Joint UAV Placement Optimization, Resource Allocation, and Computation Offloading for THz Band: A DRL Approach

Heng Wang, Haijun Zhang, *Senior Member, IEEE*, Xiangnan Liu, Keping Long, *Senior Member, IEEE*, and Arumugam Nallanathan, *Fellow, IEEE*

Abstract

With the development of internet of things (IoT), latency-sensitive applications such as telemedicine are constantly emerging. Unfortunately, due to the limited computation capacity of wireless user devices (WUDs), the real-time demands can not be met. Multi-access edge computing (MEC), which enables the deployment of edge access points (E-APs) to support computation-intensive applications, has become an effective way to meet the real-time demands. However, the number of WUDs that E-APs can serve are limited. To increase system capacity, the unmanned aerial vehicle (UAV) assisted computation offloading architecture in the terahertz (THz) band is proposed. In this paper, the problem of UAV placement optimization, resource allocation, and computation offloading is investigated considering the quality of service and resource constraints. The joint optimization problem is non-convex and hard to be solved in time by using traditional algorithms, such as successive convex approximation. Therefore, deep reinforcement learning (DRL) based approach is a promising way to solve the formulated non-convex problem of minimizing latency. Double deep Q-learning (DDQN) and deep deterministic policy gradient (DDPG) algorithms are provided to search for near-optimal solutions in highly dynamic environments. The effectiveness of the proposed algorithms is proved by simulation results in different scenarios.

H. Wang, H. Zhang, X. Liu, and K. Long are with Institute of Artificial Intelligence, Beijing Advanced Innovation Center for Materials Genome Engineering, Beijing Engineering and Technology Research Center for Convergence Networks and Ubiquitous Services, University of Science and Technology Beijing, Beijing 100083, China (email: wangheng1220@xs.ustb.edu.cn, haijunzhang@ieee.org, xiangnan.liu@xs.ustb.edu.cn, longkeping@ustb.edu.cn).

A. Nallanathan is with the School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, U.K. (e-mail: a.nallanathan@qmul.ac.uk).

Index Terms

MEC, resource allocation, UAV, THz frequency band, DRL

I. Introduction

With the advancement of internet of things (IoT), numerous computation-sensitive applications such as telemedicine, autonomous driving, virtual reality (VR), and augmented reality (AR), are gradually ingrained into our daily lives, which brings exponentially increasing traffic load and stringent latency requirements [1]. The number of wireless user devices (WUDs) will rise to 75 billion by 2025 [2]. Unfortunately, most of WUDs have limited computing resources and battery capacity. Computation-intensive tasks can not be completed independently by WUDs, which makes it difficult to support the implementation of the above computation-sensitive applications [3].

To solve the problem of insufficient computing resources and battery capacity, cloud computing technology is employed to transfer computation-intensive tasks to central cloud server for processing [4]. However, with the development of IoT, the number of computation-intensive tasks is increasing exponentially. Transferring the large amount of data generated by WUDs to central cloud server will result in significant network resource consumption [5]. As a result, traditional cloud computing is no longer sufficient to match the instantaneous computing of such huge scale data volume. To make up for the shortage of cloud computing, multi-access edge computing (MEC) is proposed. MEC transfers functions from the core network to the edge of mobile network. Deploying edge access points (E-APs) with communication and computing capabilities on the WUDs side, it not only decreases network operations and delays, but also promotes the quality of service (QoS). Additionally, the substantial growth of computation-intensive tasks impose a huge link load on the backhaul link and core network [6]. With the deployment of E-APs, the distance between WUDs and servers is greatly reduced, decreasing the bandwidth requirements on the backhaul link [7].

Traditional E-APs are deployed at fixed locations. The coverage of E-APs and the number of WUDs that E-APs can serve are limited. With the rapid breakthrough and improvement of UAV technology, equipping edge servers to UAV has become a promising way to increase system capacity. When the number of WUDs to be served exceeds the E-APs capacity limit or WUDs

exceed the coverage area, UAVs can be equipped with servers to provide services for WUDs. Compared with traditional architectures, UAV-assisted architecture is more efficiently due to the scalability and flexibility [8].

To better support computation-intensive applications, it is necessary to reduce the transmission delay from WUDs to UAV servers. Advanced modulation schemes and signal processing techniques can improve transmission rate [9]. However, due to the spectrum limitation, it is still difficult to significantly increase the transmission rate. The other solution is to employ the higher carrier frequency such as terahertz (THz) to increase channel bandwidth and provide sufficient transmission capacity. The THz band is located between infrared and microwave, and the frequency range is 0.1 THz-10 THz. THz communication can obtain tens of Gb/s wireless transmission rate, which is significantly better than the current ultra-wideband technology [10]. As a result, THz communication technology has attracted much attention and become a key wireless technology to meet the demand for real-time traffic in mobile heterogeneous network systems [11]. Due to the sensitivity to channel congestion of THz band, deploying servers on UAV can effectively reduce the impact of obstacles on the communication link. Therefore, the UAV-assisted computation offloading architecture in the THz band is promising.

The latency can be effectively reduced by the joint optimization of the UAV placement, resource allocation, and computation offloading. However, the channel state and resource requirements of each WUD are highly time-varying. It is extremely difficult to search for a near-optimal solution in time [12]. Existing conventional optimization algorithms such as successive convex approximation (SCA) and particle swarm optimization (PSO) are feasible for solving the joint optimization problem. However, SCA algorithm obtain local solution by iteratively solving a series of convex optimization problems similar to the original problem, and PSO algorithm evaluate the quality of the solution by fitness and iteratively find the optimal solution from a random solution. Each of these algorithms requires many iterations, which are not suitable for systems with high real-time requirements.

With the continuous improvement of artificial intelligence (AI), reinforcement learning (RL) and deep learning (DL) will play an essential role in wireless communications [13][14]. The traditional RL algorithm such as Q-learning employs a Q table to approximate the Q value. However, real-world networks have massive WUDs involving a large number of different actions,

which makes it impractical to evaluate every possible combination of actions. To solve the above problem, DL and RL can be combined as deep reinforcement learning (DRL). Deep neural networks (DNNs) are employed to approach the Q value, which are more scalable and flexible [15][16].

In this paper, the key point is the joint optimization of the UAV placement, resource allocation, and computation offloading. Based on the formulated model, value iteration-based RL algorithms, namely DDQN and DDPG, are provided to determine the joint policy of UAV deployment, resource allocation, and computation offloading. The main contributions are as follows.

- To support computation-intensive applications such as autonomous driving, E-APs with computing and communication capability are deployed at the edge of network. The UAV-assisted computation offloading architecture in the THz band is proposed to further increase system capacity and reduce transmission delay. Then, the physical channel model for THz link is provided.
- Based on the proposed model, the UAV placement optimization, resource allocation, and computation offloading are formulated as a joint optimization problem with the goal of minimizing total time delay. The environmental dynamics such as the highly time-varying channel state, resource requirements, and the computation capabilities of different WUDs are taken into account.
- To overcome the inability of traditional algorithms such as SCA and PSO to meet real-time requirements, DRL is applied to search for near-optimal solutions in the highly dynamic environments. Based on the markov decision process (MDP), DDQN and DDPG algorithms are supplied to transform the non-convex issue into the learning issue, which can adopt the highly dynamic environment and gain near-optimal solutions effectively.
- The provided DDQN and DDPG algorithms are compared with local computing and full offloading method under different scenarios. Simulation results verify the proposed DDQN and DDPG algorithms are effective in reducing total time delay in different scenarios.

The remainder of this paper is organized as follows. Section II introduces the related work. In Section III, the system model and problem formulation are described. The DRL approach is proposed in Section IV. Then, the simulation results are provided in Section V. Finally, the paper is concluded in Section VI.

II. Related Work

UAVs aided with servers are capable of providing services for WUDs more efficiently due to the mobility and flexibility. Therefore, the optimization problem of UAV providing services for WUDs has been widely studied. Zhou et al. investigated a UAV-enabled MEC wireless-powered system in [17]. The computation rate maximization problems are studied under both partial and binary computation offloading modes, subject to the energy-harvesting constraint and UAV speed constraint. In [18], the problem of UAV deployment, power allocation, and bandwidth allocation is investigated for the UAV-assisted wireless system operating at THz frequency. The work in [19] investigated a UAV-based computation and relay system where a joint computation offloading, bandwidth, and computing resource management issue was formulated to promote energy efficiency. Then, the block continuous upper-bound minimization algorithm was used to design reasonable strategies. The authors in [20] intended to promote energy efficiency while meeting the QoS requirements. The problem of jointly optimizing uplink and downlink communication bit allocation and UAV computing resource management with the delay and energy constraint was solved by SCA algorithm. In [21], the service request rate was maximized by optimizing the UAV placement, task offloading, and resources management while meeting tight latency as well as dependability demands. The scenario using UAVs and satellites for task offloading was considered in [22]. To reduce the latency between devices, associative control, computation task distribution, user power control, and communication resource management are achieved through block coordinate descent (BCD) and SCA algorithms. Although the above algorithms can effectively improve the system performance, they all require multiple iterations to search for a near-optimal solution. Therefore, the above algorithms are not applicable to the highly dynamic environment.

To obtain near-optimal solutions timely in the highly dynamic environment, some scholars have started to use the DRL algorithm. The authors in [23] took advantage of DRL to propose an algorithm for joint server selection, cooperative offloading, and handover in the multi-access edge wireless network. In [24], the authors proposed a multi-agent deep reinforcement learning based small base station state selection scheme for joint optimization of resource allocation and massive access in the ultra-dense network. In [25], Huang et al. investigated the wireless-powered network based on the binary offloading strategy. To promote the weighted sum transmission rate,

an offloading and resource management algorithm employing DRL was proposed. To address the privacy issue, DRL-based decentralized computation offloading algorithms are proposed without the information about network bandwidth and preference in [26]. The authors in [27] proposed a NOMA-based computation offloading system, and a joint optimization issue about NOMA and computing resource management was formulated. Considering the dynamic channel state, the DRL was used to search for the near-optimal offloading solution. In summary, the DRL algorithm has been widely used in MEC and has a good performance. To the best of our knowledge, the existing literature does not jointly consider the UAV placement optimization, resource allocation, and computation offloading in the THz band under highly time-varying environment. Therefore, the joint optimization problem of minimizing latency is formulated and solved by provided DDQN and DDPG algorithms.

III. System Model and Problem Formulation

A. UAV-assisted Computation Offloading System Model

1) Computation Task Model: The system model is shown in Fig. 1. Owing to the limited computation and battery capacity, WUDs can upload part of tasks to E-APs for processing. However, the coverage of E-APs is restricted, and the number of WUDs that can be served simultaneously is limited. Additionally, malfunction might occur in the E-APs, making it unable to provide services for WUDs. When the above situation occurs, UAV can be equipped with servers to provide services for WUDs. The set of WUDs that need to be served by UAV server is denoted by $\mathcal{N} = \{1, 2, 3.....n\}$.

Without loss of generality, the tasks for the ith WUD can be denoted as $\zeta_i \triangleq \{d_i, c_i, o_i, t_{i,max}\}$. Let d_i and c_i denote the size of the ith computation task and the number of CPU cycles required to process the ith task, respectively. d_i and c_i can be converted to each other, depending on the type of task. o_i denotes the size of the ith computation result, which is usually much smaller than d_i. $t_{i,max}$ represents the tolerable delay of the ith WUD, which means that the total time delay T_i can not be greater than $t_{i,max}$.

2) Local computing model: When WUDs chooses to execute the task locally, the task completion process is independent of the UAV server. Define α_i as the percentage of the ith WUDs' tasks uploaded to the server. $\mathcal{A} = \{\alpha_1, \alpha_2, \alpha_3, ..., \alpha_n\}$ denotes the computation offloading

TABLE I

NOTATION DEFINITION

Parameter	Definition
d_i	The size of the ith WUD computation task
c_i	The number of CPU cycles required to process the ith WUD task
o_i	The size of the ith WUD computation result
$t_{i,max}$	The tolerable delay of the ith WUD
t_i^l	The local computation latency of the ith WUD
f_i	The computation capacity of the ith WUD
E_i^l	The local computing energy consumption of the ith WUD
r_i	The uplink transmission rate of the ith WUD
B	The bandwidth of the wireless channel
G_t	The antenna gain of the transmitter
G_r	The antenna gain of the receiver
N_0	The variance of Gaussian white noise
t_i^{up}	The transmission delay for the ith WUDs to transfer task to the UAV server
E_i^{up}	The transmission energy consumption for the ith WUD to transfer task to the UAV server
t_i^o	The server processing delay of the ith WUD task
f_{uav}	The computation capacity of the UAV server
$E_{i,w}^o$	The energy consumption of the ith WUD in standby mode
r'	The downlink transmission data rate
T_i	The total time delay of the ith WUD
E_i	The total energy consumption of the ith WUD
k	The computation energy efficiency coefficient
α_i	The percentage of the ith WUDs' tasks uploaded to the server
β_i	The percentage of the computing resource allocated to the ith WUD

vector. The local computation latency t_i^l is given by

$$t_i^l = \frac{(1-\alpha_i)c_i}{f_i}, \tag{1}$$

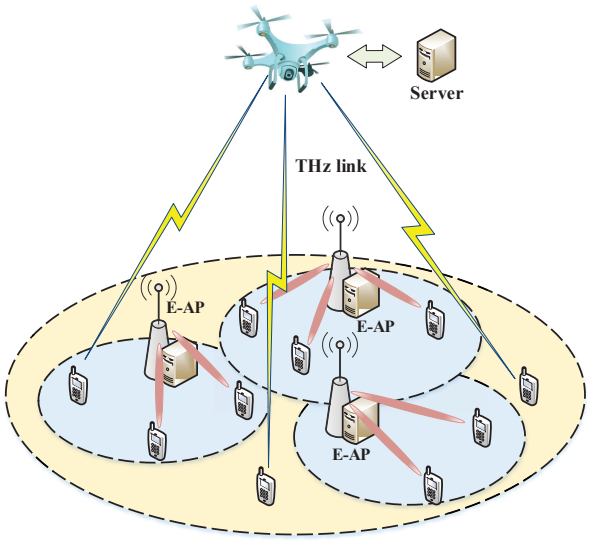

Fig. 1. UAV-assisted computation offloading system model

where f_i denotes the computation capacity of the ith WUD. Each WUD has different computation capacity.

Furthermore, the corresponding local computation energy consumption of the ith WUD can be expressed as

$$E_i^l = k(f_i)^2(1-\alpha_i)c_i, \tag{2}$$

where k denotes the computation energy efficiency coefficient [25].

3) Edge computing model: WUDs can upload tasks to UAV servers for processing to reduce computing stress. The uplink transmission rate of the ith WUD is given by

$$r_i = \frac{B}{N}\log_2(1+\frac{G_rG_tp_i|h_i|^2}{N_0B/N}), \tag{3}$$

where B denotes the bandwidth, N denotes the total number of WUDs. G_t and G_r are antenna gains of transmitter and receiver, respectively. p_i represents the transmission power of the ith WUD. h_i denotes the wireless channel gain of the ith WUD, and N_0 is variance of Gaussian white noise.

The transmission delay for the ith WUDs to transfer task to the UAV server is given by

$$t_i^{up} = \frac{d_i}{\frac{B}{N}\log_2(1+\frac{G_rG_tp_i|h_i|^2}{N_0B/N})}. \tag{4}$$

The transmission energy consumption for the ith WUD to transfer task to the UAV server is given by

$$E_i^{up} = \frac{p_i d_i}{\frac{B}{N}\log_2(1+\frac{G_r G_t p_i |h_i|^2}{N_0 B/N})}. \tag{5}$$

The UAV server processing delay of the ith WUD task is given by

$$t_i^o = \frac{\alpha_i c_i}{\beta_i f_{uav}}, \tag{6}$$

where f_{uav} denotes the computation capacity of the UAV server, β_i denotes the percentage of the computing resource allocated to the ith WUD, and $\mathcal{B} = \{\beta_1, \beta_2, \beta_3, ..., \beta_n\}$ denotes the resource allocation vector.

In addition, WUD is in standby mode while the UAV server processes the uploaded tasks. Thus, the energy consumption of the ith WUD can be expressed as

$$E_{i,w}^o = P_{i,w} t_i^o = \frac{P_{i,w}\alpha_i c_i}{\beta_i f_{uav}}. \tag{7}$$

The UAV server sends the data to WUD after the task has been processed. The transmission delay is expressed as

$$t_i^{down} = \frac{o_i}{r'}, \tag{8}$$

where r' denotes the transmission rate of UAV server to WUDs. According to [28], o_i is much smaller than d_i. Therefore, the transmission delay of the UAV server sending the calculation results to WUD is always negligible.

Accordingly, the total time delay of the ith WUD is given by

$$T_i = \max\left\{\left(t_i^{up} + t_i^o\right), t_i^l\right\}. \tag{9}$$

The total energy consumption of the ith WUD can be expressed as:

$$E_i = E_i^l + E_i^{up} + E_{i,w}^o = k(f_i)^2(1-\alpha_i)c_i + \frac{p_i d_i}{r_i} + \frac{P_{i,w}\alpha_i c_i}{\beta_i f_{uav}}. \tag{10}$$

4) THz channel model: With the electromagnetic wave frequency increases, the THz band has disparate channel characteristics from the low frequency channel [9]. The THz channel model consists of LoS propagation, reflection, scattering, and diffraction path. Due to the lower power, scattering and diffraction path can be negligible. Furthermore, the appropriate antenna gain can

avoid the influence of reflection [29][30]. Therefore, the LoS propagation path is only took into consideration in the THz band.

The coordinates of the UAV and the ith WUD can be expressed as (x_{uav}, y_{uav}) and (x_i, y_i), respectively. Thus, the distance between UAV and the ith WUD is given by

$$D_i = \sqrt{(x_{uav} - x_i)^2 + (y_{uav} - y_i)^2 + H^2}, \tag{11}$$

where H denotes the flight height of UAV.

The wireless channel gain between UAV server and WUDs is given by

$$h_i(f, D) = \sqrt{\frac{1}{PL(f, D)}}, \tag{12}$$

where $PL(f, D)$ denotes the pathloss. Within the frequency range 0.1 THz to 1 THz, the LoS propagation path is given by the combination of the molecular absorption loss $L_{abs}(f, D)$ and the free-space expansion loss $L_{spread}(f, D)$ [32]. Therefore, the LoS propagation path can be expressed by

$$\begin{aligned} PL\left(f, D\right) &= L_{abs}\left(f, D\right) \times L_{spread}\left(f, D\right) \\ &= \left(\frac{4\pi f D}{c}\right)^2 e^{k_{abs}(f)d}, \end{aligned} \tag{13}$$

or in dB:

$$\begin{aligned} PL\left(f, D\right)[dB] &= L_{abs}\left(f, D\right)[dB] + L_{spread}\left(f, D\right)[dB] \\ &= 20\log_{10}\left(\frac{4\pi f D}{c}\right) + 10k_{abs}\left(f\right) d\log_{10}e, \end{aligned} \tag{14}$$

where $k_{abs}(f)$ is denoted as the absorption coefficient and c is denoted as the speed of light.

B. Problem formulation

In this paper, the target is to minimize the total time delay of all WUDs while satisfying the resource and energy constraint. The computation offloading vector $\mathcal{A} = \{\alpha_1, \alpha_2, \alpha_3, ..., \alpha_N\}$, resource allocation vector $\mathcal{B} = \{\beta_1, \beta_2, \beta_3, ..., \beta_N\}$, and the coordinates of UAV (x_{uav}, y_{uav})

are the variables to be optimized. The corresponding optimization issue can be expressed as

$$\begin{aligned}
&\min_{\mathcal{A},\mathcal{B},x_{uav},y_{uav}} \sum_{n=1}^{N} T_n, \\
s.t.\ &C1: T_i \leqslant t_{i,max}, \forall i \in \mathcal{N}, \\
&C2: x_{\min}^{uav} \leqslant x_{uav} \leqslant x_{\max}^{uav}, y_{\min}^{uav} \leqslant y_{uav} \leqslant y_{\max}^{uav}, \\
&C3: 0 \leqslant \beta_i \leqslant 1, \forall i \in \mathcal{N}, \\
&C4: \sum_{i=1}^{N} \beta_i \leqslant 1, \\
&C5: 0 \leqslant \alpha_i \leqslant 1, \forall i \in \mathcal{N}, \\
&C6: E_i^l + E_i^{up} + E_{i,w}^o \leqslant E_{i,\max}, \forall i \in \mathcal{N}.
\end{aligned} \tag{15}$$

$C1$ ensures the completion time of each task can not be greater than the tolerable latency. $C2$ ensures that the UAV and each WUD are positioned within the specified limits. $C3$ and $C4$ are constraints on available computing resources, i.e., the sum of the computing resources distributed to each WUD can not exceed the total computing resources. $C5$ indicates that each WUD can upload any percentage of tasks to the UAV server for processing, and the rest is processed by WUD. $C6$ represents that the energy consumption of each WUD can not be greater than its available energy.

IV. DRL APPROACH

In this section, the optimization problem is modeled as MDP. The state space, action space, and reward function are defined. Based on this, DDQN and DDPG algorithms are provided to reduce the total time delay in the highly dynamic environments.

A. DRL framework

In DRL, the definition of state, action, and reward function is critical and will directly affect the optimization performance. Therefore, it is essential to define these elements according to the system model and optimization objectives.

1) State space: To accurately represent the state information of the system, task volume information and placement information of each WUD need to be added to the state space. Thus, the state space can be expressed as

$$F = [x_1, y_1, ..., x_n, y_n; d_1, ..., d_n; c_1, ..., c_n]. \tag{16}$$

2) Action space: The objective is to minimize the total time delay by determining the policy of UAV deployment, computation offloading, and resource allocation. Much literature considers binary offloading, i.e., tasks are either all processed locally or all uploaded to the server. Although the problem is simplified, binary offloading approach is not optimal. In this paper, the scope of action space is expanded. WUDs can upload any percentage of tasks to the server and be allocated any percentage of computing resources. The action space is defined as

$$G = [\alpha_1, \alpha_2, ..., \alpha_n; \beta_1, \beta_2, ..., \beta_n; x_{uav}, y_{uav}]. \tag{17}$$

3) Reward function: $R(F, G)$ is a function depending on the state F and action G. The reward function is used to measure the advantage and disadvantage of taking action G in state F. In addition, the essence of the DRL algorithm is to take a reasonable action to maximize the reward in any state F. Thus, the reward function is given by

$$R(F, G) = -\sum_{n=1}^{N} T_n. \tag{18}$$

Any action of the agent need to satisfy the constraints in (15). If the restriction is violated, a penalty value is assigned to the reward function. Based on the definition, the DDQN and DDPG algorithms will be depicted detailedly in the next section.

B. Markov Decision Process

Define the history of state as $H_t = \{F_1, F_2, F_3, ..., F_t\}$. State is Markov chain if $p(F_{t+1}|F_t) = p(F_{t+1}|H_t)$. Dynamic programming problems are usually modeled as MDP. $\pi(G|F)$ represents the policy probability.

Reward is employed to evaluate the advantage and disadvantage of performing action G_t in state F_t to reach the next state F_{t+1}. The reward of current moment and the reward of future

moment are different. Therefore, the discount factor ψ needs to be introduced. The sum of rewards can be expressed as

$$R_t^{sum} = R_{t+1} + \psi R_{t+2} + ... + \psi^{T-t-1} R_T, \tag{19}$$

where $\psi = 0$ means that only care about the immediate reward and $\psi = 1$ means future reward is equal to the immediate reward.

Different actions can be taken in state F_t. The state-value function $V^\pi(F)$ is the expected return in state F.

$$\begin{aligned} V^\pi(F) &= \mathbb{E}\left[R_t^{sum} | F_t = F\right] \\ &= \mathbb{E}\left[R_{t+1} + \psi R_{t+2} + ... + \psi^{T-t-1} R_T | F_t = F\right]. \end{aligned} \tag{20}$$

The action-value function $Q(F, G)$ is the expected return taking action G in state F.

$$\begin{aligned} Q(F, G) &= \mathbb{E}\left[R_t^{sum} | F_t = F, G_t = G\right] \\ &= \mathbb{E}\left[R_{t+1} + ... + \psi^{T-t-1} R_T | F_t = F, G_t = G\right]. \end{aligned} \tag{21}$$

According to (20) and (21), $V^\pi(F)$ and $Q(F, G)$ can be converted to each other. The relationship between $V^\pi(F)$ and $Q(F, G)$ is expressed as

$$V^\pi(F) = \sum_G \pi(G|F) Q(F, G). \tag{22}$$

In the Q-learning algorithm, the Q table is first initialized randomly and thus the action-value function is inaccurate. Therefore, the Q table needs to be updated to guide the agent's action. According to the time difference method, the target value is given by

$$Q^\pi(F, G) = R + \psi \max_{G'} Q(F', G'). \tag{23}$$

The objective is to make the predicted value $Q(F, G)$ gradually approximate the target value $Q^\pi(F, G)$. Thus, the update guideline is expressed as

$$Q(F_t, G_t) \leftarrow Q(F_t, G_t) + \alpha\left(Q^\pi(F_t, G_t) - Q(F_t, G_t)\right), \tag{24}$$

where α is learning rate, the value of the learning rate will directly affect the algorithm performance.

As the number of updates increases, Q table become more and more accurate. However, real-world networks have massive WUDs involving a large number of different actions, which makes

it impractical to evaluate every possible combination of actions. To tackle the above problem, the DDQN algorithm is provided.

C. DDQN algorithm

Inspired by DL, DDQN algorithm uses DNN to approximate action-value function as follows.

$$\hat{Q}\left(F,G,w\right)\approx Q^{\pi}\left(F,G\right), \tag{25}$$

where w denotes the parameters of the DNN and $\hat{Q}\left(F,G,w\right)$ is denoted as the predicted action-value function.

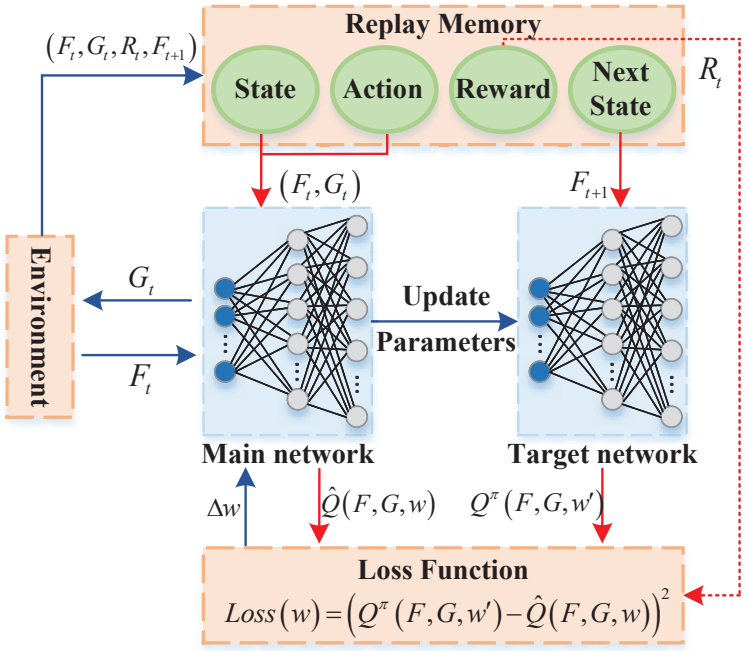

Fig. 2. The training process of DDQN

The DDQN algorithm contains two key techniques, which are experience replay and fixed targets [31]. Since the observed data are ordered, the data is dependent. It would be problematic to use such data to update the parameters. Therefore, experience replay is employed in DDQN, i.e., a memory bank is used to store the experienced data. The correlation between the data is broken by randomly selecting some data from the memory bank to modify the parameters. Each time the neural network is updated, the target is also updated, which can easily lead to

non-convergence of the algorithm. To improve stability of the algorithm, fixed target values are needed. Thus, the DDQN algorithm requires two DNNs, main network and target network. The target network is employed to output the target action-value functions $Q^{\pi}\left(F_t, G_t, w'\right)$, where the parameters are denoted by w'. The target action-value function can be expressed by

$$Q^{\pi}\left(F_t, G_t, w'\right) = R_{t+1} + \psi Q^{\pi}\left(F_{t+1}, G_{t+1}, w'\right). \tag{26}$$

To make the reasonable decision for the agent, the mean-square error (MSE) between the predicted action-value function and the target action-value function needs to be minimized as follows

$$Loss\left(w\right) = \left(\hat{Q}\left(F, G, w\right) - Q^{\pi}\left(F, G, w'\right)\right)^2. \tag{27}$$

Algorithm 1 DDQN Algorithm

Initialize the initial network parameters w and w', the number of episodes N_e, the number of steps N_t, the learning rate α, the exploration rate ε, the discount factor ψ, the update interval C, the capacity of replay memory N, and the batch size v.
for $episode = 1$ to N_e **do**
  Initialize channel state and resource requirements.
  for $step = 1$ to N_t **do**
    Obtain the state s_t.
    Choose action G_t with the random probability δ as
    if $\delta \leqslant \varepsilon$ **then**
      Choose an action G_t stochastically.
    else
      Choose an action $G_t = \arg\max_{G_t} Q\left(F_t, G_t, w\right)$.
    end if
    According to G_t, determine the UAV placement (x_{uav}, y_{uav}), computation offloading vector $\mathcal{A}$, and resource allocation vector $\mathcal{B}$.
    Execute corresponding strategy to gain next state F_{t+1} and reward R_t.
    Save transition (F_t, G_t, R_t, F_{t+1}) in memory bank.
    Sample v batches of transition from memory bank stochastically.
    Determine target action-value function $Q^{\pi}\left(F_t, G_t, w'\right)$ according to (26).
    Minimize the loss function in (27) and update the main network parameters $\mathbf{w}$ according to (28).
    Update the target network parameters w' every C steps.
  end for
end for

Subsequently, the stochastic gradient descent method is used to reduce the MSE in (27) as

follows

$$w \leftarrow w - \alpha \frac{\partial Loss\left(w\right)}{\partial w}. \tag{28}$$

In addition, the DDQN algorithm uses the ε-greedy algorithm to select actions, i.e., actions are selected randomly with probability ε. The ε-greedy algorithm motivates the agent to explore and prevents the strategy from getting trapped in local optimum. The DDQN algorithm is summarized in Algorithm 1 and presented in Fig. 2.

D. DDPG algorithm

The DDQN algorithm outputs an action-value function for each discrete action. Therefore, the algorithm can not be applied to continuous control problems. For this reason, DDPG algorithm is provided to expand the action space from the discrete domain to the continuous domain. The DDPG algorithm contains four DNNs, which are actor network $\mu\left(F,\omega\right)$, critic network $Q\left(F,G,\lambda\right)$, target actor network $\mu'\left(F,\omega'\right)$, and target critic network $Q'\left(F,G,\lambda'\right)$. The function of actor network is to generate action G based on the current state F to obtain the maximum Q value. The critic networks are employed to evaluate the advantage and disadvantage of executing action G in the state F.

The action G_t in state F_t is given by

$$G_t = \mu\left(F_t,\omega\right) + \mathcal{N}_t, \tag{29}$$

where $\mathcal{N}_t$ is exploration noise. Adding the appropriate noise can prevent local optimum.

As with the DDQN algorithm, two target networks are employed to fix target value and improve stability. The target action-value function Q_i^{target} can be expressed as

$$Q_i^{\text{target}} = R_i + \psi Q'\left(F_{i+1}, \mu'\left(F_{i+1},\omega'\right),\lambda'\right). \tag{30}$$

To promote the accuracy of the critic network, the gradient descent method is employed to reduce the MSE between the target action-value function Q_i^{target} and critic network output $Q\left(F,G,\lambda\right)$. The loss function is given by

$$Loss\left(\lambda\right) = \frac{1}{N}\sum_{i=1}^{N}\left(Q_i^{\text{target}} - Q\left(F_i, \mu\left(F_i,\omega'\right),\lambda'\right)\right)^2, \tag{31}$$

where N is the size of sampled batches.

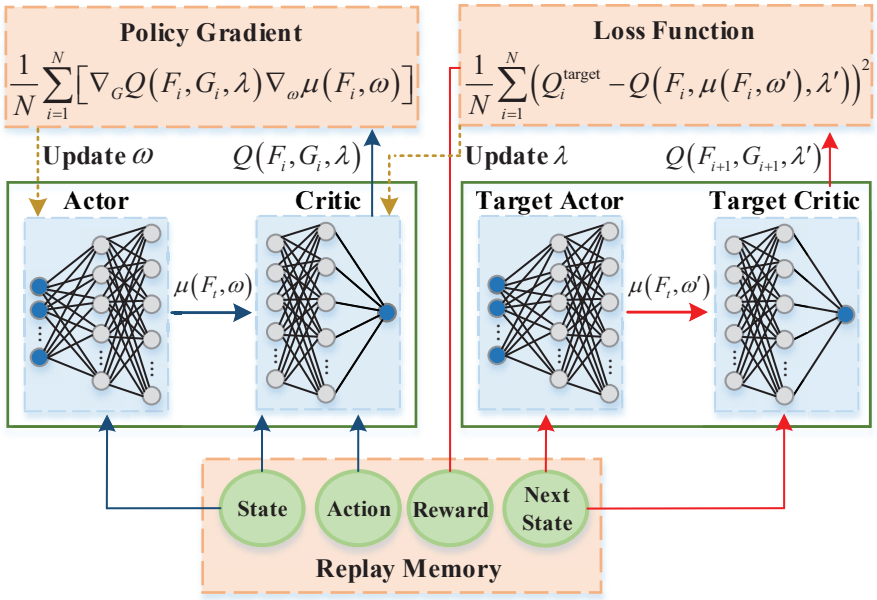

Fig. 3. The training process of DDPG

The core of the actor network is to generate the appropriate action G in any state F, so as to gain maximum state-action value function $Q(F,G)$. Therefore, the loss function of actor network can be expressed by

$$Loss(\omega) = -\frac{1}{N}\sum_{i=1}^{N} Q(F_i, G_i, \lambda). \tag{32}$$

The sampled policy gradient can be obtained by minimizing the loss function in (32).

$$\begin{aligned}\nabla Loss(\omega) &\approx \frac{1}{N}\sum_{i=1}^{N}\left[\nabla_\omega Q(F,G,\lambda)\,|_{F=F_i,G=\mu(F_i|\omega)}\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}\left[\nabla_G Q(F,G,\lambda)\,|_{F=F_i,G=\mu(F_i|\omega)}\nabla_\omega \mu(F,\omega)\,|_{F=F_i}\right]\end{aligned}. \tag{33}$$

After updating the parameters of the actor network and critic network, the parameters of two target networks also need to be updated as follows

$$\begin{aligned}\lambda &\leftarrow \tau\lambda + (1-\tau)\lambda' \\ \omega &\leftarrow \tau\omega + (1-\tau)\omega'\end{aligned} \tag{34}$$

where τ is the soft update coefficient. The DDPG algorithm is summarized in Algorithm 2 and presented in Fig. 3.

Algorithm 2 DDPG algorithm

Initialize the DNNs parameters ω, ω', λ and λ', the number of episodes N_e, the number of steps N_t, the learning rate α, the exploration noise $\mathcal{N}_t$, the discount factor ψ, the soft update coefficient τ, the learning rate α_a and α_c, the capacity of replay memory N, and the batch size v.
Let $\omega' \leftarrow , \lambda' \leftarrow \lambda$.
for episode = 1 to N_e **do**
  Initialize channel state and resource requirements.
  for step = 1 to N_t **do**
    Obtain the state F_t.
    Choose action $G_t = \mu\left(F_t|\omega\right) + \mathcal{N}_t$.
    According to G_t, determine the UAV placement (x_{uav}, y_{uav}), computation offloading vector $\mathcal{A}$, and resource allocation vector $\mathcal{B}$.
    Execute corresponding strategy to gain next state F_{t+1} and reward R_t.
    Save transition (F_t, G_t, R_t, F_{t+1}) in memory bank.
    Sample v batches of transition from memory bank stochastically.
    Determine the target action-value function Q_i^{target} according to (30).
    Minimize the loss function in (31) to update the critic network parameters λ.
    Update the actor network parameters ω employing the gradient in (33).
    Update the target actor network parameters ω' and the critic network parameters λ' according to (34).
  end for
end for

V. Simulation Results and Discussion

In this section, the simulation results of the DDQN algorithm and the DDPG algorithm are presented. To prove the advantage of the proposed algorithms, they are compared with the full local computing method and the full offloading method.

WUDs are stochastically distributed in the $10m \times 10m$ area. The flight area of the UAV is $20m \times 20m$, and the flight altitude is set to $H = 10m$. The transmission power and standby power of the ith WUD are set to p_i = 500mW, p_i^s = 100mW, respectively. The computation energy efficiency coefficient k is set to 10^{-26}. The size of task d_i is uniformly distributed in [300kbits, 500kbits], and c_i is uniformly distributed in [900 Megacycles, 1100 Megacycles]. To reduce the path loss, the carrier frequency is set to 0.55 THz. According to [32], the absorption coefficient k_{abs} is 6.7141×10^{-4}. Both the transmitter antenna gain G_t and the receiver antenna gain G_r are set to 20 dBi, which enables a high degree of directionality in the THz band. In the DDQN algorithm, the learning rate α and the discount factor ψ are set to 0.05 and 0.01, respectively.

The number of episodes N_e is 500 and the number of steps N_t is 20. The exploration rate ε is set to 0.99 and the update interval C is set to 100. The capacity of replay memory is 1000 and the batch size is 64. The parameters of the DDPG algorithm are given by Table II.

TABLE II
PARAMETERS OF DDPG ALGORITHM

Parameter	Explanations	Value
N_e	The number of episodes	500
N_t	The number of steps	20
N	The capacity of the replay memory	1000
v	The batch size	64
ε	The exploration noise	0.1
ψ	The discount factor	0.01
α_a	The learning rate of actor network	0.001
α_c	The learning rate of critic network	0.002
τ	The soft update coefficient	0.01

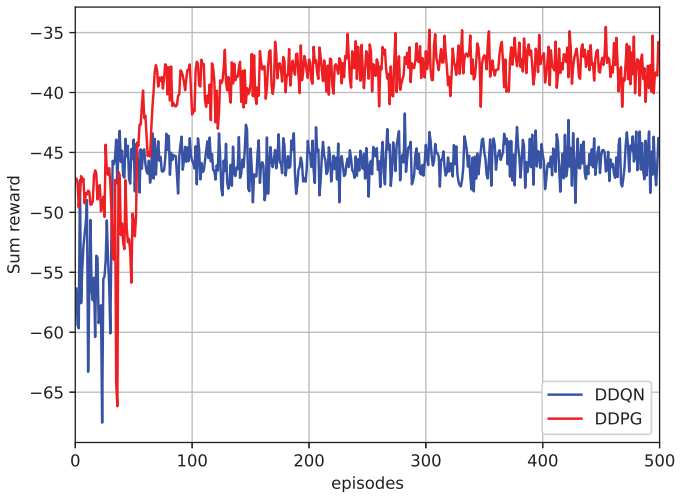

Fig. 4. The convergence of the proposed algorithms

In Fig. 4, the number of WUDs is 4 and the computation capacity of the UAV server is 5 GHz. The horizontal coordinate represents the episodes and the vertical coordinate represents

the sum of the rewards of each episode. As the time of training sessions grows, the sum reward gradually improves, indicating that the agent take action that resulted in higher rewards. The DDQN algorithm converges at about 40 episodes and DDPG algorithm converges at about 80 episodes, which proves the proposed algorithms converge simply. The convergence of sum reward indicates the end of the training process, and the agent can take appropriate action to reduce the total time delay according to the state. The convergence value of the DDQN algorithm is about -46, and the convergence value of the DDPG algorithm is about -38. From the previous definition of reward, it is clear that delay is inversely proportional to reward. Thus, the DDPG algorithm outperforms the DDQN algorithm.

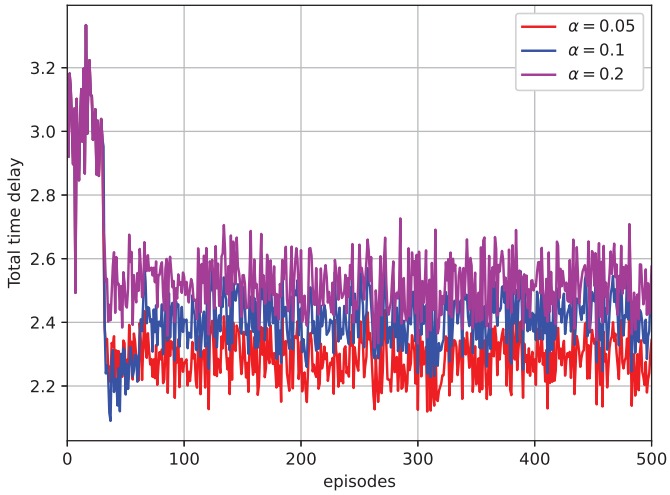

Fig. 5. The convergence of the DDQN algorithm in different learning rate

Fig. 5 and Fig. 6 show the impact of the learning rate on the DDQN and DDPG algorithm. The number of WUDs is 4 and the computation capacity of the UAV server is 5 GHz. The horizontal coordinate represents the episodes and the vertical coordinate represents the total time delay of each step. It is observed that an inappropriate learning rate will cause the algorithm to converge to a larger value. Therefore, choosing the right learning rate is very important for DDQN and DDPG algorithm. which can affect algorithm performance directly.

In Fig. 7, the effectiveness of the DDQN algorithm and DDPG algorithm are compared with local computing method and full offloading method for different numbers of WUDs. The

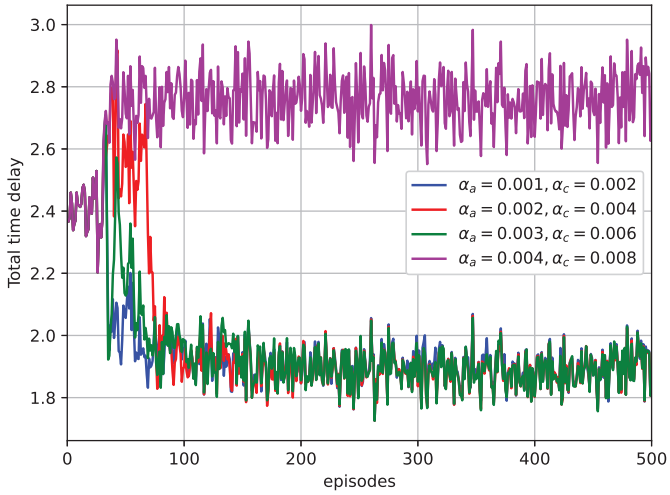

Fig. 6. The convergence of the DDPG algorithm in different learning rate

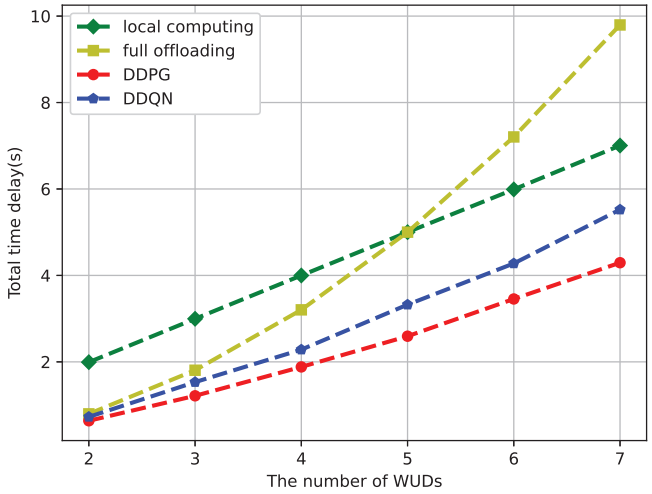

Fig. 7. The total time delay versus the number of WUDs

computation capacity of the UAV server and each WUD are 5 GHz and 1 GHz, respectively. With the increase in the number of WUDs, the total time delay under the four methods also increases. This is because that the computing resources of the UAV server are restricted, the latency of the full offloading method will be higher than the local computing method when the

number of WUDs is greater than 5. The total time delay of the DDQN and DDPG algorithms is less than the full offloading method and local computing method for different numbers of WUDs, which proves the better performance of the proposed algorithms.

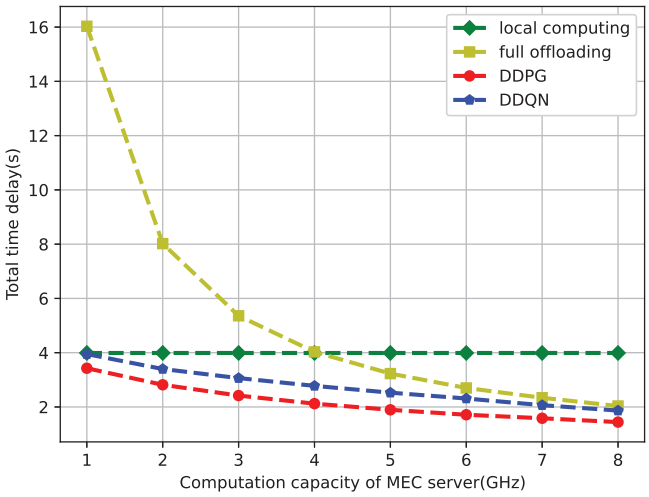

Fig. 8. The total time delay versus the computation capacity of UAV server

In Fig. 8, the effectiveness of the DDQN algorithm and DDPG algorithm is in contrast to the local computing method and full offloading method for different computation capacities of UAV server. The number of WUDs is 4 and the computation capacity of WUDs is 1 GHz. It is observed that as the computation capacity of UAV server grows, the total time delay gradually decreases. The increase of computation capacity of UAV server results in a corresponding increase in the computation resources allocated to each WUDs, which results in the reduction in total time delay. In addition, the total time delay of DDQN and DDPG algorithms in different cases is less than other methods, which proves the effectiveness of the proposed algorithm.

In Fig. 9, the effectiveness of the proposed algorithms is verified by varying the amount of tasks to be processed. The horizontal coordinate indicates the mean value of the task volume, i.e., when the horizontal coordinate is 900, c_i is uniformly distributed in [800 Megacycles, 1000 Megacycles]. As the task size grows, the total time delay increases. However, the DDQN and DDPG algorithms correspond to lower delays than the other two methods, which proves the effectiveness of the proposed algorithm.

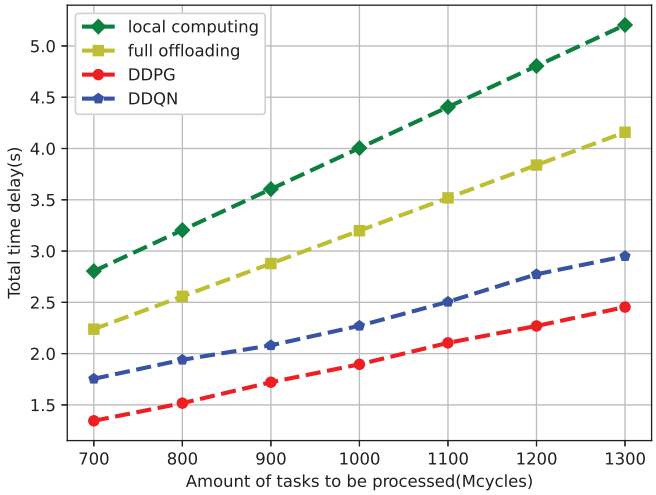

Fig. 9. The total time delay versus the amount of tasks to be processed

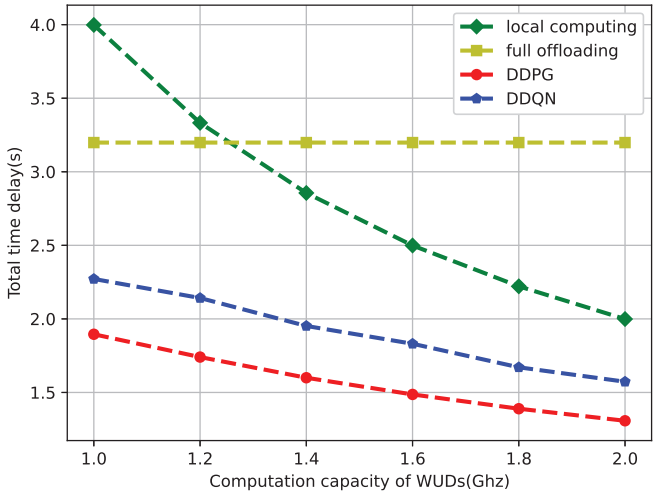

Fig. 10. The total time delay versus the computation capacity of WUDs

In Fig. 10, the horizontal coordinate represents the computation capacity of WUDs and the vertical coordinate represents the total time delay. The number of WUDs is set to 4, and the computation capacity of UAV server is set to 5 GHz. It can be clearly seen that the DDQN and DDPG algorithms are superior to other two methods for different computation capacity of

WUDs.

VI. Conclusion

In this paper, the UAV-assisted computation offloading architecture in the THz band was proposed to enhance the system capacity and QoS of WUDs. To support latency-sensitive applications, the optimization target was to minimize the total time delay. Considering the highly dynamic environment and limited resources, the DDQN and DDPG algorithms were provided to optimize the UAV placement, computation offloading, and resource allocation based on the UAV-assisted computation offloading architecture. The simulation results have shown that the proposed algorithm has good convergence performance and lower latency compared to full local computing method and full offloading method. Future work can be summarized as follows: 1) Consider more realistic scenarios, such as collaborative computation offloading of multiple UAVs and channel estimation errors. 2) Investigate the interpretability of reinforcement learning to further improve the algorithm performance.

References

[1] A. M. Zarca, J. B. Bernabe, A. Skarmeta, and J. M. Alcaraz Calero, "Virtual IoT honeynets to mitigate cyberattacks in SDN/NFV-enabled IoT networks," *IEEE J. Select. Areas Commun.*, vol. 38, no. 6, pp. 1262–1277, Jun. 2020.

[2] S. Verma, Y. Kawamoto, Z. Md. Fadlullah, H. Nishiyama, and N. Kato, "A survey on network methodologies for real-time analytics of massive IoT data and open research issues," *IEEE Commun. Surv. Tutorials*, vol. 19, no. 3, pp. 1457–1477, 2017.

[3] S. Al-Rubaye, J. Rodriguez, L. Z. Fragonara, P. Theron, and A. Tsourdos, "Unleash narrowband technologies for industrial internet of things services," *IEEE Network*, vol. 33, no. 4, pp. 16–22, Jul. 2019.

[4] M. Aazam, S. Zeadally, and K. A. Harras, "Deploying fog computing in industrial internet of things and industry 4.0," *IEEE Trans. Ind. Inf.*, vol. 14, no. 10, pp. 4674–4682, Oct. 2018.

[5] S. Khairy, P. Balaprakash, L. X. Cai, and Y. Cheng, "Constrained deep reinforcement learning for energy sustainable multi-UAV based random access IoT networks with NOMA," *IEEE J. Select. Areas Commun.*, vol. 39, no. 4, pp. 1101–1115, Apr. 2021.

[6] H. Guo, W. Huang, J. Liu, and Y. Wang, "Inter-server collaborative federated learning for ultra-dense edge computing," *IEEE Trans. Wireless Commun.*, vol. 21, no. 7, pp. 5191–5203, Jul. 2022.

[7] Y. Ye, R. Q. Hu, G. Lu, and L. Shi, "Enhance latency-constrained computation in MEC networks using uplink NOMA," *IEEE Trans. Commun.*, vol. 68, no. 4, pp. 2409–2425, Apr. 2020.

[8] H. Zhang, J. Zhang, and K. Long, "Energy efficiency optimization for NOMA UAV network with imperfect CSI," *IEEE J. Select. Areas Commun.*, vol. 38, no. 12, pp. 2798–2809, Dec. 2020.

[9] C. Castro, R. Elschner, T. Merkle, C. Schubert, and R. Freund, "Experimental demonstrations of high-capacity THz-wireless transmission systems for beyond 5G," *IEEE Commun. Mag.*, vol. 58, no. 11, pp. 41–47, Nov. 2020.

[10] H. Zhang, Y. Duan, K. Long, and V. C. M. Leung, "Energy efficient resource allocation in terahertz downlink NOMA systems," *IEEE Trans. Commun.*, vol. 69, no. 2, pp. 1375–1384, Feb. 2021.

[11] S. Krishna Moorthy and Z. Guan, "Beam learning in mmwave/Thz-band drone networks under in-flight mobility uncertainties," *IEEE Trans. on Mobile Comput.*, vol. 21, no. 6, pp. 1945–1957, Jun. 2022.

[12] X. Qiu, W. Zhang, W. Chen, and Z. Zheng, "Distributed and collective deep reinforcement learning for computation offloading: A practical perspective," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 5, pp. 1085–1101, May 2021.

[13] H. Zhang, H. Zhang, K. Long, and G. K. Karagiannidis, "Deep learning based radio resource management in NOMA networks: User association, subchannel and power allocation," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2406–2415, Oct. 2020.

[14] H. Guo, X. Zhou, J. Liu, and Y. Zhang, "Vehicular intelligence in 6G: Networking, communications, and computing," *Vehicular Communications*, vol. 33, pp. 100399, Jan. 2022.

[15] H. Zhang, N. Yang, W. Huangfu, K. Long, and V. C. M. Leung, "Power control based on deep reinforcement learning for spectrum sharing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 6, pp. 4209–4219, Jun. 2020.

[16] J. Wang, J. Liu, H. Guo, and B. Mao, "Deep reinforcement learning for securing software-defined industrial networks with distributed control plane," *IEEE Trans. Ind. Inf.*, vol. 18, no. 6, pp. 4275–4285, Jun. 2022.

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

[18] L. Xu et al., "Joint location, bandwidth and power optimization for THz-enabled UAV communications," *IEEE Commun. Lett.*, vol. 25, no. 6, pp. 1984–1988, Jun. 2021.

[19] N. N. Ei, M. Alsenwi, Y. K. Tun, Z. Han, and C. S. Hong, "Energy-efficient resource allocation in multi-UAV-assisted two-stage edge computing for beyond 5G networks," *IEEE Trans. Intell. Transport. Syst.*, pp. 1–12, 2022.

[20] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2049–2063, Mar. 2018.

[21] E. E. Haber, H. A. Alameddine, C. Assi, and S. Sharafeddine, "UAV-aided ultra-reliable low-latency computation offloading in future IoT networks," *IEEE Trans. Commun.*, vol. 69, no. 10, pp. 6838–6851, Oct. 2021.

[22] S. Mao, S. He, and J. Wu, "Joint UAV position optimization and resource scheduling in space-air-ground integrated networks with mixed cloud-edge computing," *IEEE Systems Journal*, vol. 15, no. 3, pp. 3992–4002, Sep. 2021.

[23] T. M. Ho and K.-K. Nguyen, "Joint server selection, cooperative offloading and handover in multi-access edge computing wireless network: A deep reinforcement learning approach," *IEEE Trans. on Mobile Comput.*, vol. 21, no. 7, pp. 2421-2435, Jul. 2022.

[24] Z. Shi, J. Liu, S. Zhang, and N. Kato, "Multi-agent deep reinforcement learning for massive access in 5G and beyond ultra-dense NOMA system," *IEEE Trans. Wireless Commun.*, vol. 21, no. 5, pp. 3057–3070, May 2022.

[25] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. on Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.

[26] Y. Zhan, S. Guo, P. Li, and J. Zhang, "A deep reinforcement learning based offloading game in edge computing," *IEEE Trans. Comput.*, vol. 69, no. 6, pp. 883–893, Jun. 2020.

[27] L. Qian, Y. Wu, F. Jiang, N. Yu, W. Lu, and B. Lin, "NOMA assisted multi-task multi-access mobile edge computing via

deep reinforcement learning for industrial internet of things," *IEEE Trans. Ind. Inf.*, vol. 17, no. 8, pp. 5688–5698, Aug. 2021.

[28] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Select. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[29] H. Zhang, H. Zhang, W. Liu, K. Long, J. Dong, and V. C. M. Leung, "Energy efficient user clustering, hybrid precoding and power optimization in terahertz MIMO-NOMA systems," *IEEE J. Select. Areas Commun.*, vol. 38, no. 9, pp. 2074–2085, Sep. 2020.

[30] J. M. Jornet and I. F. Akyildiz, "Channel modeling and capacity analysis for electromagnetic wireless nanonetworks in the terahertz band," *IEEE Trans. Wireless Commun.*, vol. 10, no. 10, pp. 3211–3221, Oct. 2011.

[31] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, S. U. Khan, and P. Li, "A double deep q-learning model for energy-efficient edge scheduling," *IEEE Trans. Serv. Comput.*, vol. 12, no. 5, pp. 739–749, Sep. 2019.

[32] C. Han and I. F. Akyildiz, "Distance-aware bandwidth-adaptive resource allocation for wireless systems in the terahertz band," *IEEE Trans. THz Sci. Technol.*, vol. 6, no. 4, pp. 541–553, Jul. 2016.