Joint Task Offloading, Resource Allocation, and Trajectory Design for Multi-UAV Cooperative Edge Computing with Task Priority

Hao Hao, Changqiao Xu, *Senior Member IEEE,* Wei Zhang, Shujie Yang, and Gabriel-Miro Muntean, *Fellow IEEE*

***Abstract*—Mobile edge computing (MEC) has emerged as a solution to address the demands of computation-intensive network services by providing computational capabilities at the network edge, thus reducing service delay. Due to the flexible deployment, wide coverage and reliable wireless communication, unmanned aerial vehicles (UAVs) have been employed to assist MEC. This paper investigates the task offloading problem in a UAV-assisted MEC system with collaboration of multiple UAVs, highlighting task priorities and binary offloading mode. We defined the system gain based on energy consumption and task delay. The joint optimization of UAVs' trajectory design, binary offloading decision, computation resources allocation, and communication resources management is formulated as a mixed integer programming problem with the goal of maximizing the long-term average system gain. Considering the discrete-continuous hybrid action space of this problem, we propose a novel deep reinforcement learning (DRL) algorithm based on the latent space to solve it. The evaluation results demonstrate that our proposed algorithm outperforms four state-of-the-art alternative solutions in terms of task delay and system gain.**

***Index Terms*—Mobile Edge Computing, Unmanned Aerial Vehicle (UAV), Task Offloading, Deep Reinforcement Learning**

I. INTRODUCTION

LATELY, many new computation-intensive and delay-sensitive network services which require large amounts of computation resources are emerging. Multi-access/mobile edge computing (MEC) [1] is an innovative computing paradigm that provides computational functions at the network edge to support such services. However, the current MEC solutions are not suitable for the situations with very high number of users or when network facilities are sparsely distributed [2]. In these situations, unmanned aerial vehicles (UAVs) can be employed to assist the MEC systems with their flexible deployment and large coverage potential, making UAV-assisted MEC systems promising solutions to enable execution of highly distributed computation-intensive tasks.

In terrestrial MEC networks both the location and service coverage of edge nodes are fixed and the edge servers can provide services for users in the local area only [5]. The UAV-assisted MEC systems have some unique features compared to the traditional terrestrial MEC systems [3][4]. First of all, UAVs introduce mobility and offer flexibility, and therefore can support close-range services and can increase system computing capacity dynamically. When the computation workload of an edge server is very high or users need support beyond the coverage area, UAVs' flight trajectory can be adjusted flexibly to provide service. Additionally, the scalability of the UAV-assisted MEC system is also unrivalled. Secondly, the onboard computing resources and energy supply of UAVs are very limited when compared with terrestrial edge servers, making any related task offloading decision very challenging. Therefore, unfortunately, the task offloading methods designed for terrestrial MEC-enabled networks can not be applied directly to the UAV-assisted MEC systems and new solutions are sought.

The single UAV-assisted task offloading problem has been extensively studied [6]-[10]. Due to its limited resources, a single UAV is adaptable to scenarios with small tasks, and the improvement provided to any task performance is limited. Also, it is often hard to meet the requirements for increasing task demands for computation resources. Instead, a more complex system which relies on collaboration of multiple UAVs is worth investigating [11]. It can provide rich computation resources and large service coverage. However, there are many challenges in relation to the task offloading problem in a multi-UAV MEC system, including trajectory design of multiple UAVs to avoid collisions of UAVs, management of communication resources between UAVs to improve transmission efficiency, collaborative task offloading to balance computation workload of UAVs, and so on.

In this context, few works on UAV-assisted MEC systems consider task priority. Diverse tasks have different tolerance of delay and their related services have diverse consequences [12]. For example, failure to complete navigation or road sensing tasks within the allowed delay threshold can have serious consequences (i.e. car accidents), while failure during live video streaming only affects user experience. Tasks with

H. Hao, W. Zhang are with Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan, China, and with Shandong Provincial Key Laboratory of Computer Networks, Shandong Fundamental Research Center for Computer Science, Jinan, China (e-mail: haoh@sdas.org, wzhang@sdas.org)

C. Xu, S. Yang are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. (e-mail: cqxu@bupt.edu.cn, sjyang@bupt.edu.cn).

G.-M. Muntean is with the Performance Engineering Laboratory, School of Electronic Engineering, Dublin City University, Dublin, Ireland. (e-mail: gabriel.muntean@dcu.ie).

strict delay constraints should have high priority and need to be processed first to meet their delay requirements. Using preemptive scheduling methods makes it difficult for the low-priority tasks to get their required computing resources from UAVs, causing high delay and low quality services. A different approach should be used in order to avoid the starvation of low-priority tasks. Therefore, the task offloading decision in a UAV-assisted MEC system should be carefully designed to satisfy the different task requirements.

Most existing UAV-assisted task offloading schemes focus on partial offloading, allowing some flexibility in allocating resources for smaller sub-tasks and further reducing task processing delay [3][13]. Although partial offloading has many advantages, it may be difficult to be applied to indivisible computation tasks [14]. Binary offloading may not be suitable to all cases, but is worth investigating as a complement to partial offloading, as it may provide more choices in the quest to achieve a good performance of task offloading in many scenarios. However, employing binary offloading may turn the task offloading problem into a joint optimization problem of continuous and discrete variables, which further increases the difficulty of solving the problem. Besides, both the requests and resource requirements associated with each UAV are highly time-varying. In this context, to improve the system performance, long-term average optimization is essential. However, it is hard to solve this problem of a non-convex nature, in a dynamic environment and with incomplete future information. Following the recent improvements of artificial intelligence (AI) approaches, deep reinforcement learning (DRL) has demonstrated good results in long-term optimization problem solving, which is very useful in wireless communications. Through training on historical data and exploring the dynamic environments, DRL can help take appropriate actions to get the optimal long-term average reward and make intelligent decisions under uncertainty, which can help solve our problem.

In order to address the above challenges, this paper focuses on task offloading in collaborative UAV-assisted MEC systems while considering task priorities and binary offloading. We optimize the long-term average system gain which is defined as being composed of task delay and energy consumption. The problem is formulated as a Markov Decision Process (MDP) with a discrete-continuous hybrid action space, and a novel DRL algorithm is proposed to solve it. The major contributions of this paper are as follows:

- We investigate the priority-aware task offloading problem in a collaborative multi-UAV-assisted system, whose goal is to maximize the long-term average system gain. The joint optimization of UAVs' trajectory design, offloading decision, computation resources allocation, and communication resources management is formulated as a mixed integer programming problem with the constraints of transmit power, computation capacity and task delay. Furthermore, this problem is transformed into a MDP.
- Considering that the traditional DRL algorithms are not compatible with a discrete-continuous hybrid action space, we introduce an embedding table for discrete actions and a conditional variational auto-encoder for continuous actions. Using the encoder, we construct a latent space for hybrid actions. Combining the latent space and a twin delayed deep deterministic policy gradient (TD3) algorithm, a novel DRL algorithm which can deal with a discrete-continuous hybrid action space is proposed to solve our joint optimization problem.
- We evaluate comparatively the proposed algorithm and experimental results show that our algorithm has better performance than four alternative solutions in terms of task delay and system gain.

The rest of this paper is organized as follows. The related works are discussed in Section II. The system model is introduced in Section III and the optimization problem formulation is shown in Section IV. Algorithm design and analysis of solutions are given in Section V. Section VI shows the performance of the proposed algorithm in terms of experimental results. Finally, Section VII concludes this paper.

II. Related Works

Employing UAV-assisted MEC systems is a promising approach to dynamically expand network computing capacity and support emergency events. Task offloading in a UAV-assisted MEC system is a key issue and is becoming the focus of the latest research. From the optimization scenario point of view, the existing works can been mainly divided into single UAV solution and multi-UAV cooperative approaches.

For single UAV-assisted MEC networks, authors of [6] designed a resource allocation framework, which maximizes the computation rate by jointly optimizing computation resources, communication resources and UAV trajectory. An alternative algorithm was proposed to solve the non-convex problem, and the successive convex approximation (SCA) method was used to optimize the UAV trajectory. Authors of [7] designed a UAV-assisted MEC system to reduce terrestrial signal blockage and shadowing. The joint optimization of UAV position, task offloading decision and resource allocation was formulated as a problem with the goal of minimizing task delay and UAV energy consumption. To solve the problem, an algorithm based on SCA was proposed. In [8], UAVs were applied to a 5G-enabled community task offloading system. The authors clustered users into communities based on geographic locations, and formulated the UAV-assisted task offloading problem as a mixed integer non-linear programming problem to maximize the average throughput. Researchers in [9] proposed an evolutionary multi-objective reinforcement learning algorithm to solve the UAV trajectory design and task offloading problem. They focused on three optimization sub-problems: minimize task delay, minimize energy consumption of UAV, and maximize the number of tasks collected by the UAV. The dependency among different tasks was highlighted in [10]. The joint optimization problem of resource allocation and trajectory design was formulated to minimize the system energy consumption with the constraints of task delay and dependency. The problem was further decomposed into two sub-problems iteratively and a joint dynamic programming and convex optimization algorithm was proposed to solve it. However, as mentioned, single UAV has limited power and computation resources, and may not significantly improve the system performance to meet users' requirements.

Multiple UAVs collaborative MEC systems which can utilize the resources of multiple UAVs have attracted increasing attention. Authors of [15] focused on the task offloading optimization problem in multiple UAV-enabled wireless networks. A two-layer cooperative framework based on software-defined networks to optimize the computation resources was designed, and a queue-based algorithm was proposed to minimize task delay. Authors of [16] formulated a delay minimization problem for the multi-UAVs enabled MEC networks. The load balancing of multiple UAVs was formulated as a no-convex problem. To get online task offloading decisions, authors further transformed the problem and utilized Lyapunov stochastic optimization to address it. DRL was used for the multi-UAV MEC systems in [17]. Authors formulated a MDP by jointly optimizing UAV trajectories, task offloading and transmit power. Considering the high-dimensional continuous action space, a multi-agent DRL based on the TD3 algorithm was proposed to minimize total system cost. In addition to the task offloading problem in UAV-enabled systems, service caching problem was also taken into account in [18]. UAVs made task offloading decision and service caching decision at different time frequencies where caching decision had a longer time window. The energy consumption is formulated as a virtual queue and an algorithm based on the Lyapunov optimization was proposed to minimize the long-term average service delay. The vehicular fog computing based UAV system which combined unmanned ground vehicles and UAVs was introduced in [19]. The task offloading was transformed into a two-sided matching problem, then the authors designed a distributed algorithm by the dynamic of UAVs to reduce task delay. The UAVs-assisted MEC technology was combined with intelligent transportation systems in [20]. The authors proposed a UAV-enabled multi-hop collaborative framework to maximize user experience and task delay in each time slot. However, the above works do not consider the priority of tasks when making task offloading decisions. Different tasks have different delay requirements. If we schedule all tasks equally, some important tasks may not be finished within the allowed delay threshold, which has serious consequences.

Few works on computing offloading consider task priority. The authors of [21] assigned a priority to each task based on its deadline and proposed a new delay-dependent priority-aware task offloading strategy for scheduling tasks, which can reduce the waiting time of the delay-sensitive tasks. The researchers who published [22] studied the priority-aware task offloading problem in a vehicular fog computing context. They formulated this problem as a MDP and proposed a DRL algorithm to solve it. Unfortunately, the research solutions proposed these papers rely on the fog computing framework and cannot be directly applied to UAV-enabled MEC networks. The authors of [23] studied the priority-aware task offloading problem with one UAV providing service. They employed a deep Q-learning algorithm for the problem and considered the scenario of a single UAV only, without any cooperation between multiple UAVs. The authors of [24] paid more attention to users' satisfaction of servers in UAV-enabled MEC networks and considered the task priority based on the delay requirements of users' tasks and remaining energy status of users. By jointly optimizing task offloading decisions and UAV scheduling strategy, the multi-UAVs enabled task offloading problem is formulated to maximize the total user satisfaction with constraints related to UAV energy consumption. This work mainly focused on the design of offloading decisions and UAV scheduling strategy, but did not consider the allocation of transmit power and computation resources. Additionally, the authors of [24] studied the partial task offloading problem, which applies to many scenarios. In real world, there are many indivisible computation tasks, and the study of binary task offloading is still highly valuable.

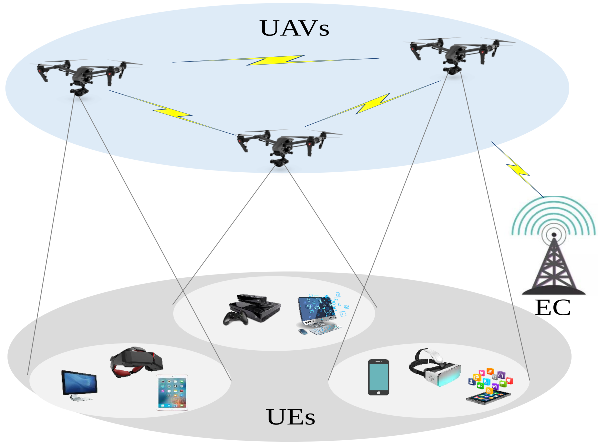


Fig. 1: UAV-assisted MEC system

III. System Model and Problem Formulation

A UAV-assisted MEC-based system with N UAVs, M user equipments (UEs) and access to an edge cloud server (EC) is considered as shown in Fig. 1. UAVs have two main roles related to data transmission and computation, respectively. On one hand, UAVs can forward computing tasks to other UAVs or the EC. On the other hand, UAVs can also provide computation resources to help UEs accomplish their tasks. Without loss of generality, the time is slotted, i.e., $\mathcal{T} = \{1, 2, ..., T\}$. A time slot refers to a short period of time, which can be in the region of several hundred milliseconds. Time slots are used to describe small time intervals in the proposed model design. Each UE m needs to handle computation-intensive tasks in each time slot. This can be defined via a four tuple $\mathbf{q}_m(t) = (c_m(t), u_m(t), v_m(t), o_m(t))$, where $c_m(t)$ is the computing workload (the number of CPU cycles), $u_m(t)$ is the transmitted data size, $v_m(t)$ is the allowed delay threshold and $o_m(t)$ is the task priority.

UAVs are equipped with multiple antennae, and can serve multiple UEs at the same time [25]. There are three transmission modes: ground-to-air (G2A) transmission from UE to UAV, air-to-air (A2A) transmission from UAV to UAV, and air-to-ground (A2G) transmission from UAV to EC.

A. UAVs Movement

We design the 3D coordinate of UAV n as $\mathbf{w}_n(t) = [x_n(t), y_n(t), z_n(t)]^T$, where $x_n(t)$, $y_n(t)$ and $z_n(t)$ are the X, Y, Z coordinates of UAV n at time slot t, respectively. Denote the $\mathbf{v}_n(t) = [x_n(t), y_n(t)]^T$ as the 2D coordinate of UAV n. UAVs always have limited flight distances because of their limited horizontal and vertical flight speeds, which can be given by:

$$\triangle v_n(t) = ||\mathbf{v}_n(t+1) - \mathbf{v}_n(t)|| \le L^h_{max} \tag{1}$$

$$\triangle z_n(t) = |z_n(t+1) - z_n(t)| \le L^v_{max} \tag{2}$$

$$Z_{min} \le z_n(t) \le Z_{max} \tag{3}$$

where $\triangle v_n(t)$ and $\triangle z_n(t)$ denote the horizontal travel distance and vertical travel distance, respectively; L^h_{max} and L^v_{max} are the maximum horizontal and vertical distances of the UAVs, respectively; Z_{min} and Z_{max} denote the minimum and maximum heights of UAVs.

To avoid collision between any two UAVs, the distance between UAVs should not be less than a minimum distance D_{min}. The collision constraint is:

$$||\mathbf{w}_n(t) - \mathbf{w}_j(t)|| \ge D_{min}, \forall n, j, n \ne j \tag{4}$$

When a rotary-wing UAV flies, its flight energy power is related to the speed v [26], which is defined as:

$$P^{fly}_n(v) = \frac{W_n}{2} v^2 \tag{5}$$

where W_n is the mass of UAV n. The flight energy consumption of UAV is obtained by:

$$E^{fly}_n(t) = P^{fly}_n\left(\frac{||\mathbf{w}_n(t+1) - \mathbf{w}_n(t)||}{\triangle t}\right)\triangle t \tag{6}$$

where $\triangle t$ is the interval duration of time slot.

B. Communication Model

We consider a UAV-enhanced MEC system which involves collaboration between multiple UAVs, which can communicate with each-other. We denote the bandwidths of the three main links as follows: B^G for the G2A links, B^A for the A2A links and B^E for the A2G links. As the output data size of sub-tasks is usually much smaller than the input data, next we ignore the cost of result downloading [8]. Beside, the cross-interference between UAVs and UEs is also neglected in this paper [27]. This can be the focus of future work.

1) G2A Transmission: For G2A communication links, there are many scatters or obstacles in the real environment. So the radio signals do not propagate in free space because of the shadowing or scattering caused by obstacles, which results in additional path loss. As a result, the use of the simplified free space path loss (FSPL) model [26] is not accurate enough to model the communication between ground UEs and air UAVs. Instead, a probabilistic path loss model which considers the occurrence probabilities and path loss of LoS and Non-LoS (NLoS) communication is introduced to model the G2A communications.

The occurrence probabilities of LoS and NLoS communications between UE m and UAV n are:

$$P^{LoS}_{m,n}(t) = \frac{1}{1 + ae^{-b((180/\pi)arcsin(z_n(t)/d_{m,n}(t)) - a)}} \tag{7}$$

$$P^{NLoS}_{m,n}(t) = 1 - P^{LoS}_{m,n} \tag{8}$$

where $d_{m,n}(t) = ||\mathbf{w}_n(t) - \mathbf{w}_m(t)||$ is the distance between UE m and UAV n, a and b are constant values related to the environment. Thus, the path loss between UE m and UAV n for LoS and NLoS communication is modeled as follows:

$$PL^{\zeta}_{m,n}(t) = L_{m,n}(t) + \eta_\zeta, \quad \zeta \in \{LoS, NLoS\} \tag{9}$$

where $L_{m,n}(t) = 20lg(4\pi/c) + 20lg(fr_c) + 20lg(d_{m,n}(t))$ denotes the free space path loss, lg is log_{10}, fr_c means the carrier frequency, c means the speed of light, and η_ζ is excessive path loss of LoS or NLoS links. We get the average path loss for the G2A links next:

$$\bar{PL}_{m,n}(t) = PL^{LoS}_{m,n}(t)P^{LoS}_{m,n}(t) + PL^{NLoS}_{m,n}(t)P^{NLoS}_{m,n}(t) \tag{10}$$

The channel gain between UE m and UAV n is

$$g_{m,n}(t) = 1/\bar{PL}_{m,n}(t) \tag{11}$$

Therefore, we denote uplink transmission rate from UE m to UAV n as follows:

$$r^{G2A}_{m,n}(t) = B^G log_2\left(1 + \frac{p_m(t) g_{m,n}(t)}{IN_{m,n} + N_G}\right) \tag{12}$$

where $IN_{m,n} = \sum^{m_0 \in N_n}_{m_0 \ne m} p_{m_0}(t) g_{m_0,n}(t)$ is the interference power signal from other UEs in the coverage area of UAV n, $p_m(t)$ is the transmit power of UE m, and N_G is the noise power.

Due to the limited coverage of a UAV, if UE m communicates with UAV n, the distance between UE m and UAV n cannot exceed the specified communication distance R_{G2A}, which is expressed as follows:

$$d_{m,n}(t) \le R_{G2A} \tag{13}$$

2) A2A Transmission: When the offloading target of UE m is UAV n' rather than UAV n which it belongs to, UE m transmits data to UAV n and UAV n forwards it to UAV n'. As UAVs can communicate in full duplex mode. UAV n can receive data from UE m while forwarding data to UAV n'. Considering the high hovering altitude of UAVs, the LoS link is the dominant one in A2A communications, and the communication environment between UAVs can be approximated as a free space. So, we apply the FSPL model to describe the A2A communications [28], where the path loss between UAV n and UAV n' is given as

$$PL^{A2A}_{n,n'} = 32.45 + 20lg(fr_c) + 20lg(d_{n,n'}(t)) \tag{14}$$

where $d_{n,n'}(t) = ||\mathbf{w}_n(t) - \mathbf{w}_{n'}(t)||$ is the distance between UAV n' and UAV n.

The data rate between UAV n and n' is expressed as

$$r_{n,n'}^{A2A}(t) = B^A log_2(1 + \frac{p_n(t)10^{-\frac{PL_{n,n'}^{A2A}}{10}}}{N_A}) \tag{15}$$

where $p_n(t)$ is the transmit power of UAV n, N_A is the noise power.

3) A2G Transmission: We denote the fixed location of EC as $\mathbf{w}^{EC} = [x^{EC}, y^{EC}, z^{EC}]^T$. The distance between UAV n and EC at time slot t is:

$$d_n^{EC}(t) = ||\mathbf{w}_n(t) - \mathbf{w}^{EC}|| \tag{16}$$

Similar to the G2A transmissions from UEs to UAVs, the channel gain between UAV n and the EC at time slot t is:

$$g_n(t) = \frac{1}{PL_n^{LoS} P_n^{LoS} + PL_n^{NLoS} P_n^{NLoS}} \tag{17}$$

where PL_n^{LoS} and PL_n^{NLoS} are the path loss of LoS and NLoS, P_n^{LoS} and P_n^{NLoS} are the occurrence probabilities of LoS and NLoS communication between UAV n and the EC, respectively. For the calculation of these parameters, refer to eq. (7)-(9).

The transmission rate from UAV n to the EC is:

$$r_n^{A2G}(t) = B^E log_2(1 + \frac{p_n(t) g_n(t)}{\sum_{n_0 \neq n}^{n_0 \in \mathbb{N}} p_{n_0}(t) g_{n_0}(t) + N_E}) \tag{18}$$

where N_E is the noise power.

C. Computation Model

We denote the task offloading decision as $\gamma_m^n(t) \in \{0, 1\}$, where $\gamma_m^n(t) = 1$ if UE m offloads task to computation location n at time t, otherwise, $\gamma_m^n(t) = 0$. Here, $n \in \{0, 1, ..., N, N+1\}$ indicates the computation location. If $n = 0$, the location is UE itself; if $1 \leq n \leq N$, the location is UAV n; if $n = N+1$, the location is EC. For example, if UE m completes the task locally, then $\gamma_m^0(t) = 1$. So, the tasks from UE m have $N+2$ options for computation locations: local device, anyone of N UAVs, and edge cloud server. In other words, a UAV can offload computing tasks from users within its own coverage area to other UAVs; this illustrates the collaboration between multiple UAVs. In a classic model without the collaboration between UAVs, tasks have three options only for computation locations: local device, the UAV they belongs to, and edge cloud server. In that case, a UAV cannot offload any task to other UAVs, even if they are free.

We assume that computing tasks are indivisible, and a task can only be processed at one location in each time slot. The constraints of tasks are as follows:

$$\sum_{n=0}^{N+1} \gamma_m^n(t) = 1 \tag{19}$$

The computation delay of UEs is:

$$t_m^{UE}(t) = \frac{\gamma_m^0(t) c_m(t)}{f_m^0} \tag{20}$$

where f_m^0 is the computing capability of UE m. The computation delay of UAVs is

$$t_m^{UAV}(t) = \sum_{n=1}^{N} \frac{\gamma_m^n(t) c_m(t)}{f_m^n(t)} \tag{21}$$

where $f_m^n(t)$ is the computing capability that UAV n allocates to UE m at time slot t. UAV n has limited computing resources [29], the constraint is:

$$\sum_{m=1}^{M} f_m^n(t) \leq F^n \tag{22}$$

where F^n is the computing capacity of UAV n. If tasks are offloaded to EC, the computation delay is:

$$t_m^{EC}(t) = \frac{\gamma_m^{N+1}(t) c_m(t)}{f_m^{N+1}} \tag{23}$$

where f_m^{N+1} is the fixed computing power allocated to UE m by the EC.

D. Task Priority Model

Tasks are classified into high-priority tasks and low-priority tasks according to their allowed delay threshold. High-priority tasks have strict delay constraint (i.e. navigation, road-sensing in vehicular). If we cannot finish a high-priority task within its maximum tolerable delay, the task will be failed and results in severe impact. The tasks with tolerant delay are classified as low priority tasks, such as entertainment applications. If the allotted time for a low-priority task surpasses the allowed delay threshold, it might solely impact the user experience without compromising the overall usefulness of the result.

To prevent low-priority computational tasks from starvation, we utilize distinct utility functions to represent task priorities, rather than employing preemptive scheduling methods directly. Similar to [22][30], we consider a definition of task utility based on priority, completion time (task delay), and allowed delay threshold. For high-priority tasks, it is mandatory that they are completed within the designated delay threshold. When a high-priority task satisfies its allowed delay threshold, it is considered available, and its utility is non-negative and inversely proportional to the completion time. However, if a high-priority task exceeds the allowed delay threshold and cannot be completed in time, it is deemed a failure and incurs a negative utility as a penalty. We establish the utility function of a high-priority task following the principles mentioned above, as follows:

$$U_m^H(t) = \begin{cases} log_2(1 + v_m(t) - T_m(t)), & T_m(t) \leq v_m(t) \\ -P^H, & T_m(t) > v_m(t) \end{cases} \tag{24}$$

where $T_m(t)$ is the completion time, and $-P^H$ is a negative constant, which represents the penalty for not completing the high-priority task within its allowed delay threshold.

For a low-priority task, the completion time requirement is relatively lenient. If a low-priority task cannot be completed within its allowed delay threshold, it is still considered available, but the utility decreases exponentially with time. On the other hand, if a low-priority task is completed before the deadline, the utility is a positive constant as a reward. We define the utility function for a low-priority task next:

$$U_m^L(t) = \begin{cases} P^L, & T_m(t) \leq v_m(t) \\ P^L e^{-\rho(T_m(t)-v_m(t))}, & T_m(t) > v_m(t) \end{cases} \quad (25)$$

where P^L is a fixed positive value that represents the reward for successfully completing a low-priority task within its specified time limit, and $\rho > 0$ is a constant. Specifically, if a low-priority task cannot be completed (i.e. $t_n = -\infty$), then the utility is zero.

The task priority model employed, which uses logarithmic and negative exponential expressions, is appropriate. Logarithmic and negative exponential forms have long tail effects which are close to how user experience manifests. For example, if the latency of a service changes from 0.1s to 1s, it will have a big impact on the user experience. However, if the latency of a service changes from 10.1s to 11s, it will have little impact on the user experience. Logarithmic and negative exponential forms can describe this property very well. Besides, the minimum value of a logarithmic expression is 0 if the high-priority task can been completed within the allowed delay threshold, which guarantees that the utility of on-time completion is higher than the utility of a task overtime. Similarly, the maximum value of a negative exponential expression is 1 if the low-priority task cannot be completed within the allowed delay threshold, which guarantees that the utility of a task overtime is lower than the utility of the on-time completion.

IV. Problem Optimization

A. Multi-UAV Cooperative Computation Model

Based on location, there are three computation types: computation at UEs, computation at UAVs and computation at the EC.

1) Computation at UEs: There is no transmission delay if UEs finish tasks locally, so the total delay is equal to the computation delay $T_m^0(t) = t_m^{UE}(t)$. There is only energy consumption of local computation.

$$E_m^0(t) = \kappa_0 (f_m^0)^3 t_m^{UE}(t) \quad (26)$$

where $\kappa_0 \leq 0$ is the effective switched capacitance of UEs.

2) Computation at UAVs: We assume that UE m offloads data to UAV n' in time slot t. UE m first needs to transfer the data to UAV n that it belongs to. If the data target is not n, which means $n' \neq n$, UAV n has to further transfer the data to UAV n'. As the UAVs communicate in full duplex. UAV n can receive the data from UE m while also can forward the received data to the target UAV n'. In this process, UAV n assumes the role of a transmission relay, and the G2A and A2A data transmissions are done in parallel. Therefore, the transmission delay takes the maximum values of the time needed for G2A and A2A communications. Otherwise, UAV n allocates computing resource to UEs m directly. The transmission delay is:

$$t_m^{n'}(t) = max\{\frac{o_k}{r_{m,n}^{G2A}(t)}, \frac{o_k}{r_{n,n'}^{A2A}(t)}\} \quad (27)$$

where $o_k / r_{n,n'}^{A2A}(t) = 0$ if $n = n'$. The total delay is:

$$T_m^{n'}(t) = t_m^{n'}(t) + t_m^{UAV}(t) \quad (28)$$

The transmission energy consumption from UE m to UAV n can be obtained as follows:

$$e_m^n(t) = \frac{p_m(t) o_k}{r_{m,n}^{G2A}(t)} \quad (29)$$

Similarly, if the target UAV n' is not n, the transmission energy consumption from UAV n to UAV n' can be obtained as follows:

$$e_n^{n'}(t) = \frac{p_n(t) o_k}{r_{n,n'}^{A2A}(t)} \quad (30)$$

The computation energy consumption of UAV n' is:

$$e_{m,n'}^{UAV}(t) = \kappa_{n'} [f_m^{n'}(t)]^3 t_m^{UAV}(t) \quad (31)$$

where $\kappa_{n'}$ is the effective switched capacitance of UAV n'.

The total energy consumption if UE m offloads task to UAV n' can be obtained as follows:

$$E_m^{n'}(t) = e_m^n(t) + e_n^{n'}(t) + e_{m,n'}^{UAV}(t) \quad (32)$$

3) Computation at EC: Similar to computation at UAVs, UE m transmits data to the EC through UAV n. The transmission delay is:

$$t_m^{N+1}(t) = max\{\frac{o_k}{r_{m,n}^{G2A}(t)}, \frac{o_k}{r_n^{A2G}(t)}\} \quad (33)$$

and the total delay is:

$$T_m^{N+1}(t) = t_m^{N+1}(t) + t_m^{EC}(t) \quad (34)$$

The transmission energy consumption from UAV n to the EC is as follows:

$$e_n(t) = \frac{p_n(t) o_k}{r_n^{A2G}(t)} \quad (35)$$

Considering that the EC has sufficient power, we do not incorporate the energy consumption of EC into the optimization. The total energy consumption is then:

$$E_m^{N+1}(t) = e_m^n(t) + e_n(t) \quad (36)$$

B. Problem Design

The service delay of UE m at time slot t is:

$$\begin{aligned} T_m(t) \quad = \quad & \gamma_m^0(t) T_m^0(t) + \gamma_m^{N+1}(t) T_m^{N+1}(t) \\ & + \sum_{n'=1}^{N} \gamma_m^{n'}(t) T_m^{n'}(t) \end{aligned} \quad (37)$$

As previously mentioned, computational tasks of varying priorities have distinct requirements regarding task delay. Rather than directly optimizing task delay, we optimize the priority-based utility function of task delay, which is defined as follows:

$$U_m(t) = (1 - o_m(t))U_m^H(t) + o_m(t)U_m^L(t) \tag{38}$$

where $o_m(t) = 0$ is the high-priority task, and $o_m(t) = 1$ is the low-priority task.

The total energy consumption is:

$$\begin{aligned} E_m(t) &= \gamma_m^0(t)E_m^0(t) + \gamma_m^{N+1}(t)E_m^{N+1}(t) \\ &\quad + \sum_{n'=1}^{N}(\gamma_m^{n'}(t)E_m^{n'}(t) + E_n^{fly}(t)) \end{aligned} \tag{39}$$

Task delay and energy consumption are two main factors in UAV-assisted MEC systems, which are also our optimization objectives. Similar to [7][15][17], we define the system gain as a weighted sum of the energy consumption $E_m(t)$ and the priority-based utility function $U_m(t)$ which combines task delay and priority. The utility function of system gain is defined as follows:

$$F_m(t) = w_1 U_m(t) - w_2 E_m(t) \tag{40}$$

where w_1 and w_2 are weight parameters. We can adjust the weight parameters according to the system deployment scenario. For example, in delay-sensitive systems, we can increase the weight parameter w_1 or decrease the weight parameter w_2. Even we can optimize the task delay only by setting $w_2 = 0$.

Thus, by jointly optimizing offloading decision γ, UAVs position $\mathbf{w}$, transmit power $\mathbf{p}$, and the computation resource allocation of UAVs $\mathbf{f}$, the task offloading optimization problem can be designed to maximize the total system gain. The problem is formulated as follows:

$$\max_{\gamma,\mathbf{w},\mathbf{p},\mathbf{f}} \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\sum_{m=1}^{M} F_m(t)$$

$$s.t.\quad 0 \le p_n(t) \le P_{max}^{UAV}, \quad \forall n \in \mathcal{N} \tag{41a}$$

$$0 \le p_m(t) \le P_{max}^{UE}, \quad \forall m \in \mathcal{M} \tag{41b}$$

$$\gamma_m^n(t) \in \{0, 1\} \tag{41c}$$

$$x_{min} \le x_n(t) \le x_{max},\ y_{min} \le y_n(t) \le y_{max} \tag{41d}$$

$$\triangle w_n(t) \le v_{max}\triangle t \tag{41e}$$

$$(1) - (4), (13), (19), (22) \tag{41f}$$

where the optimization goal is to maximize the long-term average system gain. Constraints (41a) and (41b) indicate that the transmit power of UAVs and UEs are limited. Constraint (41c) denotes the constraints of task offloading, eq. (41d) and eq. (41e) are the constraints related to the movement area and movement speed of UAVs, respectively. Eq. (1)-(4) describe the position constraints of UAVs, (13) denotes UE is within the coverage range of the UAV, (19) denotes that there is one and only one device available to process the task, and (22) is the constraints about the limited computing resources of UAVs.

Generally, it is intractable to solve the optimization problem (41). The optimization objective is the long-term average system gain, which always need the future information in traditional methods (i.e. dynamic programming). However, it is challenging to predict system state in dynamically networks. Moreover, DRL can achieve model-free learning by data sampling instead of state transition. Although DRL is an effective method to solve long-term average optimization problem, this is a discrete-continuous hybrid optimization problem. There are scalability issue and additional approximation difficulty which may decrease the model performance if we use traditional DRL methods directly. To address these challenges, a novel DRL method will be investigated to learn the near-optimal policy with discrete-continuous hybrid action space in the next section.

C. MDP Formulation

In UAV-assisted MEC systems, we optimize the offloading decision, UAVs position, transmit power and computation resource allocation to maximize the system gain. The system state in the next time slot depends on the state and action at the current time only. In this case, the UAV-assisted task offloading problem (41) can be formulated as a MDP. In time slot t, we observe system state and then select the action. The system will generate a corresponding reward to reflect the action. The goal is to maximize the long-term system reward by employing an optimization strategy that maps states to actions.

1) State Space $\mathcal{S}$: If we add the channel quality of each transmission link into the state space, the state space will increase rapidly with the number of UAVs, increasing the complexity of any associated algorithm to $\mathcal{O}(N^2)$. In order to reduce the size of the state space, we noted that the channel quality is related to the positions of UAVs due to the time-invariant signal interference (i.e. noise power) between two positions in the model. In other words, the channel quality of links varies with the positions of UAVs, and we can calculate the channel quality of links according to the positions of UAVs if the signal interference is fixed. Therefore, we only add the positions of UAVs into the state space to describe the channel quality and the associated complexity is $\mathcal{O}(N)$. Additionally, we do not add the variables that are not time-varying (such as L_{max}^h and L_{min}^v) to the state space. We can use these variables directly during training without them being part of the state space. Therefore in the optimization problem, the state $s(t)$ is composed of properties of computing tasks and 3D coordinate positions of UAVs, that is:

$$s(t) = \{\mathbf{q}(t), \mathbf{w}(t)\} \tag{42}$$

where $\mathbf{q}(t) = [\mathbf{q}_1(t), \mathbf{q}_2(t), ..., \mathbf{q}_M(t)]$ and $\mathbf{w}(t) = [\mathbf{w}_1(t), \mathbf{w}_2(t), ..., \mathbf{w}_N(t)]$. Since the total dimension of computing tasks' properties is $4M$ and the total dimension of UAVs' positions is $3N$, the total dimension of state $s(t)$ is $4M + 3N$, where N is the number of UAVs and M is the number of UEs.

2) Action Space $\mathcal{A}$: If we directly use the $\gamma_m^n(t)$ as the action, the action space is $M(N+2)$. This both increases the number of output neurons and leads to additional consideration of constraint (19), which increases the complexity of training. For each computing task of UE m, there are $N+2$ positions to choose from. We can complete it in UE locally, or offload it to UAV n, or offload it to EC. To simplify the discrete action in action space, we use $i_m(t) \in \{0, 1, ..., N+1\}$ to denote the computation position, where $i_m(t) = 0$ means we complete the task at the UE locally, $i_m(t) = N+1$ means the task is offloaded to EC, otherwise, the task is offloaded to UAV $i_m(t)$. In this way, we can reduce the number of neurons for task offloading variable to M, and do not need to consider constraint (19) during training.

In addition to task offloading variable, we have to determine the mobility of UAVs , the transmit power and the allocation of computation resources. To be specific, the action at time slot t is defined as:

$$a(t) = \{\mathbf{i}(t), \triangle\mathbf{w}(t), \mathbf{p}(t), \mathbf{f}(t)\} \tag{43}$$

where $\mathbf{i}(t) = [i_1(t), i_2(t), ..., i_M(t)]$ is the decision of task offloading, $\triangle\mathbf{w}(t) = \{\triangle\mathbf{w}_1(t), \triangle\mathbf{w}_2(t), ..., \triangle\mathbf{w}_N(t)\}$ is the mobility of all UAVs, $\mathbf{p}(t) = [p_1(t), i_2(t), ..., p_{M+N}(t)]$ is the transmit power of all UEs and UAVs, $\mathbf{f}(t) = [f_m^n(t)], \forall m \in \{1, ..., M\}, \forall n \in \{1, ..., N\}$ is the allocated computation resources from UAV n to UE m. The dimension of action $a(t)$ is $M + 3N + (M+N) + MN = 4N + 2M + MN$.

3) Reward Function: The goal of the formulated task offloading optimization problem (41) is to maximize the system gain while satisfying certain constraints. Therefore, an action has a larger reward if it can bring a higher system gain and satisfies all constraints [31]. Otherwise, if certain constraints are not satisfied, there will be corresponding penalties in the reward function. The reward function is defined as follows:

$$r(t) = \begin{cases} \sum_{m=1}^{M} F_m(t), & if\ sastifies\ constraints \\ -Pu, & otherwise \end{cases} \tag{44}$$

where Pu is a positive value and $-Pu$ is the penalty for actions that do not satisfy constraints. Notable is that, we can influence the reward function by adjusting the value of P^H in Eq.(24) and P^L in Eq.(25), which affect the completion ratio of high-priority tasks and low-priority tasks. For example, if we increase the value of P^L, the reward for completing a low-priority task increases, and the model will allocate more resources to low-priority tasks. As a result, the completion rate of low-priority tasks will increase. However, as the total amount of resources is limited, improving the completion rate of low-priority tasks is expected to reduce the completion rate of high-priority tasks.

V. DRL-based Algorithm Design

Because the above-described MDP has a discrete-continuous hybrid action space, conventional DRL algorithms are not suitable for it. If we convert the hybrid action space into either a discrete or a continuous action space directly, it may lead to a degradation in model performance due to scalability issues and increased approximation complexity. To address this problem, we propose a novel algorithm, which is based on a hybrid action representation, as introduced in [32].

A. Latent Space

In terms of the formulated MDP, there are discrete variable $\mathbf{i}(t)$ and continuous variables $\{\mathbf{w}(t), \mathbf{p}(t), \mathbf{f}(t)\}$. Hybrid action representation can convert the discrete-continuous hybrid action space problem into a continuous policy learning problem which considers the dependence between the two heterogeneous components. With some abuse of notation, we use p to uniformly refer to continuous actions, and we get rid of the subscript t (i.e., action $a = (i_1, i_2, ..., i_M, p)$) to help clarify the algorithm. We detail the method from dependence-aware encoding and decoding of hybrid action.

There are $N+2$ locations for computation offloading for each task. We first establish an embedding table $G_\omega \in \mathbb{R}^{(N+2)\times l_1}$ with learnable parameters ω to denote the $N+2$ discrete actions. In the table, each row $g_{\omega,i_m} = G_\omega(i_m)$ is a l_1-dimensional continuous vector for the discrete action i. Note that there are M UEs to make decisions in each time slot, so M embedding tables should be established for learning. However, the action space for each UE and the meaning represented by each action are both consistent, which means all UEs can share a common embedding table.

To construct a l_2-dimensional latent representation space for the continuous parameters, a conditional Variational Auto-Encoder (VAE) [33] is utilized. In the mathematical formulation, given a hybrid action $a = (i_1, i_2, ..., i_M, p)$ and a state s, the encoder $q_\phi(z|p, s, g_{\omega,i_m})$ with parameters ϕ maps p to the latent variable $z \in \mathbb{R}^{l_2}$ conditioned on s and g_{ω,i_m}. In this case, a Gaussian latent distribution $\Gamma(\mu_q, \sigma_q)$ is employed to describe the encoder $q_\phi(z|p, s, g_{\omega,i_m})$. The encoder outputs the mean μ_q and standard deviation σ_q of the latent distribution. By sampling from this distribution, we obtain the latent representation $z \sim \Gamma(\mu_q, \sigma_q)$.

Under the same condition, the decoder $q_\psi(\tilde{p}|z, s, g_{\omega,i_m})$ with parameters ψ reconstructs the continuous parameter $\tilde{p}$ from z. Given a sample $z \sim \Gamma(\mu_q, \sigma_q)$, the decoder deterministically decodes it, resulting in $\tilde{p} = q_\psi(z, s, g_{\omega,i_m})$. Furthermore, through nearest-neighbor lookup in the embedding table for g_{ω,i_m}, we can decode the discrete parameter i_m.

We use the encoder to construct a hybrid action representation space ($\in \mathbb{R}^{Ml_1+l_2}$) for hybrid actions. Additionally, we can decode latent variables $g \in \mathbb{R}^{Ml_1}$ and $z \in \mathbb{R}^{l_2}$ into a hybrid action $(i_1, ..., i_M, p)$ based on the decoder. To formalize this, the encoding and decoding processes are summarized as follows:

Encoding:

$$g_{\omega,i_m} = G_\omega(i_m), z \sim q_\phi(\cdot|p, s, g_{\omega,i_m}) \tag{45}$$

Decoding:

$$i_m = argmin_{i' \in \mathcal{I}} ||g_{\omega,i'} - g||_2, \tilde{p} = q_\psi(z, s, g_{\omega,i_m}) \tag{46}$$

We train G_ω and q_ϕ, q_ψ together using experiences from buffer $\mathcal{D}$ by minimizing the loss function:

$$\begin{aligned} L_V(\psi, \phi, \omega) \quad &= \quad \mathbb{E}[||p - \tilde{p}||_2^2 + \\ &\quad D_{KL}(q_\phi(\cdot|p, s, g_{\omega,i_m})||\Gamma(0, I))] \end{aligned} \tag{47}$$

where the first term represents the squared L_2-norm reconstruction error, and the second term represents the Kullback-Leibler divergence (DKL) between the variational posterior of the latent representation z and the standard Gaussian prior.

Because hybrid actions have varying impacts on the environment, we incorporate a cascaded structure that follows the transformation network of the conditional VAE decoder. For any experience sample (s, a, s'), we define the state residual as $\delta_{s,s'} = s' - s$. By introducing the cascaded structure into the decoder, we can generate predictions according to the following process:

$$\bar{\delta}_{s,s'} = q_\psi(z, s, g_{\omega,i_m}), \quad for \;\; z, s, g_{\omega,i_m} \tag{48}$$

Then the L_2-norm square prediction error is:

$$L_D(\psi, \phi, \omega) = \mathbb{E}[||\bar{\delta}_{s,s'} - \delta_{s,s'}||_2^2] \tag{49}$$

So, we minimize the ultimate training loss:

$$L_H(\psi, \phi, \omega) = L_V(\psi, \phi, \omega) + \alpha L_D(\psi, \phi, \omega) \tag{50}$$

where α is a weight parameter that depends on the importance of the loss associated with the dynamics predictive representation. We denote the dimension of the system state s as dim, and the dimension of the continuous policy p as $\mathcal{X}_p$. The network structures of the encoder and decoder are illustrated in Table I.

Although the latent space for hybrid action representation increases the complexity of the algorithm, it is necessary. Take the DRL algorithm DDPG as an example. The actions outputted by DDPG are continuous. For the discrete variable in hybrid action, we need to convert the continuous actions outputted by the model into discrete values by crude methods such as rounding. In this case, even though the model outputs for instance 4.6 and 4.9, the results will be the same (both are rounded to 5), which leads to a degradation in the model's performance. Therefore, using a latent space which can convert between continuous output values and discrete variables is more accurate. Considering that discrete variables and continuous variables in a hybrid action space are coupled with each other, we did not encode only discrete variables, but the whole hybrid action, to ensure the correlation of variables

B. Cooperative Long-term Average Optimization Algorithm

In the previous section, we discussed the construction of the hybrid action representation space. Now, this representation space will be combined with the model-free TD3 algorithm [34] to solve the task offloading problem.

TD3 is an algorithm for deterministic strategy reinforcement learning that is well-suited for continuous action spaces with high dimensions. It utilizes two types of networks: the actor and the critic. The actor network maps various states to their corresponding actions, influencing the decision-making process. The critic network estimates the potential rewards associated with different actions given specific states, influencing the action's value. The actor and critic networks are implemented separately using distinct neural networks, which are shown in Table II.

TABLE I: Network Structures of Encoder and Decoder

Component	Layer	Structure
Discrete Action Embedding Table G_ω	Parameterized Table	$(\mathbb{R}^{N+2}, \mathbb{R}^{l_1})$
Conditional Encoder Network q_ϕ	Fully Connected(encoding) Fully Connected(condition) Element-wise Product Fully Connected Activation Fully Connected(mean) Activation Fully Connected(log_std) Activation	$(\mathcal{X}_p, 512)$ $(dim + \mathbb{R}^{Ml_1}$,512) ReLU · ReLU (512,512) ReLU (512,$\mathbb{R}^{l_2}$) None (512,$\mathbb{R}^{l_2}$) None
Conditional Decoder & Prediction Network q_ψ	Fully Connected(latent) Fully Connected(condition) Element-wise Product Fully Connected Activation Fully Connected (reconstruction) Activation Fully Connected Activation Fully Connected(prediction) Activation	$(\mathbb{R}^{l_2}, 512)$ $(dim + \mathbb{R}^{Ml_1}$,512) ReLU · ReLU (512,512) ReLU (512,$\mathcal{X}_p$) None (512,512) ReLU (512,dim) None

TABLE II: Network Structures of TD3

Model Component	Layer	Structure
Actor Network π_ζ	Fully Connected Activation Fully Connected Activation Fully Connected Activation	$(dim, 512)$ ReLU (512,512) ReLU (512,$\mathbb{R}^{Ml_1+l_2}$) Tanh
Critic Network Q_{θ_j}	Fully Connected Activation Fully Connected Activation Fully Connected Activation	$(dim + \mathcal{X}_p + M, 512)$ ReLU (512,512) ReLU (512,1) None

The actor network takes the state s as input and produces a latent action vector, represented as g and z, (i.e. $g, z = \pi_\zeta(s)$ where $g \in \mathbb{R}^{Ml_1}, z \in \mathbb{R}^{l_2}$). Next, we utilize a decoder to decode this latent action vector (g, z) into a corresponding discrete-continuous hybrid action $a = (i_1, ...i_M, p)$. To approximate the hybrid-action value function Q^{π_ζ}, we employ twin critic networks $Q_{\theta_1}, Q_{\theta_2}$. These networks take the hybrid action a as input. In training, we use the collected experience (s, a, r, s') stored in the buffer $\mathcal{D}$ to train the critics using the Clipped Double Q-Learning algorithm. The loss function for training the critics is as follows:

$$L_{CDQ}(\theta_j) = \mathbb{E}[(\varsigma - Q_{\theta_j}(s, g, z))^2], \quad for \;\; j = 1, 2 \tag{51}$$

where $\varsigma = r + \gamma min Q_{\bar{\theta}_j}(s', \pi_{\bar{\zeta}}(s'))$ and $\bar{\theta}_j, \bar{\zeta}$ are the target network parameters. The actor (latent policy) is updated with

Deterministic Policy Gradient [35] as follows:

$$\nabla_\zeta J(\zeta) = \mathbb{E}[\nabla_{\pi_\zeta(s)} Q_{\theta_1}(s, \pi_\zeta(s)) \nabla_\zeta \pi_\zeta(s)] \tag{52}$$

Algorithm 1 CLP training Algorithm

```
1: Initialize actor $\pi_\zeta$ and critic networks $Q_{\theta_1}, Q_{\theta_2}$ with random parameters $\zeta, \theta_1, \theta_2$;
2: Initialize discrete action embedding table $G_\omega$ and conditional VAE $q_\phi, q_\psi$ with random parameters $\omega, \phi, \psi$;
3: Initialize state information $s_1$;
4: Prepare replay buffer $\mathcal{D}$;
5: while not reach maximum warm-up training times do
6:     Update $\omega, \phi, \psi$ using samples in $\mathcal{D}$ by Eq.(50);
7: end while
8: while not reach maximum total environment steps do
9:     Observe current system state $s$;
10:    /* select latent actions by actor network */
11:    $g, z = \pi_\zeta(s) + \epsilon_g$ with $\epsilon_g \sim \Gamma(0, \sigma)$;
12:    /* decode into original hybrid actions */
13:    Decode $i = f_D(g), p = q_\psi(z, s, g)$ by decoder;
14:    Execute $(i, p)$, get reward $r$ and new state $s'$;
15:    Store $(s, i, p, g, z, r, s')$ in replay buffer $\mathcal{D}$;
16:    /* evaluate hybrid actions by critic network */
17:    Sample a mini-batch experience from $\mathcal{D}$;
18:    Update $Q_{\theta_1}, Q_{\theta_2}$ according to the loss function Eq.(51);
19:    Update $\pi_\zeta$ with policy gradient according to Eq.(52);
20:    while not reach representation training times do
21:        Update $\omega, \phi, \psi$ using samples in $\mathcal{D}$ by Eq.(50);
22:    end while
23: end while
```

Combining the latent representation space and TD3, we propose the *Cooperative Long-term average oPtimization (CLP)* algorithm to solve the joint optimization problem of UAV placement and resource allocation in UAV-assisted MEC system. The proposed CLP algorithm is detailed in **Algorithm 1**. We first initialize the parameters of networks and embedding table randomly, and initialize the system state $s(1)$ with the UAV start positions. There are two major stages in training: warm-up stage and learning stage. In the warm-up stage, the encoder and decoder are pre-trained by experiences found in the replay buffer $\mathcal{D}$ (line 5-7). In the learning stage, the actor outputs a latent action g, z perturbed by a Gaussian exploration noise based on current s. Then the decoder decodes the latent action g, z into the original hybrid action i, p to interact with the environment and get the reward r and the new state s'. The experience (s, i, p, g, z, r, s') is stored in the replay buffer $\mathcal{D}$. To avoid the correlation of input samples, we randomly sample a mini-batch experience from $\mathcal{D}$. We will calculate the loss function according the evaluation of critic network, and update parameters of the actor network and critic network with a policy gradient (lines 18-19). In addition, the encoder and decoder are updated concurrently in the training stage to adapt the change of data distribution as shown in lines 20-22. Note that the actor network can be used without the critic network (lines 9-15) when the model has been trained. The CLP framework is illustrated in Fig.2.

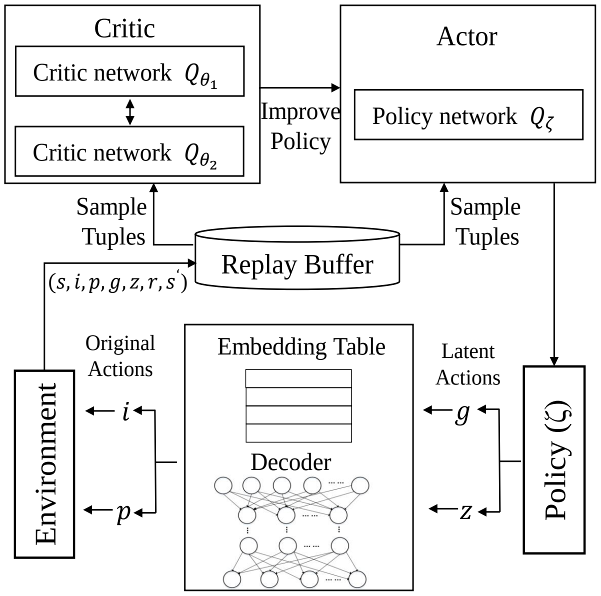


Fig. 2: Framework of CLP Algorithm

C. *Complexity Analysis*

The complexity of our proposed algorithm can be analysed after considering its two main aspects. First, there is the complexity related to the encoding and decoding of hybrid actions. Secondly, there is the complexity associated with training the actor and critic networks. As referenced in [36], the computational complexity of back-propagation algorithm for a fully-connected neural network with fixed number of hidden layers and neurons is proportional to the product of input size and output size.

In the encoding and decoding of hybrid actions, the input size of encoder is $dim + Ml_1 + \mathcal{X}_p = MN + Ml_1 + 5M + 7N$ where $dim = 4M + 3N$ and $\mathcal{X}_p = MN + 4N + M$. The output size of encoder is l_2, so the computational complexity of encoder is $\mathcal{O}((MN + Ml_1)l_2)$. The input size of decoder is $dim + Ml_1 + l_2 = 4M + 3N + Ml_1 + l_2$, and the output size is $dim + \mathcal{X}_p = 3MN + 5M + 7N$. So the decoder complexity is $\mathcal{O}(M^2Nl_1 + MNl_2 + MN^2)$.

In the training actor and critic networks, the input size of actor is the dimensions of system space $dim = 4M + 3N$, the output size is the dimensions of hybrid action representation space $Ml_1 + l_2$, so the complexity of the actor is $\mathcal{O}((Ml_1 + l_2)(N + M))$. The input size of critic is $dim + \mathcal{X}_p + M$, the output is 1, so the critic complexity is $\mathcal{O}(MN)$.

Finally, the overall complexity of our algorithm is $\mathcal{O}((MN+Ml_1)l_2)+\mathcal{O}(M^2Nl_1+MNl_2+MN^2)+\mathcal{O}((Ml_1+l_2)(N+M)) + \mathcal{O}(MN) = \mathcal{O}(M^2Nl_1 + MNl_2 + MN^2 + Ml_1l_2)$.

TABLE III: Parameter Settings for Simulations

Parameters	Value
Number of UEs M	10, 20, **30**, 40, 50, 60
Number of UAVs N	1,2, **3**, 4, 5, 6
Minimum height of UAVs Z_{min}	50m
Maximum height of UAVs Z_{max}	100m
Maximum horizontal distance L^h_{max}	49m
Maximum vertical distance L^v_{max}	12m
Minimum distance of UAVs D_{min}	50m
Maximum transmit power of UAVs P^{UAV}_{max}	5W
Maximum transmit power of UEs P^{UE}_{max}	1W
Computation resource of UAVs F_n	[10,20] Gigacycles
Computation resource of UEs F^0_m	1.5 Gigacycles
Data size of tasks $u_m(t)$	[1,3] MB
Computing workload of tasks $c_m(t)$	[300,500] Megacycles
Allowed delay threshold $v_m(t)$	[250,300]ms
Channel Bandwidth of G2A B^G	20MHz
Channel Bandwidth of A2A B^A	40MHz
Channel Bandwidth of A2G B^E	10MHz
Effective switched capacitance κ	10^{-28}
Noise power N_g,N_A,N_E	-100dBm
Constant values a,b	9.61, 0.16
Excessive path loss η_{LoS}, η_{NLoS}	1, 20
Actor learning rate (γ_1)	10^{-2}, 10^{-3}, 10^{-4}
Critic learning rate (γ_2)	10^{-3}, 10^{-2}, 10^{-4}
Representation model learning rate (γ_3)	10^{-2}, 10^{-3}, 10^{-4}
Penalty for actions unsatisfy constraints ($-Pu$)	-1000

VI. Performance Evaluation

In this section, we describe the experimental setup and introduce the alternative solutions used for comparison-based assessment. Then the experimental results and related analysis are presented to validate the performance of the proposed algorithm.

A. Experimental Setup

We consider a UAVs-assisted MEC scenario with 30 UEs randomly distributed in an area of $1000 \times 1000m^2$ as set in [37]. Three UAVs with random initial positions can help UEs to complete their computing tasks. For UE tasks, the data size $u_m(t)$ is set from 1 to 3 MB [37] and the computing workload $c_m(t)$ is generated randomly within [300,500] Megacycles [38]. For the UAVs, the computing capability F^n is set from 10 to 20 Gigacycles [16]. According to [17], we set the minimum height for UAVs Z_{min} to 50m, maximum height Z_{max} to 100m, maximum horizontal distance L^h_{max} to 49m, maximum vertical distance L^v_{max} to 12m, the maximum transmit power P^{UAV}_{max} to 5W, effective switched capacitance κ to 10^{-28}, and noise power N_g,N_A,N_E to -100dBm. Constant values and excessive path loss a, b, η_{LoS}, and η_{NLoS} are set to 9.61, 0.16, 1, and 20, respectively [37]. The channel bandwidth values for G2A, A2A, and G2A communications are set to 20MHz, 40MHz, and 10MHz, respectively [15][38]. The actor learning rate γ_1, critic learning rate γ_2 and representation model learning rate γ_3 are set based on [32]. Table III presents the values of system parameters, the numbers in bold are the default values.

B. Alternative Solutions

CLP, our proposed algorithm, is compared with the following four alternative algorithms.

- Optimization of Single UAV (OSU) [39]: This solution studies the task offloading problem in a single UAV scenario, which takes the energy consumption as a constraint and task delay as the optimization objective. It employs an algorithm based on deep deterministic policy gradient (DDPG) to search for near-optimal solutions in highly dynamic environments.
- No cooperation between UAVs (NCO) [9]: This solution also considers a single UAV scenario and therefore there is no cooperation between multiple UAVs. Its optimization objective considers task delay, energy consumption and number of tasks collected by the UAV. The proposed solution is based on the multi-task multi-objective proximal policy optimization (PPO) algorithm.
- Cooperation without long-term optimization (CNL) [40]: This solution involves some cooperation between UAVs. Its authors decomposed the UAV-assisted MEC problem into three subproblems and proposed a greedy approximation algorithm as a solution. Rather than optimizing the long-term average system performance, this solution only focuses on achieving the optimal performance in the current time slot.
- Cooperation with multi-agent reinforcement (CMA) [17]: This solution employs a partial task offloading strategy which considers cooperation between UAVs and optimization of long-term performance. A multi-agent TD3 algorithm is designed to find the efficient UAVs' movements, task offloading allocation, and communication resource management based on dynamic MEC environments. In order to accommodate binary computing offloads, the node with the largest offload proportion to offload is chosen.

C. Experiment Results

We show the convergence of our proposed CLP algorithm with different learning rates in Fig. 3. Different learning rates lead to different training performance results. When learning rates are very large (i.e. 10^{-2}), there are great fluctuations in the process of model convergence. Additionally, the convergence points are also often local optimal solutions. When learning rates are very small (i.e. 10^{-4}), the convergence state is stable, but the convergence is slow, taking about 1500 episodes. When learning rates are set to 10^{-3}, the model converges quickly (almost 600 episodes) and has a relatively stable convergence state. Therefore, the learning rates are set to $\gamma_1 = 10^{-3}, \gamma_2 = 10^{-3}, \gamma_3 = 10^{-3}$ in our model training.

To show the effectiveness of the hybrid action representation method in CLP, we perform ablation experiments. The representation method is to transform discrete variables in action space into continuous values to improve the training performance of the model. Considering that discrete variables and continuous variables in the action space are interrelated,

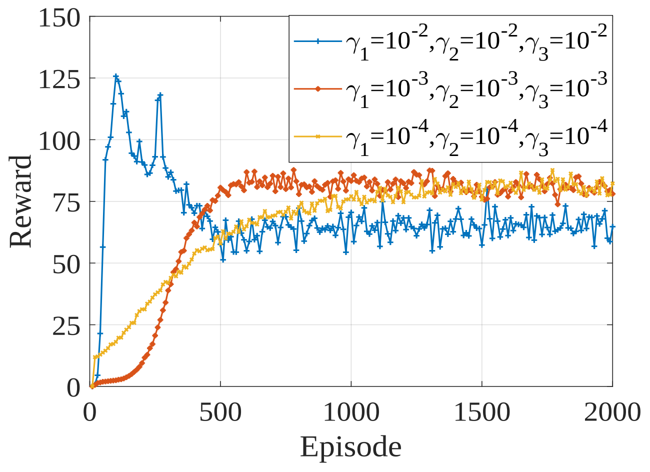


Fig. 3: Convergence

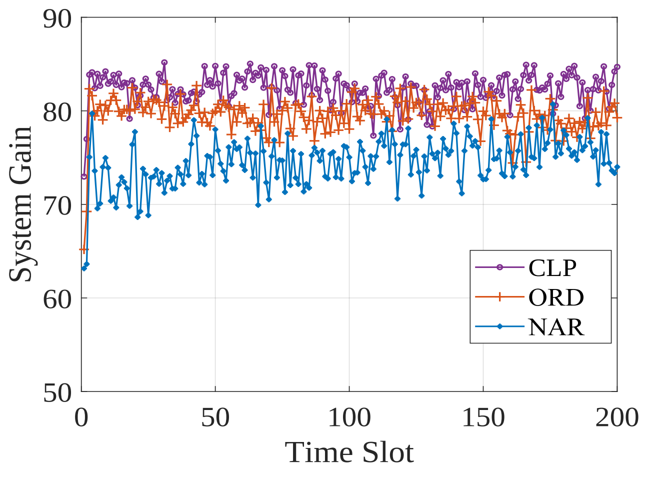


Fig. 4: Ablation experiment

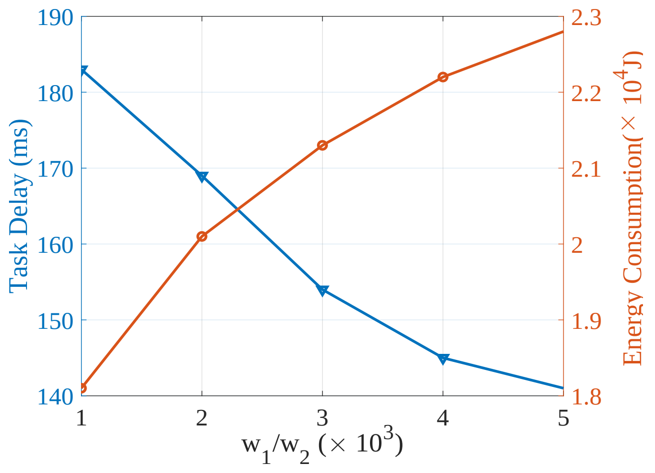


Fig. 5: The effect of weight

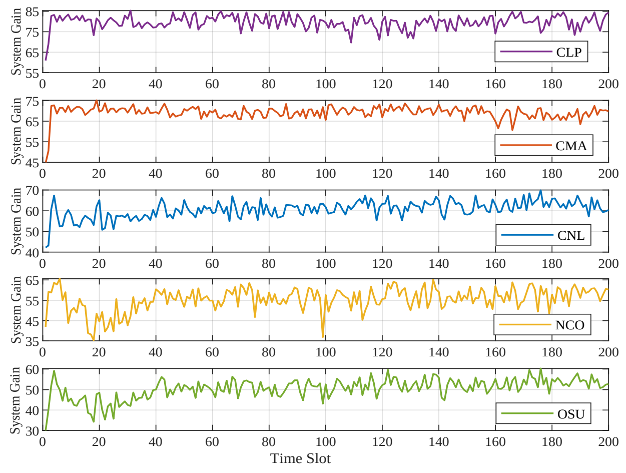
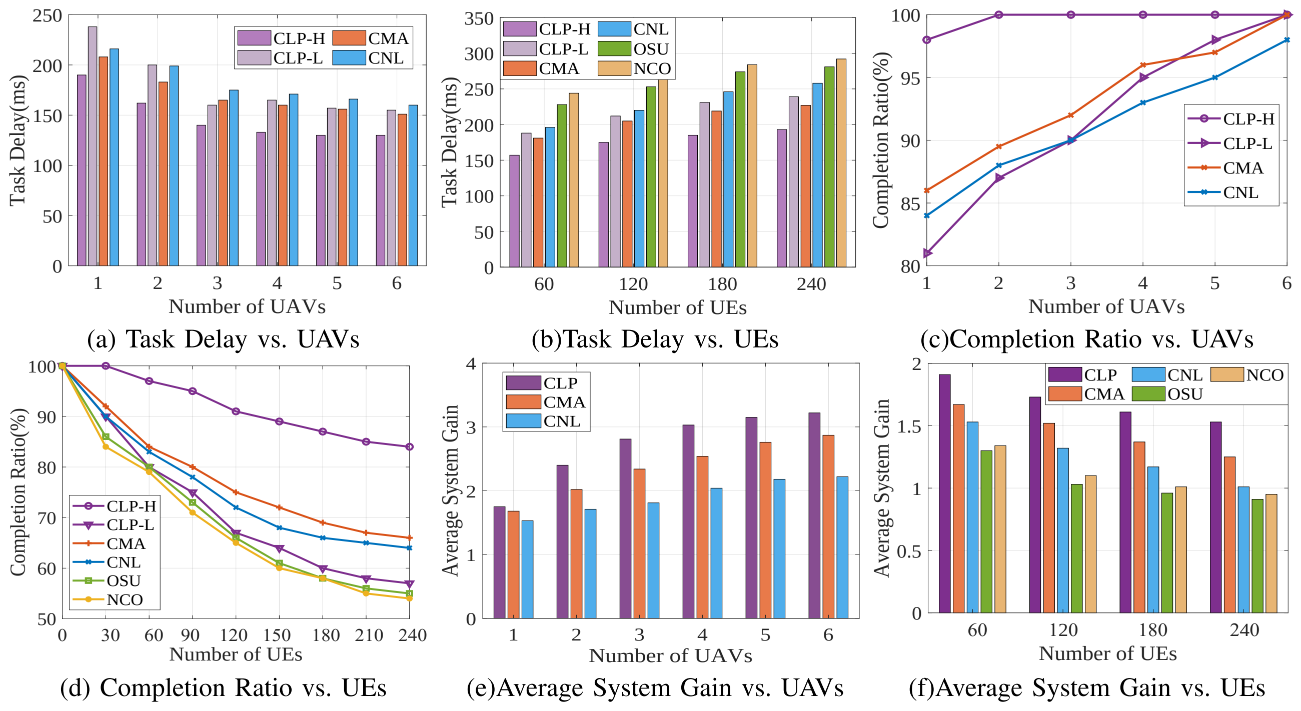


Fig. 6: System gain

the hybrid action representation method in CLP jointly trains the whole action space. Therefore, we use two comparison methods in our ablation experiments. Comparison one employs a no action representation (NAR) method. NAR only discretizes the variables directly by rounding, without any action representation algorithm. Comparison two uses an ORD method, which only represents discrete variables. The method ignores the correlation between discrete variables and continuous variables in the action space, and only represents discrete variables instead of the whole action space. Considering that the goal of the optimization problem is to maximize average system gain, we show the system gain in each time slot for the three algorithms in Fig. 4. We note that NAR has the worst performance and greatest fluctuations, as the crude approximation method leads to a degradation in model performance. ORD only represents discrete variables and ignores the correlation between discrete variables and continuous variables in the action space, so it performs better than NAR, but not as well as CLP. CLP represents the whole action space and has the best performance in terms of system gain from the three methods.

Fig. 5 shows the impact of weight parameters. The optimization objective of our problem is the combination of task delay and system energy consumption by weight parameters. When w_1/w_2 is larger, task delay accounts for more weight and becomes more important. Accordingly, the task delay is reduced but the system energy consumption is increased. When w_1/w_2 is smaller, system energy consumption is more important. Our solution tends to sacrifice the task delay to obtain smaller system energy consumption. In practice, the weight parameters can be adjusted according to the system requirements.

The system gains of the five algorithms in the experiment are illustrated in Fig. 6. The subfigures of Fig. 6 show that the average system gain of CLP in 200 time slots is around 78, CMA's is around 70, CNL's is about 64, NCO's is around 58, and OSU's is approximately 54. Our optimization goal is to maximize the long-term average system gain. The average system gain of CLP is the largest of the five algorithms, demonstrating that our algorithm CLP has the best performance. As OSU is a task offloading algorithm in a single UAV scenario and its goal is to optimize task delay only, it has the worst system gain of all tested solutions. NCO also lacks the cooperation between multiple UAVs, but optimizes both



(a) Task Delay vs. UAVs (b)Task Delay vs. UEs (c)Completion Ratio vs. UAVs

(d) Completion Ratio vs. UEs (e)Average System Gain vs. UAVs (f)Average System Gain vs. UEs

Fig. 7: The effect of UAVs and UEs

task delay and energy consumption, so it has better performance than OSU. CNL considers the multi-UAV cooperative scenario, but only optimizes the current time decision, which easily leads to finding local optimal solutions only. CMA takes both the cooperation between UAVs and long-term average optimization into account, and has the best performance among the alternative solutions. Unfortunately, some performance is lost when converting partial offloading to binary offloading, so CMA is slightly worse than CLP.

Fig. 7 presents the effects of variations in the numbers of UAVs and UEs. Considering that tasks with different priorities have different performance in our algorithm, we will analyze separately the high-priority tasks in CLP (CLP-H) and low-priority tasks in CLP (CLP-L). Fig. 7(a) and Fig. 7(b) show the impact of the number of UAVs on the task delay. In general, as the number of UAVs increases, the task delay gradually decreases. More UAVs means more edge computation resources, and consequently more tasks can be completed on UAVs, which determines a reduction in task delay. It is worth noting that CLP-L performs the worst among all algorithms when the number of UAVs is 1. When there is a single UAV, the available computing resources are very limited. To ensure the completion of high priority tasks, CLP allocates most resources to high-priority tasks which leads to the best performance when completing these tasks. Unfortunately, low-priority tasks cannot be allocated sufficient computing resources, so the task delay associated with these tasks is the highest. As the number of UAVs increases, so do computing resources. Although CLP still allocates most resources to high-priority tasks, lower-priority tasks can also receive more resources. Therefore, the task delay of CLP-L gradually approaches the values experienced by other algorithms.

To show the effect of the number of UEs, we set different numbers of users in the experiment, with a maximum value of 240. Fig. 7(b) shows that the increase in the number of UEs leads to an increase in task delay. More UEs imply more tasks, but due to the limited computing resources of UAVs, some tasks must be offloaded to the remote cloud server, which determines longer task delay. It is worth noting that OSU focuses on the optimization of task delay, while NCO optimizes both task delay and energy consumption, so OSU performs better than NCO in terms of task delay, but worse in terms of system gain. Additionally, OSU and NCO only consider the scenario with a single UAV, so they cannot be compared against when analyzing the impact of the number of UAVs.

We define the completion rate as the number of tasks completed within the allowed delay threshold divided by the total number of tasks. A similar metric is the task completion rate, as shown in Fig. 7(c) and Fig. 7(d). The increase in the number of UAVs improves the completion rate, while the increase in the number of UEs decreases the completion rate. However, as the number of users continues to increase, the completion rate will also level off. The reason is that limited resources of UAVs are difficult to meet the needs of a large number of users. As the number of users increases, most computing tasks will be offloaded to cloud servers and the completion rate will be stable. Note that the high-priority tasks in our algorithm benefit in terms of performance in both

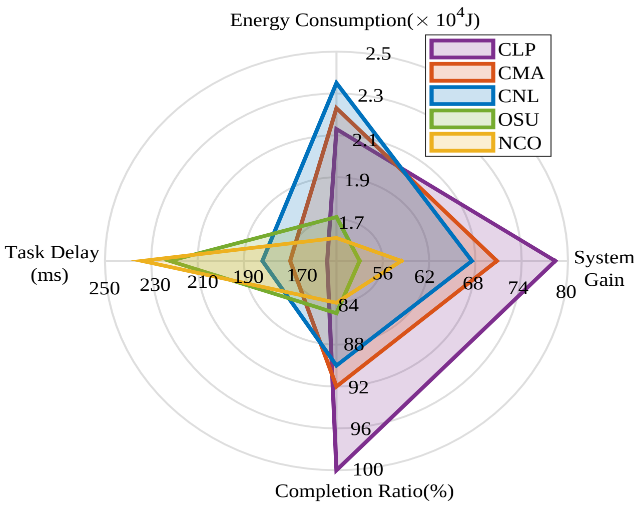


Fig. 8: Performance in terms of four metrics

task delay and completion rate, while low-priority tasks in our algorithm perform worse in many cases. This is because we set different reward functions for different priority tasks, and the CLP algorithm is more inclined to complete the high-priority tasks first, which sacrifices the performance of low-priority tasks. However, the alternative solutions have no priority consideration, and there is no difference in task performance. Considering that the system gain is the sum of all UEs $\sum_{m=1}^{M} F_m(t)$, which is related to the number of UEs, we use the average performance $\sum_{m=1}^{M} F_m(t)/M$ to show the impact of the number of UAVs and UEs, as shown in Fig. 7(e) and Fig. 7(f), respectively. Similar, the increase of UAVs will improve average system gain and the increase of UEs will decrease average system gain.

Fig. 8 compares our CLP with the four alternative solutions in terms of four performance indicators. We find that our proposed CLP algorithm has obvious advantages in terms of task delay, completion rate and system gain. In terms of energy consumption, CLP is better than CMA and CNL, but is inferior to NCO and OSU. Note that NCO and OSU consider the scenario with only one UAV, so they have the lowest energy consumption. This also causes the task delay of NCO and OSU to be far inferior to that of the other algorithms. OSU focuses on optimizing task delay, so it performs better than NCO in terms of task delay and completion ratio, but worse in terms of system gain and energy consumption. The performance of CMA is better than that of CNL due to its long-term average optimization. Our CLP considers task priority, long-term average optimization and binary optimization, which leads to the maximum system gain, and is an excellent result.

VII. Conclusions

In this paper, we focused on the UAV-assisted task offloading problem with task priority. A long-term average problem with the collaboration between multiple UAVs was formulated to optimize task delay and energy consumption by jointly designing the UAV trajectories, task offloading, computation resources allocation, and communication resource management. To solve this problem, we transformed it to a MDP. Considering a discrete-continuous hybrid action space, the Cooperative Long-term average oPtimization (CLP), a novel DRL algorithm was proposed. Following detailed experimental testing, our algorithm CLP outperforms four state-of-the-art optimization approaches in terms of task delay, system gain and system energy consumption.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (NSFC) via grant 62225105, the National Natural Science Foundation of Shandong Province under grant ZR2022QF040 and the QLU Talent Research Project under grant 2023RCKY138. G.-M. Muntean wishes to acknowledge the support of the Science Foundation Ireland (SFI) grant 12/RC/2289_P2 (INSIGHT).

References

[1] Y. Mao, C. You, J. Zhang, K. Huang and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322-2358, 2017, doi: 10.1109/COMST.2017.2745201.

[2] Q. Hu, Y. Cai, G. Yu, Z. Qin, M. Zhao and G. Y. Li, "Joint Offloading and Trajectory Design for UAV-Enabled Mobile Edge Computing Systems," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1879-1892, 2019, doi: 10.1109/JIOT.2018.2878876.

[3] H. Guo, J. Li, J. Liu, N. Tian and N. Kato, "A Survey on Space-Air-Ground-Sea Integrated Network Security in 6G," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 1, pp. 53-87, 2022, doi: 10.1109/COMST.2021.3131332.

[4] G. Yang, Y. -C. Liang, R. Zhang and Y. Pei, "Modulation in the Air: Backscatter Communication Over Ambient OFDM Carrier," *IEEE Transactions on Communications*, vol. 66, no. 3, pp. 1219-1233, 2018, doi: 10.1109/TCOMM.2017.2772261.

[5] H. Xiao, C. Xu, Y. Ma, S. Yang, L. Zhong and G. -M. Muntean, "Edge Intelligence: A Computational Task Offloading Scheme for Dependent IoT Application," *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 7222-7237, 2022, doi: 10.1109/TWC.2022.3156905.

[6] F. Zhou, Y. Wu, R. Q. Hu and Y. Qian, "Computation Rate Maximization in UAV-Enabled Wireless-Powered Mobile-Edge Computing Systems," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 1927-1941, 2018, doi: 10.1109/JSAC.2018.2864426.

[7] Z. Yu, Y. Gong, S. Gong and Y. Guo, "Joint Task Offloading and Resource Allocation in UAV-Enabled Mobile Edge Computing," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3147-3159, 2020, doi: 10.1109/JIOT.2020.2965898.

[8] Z. Ning et al., "5G-Enabled UAV-to-Community Offloading: Joint Trajectory Design and Task Scheduling," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 11, pp. 3306-3320, 2021, doi: 10.1109/JSAC.2021.3088663.

[9] F. Song et al., "Evolutionary Multi-Objective Reinforcement Learning Based Trajectory Control and Task Offloading in UAV-Assisted Mobile Edge Computing," *IEEE Transactions on Mobile Computing*, early access, 2023, doi: 10.1109/TMC.2022.3208457.

[10] B. Xu, Z. Kuang, J. Gao, L. Zhao and C. Wu, "Joint Offloading Decision and Trajectory Design for UAV-Enabled Edge Computing with Task Dependency," *IEEE Transactions on Wireless Communications*, early access, 2023, doi: 10.1109/TWC.2022.3231408.

[11] W. Ma, X. Liu and L. Mashayekhy, "A Strategic Game for Task Offloading among Capacitated UAV-Mounted Cloudlets," in *Proc. of ICIOT*, 2019, doi: 10.1109/ICIOT.2019.00022.

[12] C. -Y. Hsieh, Y. Ren and J. -C. Chen, "Edge-Cloud Offloading: Knapsack Potential Game in 5G Multi-Access Edge Computing," *IEEE Transactions on Wireless Communications*, 2023, doi: 10.1109/TWC.2023.3248270.

[13] Z. Ning et al., ”Partial Computation Offloading and Adaptive Task Scheduling for 5G-Enabled Vehicular Networks,” *IEEE Transactions on Mobile Computing*, vol. 21, no. 4, pp. 1319-1333, 2022, doi: 10.1109/TMC.2020.3025116.

[14] 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 Journal on Selected Areas in Communications*, vol. 41, no. 2, pp. 349-365, 2023, doi: 10.1109/JSAC.2022.3228605.

[15] S. Goudarzi, S. A. Soleymani, W. Wang and P. Xiao, ”UAV-Enabled Mobile Edge Computing for Resource Allocation Using Cooperative Evolutionary Computation,” *IEEE Transactions on Aerospace and Electronic Systems*, early access, 2023, doi: 10.1109/TAES.2023.3251967.

[16] Z. Bai, Y. Lin, Y. Cao and W. Wang, ”Delay-Aware Cooperative Task Offloading for Multi-UAV Enabled Edge-Cloud Computing,” *IEEE Transactions on Mobile Computing*, early access, 2023, doi: 10.1109/TMC.2022.3232375.

[17] N. Zhao, Z. Ye, Y. Pei, Y. -C. Liang and D. Niyato, ”Multi-Agent Deep Reinforcement Learning for Task Offloading in UAV-Assisted Mobile Edge Computing,” *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 6949-6960, 2022, doi: 10.1109/TWC.2022.3153316.

[18] R. Zhou, X. Wu, H. Tan and R. Zhang, ”Two Time-Scale Joint Service Caching and Task Offloading for UAV-assisted Mobile Edge Computing,” in *Proc. of INFOCOM*, 2022, doi: 10.1109/INFOCOM48880.2022.9796714.

[19] W. Chen, Z. Su, Q. Xu, T. H. Luan and R. Li, ”VFC-Based Cooperative UAV Computation Task Offloading for Post-disaster Rescue,” in *Proc. of INFOCOM*, 2020, doi: 10.1109/INFOCOM41043.2020.9155397.

[20] S. Tong, Y. Liu, J. Mišić, X. Chang, Z. Zhang and C. Wang, ”Joint Task Offloading and Resource Allocation for Fog-Based Intelligent Transportation Systems: A UAV-Enabled Multi-Hop Collaboration Paradigm,” *IEEE Transactions on Intelligent Transportation Systems*, early access, 2023, doi: 10.1109/TITS.2022.3163804.

[21] M. Adhikari, M. Mukherjee and S. N. Srirama, ”DPTO: A Deadline and Priority-Aware Task Offloading in Fog Computing Framework Leveraging Multilevel Feedback Queueing,” *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5773-5782, 2020, doi: 10.1109/JIOT.2019.2946426.

[22] J. Shi, J. Du, J. Wang, J. Wang and J. Yuan, ”Priority-Aware Task Offloading in Vehicular Fog Computing Based on Deep Reinforcement Learning,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16067-16081, 2020, doi: 10.1109/TVT.2020.3041929.

[23] W. Zhou et al., ”Priority-Aware Resource Scheduling for UAV-Mounted Mobile Edge Computing Networks,” *IEEE Transactions on Vehicular Technology*, vol. 72, no. 7, pp. 9682-9687, 2023, doi: 10.1109/TVT.2023.3247431.

[24] J. Tian, D. Wang, H. Zhang and D. Wu, ”Service Satisfaction-Oriented Task Offloading and UAV Scheduling in UAV-Enabled MEC Networks,” *IEEE Transactions on Wireless Communications*, early access, 2023, doi: 10.1109/TWC.2023.3267330.

[25] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam and L. Hanzo, ”Multi-Agent Deep Reinforcement Learning-Based Trajectory Planning for Multi-UAV Assisted Mobile Edge Computing,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 73-84, 2021, doi: 10.1109/TCCN.2020.3027695.

[26] X. Zhang, J. Zhang, J. Xiong, L. Zhou and J. Wei, ”Energy-Efficient Multi-UAV-Enabled Multiaccess Edge Computing Incorporating NOMA,” *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5613-5627, 2020, doi: 10.1109/JIOT.2020.2980035.

[27] M. H. Kumar, S. Sharma, K. Deka and M. Thottappan, ”Reconfigurable Intelligent Surfaces Assisted Hybrid NOMA System,” *IEEE Communications Letters*, vol. 27, no. 1, pp. 357-361, 2023, doi: 10.1109/LCOMM.2022.3211292.

[28] Y. Zhou et al., ”Secure Communications for UAV-Enabled Mobile Edge Computing Systems,” *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 376-388, 2020, doi: 10.1109/TCOMM.2019.2947921.

[29] H. Hao, C. Xu, W. Zhang, S. Yang and G. -M. Muntean, ”Computing Offloading with Fairness Guarantee: A Deep Reinforcement Learning Method,” *IEEE Transactions on Circuits and Systems for Video Technology*, early access, 2023, doi: 10.1109/TCSVT.2023.3255229.

[30] J. Zhao, Q. Li, Y. Gong and K. Zhang, ”Computation Offloading and Resource Allocation For Cloud Assisted Mobile Edge Computing in Vehicular Networks,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944-7956, 2019, doi: 10.1109/TVT.2019.2917890.

[31] H. Hao, C. Xu, L. Zhong and G.B. Muntean , ”A Multi-Update Deep Reinforcement Learning Algorithm for Edge Computing Service Offloading,” in *Proceedings of ACM MM*, 2020, doi: 10.1145/3394171.3413702.

[32] B. Li, H. Tang, Y. Zheng, et al., ”HyAR: Addressing Discrete-Continuous Action Reinforcement Learning via Hybrid Action Representation,” *ArXiv Preprint*, 2022, doi: 10.48550/arXiv.2109.05490.

[33] D. P. Kingma and M. Welling. ”Auto-encoding Variational Bayes,” in *ArXiv Preprint*, 2014, doi: 10.48550/arXiv.1312.6114.

[34] S. Fujimoto, H. v. Hoof, and D. Meger. ”Addressing Function Approximation Error in Actor-Critic Methods,” *ArXiv Preprint*, pp. 1582–1591, 2018, doi: arXiv.1802.09477.

[35] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller. ”Deterministic Policy Gradient Algorithms,” in *Proc. of PMLR*, pp. 387–395, 2014.

[36] M. Sipper, ”A serial complexity measure of neural networks,” in *Proc. of ICNN*, pp. 962-966 vol.2, 1993, doi: 10.1109/ICNN.1993.298687.

[37] H. Guo, Y. Wang, J. Liu and C. Liu, ”Multi-UAV Cooperative Task Offloading and Resource Allocation in 5G Advanced and Beyond,” *IEEE Transactions on Wireless Communications*, early access, 2023, doi: 10.1109/TWC.2023.3277801.

[38] X. Dai, Z. Xiao, H. Jiang and J. C. S. Lui, ”UAV-Assisted Task Offloading in Vehicular Edge Computing Networks,” *IEEE Transactions on Mobile Computing*, early access, 2023, doi: 10.1109/TMC.2023.3259394.

[39] H. Wang, H. Zhang, X. Liu, K. Long and A. Nallanathan, ”Joint UAV Placement Optimization, Resource Allocation, and Computation Offloading for THz Band: A DRL Approach,” *IEEE Transactions on Wireless Communications*, vol. 22, no. 7, pp. 4890-4900, 2023, doi: 10.1109/TWC.2022.3230407.

[40] L. Zhang and N. Ansari, ”Latency-Aware IoT Service Provisioning in UAV-Aided Mobile-Edge Computing Networks,” *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10573-10580, 2020, doi: 10.1109/JIOT.2020.3005117.

BIOGRAPHIES

Hao Hao received the Ph. D degree in computer science and technology from Beijing University of Posts and Telecommunications, Beijing, China,in 2021. He is currently a lecturer with the Shandong Computer Science Center (National Supercomputing Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences). His research interests include MEC and content caching over the wireless network, Multimedia Communications.

Changqiao Xu (SM'15) received the Ph.D. degree from the Institute of Software, Chinese Academy of Sciences (ISCAS) in Jan. 2009. He was an Assistant Research Fellow and R&D Project Manager in ISCAS from 2002 to 2007. He was a researcher at Athlone Institute of Technology and joint PhD at Dublin City University, Ireland during 2007-2009. He joined Beijing University of Posts and Telecommunications (BUPT), China, in Dec. 2009. Currently, he is a Full Professor with the State Key Laboratory of Networking and Switching Technology, and Director of the Next Generation Internet Technology Research Center at BUPT. His research interests include Future Internet Technology, Mobile Networking, Multimedia Communications, and Network Security. He has published over 200 technical papers in prestigious international journals and conferences, including IEEE Comm. Surveys & Tutorials, IEEE Wireless Comm., IEEE Comm. Magazine, IEEE/ACM ToN, etc. He has served many international conferences and workshops as Co-Chair or Technical Program Committee member. He is currently serving as the Editor-in-Chief of Transactions on Emerging Telecommunications Technologies (Wiley).

Wei Zhang received the B.E. degree from Zhejiang University in 2004, the M.S. degree from Liaoning University in 2008, and the Ph.D. degree from Shandong University of Science and Technology in 2018. He is currently a Professor with the Shandong Computer Science Center (National Supercomputing Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences). His research interests include future generation network architectures, edge computing and edge intelligence.

Shujie Yang received the Ph.D. degree from the Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing, China, in 2017, where he is currently a Lecturer with the State Key Laboratory of Networking and Switching Technology. His major research interests are in the areas of wireless communications and wireless networking.

Gabriel-Miro Muntean (F'23) is Professor with the School of Electronic Engineering, Dublin City University (DCU), Ireland, and co-Director of the DCU Performance Engineering Laboratory. He has published over 500 papers in top-level international journals and conferences, authored 4 books and 28 book chapters, and edited 6 additional books. He has supervised to completion 26 PhD students and has mentored 20 post-doctoral researchers and fellows. His research interests include quality, performance, and energy saving issues related to rich media content delivery, technology enhanced learning, and other data communications over heterogeneous networks. He is an Associate Editor of the IEEE TRANSACTIONS ON BROADCASTING, Multimedia Communications Area Editor of the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, and chair and reviewer for important international journals, conferences, and funding agencies. He was Project Coordinator and DCU team leader for the EU projects NEWTON and TRACTION.