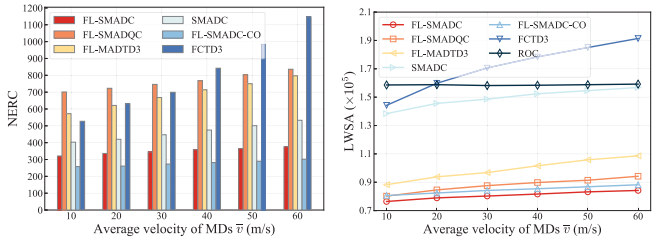
Fig. 10 shows the normalized cumulative reward under FL-SMADC with different values of aggregation interval d_{agg} versus episodes. We observe that d_{agg} has an impact on the convergence performance but has no impact on the reward. Moreover, the convergence performance becomes better $d_{\text{agg}} \in [1, 5]$ and becomes worse with $d_{\text{agg}} \in [5, 25]$. The reason is that, there exists a tradeoff between the aggregation frequency of local DRL models and the training adequacy of local DRL models. Specifically, for small values of d_{agg}, although the frequent aggregation accelerates the integration of local DRL model parameters into the global DRL model, the under-trained local DRL model parameters degrade the convergence performance. For large values of d_{agg}, although local DRL models are trained adequately, low aggregation frequency degrades the convergence performance.

Fig. 11 shows the convergence performance, measured by the NERC and the normalized AoI versus the number of MDs I. It is meaningless to discuss the convergence performance of ROC, so we do not present its curve about NERC in Fig. 11 (a) and Fig. 12 (a). We observe that both the NERC and normalized AoI under all the algorithms increase with I. The reason why the NERC increases with I is that, for the multi-agent DRL algorithms, i.e., FL-SMADC, FL-SDQNC, FL-MADTD3, SMADC, and FL-SMADC-CO, the increase of I leads to the increase of the dynamic complexity of the



(a) (b)

Fig. 11. (a) The NERC versus I. (b) The normalized AoI versus I.



(a) (b)

Fig. 12. (a) The NERC versus $\bar{v}$. (b) The LWSA versus $\bar{v}$.

environment and the dimensionality of the policy space, which results in longer time for convergence. For the centralized DRL algorithms, i.e., the centralized TD3 in FL-MADTD3 and FCTD3, the increase of I leads to the increase of state and action space dimensions, which also results in longer time for convergence. The reason why the normalized AoI increases with I is that, the resource allocated to each MD that determines to offload decreases with I, and on the other hand, the number of MDs that determine not to offload increases with I due to the limited number of MDs that could offload in each time slot. We also observe that both the NERC and normalized AoI under FL-SMADC and FL-SMADC-CO increase more slowly than other algorithms, and when the number of MDs reaches 80, only FL-SMADC and FL-SMADC-CO could converge within a reasonable number of episodes. These observations strongly demonstrate the superiority of the proposed algorithm in terms of the scalability.

Fig. 12 shows the NERC and LWSA under seven algorithms versus the average velocity of MDs $\bar{v}$. We observe that both the NERC and LWSA under all the algorithms increase with $\bar{v}$. The reason is that, the dynamicity of the network increases with $\bar{v}$, which increases the difficulty of the training process of DRL algorithms. We also observe that both the NERC and LWSA

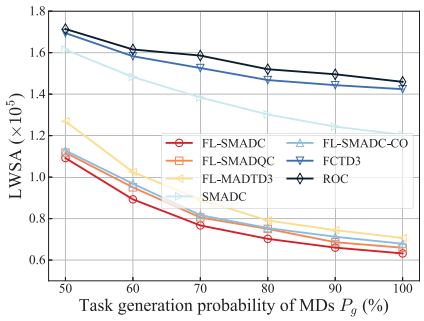


Fig. 13. The LWSA versus P_g.

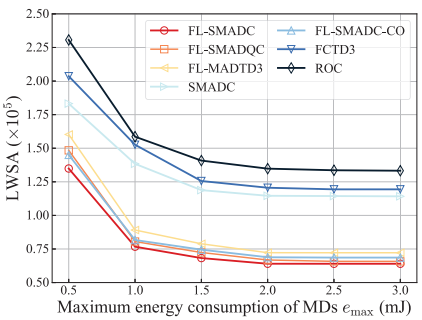


Fig. 14. The LWSA versus $e_{\max}$.

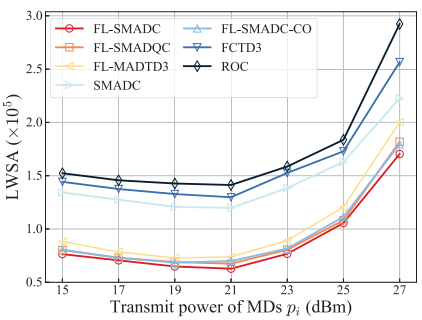


Fig. 15. The LWSA versus p_i.

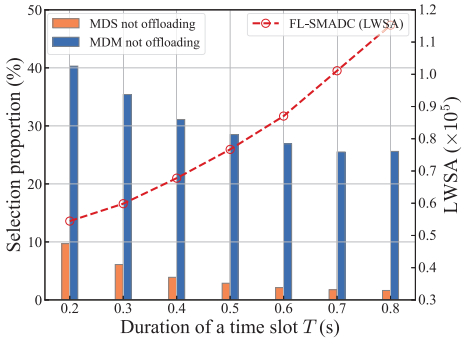


Fig. 16. The proportion of MDs that select not to offload tasks, and the LWSA under FL-SMADC versus T.

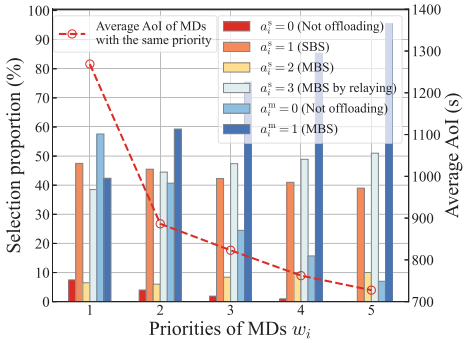


Fig. 17. The selection proportion of MDs on the offloading decisions, and the average AoI of MDs under FL-SMADC versus w_i.

under FL-SMADC and FL-SMADC-CO increase more slowly than other algorithms, which demonstrates the robustness of the proposed algorithm in mobile or dynamic network environment. It is worth noting that the LWSA under ROC keeps unchanged with the increase of $\bar{v}$, since ROC employs the same random policy for every time slot. Hence, it is meaningless to compare the impact of mobility on the performance under the proposed algorithm with that under ROC.

Fig. 13 shows the LWSA under seven algorithms versus the task generation probability of MDs P_g. We observe that the LWSA decreases rapidly first and then slowly with P_g. The reason is that, for small values of P_g, the number of MDs with tasks to offload is less than the number of MDs that the network resource could support to offload. In other words, the network resource is not fully utilized due to the small number of MDs with tasks to offload. Hence, the LWSA first decreases rapidly with P_g. For large values of P_g, when the number of MDs with tasks to offload reaches a certain value, it will no longer affect the LWSA intuitively. However, the number of MDs with high priorities still increases with P_g, and more tasks from the MDs with high priorities would be offloaded. Accordingly, the LWSA would still decrease with P_g, but slowly.

Fig. 14 shows the LWSA under seven algorithms versus the maximum energy consumption of MDs $e_{\max}$. We observe that the LWSA first decreases and then keeps unchanged with $e_{\max}$. According to (40), the energy constraint of each MD is equivalent to the minimum bandwidth constraint of each MD. For small values of $e_{\max}$, some MDs with tasks to offload cannot transmit to BSs due to insufficient energy (i.e., insufficient bandwidth resources). Hence, the LWSA decreases with $e_{\max}$ for $e_{\max} < 2$ mJ. For large values of $e_{\max}$, all the MDs with tasks to offload could transmit to BSs, and the bottleneck that limits the LWSA becomes the computation resources of BSs. Hence, the LWSA keeps unchanged with $e_{\max}$ for $e_{\max} \geq 2$ mJ.

Fig. 15 shows the LWSA under seven algorithms versus the transmit power of MDs p_i. We observe that the LWSA first decreases and then increases with p_i. This observation can be illustrated as follows. According to (7), (8) and (11), the transmission delay of the MDs that offload tasks to BSs decreases with p_i, which reduces the AoI of the MDs based on (20). Accordingly, the LWSA would decrease. On the other hand, according to (40), the required minimum bandwidth of MDs increases with p_i. When p_i increases to a certain value, the sum of the required minimum bandwidth of MDs that offload tasks to a given BS at each time slot would exceed the bandwidth of the BS. When p_i continues to increase, the number of MDs that offload tasks at each time slot would decrease. As a result, the LWSA would increase. In other words, there is a tradeoff between the transmission delay of the MDs and the number of MDs that offload tasks to BSs. As the impact of p_i on the number of MDs that offload tasks takes effect only when p_i increases to a certain value, for small values of p_i, i.e., $p_i < 21$ dBm, the transmission delay is the key factor that influences the LWSA. Hence the LWSA decreases with p_i. For large values of p_i, i.e., $p_i \geq 21$ dBm, the number of MDs that offload tasks becomes the key factor that influences the LWSA, hence the LWSA increases with p_i.

Fig. 16 shows the proportion of MDs that select not to offload tasks and the LWSA under FL-SMADC versus the duration of a time slot T. We observe that the proportion of MDs that select not to offload tasks first decreases and then keeps unchanged with T. The reason is that, for $T < 0.7$ s, there are some MDs that select not to offload tasks due to the service time constraint. Hence, the proportion first decreases with T

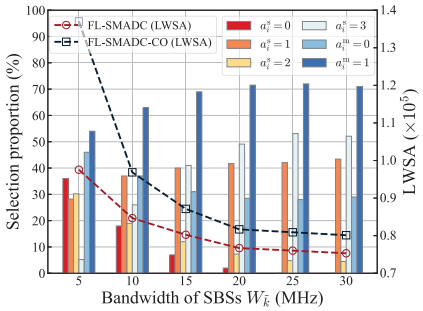


Fig. 18. The selection proportion of MDs on the offloading decisions under FL-SMADC, and the LWSA under FL-SMADC and FL-SMADC-CO versus $W_{\tilde{k}}$.

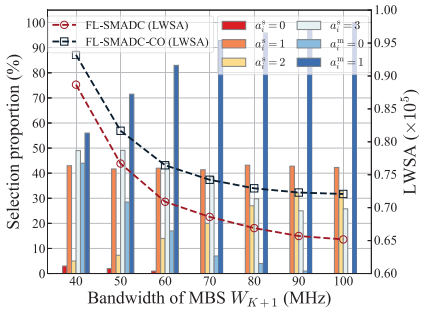


Fig. 19. The selection proportion of MDs on the offloading decisions under FL-SMADC, and the LWSA under FL-SMADC and FL-SMADC-CO versus W_{K+1}.

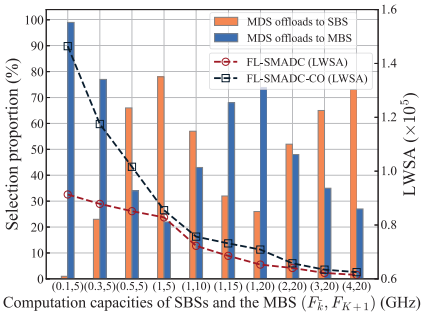


Fig. 20. The selection proportion of MDSs on the offloading decisions under FL-SMADC, and the LWSA under FL-SMADC and FL-SMADC-CO versus $(F_{\tilde{k}}, F_{K+1})$ when $e_{\max} = 100\,\text{mJ}$.

for $T < 0.7\,\text{s}$. For $T \geq 0.7\,\text{s}$, the bottleneck that limits MDs to offload tasks becomes the energy or bandwidth resource. Hence, the proportion keeps unchanged with T for $T \geq 0.7\,\text{s}$. We also observe that the LWSA increases with T. The reason is that the AoI is positively associated with T according to the evolution of AoI in (20), and the AoI reduction resulting from the MDs that select to offload tasks due to the increase of T cannot compensate for the AoI increment resulting from the increase of T.

Fig. 17 shows the selection proportion of MDs on the offloading decisions and the average AoI of MDs with the same priority under FL-SMADC versus the priorities of MDs w_i, where a_i^{s} and a_i^{m} presented in (26) denote the offloading decisions of MDSs and MDMs, respectively. We observe that for MDSs, both the selection proportion of not offloading and that of offloading to SBSs decrease with w_i, while both the selection proportion of offloading to the MBS directly and that of offloading to the MBS by relaying increase with w_i. For MDMs, the selection proportion of not offloading decreases with w_i, while the selection proportion of offloading to the MBS increases with w_i. We also observe that the average AoI of MDs with the same priority decreases with w_i. The common reason for the above observations is that, MDs with higher priorities have more important impacts on the LWSA and accordingly would be allocated more bandwidth and computation resources. For MDSs, the reason why the selection proportion of offloading to SBSs decreases with w_i is that SBSs need to use parts of their bandwidth for relaying, thereby utilizing the computation capacity of the MBS. Moreover, we notice that the selection proportion of not offloading for MDSs is significantly smaller than that for MDMs. This is due to the reason that MDSs have more offloading options, which allow them to utilize the resources of both SBSs and MBS.

Fig. 18 shows the selection proportion of MDs on the offloading decisions under FL-SMADC and the LWSA under FL-SMADC and FL-SMADC-CO versus the bandwidth of SBSs $W_{\tilde{k}}$. We observe that when the value of $W_{\tilde{k}}$ is small (i.e., $W_{\tilde{k}} < 25\,\text{MHz}$ in Fig. 18), for MDSs, the selection proportion of not offloading decreases with $W_{\tilde{k}}$, while both the selection proportion of offloading to SBSs and that of offloading to the MBS by relaying increase with $W_{\tilde{k}}$. The reason is that, for small values of $W_{\tilde{k}}$, the bandwidth of SBSs limits the number of MDSs offloading to SBSs and that offloading to the MBS by the relaying of SBSs, resulting in the underutilization of the computation capacities of both SBSs and MBS. We also observe that when the value of $W_{\tilde{k}}$ is small (i.e., $W_{\tilde{k}} < 25\,\text{MHz}$ in Fig. 18), for MDSs, the selection proportion of offloading to the MBS directly decreases with $W_{\tilde{k}}$; for MDMs, the selection proportion of not offloading decreases with $W_{\tilde{k}}$, while the selection proportion of offloading to the MBS increases with $W_{\tilde{k}}$. This is due to the reason that, for MDSs, offloading to the MBS by the relaying of SBSs has no minimum bandwidth constraint on MBS at the cost of SBSs' bandwidth. Hence, with the increase of $W_{\tilde{k}}$, MDSs are more likely to utilize the computation capacity of the MBS by the relaying of SBSs rather than direct offloading, and at the same time, more bandwidth of MBS would be released to MDMs. We notice that when the value of $W_{\tilde{k}}$ is large (i.e., $W_{\tilde{k}} \geq 25\,\text{MHz}$ in Fig. 18), all the selection proportions for both MDSs and MDMs keep nearly unchanged with $W_{\tilde{k}}$, showing that the offloading decisions are no longer affected by the bandwidth resource of the network, but other factors such as the computation resource. This is also the reason why the LWSA under FL-SMADC and FL-SMADC-CO first decreases and then keeps nearly unchanged with $W_{\tilde{k}}$. By observing the curves about the LWSA, we find that the gap between FL-SMADC and FL-SMADC-CO increases with the decrease of $W_{\tilde{k}}$, indicating that the superiority of the proposed offloading model becomes more prominent with the decrease of $W_{\tilde{k}}$.

Fig. 19 shows the selection proportion of MDs on the offloading decisions under FL-SMADC and the LWSA under FL-SMADC and FL-SMADC-CO versus the bandwidth of the MBS W_{K+1}. We observe that when the value of W_{K+1} is small (i.e., $W_{K+1} < 90\,\text{MHz}$ in Fig. 19), for MDSs, the selection proportion of offloading to MBS directly increases with W_{K+1}, while the selection proportion of offloading to MBS by relaying decreases with W_{K+1}. This is due to the reason that, the transmission delay is positively related to the AoI and the transmission delay by one hop is smaller than that by two hop. We also observe that when the value of W_{K+1} is small (i.e., $W_{K+1} < 90\,\text{MHz}$ in Fig. 19), for MDSs, the selection proportion of offloading to SBSs keeps nearly unchanged with W_{K+1}. The reason is that, although the bandwidth of SBSs is released by the MDSs that originally offload tasks to MBS by the relaying of SBSs, the available computation resources of SBSs do not increase. When the value of W_{K+1} is large (i.e., $W_{K+1} \geq 90\,\text{MHz}$ in Fig. 19), all the selection proportions keep nearly unchanged with W_{K+1}, which is the

same as the observation in Fig. 15. The reason is that, the key factor affecting the offloading decisions is no longer the bandwidth resource but the computation resource. This is also the reason why the LWSA under FL-SMADC and FL-SMADC-CO first decreases and then keeps nearly unchanged with W_{K+1}. By observing the curves about the LWSA, we find that the gap between FL-SMADC and FL-SMADC-CO increases with W_{K+1}, indicating that the superiority of the proposed offloading model becomes more prominent with the increase of W_{K+1}.

Fig. 20 shows the selection proportion of MDSs on the offloading decisions under FL-SMADC and the LWSA under FL-SMADC and FL-SMADC-CO versus $(F_{\tilde{k}}, F_{K+1})$ when $e_{\max} = 100\,\text{mJ}$, where $F_{\tilde{k}}$ and F_{K+1} are computation capacities of SBSs and the MBS, respectively. We observe that the selection proportion of offloading to SBSs/MBS increases with the computation capacity $F_{\tilde{k}}/F_{K+1}$. The reason is straightforward: with the aim of minimizing the LWSA, MDSs tend to offload tasks to the BSs with more computation resources to allocate. Besides, we observe from the curves about the LWSA that, the gap between FL-SMADC and FL-SMADC-CO decreases with $F_{\tilde{k}}$ while increases with F_{K+1}. This observation indicates that the superiority of the proposed offloading model becomes more prominent with the decrease of $F_{\tilde{k}}$ and the increase of F_{K+1}.

VI. Conclusion

This paper studied the LWSA minimization problem with the energy, delay and peak AoI constraints in a dynamic HMECN by jointly optimizing the offloading decisions of MDs as well as the bandwidth and computation resource allocation of BSs. To efficiently address the problem, we proposed a FL-SMADC algorithm that combines the advantages of multi-agent DRL, convex optimization and FL, by decomposing it into the ODO top-problem and the RAO sub-problem. Simulation results showed that the proposed FL-SMADC algorithm performs better than other comparative algorithms in terms of the LWSA, convergence, scalability and robustness in dynamic environments.

References

[1] W. Zhou, L. Fan, F. Zhou, F. Li, X. Lei, W. Xu, and N. Arumugam, "Priority-aware resource scheduling for UAV-mounted mobile edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 7, pp. 9682-9687, Jul. 2023.

[2] J. Li, B. Gu, Z. Qin, and Y. Han, "Graph tasks offloading and resource allocation in multi-access edge computing: A DRL-and-optimization-aided approach," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 6, pp. 3707-3718, Nov.-Dec. 2023.

[3] J. Zhu and J. Gong, "Optimizing peak age of information in MEC systems: Computing preemption and non-preemption,"*IEEE/ACM Trans. Netw.*, vol. 32, no. 4, pp. 3285-3300, Aug. 2024.

[4] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177-4190, Jun. 2018.

[5] Y. Chen, Z. Liu, Y. Zhang, Y. Wu, X. Chen, and L. Zhao, "Deep reinforcement learning based dynamic resource management for mobile edge computing in industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4925-4934, Jul. 2021.

[6] L. Tan, Z. Kuang, L. Zhao, and A. Liu, "Energy-efficient joint task offloading and resource allocation in OFDMA-based collaborative edge computing," *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1960-1972, Mar. 2022.

[7] G. Chen, Q. Wu, R. Liu, J. Wu, and C. Fang, "IRS aided MEC systems with binary offloading: A unified framework for dynamic IRS beamforming," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 2, pp. 349-365, Feb. 2023.

[8] Y. Li, H. Ma, L. Wang, S. Mao, and G. Wang, "Optimized content caching and user association for edge computing in densely deployed heterogeneous networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 2130-2142, Jun. 2022.

[9] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Edge intelligence for energy-efficient computation offloading and resource allocation in 5G beyond," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12175-12186, Oct. 2020.

[10] C. Xu, G. Zheng, and X. Zhao, "Energy-minimization task offloading and resource allocation for mobile edge computing in NOMA heterogeneous networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16001-16016, Dec. 2020.

[11] Y. Liu, Y. Mao, Z. Liu, and Y. Yang, "Deep learning-assisted online task offloading for latency minimization in heterogeneous mobile edge," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 4062-4075, May 2024.

[12] M. Senthil Kumar, A. Dadlani, and H. Tabassum, "Age of information in unreliable tandem queues,"*IEEE Commun. Lett.*, vol. 29, no. 10, pp. 2308-2312, Oct. 2025.

[13] J. Li, S. Guo, W. Liang, J. Wu, Q. Chen, Z. Xu, W. Xu, and J. Wang, "AoI-aware, digital twin-empowered IoT query services in mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 32, no. 4, pp. 3636-3650, Aug. 2024.

[14] K. Peng, P. Xiao, S. Wang, and V. C. M. Leung, "AoI-aware partial computation offloading in IIoT with edge computing: A deep reinforcement learning based approach," *IEEE Trans. Cloud Comput*, vol. 11, no. 4, pp. 3766-3777, Oct.-Dec. 2023.

[15] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 2731-2735.

[16] H. Lv, Z. Zheng, F. Wu, and G. Chen, "Strategy-proof online mechanisms for weighted AoI minimization in edge computing," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1277-1292, May 2021.

[17] F. Song, Q. Yang, M. Deng, H. Xing, Y. Liu, X. Yu, K. Li, and L. Xu, "AoI and energy tradeoff for aerial-ground collaborative MEC: A multi-objective learning approach," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 11278-11294, Dec. 2024.

[18] S. Goudarzi, S. Ahmad Soleymani, M. Hossein Anisi, A. Jindal, and P. Xiao, "Optimizing UAV-assisted vehicular edge computing with age of information: An SAC-based solution," *IEEE Internet Things J.*, vol. 12, no. 5, pp. 4555-4569, Mar. 2025.

[19] J. Wei, C. Shi, Y. Zhu, Y. Hu, and A. Schmeink, "Age-of-Information minimization in multi-server edge computing networks for URLLC services," *IEEE Trans. Veh. Technol.*, vol. 74, no. 9, pp. 14317-14330, Sep. 2025.

[20] S. Shen, H. Yang, K. Yang, K. Wang, and G. Zhang, "AoI-aware joint resource allocation in multi-UAV aided multi-access edge computing systems," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 3, pp. 2596-2609, May-Jun. 2024.

[21] Z. Zhu, S. Wan, P. Fan, and K. B. Letaief, "Federated multiagent actor-critic learning for age sensitive mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1053-1067, Jan. 2022.

[22] Z. Qin, Z. Wei, Y. Qu, F. Zhou, H. Wang, D. W. K. Ng, and C. -B. Chae, "AoI-aware scheduling for air-ground collaborative mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 22, no. 5, pp. 2989-3005, May 2023.

[23] X. Chen, C. Wu, T. Chen, Z. Liu, H. Zhang, M. Bennis, H. Liu, and Y. Ji, "Information freshness-aware task offloading in air-ground integrated edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 243-258, Jan. 2022.

[24] H. Quan, Q. Zhang, and J. Zhao, "Federated learning assisted intelligent IoV mobile edge computing," *IEEE Trans. Green Commun. Netw.*, vol. 9, no. 1, pp. 228-241, Mar. 2025.

[25] J. Chi, X. Zhou, F. Xiao, Y. Lim, and T. Qiu, "Task offloading via prioritized experience-based double dueling DQN in edge-assisted IIoT," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 14575-14591, Dec. 2024.

[26] C. Wang, T. Yao, T. Fan, S. Peng, C. Xu, and S. Yu, "Modeling on resource allocation for age-sensitive mobile-edge computing using federated multiagent reinforcement learning," *IEEE Internet Things J.*, vol. 11, no. 2, pp. 3121-3131, Jan. 2024.

[27] A. A. Ayeni, A. A. Yusuf, and S. O. Onidare, "Correlation between the electromagnetic properties of building materials and wireless signal penetration loss," *IEEE Trans. Antennas Propag.*, vol. 70, no. 12, pp. 12040-12048, Dec. 2022.

[28] Z. Ji, S. Wu, and C. Jiang, "Cooperative multi-agent deep reinforcement learning for computation offloading in digital twin satellite edge networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3414-3429, Nov. 2023.

[29] "IEEE standard for floating-point arithmetic," *IEEE Std 754-2019* (Revision of IEEE 754-2008), pp. 1-84, Feb. 2019.

[30] V. Freschi and E. Lattanzi, "A study on the impact of packet length on communication in low power wireless sensor networks under interference," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3820-3830, Apr. 2019.

[31] H. Saito, "Theoretical analysis of nonlinear energy harvesting from wireless mobile nodes," *IEEE Wireless Commun. Lett.*, vol. 10, no. 9, pp. 1914-1918, Sep. 2021.

[32] X. Zhou, L. Huang, T. Ye, and W. Sun, "Computation bits maximization in UAV-assisted MEC networks with fairness constraint," *IEEE Internet Things J.*, vol. 9, no. 21, pp. 20997-21009, Nov. 2022.

[33] J. Tan, Y.-C. Liang, L. Zhang, and G. Feng, "Deep reinforcement learning for joint channel selection and power control in D2D networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1363-1378, Feb. 2021.

[34] L. Cheng, G. Feng, Y. Sun, S. Qin, F. Wang, and T. Q. S. Quek, "Energy-constrained satellite edge computing for satellite-terrestrial integrated networks," *IEEE Trans. Veh. Technol.*, vol. 74, no. 2, pp. 3359-3374, Feb. 2025.

[35] S. S. Hassan, Y. M. Park, Y. K. Tun, W. Saad, Z. Han, and C. S. Hong, "Satellite-based ITS data offloading & computation in 6G networks: A cooperative multi-agent proximal policy optimization DRL with attention approach," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 4956-4974, May 2024.

[36] C. Liu, M. Xu, Y. Yang, and N. Geng, "DRL-OR: Deep reinforcement learning based online routing for multi-type service requirements," in *Proc. IEEE INFOCOM*, Vancouver, BC, Canada, May 2021, pp. 1-13.

[37] Z. Li, G. Li, M. Bilal, D. Liu, T. Huang, and X. Xu, "Blockchain-assisted server placement with elitist preserved genetic algorithm in edge computing," *IEEE Internet Things J.*, vol. 10, no. 24, pp. 21401-21409, Dec. 2023.

[38] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, and S. Russell, "Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient,"*AAAI*, vol. 33, no. 01, pp. 4213-4220, Jul. 2019.

[39] J. Lan, X. Jia, and Z. Kou, "Age of information and energy-efficiency optimization in RIS-assisted vehicular edge computing via hierarchical deep reinforcement learning,"*IEEE Internet Things J.*, vol. 12, no. 20, pp. 43681-43695, Oct. 2025.

[40] N. Qu, C. Wang, Z. Li, F. Liu, and Y. Ji, "A distributed multi-agent deep reinforcement learning-aided transmission design for dynamic vehicular communication networks," *IEEE Trans. Veh. Technol.*, vol. 73, no. 3, pp. 3850-3862, Mar. 2024.

[41] F. A. Potra and S. J. Wright, "Interior-point methods," *J. Comput. Appl. Math.*, vol. 124, no. 1-2, pp. 281-302, Dec. 2000.

[42] J. Xu, K. Ota, and M. Dong, "Energy efficient hybrid edge caching scheme for tactile internet in 5G," *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 2, pp. 483-493, Jun. 2019.

[43] Z. Wang, Y. Wei, F. R. Yu, and Z. Han, "Utility optimization for resource allocation in multi-access edge network slicing: A twin-actor deep deterministic policy gradient approach," *IEEE Trans. Wireless Commun.*, vol. 21, no. 8, pp. 5842-5856, Aug. 2022.

Xiaoying Liu (Senior Member, IEEE) received the B.E. degree in electronic engineering from the Nanjing University of Science and Technology, Nanjing, China, in 2013 and the Ph.D. degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2018. She is currently an Associate Professor with the School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China. Her research interests include cognitive radio, energy-efficient wireless communications, mobile computing, and the Internet of Things. Dr. Liu was the recipient (as the corresponding author) of the Best Paper Award at the International Conference on Networking and Network Applications in 2021.

Junhao Zheng received the B.E. degree in computer science and technology from Zhejiang University of Technology, Hangzhou, China, in 2021. He is currently pursuing the M.E. degree in computer technology from Zhejiang University of Technology. His current research interests include edge computing and AoI.

Kechen Zheng (Senior Member, IEEE) received the B.E. and Ph.D. degrees in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2013 and 2018, respectively. He is currently an Associate Professor with the School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou, China. He has published more than 50 technical papers in journals and conferences, including IEEE Transactions on Mobile Computing, IEEE Transactions on Wireless Communications, and IEEE Transactions on Communications. His research interests include symbiotic radio, energy harvesting, wireless communication, and edge computing. Dr. Zheng was the recipient of the Best Paper Award at the International Conference on Networking and Network Applications in 2021.

Jia Liu (Senior Member, IEEE) received the B.E. degree from the School of Telecommunications Engineering, Xidian University, Xi'an, China, in 2010, and the Ph.D. degree from the School of Systems Information Science, Future University Hakodate, Japan, in 2016. His research interests include wireless systems security, space-air-ground integrated networks, the Internet of Things, and 6G. He received the 2016 and 2020 IEEE Sapporo Section Encouragement Award.

Tarik Taleb (Senior Member, IEEE) received the B.E. degree (with distinction) in information engineering, and the M.Sc. and Ph.D. degrees in information sciences from Tohoku University, Sendai, Japan, in 2001, 2003, and 2005, respectively. He is currently a full professor with Ruhr University Bochum, Germany. He was a professor with the Center of Wireless Communications, University of Oulu, Oulu, Finland. He is the founder of ICTFICIAL Oy, and the founder and the director with the MOSA!C Lab, Espoo, Finland. From October 2014 to December 2021, he was an associate professor with the School of Electrical Engineering, Aalto University, Espoo, Finland. Prior to that, he was working as a senior researcher and a 3GPP standards expert with NEC Europe Ltd., Heidelberg, Germany. Before joining NEC and till March 2009, he worked as assistant professor with the Graduate School of Information Sciences, Tohoku University, in a lab fully funded by KDDI. From 2005 to 2006, he was a research fellow with the Intelligent Cosmos Research Institute, Sendai. He has been directly engaged in the development and standardization of the Evolved Packet System as a member of the 3GPP System Architecture Working Group. His current research interests include AI-based network management, architectural enhancements to mobile core networks, network softwarization and slicing, mobile cloud networking, network function virtualization, software-defined networking, software-defined security, and mobile multimedia streaming.

Norio Shiratori (Life Fellow, IEEE) received the Ph.D. degree from Tohoku University in 1977. Since 2017, he has been a Professor with the Research and Development Initiative, Chuo University. He has published over 15 books and over 600 refereed articles in computer science and related fields. He is a fellow of Japan Foundation of Engineering Societies (JFES), the Information Processing Society of Japan (IPSJ), and the Institute of Electronics, Information and Communication Engineers (IEICE). Recognizing his significant contributions, he was honored with the title of IEEE Life Fellow in 2017. He was a recipient of the Minister of MEXT Award from Japanese Government in 2016; the Science and Technology Award from the Ministry of Education, Culture, Sports, Science, and Technology (MEXT), in 2009; the IEICE Achievement Award in 2001; the IEICE Contribution Award in 2011; the IPSJ Contribution Award in 2008; and the IEICE Honorary Member in 2012.