Graph Convolutional Reinforcement Learning-Guided Joint Trajectory Optimization and Task Offloading for Aerial Edge Computing

Huaming Wu, *Senior Member, IEEE*, Lei Tian, Huijun Tang, *Member, IEEE*, Ruidong Li, *Senior Member, IEEE*, and Pengfei Jiao, *Member, IEEE*

***Abstract*— The unique capabilities of Unmanned Aerial Vehicles (UAVs), including their superior mobility, flexibility, and line-of-sight transmission, have made them well-suited for facilitating Aerial Edge Computing (AEC). This computing paradigm is particularly beneficial for meeting the computing demands of User Equipments (UEs) in emergency situations, as it offers efficient support for task offloading. Considering the service requirements of UEs, it is essential to minimize the processing delay experienced by UEs in AEC systems. This is accomplished through the joint optimization of the UAV trajectory, flight speed, and task offloading ratio allocation for UEs. Due to the non-convex nature and the continuous action space of the problem, recent studies have turned to the Deep Deterministic Policy Gradient (DDPG) to tackle similar challenges. However, Deep Neural Networks (DNNs) employed in DDPG are limited to extracting latent information solely from Euclidean data, and are similarly constrained by the highly dynamic changes in channel states within AEC networks, thereby disregarding the valuable features inherent in the structural information. In order to alleviate the task offloading problem in AEC systems, we propose a novel Graph Convolutional Pooling-DDPG (GCP-DDPG) algorithm by exploiting the graph-based multi-relational derivation capability of the multi-Relational Graph Convolutional Network (R-GCN) and employing the reinforcement learning technique. Extensive simulation experiments are conducted to evaluate the superiority and effectiveness of the GCP-DDPG algorithm. The results demonstrate a remarkable performance improvement of 34.6% compared to state-of-the-art approaches.**

***Index Terms*— Aerial edge computing, unmanned aerial vehicles, Internet of Things, graph neural networks, deep reinforcement learning.**

I. Introduction

With the rapid advancement of wireless communication networks and intelligent Internet of Things (IoT) technologies, it has gained increasingly popular to deploy compute-intensive and delay-sensitive applications on User Equipments (UEs) [1], e.g., online games [2], autonomous driving [3], [4] and telemedicine [5]. Meanwhile, a substantial multitude of mobile devices are interconnected to wireless networks, heavily relying on online resources to carry out diverse tasks. These scenarios require more computing power and longer battery life, which are seriously insufficient in current smart devices.

Mobile Cloud Computing (MCC) has attracted widespread attention as a promising approach to augment the computing and storage capabilities of UEs, which redeploys computing and storage resources on cloud servers, and transmits computing results through downlinks. The ensuing problem is that the cloud server is far away from UEs, which requires numerous intermediate nodes to ensure the timeliness of traffic. To alleviate the backhaul link latency, Mobile Edge Computing (MEC) has emerged as a promising method in providing quick and efficient computing services to UEs [6], [7]. Supported by the considerable geographical data acquisition ability of IoTs, Mobile Edge Networks (MENs) provide spatial locations and services to mobile users. Moreover, MEC enables various compute-intensive and delay-sensitive applications to run on multiple MEC servers without congestion, and supports real-time data computing and analysis. Consequently, tasks can be performed on MEC servers to enhance the Quality of Service (QoS) [8].

While MEC presents numerous advantages, its effectiveness may be constrained by the immobility of towers [9], where IoT devices are deployed in unattended or challenging environments such as forests, mountains, underwater locations, as well as temporary hotspot areas or emergency situations [10]. In such scenarios, relying solely on MEC may prove inadequate for fulfilling requirements and delivering optimal services to UEs. To tackle this challenge, researchers have extensively investigated Unmanned Aerial Vehicle (UAV)-assisted MEC, also referred to as Aerial Edge Computing (AEC), which can fully exploit the benefits of the synergy within the air-space-ground integrated network in the era of super 5G/6G. This computing paradigm is capable of satisfying the escalating demand for accelerated and highly dependable wireless connectivity, while also offering support for emerging technologies in the domain of AEC Compared to traditional fixed-position MEC, this approach capitalizes

Received 24 April 2024; revised 28 August 2024 and 3 October 2024; accepted 30 October 2024. Date of publication 14 November 2024; date of current version 21 October 2025. This work was supported in part by the National Key Research and Development Program of China under Grant 31400, in part by the National Natural Science Foundation of China under Grant 62071327 and Grant 62401190, and in part by Tianjin Science and Technology Planning Project under Grant 22ZYYYJC00020. The Associate Editor for this article was I. Ashraf. *(Corresponding author: Huijun Tang.)*

Huaming Wu and Lei Tian are with the Center for Applied Mathematics, Tianjin University, Tianjin 300072, China (e-mail: whming@tju.edu.cn; tianlei@tju.edu.cn).

Huijun Tang and Pengfei Jiao are with the School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, China (e-mail: tanghuijune@hdu.edu.cn; pjiao@hdu.edu.cn).

Ruidong Li is with the Institute of Science and Engineering, Kanazawa University, Kanazawa 920-1192, Japan (e-mail: liruidong@ieee.org).

Digital Object Identifier 10.1109/TITS.2024.3490533

on the fact that UAVs can carry computational resources and have the freedom of flight, utilizing their efficient mobility to provide on-demand communication and real-time computing services for UEs.

Despite their potential benefits, AEC systems still suffer from several challenges related to network deployment and operation [11]. On the one hand, UAVs commonly encounter limitations in terms of their battery energy capacity, and both their flight operations and task offloading activities are characterized by substantial energy consumption [12]. Thus, UAVs need to select UEs for partial task offloading within a limited service time. On the other hand, another challenge in AEC systems is the development of energy-aware UAV/Vehicle trajectory planning, which has been extensively investigated in recent research [13], [14]. Specifically, the challenge lies in real-time control of the trajectory for each UAV in a dynamic environment where both the UAVs and UEs may be obstructed by buildings. Traditional techniques such as exhaustive search are not applicable in this scenario due to the continuous nature of decision variable space [15], such as determining the optimal trajectory and resource allocation. To this end, we consider an energy-constrained AEC system as a potential solution, aiming to tackle the aforementioned challenges by UAVs in conjunction with MEC capabilities. Specifically, UEs can access and offload a portion of their computing tasks to UAVs that are equipped with computing resources, based on their specific service requirements.

So far, many research works have been devoted to addressing the challenges mentioned above in UAV-assisted MEC systems [16]. Deep Reinforcement Learning (DRL) has gained significant popularity in tackling a wide range of problems [17], [18], which can learn and optimize algorithms by interacting with the MEC environment to obtain training data. These DRL algorithms typically employ Deep Neural Networks (DNNs) as feature representations and function approximators [19], e.g., Multilayer perceptron (MLP) [20], Fully Convolutional Network (FCN) [21], which can abstract the features of input data, calculate and the offloading decision of the agent. However, the neural network architectures mentioned above can only extract latent representations from Euclidean data, resulting in the neglect of position information of IoT devices and highly dynamic channel state information in AEC systems. In practice, task offloading and resource allocation problems in AEC systems often involve structured data that can be represented as graphs.

Over the past few years, Graph Neural Networks (GNNs) have achieved significant success and have been successfully applied in various domains, including social network analysis, recommendation systems, knowledge graphs, and bioinformatics [22], [23]. Compared to other neural networks, GNNs exhibit superior performance in learning graph features, demonstrating immense potential for solving task offloading and resource allocation problems in AEC systems. The key idea behind GNNs is to learn node representations by aggregating and propagating information across the graph, where each node updates its representation based on the representations of its neighbors, enabling the capturing of complex relationships among nodes and global topology information [24]. This allows for better utilization of both local structure and global features of the graph, resulting in more accurate modeling and prediction for tasks such as task offloading and resource allocation.

In comparison to other network architectures like Graph Convolutional Networks (GCNs) [25] and Graph Attention Networks (GATs) [26], which handle single-relation graph data, the multi-relational GCN (R-GCN) [27] is designed specifically for graphs with multiple types of relationships, such as entities and relations in knowledge graphs. It achieves this by introducing relationship-specific parameters that capture differences between various relationships, enabling better generalization to unknown graph data. This is particularly relevant in capturing the highly dynamic position information and channel state information of IoT devices in AEC systems. Inspired by the successful and powerful performance of GNNs, we further employ R-GCN to overcome the limitations of DDPG in handling graph data. The proposed Graph Convolutional Pooling-DDPG (GCP-DDPG) algorithm not only uses R-GCN to model AEC scene information, but also preserves the advantages of the DDPG algorithm.

The main contributions of this paper are as follows:

- *Partial Offloading for Energy-Constrained AEC Systems:* We develop an energy-constrained AEC system that enables UEs to partially offload their computational tasks to the UAV equipped with computation resources, based on their service requirements. The UAV plans the flight trajectory according to the environment state, collects task data, processes calculation tasks, and sends the results back to UEs. The approach aims to achieve the most effective reduction in latency for all UEs by jointly optimizing factors such as the task offloading ratio of UEs, and the flight angle and speed of the UAVs.
- *Multi-relational Graph Embedding for AEC Networks:* we model the time-varying channel state and device's attributes in the proposed AEC system as multi-relational graph-structured data, utilize R-GCN to model complex relational interactions in graphs with different edges, and fuse the unstructured features of both the UAV and the users into meaningful embeddings, which integrates the state information of devices and their neighboring devices.
- *GCP-DDPG Offloading Algorithm*: Considering the complexity and dynamics of AEC systems, we model the task offloading and trajectory control problems in the scenario as Markov Decision Process (MDP), and input the Multi-relational graph embeddings into graph pooling layer and DDPG module to learn the optimal offloading ratio and trajectory control strategy, which is called GCP-DDPG algorithm.
- *Effective Performance*: We conduct extensive simulations to verify the robust performance of our proposed GCP-DDPG algorithm. The results consistently demonstrate that GCP-DDPG outperforms the baseline algorithms. This confirms the superiority and robustness of the GCP-DDPG algorithm in optimizing task-offloading processes in AEC systems.

TABLE I
THE QUALITATIVE COMPARISON OF THE CURRENT LITERATURE ON AEC SYSTEMS

Approaches	Trajectory optimization	Continuous flight angles	Speed control	Dynamic Offloading	Partial Offloading	Objectives	Methods
Zhang *et al.* [11]	✓	✗	✓	✗	✓	Energy consumption	Lagrangian duality
Wang *et al.* [28]	✓	✓	✗	✗	✗	Energy consumption	MADDPG
Cheng *et al.* [17]	✗	✗	✗	✓	✗	Delay	DDPG
Li *et al.* [20]	✗	✗	✗	✓	✗	Weighted cost	SAC
Wang *et al.* [29]	✓	✓	✓	✗	✗	Energy consumption	BCD and DDPG
Li *et al.* [36]	✓	✗	✓	✗	✗	Task computation	GNN-A2C
Pamuklu *et al.* [35]	✗	✓	✓	✓	✗	QoS	GNN-DQN
Ours	✓	✓	✓	✓	✓	Delay	proposed GCP-DDPG

The remainder of this paper is outlined as follows. Section II summarizes the related work on AEC systems and GNNs. In Section III, we present the system model and define the problem formulation. The GCP-DDPG-based computation offloading training algorithm is introduced in Section IV, followed by the simulation results and discussions in Section V. Section VI concludes this paper.

II. RELATED WORK

A. AEC Systems

In recent years, AEC has become a research hotspot for many scholars. Table I shows some works in AEC systems. To minimize the overall energy consumption of communication in AEC systems, Zhang et al. [11] proposed an approach that involves optimizing various factors including bit allocation, slot scheduling, power allocation, and UAV trajectory. Wang et al. [28] introduced a framework for AEC that enables the support of multiple UAVs with diverse trajectories, where a multi-agent DRL-based trajectory control algorithm was designed to independently manage the trajectory of each UAV and optimize offloading decision-making. Chen et al. [17] presented an integrated network architecture for air-ground communication, specifically designed for offloading applications with considerations for long-range energy and computational limitations. Additionally, an actor-critic approach was designed to speed up the learning process and enable real-time learning of optimal offloading policies. Li et al. [20] conducted a study on task offloading decisions and resource allocation in an AEC environment involving multiple users and servers, and employed a soft actor-critic algorithm to optimize factors such as latency, energy consumption, and a weighted total cost, aiming to enhance the overall system performance. Wang et al. [29] introduced a novel flight MEC platform and devised a trajectory control algorithm based on DRL, which synergistically optimizes user association, resource allocation, and UAV trajectory within the AEC system.

B. GNN-Based MEC Systems

GNNS have demonstrated significant advantages and potential in processing graph data, which has driven many researchers to actively explore their application in wireless communication.

Chowdhury et al. [30] proposed an approach that utilizes GNNs to parameterize the iterative weighted minimum mean square error method. By leveraging GNNs, they aimed to optimize the power allocation process and achieve improved performance in terms of power efficiency and network capacity. Chen et al. [31] tackled the issue of dependent task offloading in multi-user scenarios in MEC environments. They proposed a GNN approach for Directed Acyclic Graph (DAG)-based multi-dependency task offloading, leveraging the capabilities of GNNs for improved performance and efficiency. Sheng et al. [32] introduced a unified framework that addresses various design problems in wireless networks. The use of GNNs in wireless network design demonstrates near-optimal performance, further emphasizing the efficacy of GNNs as a powerful tool in this domain. Li et al. [33] designed a distributed scheduling algorithm for task offloading in MEC scenarios. This algorithm is based on a multi-agent collaborative DRL approach, utilizing a GAT to capture the spatial location relationship among UEs in the environment, this algorithm effectively reduces average delay and packet drop rate, and improves link utilization, leading to improved performance in task scheduling. Huang and Wang et al. [34] designed a user preference prediction and recommendation model for GNN-based MEC systems called GCN-GAN. The model utilizes the user's browsing history and preferences to prioritize recommending high-quality video to the local MEC server. However, these studies focus solely on optimization within MEC scenarios and do not account for UAV-assisted environments, which are inherently more dynamic and complex compared to the former.

C. GNN-Based AEC Systems

Recent efforts focused on using GNN to extract environmental features, which has proven to be an effective approach that thoroughly captures the complex relationships between various devices in UAV-assisted scenarios and can aid in decision-making. Pamuklu et al. [35] proposed a Graph Reinforcement Learning(GRL) method to optimize QoS by offloading from IoT devices to the UAVs without trajectory optimization of UAVs. Li et al. [36] proposed a GRL method to maximize the total computation of tasks offloaded to the UAV. Different from the previous studies, we are the first to construct graph data based on an AEC scenario to optimize total delay by partial offloading and trajectory optimization, where the UAV and UEs are constructed as nodes, and the channel states of different time slots are constructed as edges with different relationships. The proposed GCP-DDPG completes preliminary

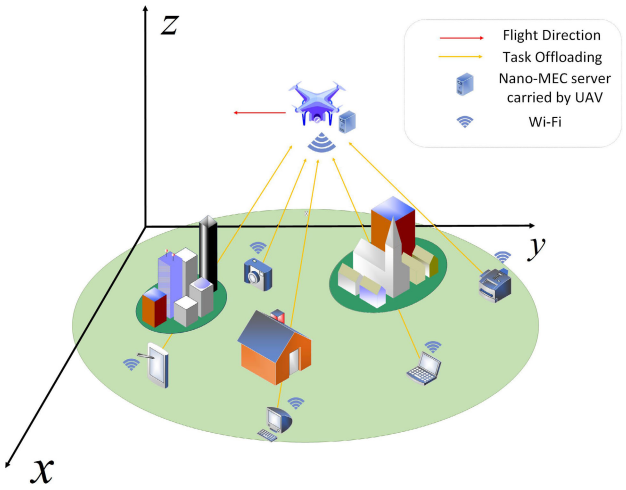


Fig. 1. Architecture of the considered AEC system.

feature learning through FC and employs R-GCN to capture the spatial location features of UEs and the UAV, obtaining graph embeddings by the GAP method. Finally, the encoded state information is inputted into DDPG to learn the optimal real-time task offloading and UAV trajectory control strategy.

III. System Model and Problem Formulation

In this section, we introduce the scenario, communication model, computational model, and performance metrics used to formulate the problem of evaluating the performance of task-offloading processes.

A. AEC System Model

In this paper, we consider an AEC system model in the IoT scenario, as illustrated in Fig. 1. In this scenario, a single UAV and N UEs are deployed in 3-D Cartesian coordinates, and the UAV carries a nano-MEC server that flies at a fixed height to provide computing services for UEs. Considering the limited computational resources of UEs, it is feasible to partially offload compute-intensive and delay-sensitive tasks to the nano-MEC server of the UAV for execution. Meanwhile, the remaining tasks are handled locally by the UEs themselves. To partition each episode L into T discrete time slots of equal length, we denote the set of time slots as $\mathcal{T} \in \{1, 2, \cdots, T\}$. The list of symbols and their definitions is provided in Table II.

At the beginning of each episode, the UE executes a computing task in a predetermined area and sends an offload request to the nano-MEC server installed on the UAV. It is assumed that UEs can communicate with the UAV in real time during the episode. The UAV and UEs jointly determine the task offloading ratio and the UAV trajectory strategy for that time slot. The UAV follows the designed trajectory and reaches the designated location, providing computing services to the UEs for the remaining time slots. The flow for each time slot is depicted in Fig. 2.

B. Communication Model

The positions of the UAV and UEs directly impact the quality of the communication link between them. To express

TABLE II
Symbols and Their Definitions

Symbol	Definition
t	Index of time slot
N	Number of UEs
n	Index of UEs
T	Number of time slots
$\mathcal{T}$	A collection of time slot divisions
$q_n(t)$	Current position coordinates of n-th UE during time slot t
$Q(t)$	Current position coordinates of UAV during time slot t
g_0	Channel gain between receiver and sender at 1 m distance
H	Flying height of UAV
v_{max}	Maximum speed at which the UAV can fly
$R_n(t)$	Uplink rate of n-th UE during time slot t
B	Channel bandwidth
$P_n(t)$	Transmission power of n-th UE to upload task during t
$D_n(t)$	Data size of the task generated by the n-th UE during time slot t
$P_n^{nlos}(t)$	Transmission loss of the n-th UE during time slot t
$T_n^{trans}(t)$	Transmission delay of the n-th UE during time slot t
s	Number of CPU cycles required to process each unit byte
f_{UAV}	Computing frequency of the nano-MEC server
$T_{UAV,n}^{com}(t)$	Computation delay generated by nano-MEC server of n-th UE
$T_{UE,n}^{com}(t)$	Computation delay of the task locally of n-th UE
$o_n(t)$	Task offloading ratio of the n-th UE to nano-MEC server
$f_{UE,n}$	Computing frequency of the n-th UE
$E_{fly}(t)$	Energy consumption of UAV during t for flight-related activities
M_{fly}	Quality of the UAV
k	The switched capacitance
$E_{UAV,n}(t)$	Energy consumption of n-th UE during tasks unloading of UAV
$E_{UAV}(t)$	Total energy consumption of the UAV
$E_r(t)$	Remaining power of the UAV
$S(t)$	State of AEC system
s_t	State embedding of AEC system

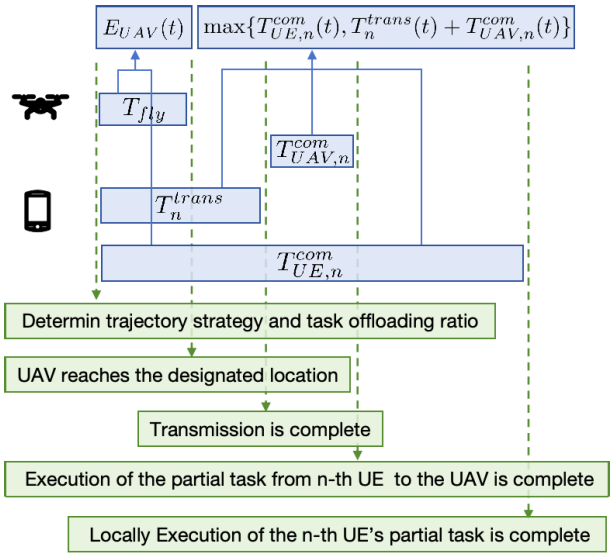


Fig. 2. Schematic diagram of computation offloading time slots in the considered AEC system.

their positions clearly, we establish a 3D Cartesian coordinate system. In the time slot $t \in \mathcal{T}$, the position of the n-th UE can be denoted as $q_n(t) = [x_n(t), y_n(t)]$. Additionally, we assume that the UAV flies at a fixed height H and its coordinates are expressed as $Q(i) = [X_{(t)}, Y_{(t)}]$. The channel gain between the n-th UE and the UAV can be denoted as [37]:

$$g_n(t) = \frac{g_0}{||Q(t+1) - q_n(t)||^2 + H^2}, \tag{1}$$

where $||\cdot||^2$ represents the L2 norm and g_0 represents the channel gain at a distance of 1m between the receiver and the sender. During the UAV flight, the flying speed

$v(t) \in [0, v_{max}]$ and angle $\beta(t) \in [0, 2\pi]$ of the UAV in time slot t are given by the agent, and the next hover position of the UAV can be expressed as:

$$Q(t+1) = [X(t) + v(t)T_{fly}\cos\beta(t), Y(t) + v(t)T_{fly}\sin\beta(t)]^T. \quad (2)$$

Here, we consider the occlusion effects caused by obstacles in the real-world environment. As a result of these occlusions, the uplink rate $R_n(t)$ is calculated by:

$$R_n(t) = B\log_2\left(1 + \frac{P_n(i)g_n(t)}{\sigma^2 + i_n(t)P_n^{nlos}(t)}\right), \quad (3)$$

where B represents the channel bandwidth. The variables $P_n(t)$ and $P_n^{nlos}(t)$ respectively stand for the transmission power and the transmission loss of the n-th UE during task uploading in the time slot t. Moreover, $i_n(t)$ is an indicator variable, which signifies the presence or absence of any blocking between the UAV and the n-th UE during the t-th time slot.

The battery and computing capabilities of the UAV and the nano-MEC server are limited. Therefore, only a portion of the tasks from user equipment can be offloaded and executed on the nano-MEC server. We assume that the n-th UE needs to calculate $D_n(t)$ bits of data in time slot t. Thus, the transmission delay can be mathematically represented as follows:

$$T_n^{trans}(t) = \frac{D_n(t)o_n(t)}{R_n(t)}, \quad (4)$$

where $o_n(t) \in [0, 1]$ denote the task offloading ratio of the n-th UE to the nano-MEC server in the time slot t, while $1 - o_n(t)$ indicates that the rest tasks are executed locally on the n-th UE.

C. Computational Model

In the time slot t, when the n-th UE offloads a task to the nano-MEC server, the resulting computational delay can be calculated using the following formula:

$$T_{UAV,n}^{com}(t) = \frac{D_n(t)o_n(t)s}{f_{UAV}}, \quad (5)$$

where s represents the number of CPU cycles required to process each unit byte and f_{UAV} is denoted as the computing frequency of the nano-MEC server. Correspondingly, the computation latency of a task executed locally is expressed as:

$$T_{UE,n}^{com}(t) = \frac{D_n(t)(1 - o_n(t))s}{f_{UE,n}}, \quad (6)$$

where $f_{UE,n}$ is the computing frequency of the n-th UE.

Correspondingly, the energy consumed by the UAV can be categorized into two parts in a given time slot t, one of which is attributed to the UAV's flight, while the other is due to the computational tasks it performs. Specifically, the energy consumed by the UAV for its flight can be expressed as:

$$E_{fly}(t) = \varphi||V(t)||^2, \quad (7)$$

where $\varphi = 0.5M_{fly}T_{fly}$, M_{fly} denotes the weight of the UAV, and T_{fly} denotes the duration of the UAV's flight within each time slot.

In addition, the energy consumed by the nano-MEC server due to computing tasks in the time slot t is denoted as:

$$E_{UAV,n}(t) = kf_{UAV}^2 D_n(t)o_n(t)s, \quad (8)$$

where k is the switched capacitance [31].

The total energy consumption of the UAV in a given time slot t can be calculated as the sum of the energy consumed for its flight and the energy consumed for computational tasks, expressed as:

$$E_{UAV}(t) = E_{fly}(t) + \sum_{n=1}^{N} E_{UAV,n}(t). \quad (9)$$

D. Problem Formulation

Based on the AEC system described above, we design a joint optimization approach for determining the task offloading ratio of UEs and real-time scheduling of the UAV, aiming to ensure efficient utilization of limited computing resources in the presence of UAV energy constraints. Specifically, the optimization problem aims to minimize the processing delay of tasks for all UEs. The formulation of this optimization problem can be expressed as follows:

$$\min_{Q(t+1), o_n(t))} \sum_{t=1}^{T}\sum_{n=1}^{N} \max\{T_{UE,n}^{com}(t), T_n^{trans}(t) + T_{UAV,n}^{com}(t)\},$$

$$\text{s.t.} \quad \sum_{t=1}^{K} E_{UAV}(t) \le E_{Battery}, \quad (10a)$$

$$0 \le o_n(t) \le 1, \forall t, \forall n, \quad (10b)$$

$$0 \le v(t) \le v_{max}, \forall t, \quad (10c)$$

$$0 \le \beta(t) \le 2\pi, \forall t, \quad (10d)$$

where constraint (10a) guarantees that the total energy consumption of the UAV in all time slots does not exceed the maximum battery capacity $E_{Battery}$. Constraints (10b), (10c), and (10d) restrict the feasible range of task offloading ratio, UAV flight speed, and flight angle, respectively.

IV. GCP-DDPG

We first propose a novel GCP-DDPG scheme that effectively combines DRL and GNNs to obtain a near-optimal task offloading strategy and real-time scheduling strategy of the UAV in the AEC system. The framework is illustrated in Fig. 3. The GCP-DDPG algorithm-based agent observes the state of the AEC system and makes informed decisions. Subsequently, it receives rewards as feedback from the AEC system. Notably, our study distinguishes itself from previous research by introducing a novel approach where the AEC system is represented as multi-relational graph data for the first time. Specifically, we learn the initial features through the fully connected layer (FC) and extract state embeddings S_t by R-GCN and graph pooling modules. After that, we use the policy network to output the real-time action a_t and execute the action in the current state $S(t)$. The action a_t results in the next new state $S(t+1)$.

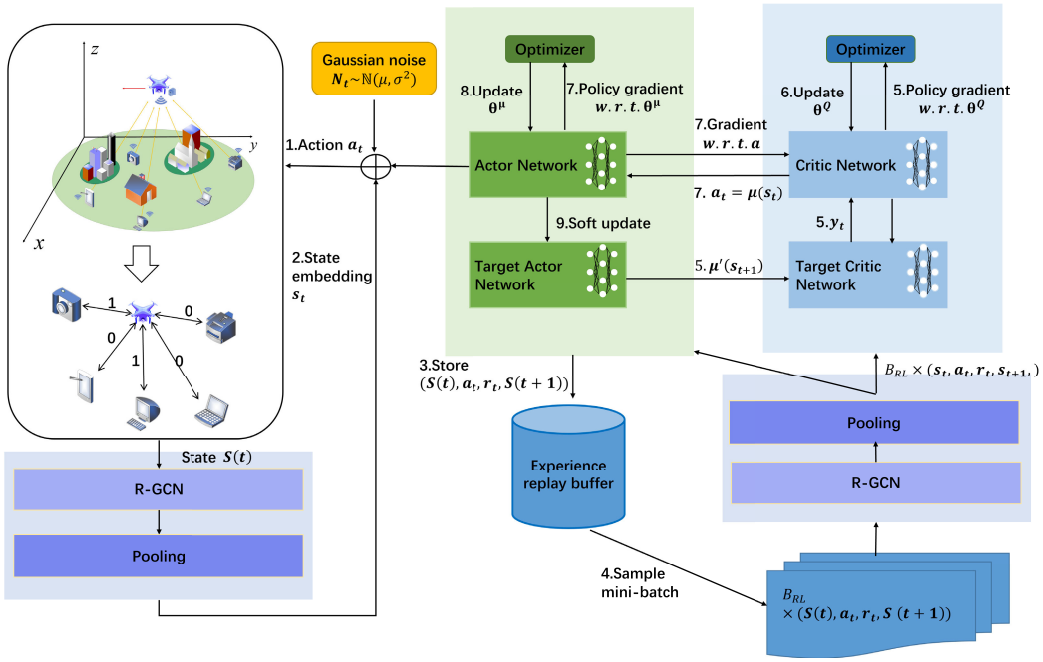


Fig. 3. The overall architecture of the proposed GCP-DDPG algorithm in AEC systems. We fuse features from different devices by R-GCN [27] and Pooling layer as embeddings and input embeddings to DDPG [38] to learn action policies in continuous action space.

A. State Space and State Embedding

1) State Space: We represent UEs and the UAV in AEC systems as graph data. Each node has its feature, which refers to a set of attributes that provides additional information about the node. Feature space refers to the set of all possible features that an environment can be in at any given time. In the graph composed of UEs and the UAV, we treat the potential factors influencing decisions as features of each node. Specifically, the features of user nodes consist of location information, task information, and obstacle occlusion information, while the features of the UAV node include remaining power and location information. We refer to the set of all possible features as the Feature Space. Additionally, since we use this information as state inputs in reinforcement learning, we also refer to this Feature Space as the State Space. The features of UE nodes in time slot t can be expressed as:

$$s_n(t) = [q_n(t), D_n(t), i_n(t)], \tag{11}$$

where $q_n(t)$ indicates the position of the n-th UE in the time slot t, $D_n(t)$ represents the size of the calculation task of the n-th UE in the time slot t, and $i_n(t)$ indicates the indicative function of the obstacle occlusion between the UAV and the n-th UE in the time slot t. If $i_n(t) = 0$, it signifies the absence of transmission loss, and we construct an edge of type 0. Conversely, if $i_n(t) = 1$, it indicates the presence of transmission loss due to obstructing buildings, and we construct an edge of type 1.

Similarly, the features of UAV nodes can be expressed as:

$$s_{UAV}(t) = [E_r(t), Q(t)], \tag{12}$$

where $E_r(t)$ represents the remaining power of the UAV in time slot t and $Q(t)$ indicates the real-time position of the UAV in time slot t. For ease of expression, we denote the state of the AEC system as follows:

$$S(t) = \{s_n(t), \forall n \in [1, N], s_{UAV}(t)\}. \tag{13}$$

2) State Embedding: In this part, we input $S(t)$ composed of UAV and UEs information in time slot t into R-GCN [27] and the Pooling module to obtain embeddings, which transforms entire graphs into low-dimensional vectors. First, we perform preliminary feature extraction on the nodes by applying the full connection (FC) layer.

$$H_n(t) = FC_1(s_n(t)), \tag{14}$$

$$H_{UAV}(t) = FC_2(s_{UAV}(t)), \tag{15}$$

where FC_1 represents FC layer for UEs, and FC_2 represents FC layer for the UAV.

Subsequently, we construct different types of edges to represent the different channel states between the UAV and the UE in each time slot t. Specifically, we employ a binary variable $i_n(t)$ to denote whether there is a building block obstructing the transmission between the UAV and the UE n at time slot t. By adopting this approach, we construct two-state edges that capture the diverse channel states and further utilize the R-GCN to efficiently process the multi-relational graph features. The process can be expressed as follows:

$$H_n(t) = \sigma(W_{n,UAV} H_{UAV}(t) + H_n(t)), \tag{16}$$

$$H_{UAV}(t) = \sigma\Big(\sum_{n=1}^{m} \frac{1}{m} W_{UAV,n} H_n(t) + H_{UAV}(t) + \sum_{n=m+1}^{N} \frac{1}{N-m} W_{UAV,n} H_n(t)\Big), \tag{17}$$

Algorithm 1 State Embedding

Input: $S(t) = \{s_n(t), \forall n \in [1, N], s_{UAV}(t)\}$
Output: s_t

for $n \in [1, N]$ **do**
  $H_n(t) = FC_1(s_n(t))$
end for
$H_{N+1}(t) = FC_2(s_{UAV}(t))$
for $n \in [1, N]$ **do**
  $H_n(t) = \sigma(W_{n,UAV} H_{UAV}(t) + H_n(t)))$
end for
$H_{UAV}(t) = \sigma(\sum_{n=1}^{m} \frac{1}{m} W_{UAV,n} H_n(t) + \sum_{n=m+1}^{N} \frac{1}{N-m} W_{UAV,n} H_n(t) + H_{UAV}(t))$
$s_t = pooling(H_1(t), H_2(t), \cdots, H_N(t), H_{UAV}(t))$
return s_t

where σ represents a nonlinear activation function. $W_{n,UAV}$ and $W_{UAV,n}$ represent their respective learnable weight matrices. m represents the number of channel states $i_n(t) = 0$. This step involves the aggregation of messages between nodes.

After message passing, we use graph pooling to construct the graph embedding based on node embeddings. Specifically, the state embedding of the AEC system can be expressed as:

$$s_t = pooling(H_1(t), H_2(t), \cdots, H_N(t), H_{UAV}(t)). \quad (18)$$

In general, graph pooling approaches, are categorized into three types, including concatenate (cat) pooling, mean pooling, and add pooling [39], as follows:

$$cat\ pooling: s_t = concatenate(H_1(t), H_2(t), \ldots, H_N(t), H_{UAV}(t)), \quad (19)$$

$$mean\ pooling: s_t = \frac{\sum_{n=1}^{N} H_n(t) + H_{UAV}(t)}{N+1}, \quad (20)$$

$$add\ pooling: s_t = \sum_{n=1}^{N} H_n(t) + H_{UAV}(t). \quad (21)$$

The state embedding algorithm is specifically described in **Algorithm 1**.

B. Action Space

With the observed environment and obtained state embedding generated by **Algorithm 1**, the agent selects actions in time slot t that include the task offloading rate of each UE, the flight angle, and the flight speed of the UAV. It is important to mention that the agent operates in a continuous action space. The action a_t can be expressed as follows:

$$a_t = (o_1(t), o_2(t), \cdots, o_N(t), v(t), \beta(t)). \quad (22)$$

where $o_n(t) \in [0, 1], \forall n \in [1, N]$ indicates the task offloading rate of all UEs in time slot t, $v(t) \in [0, v_{max}]$ indicates the flight speed of the UAV and $\beta(t) \in [0, 2\pi]$ indicates the flight angle of the UAV are continuous variables. These continuous variables work together to optimize the latency of all UEs.

C. Reward

By assigning a negative reward proportional to the delay, the agent is incentivized to take actions that reduce delay, because a smaller delay will result in a less negative reward. Our primary objective is to maximize the reward by minimizing the delay, as defined in the problem statement. Therefore, we set the reward r_t as follows:

$$r_t = -\sum_{n=1}^{N} \max\left\{T_{UE,n}^{com}(t), T_n^{trans}(t) + T_{UAV,n}^{com}(t)\right\}. \quad (23)$$

D. GCP-DDPG Algorithm

In order to optimize UAV trajectory control and task offloading ratios, and enhance the performance of AEC systems, we employ DDPG [38], which combines DNNs and deterministic policy gradients to learn action policies in continuous action space. To estimate the policy and Q-value functions, DDPG utilizes DNNs to approximate the actor-network $\mu(s, a; \theta^{\mu})$ with learnable parameters θ^{μ} and the critic network $Q(s, a; \theta^{Q})$ with parameters θ^{Q}. Moreover, to enhance the stability and efficiency of the learning process, DDPG employs a dual neural network architecture for both the policy and value functions. This architecture consists of a target actor-network denoted as μ', with parameters $\theta^{\mu'}$ and a target critic network referred to as Q', with parameters $\theta^{Q'}$.

By utilizing deterministic policy gradients and the dual network architecture, we can effectively handle high-dimensional action spaces and avoid the high variance that is often associated with stochastic policy gradients. According to the chain rule [40], the policy gradient can be updated in the following manner:

$$\begin{aligned}\nabla_{\theta_{\mu}} J &\approx E_{\mu'}[\nabla_{\theta^{\mu}} Q(s, a; \theta^{Q})|_{s=s_t, a=\mu(s_t;\theta^{\mu})}] \\ &= E_{\mu'}[\nabla_a Q(s, a; \theta^{Q})|_{s=s_t, a=\mu(s_t;\theta^{\mu})} \nabla_{\theta^{\mu}} \mu(s; \theta^{\mu})|_{s=s_t}],\end{aligned} \quad (24)$$

which employs the gradient ascent algorithm for optimization calculations to increase the expectation of discounted cumulative rewards:

$$J(\mu) = E_{\mu}[r_1 + \gamma r_1 + \gamma^2 r_2 + \cdots + \gamma^n r_n], \quad (25)$$

where γ is the reward discount factor.

The critic network is updated through updating the value of Deep Q-Network (DQN) [41], and the gradient is expressed as:

$$L(\theta^{Q}) = E_{\mu'}[(y_t - Q(s_t, a_i; \theta^{Q}))^2], \quad (26)$$

where $y_t = \gamma Q(s_{i+1}, \mu(s_{t+1}); \theta^{Q})$.

First of all, we need to initialize the policy online network $\mu(s, a; \theta^{\mu})$ and the critic network $Q(s, a; \theta^{Q})$, respectively. The parameters of the online network are assigned to their corresponding target network parameters, i.e., $\theta^{Q'} \longleftarrow \theta^{Q}$ and $\theta^{\mu'} \longleftarrow \theta^{\mu}$. At the same time, reset the state $S(1)$ of the AEC MEC system.

During a single experience trajectory episode phase, an action a_t is generated by adding behavior noise N_t as follows:

$$a_t = \mu(s, a : \theta^{\mu}) + N_t, \quad (27)$$

where N_t samples the Gaussian distribution $N_t \sim \mathbb{N}(\mu_n, \sigma_n^2)$, μ_n is the mean and σ_n^2 is the variance.

The UAV's trajectory strategy and UEs' task offloading strategy depend on a_t generated by the agent, returning rewards r_t and new states $S(t+1)$. Next, the agent stores the generated information tuple $(S(t), a_t, r_t, S(t+1))$ in R as training samples for the online network. After that, the agent randomly samples $(S(t), a_t, r_t, S(t+1))$ from R as training sample data in mini-batch. Execute **Algorithm 1** to obtain mini-batches (s_t, a_t, r_t, s_{t+1}) from $(S(t), a_t, r_t, S(t+1))$. The actor target network μ' outputs actions $\mu'(s_{t+1})$, the θ^Q calculate y_i based on $y_t = \gamma Q(s_{i+1}, \mu(s_{t+1}); \theta^Q)$. Update parameter θ^Q of the critic network using the stochastic gradient descent algorithm.

After that, the actor-network $\mu(s, a; \theta^\mu)$ is updated as follows:

$$\nabla_{\theta_\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a; \theta_Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_\mu} \mu(s; \theta^\mu)|_{s_t}. \tag{28}$$

In DDPG, the target actor network μ' and target critic network Q' are updated using the soft update method. This update process can be formulated as follows:

$$\theta^{Q'} \longleftarrow \tau\theta^{Q'} + (1-\tau)\theta^Q, \tag{29}$$

$$\theta^{\mu'} \longleftarrow \tau\theta^{\mu'} + (1-\tau)\theta^\mu, \tag{30}$$

where the default value of the soft update factor τ is typically set to 0.001. This value determines the rate at which the target networks are blended with the local networks during the update process.

In general, the proposed GCP-DDPG algorithm is illustrated in **Algorithm 2**, where the computation offloading training process is described.

Algorithm 2 GCP-DDPG-Based Computation Offloading Training Algorithm

Input: Number of training episode E; Critic network learning rate α_{Critic}; Actor-network learning rate α_{Critic}; Experience relay buffer R; Mini-batch size B_{RL};

Initialize the weight parameters θ^μ and θ^Q of actor-network $\mu(s, a; \theta^\mu)$ and critic network $Q(s, a; \theta^Q)$, respectively.
Initialize the weight parameters $\theta^{Q'} \longleftarrow \theta^Q$ and $\theta^{\mu'} \longleftarrow \theta^\mu$ of target critic network Q' and actor network μ', respectively.
Initialize the networks FC_1 and FC_2
Initialize the parameters $W_{n,UAV}$ and $W_{UAV,n}$,$\forall n \in [1, N]$
Empty experience relay buffer R
while each $episode = 1, 2, \cdots, E$ **do**
  Initialize the state $S(1)$ of the AEC system.
  for $t = 1, 2, 3, \cdots, T$ **do**
    Execute Algorithm 1 to obtain the state embedding S_t of the AEC system.
    Calculate the action $a_t = \mu(s, a : \theta^\mu) + N_t$ in the current time slot based on the current policy $\mu(s, a : \theta^\mu)$ with noise.
    Execute action a_t, and record the reward r_t and the next state $S(t+1)$.
    Store transfer tuple $(S(t), a_t, r_t, S(t+1))$ in experience relay buffer R.
    Randomly sample mini-batches B_{RL} of transitions $(S(t), a_t, r_t, S(t+1))$ from R.
    Execute Algorithm 1 to obtain mini-batches (s_t, a_t, r_t, s_{t+1}) from $(S(t), a_t, r_t, S(t+1))$.
    $y_i = r_i + Q'(s_{t+1}, \mu'(s_{t+1}; \theta^{\mu\prime}); \theta^{Q'})$
    Minimizing the loss to update the critic network $Q(s, a : \theta^Q)$.
    Update actor-networks using gradient policy algorithm.
    $\nabla_{\theta_\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a; \theta_Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_\mu} \mu(s; \theta^\mu)|_{s_t}$
    Soft update the target network via Eq. (29) and Eq. (30).
  end for
end while

E. Complexity Analysis

As described in Algorithm 2, in the DCP-DDPG architecture, let's consider that the number of neurons in the FC layer is denoted as N_{fc} and the number of neurons in the R-GCN graph convolutional layer is denoted as N_g. The time complexity for the FC layer would be $\mathcal{O}((N+1) \cdot N_{fc})$, due to linear transformations, and the time complexity for the R-GCN graph convolutional layer would be $\mathcal{O}((N+1) \cdot N_g)$ since it involves linear transformations as well.

In the Actor and Critic networks with N_L layers and N_{ac} neurons per layer, the forward propagation process includes matrix multiplication and activation function computations. Therefore, the overall time complexity for the forward propagation can be approximated as $\mathcal{O}(2 \cdot N_L \cdot N_{ac}^2)$. As a result, the time complexity of the DCP-DDPG algorithm can be estimated as $\mathcal{O}((N+1) \cdot (N_{fc}+N_g) + 2 \cdot N_L \cdot N_{ac}^2)$, considering the computations involved in both the FC and R-GCN layers, as well as the forward propagation in the Actor and Critic networks.

V. Performance Evaluation

In this section, we conduct extensive simulation experiments to exhibit the superiority of our proposed GCP-DDPG algorithm.

A. Parameters Setting

This paper considers an AEC system, where a UAV and $N = 4$ UEs are randomly placed within a 100×100 m^2 area. The UAV's flight altitude is set at a fixed value of $H = 100$ m, and its quality is set to $M_{fly} = 9.65$ kg. The maximum flight speed, denoted as v_{max}, is set to 50 m/s. The entire week of time $L = 400$ seconds is divided into $T = 40$ time slots, with each time slot having a UAV flight time T_{fly} of 1 second. The task data $D_n(t)$ generated by the n-th UE follows a uniform distribution within the range of 2.5 to 3 Mbits. The computation frequency $f_{UE,n}$ of each UE is uniformly distributed between 0.5 $\sim$ 1 GHz. The initial position of the UAV is set to [50, 50]. Further details of the simulation parameters are provided in Table III [21], [31].

The neural network architecture is configured by specifying the number of neurons in FC_1 and FC_2 as 64. The dimension of input for FC_1 is 4, where the first and second dimensions are the position coordinates of the UE, the third dimension represents the size of the calculation task of the UE, and the fourth dimension represents the indicative value of the obstacle occlusion. The input dimension for FC_2 is 3, where the first dimension is the remaining power and the second and third dimensions are the position coordinates of the UAV.

TABLE III
SIMULATION PARAMETERS

Parameter	Defaults	Parameter	Defaults
N	4	T	40
g_0	-50 dB	H	100 m
B	1 MHz	$P_n(t)$	$0.1 \sim 0.5$ W
$D_n(t)$	$2.5 \sim 3$ Mbits	$P_n^{nlos}(t)$	$10 \sim 50$ dB
s	1000 cycles/bit	f_{UAV}	1.2 GHz
$f_{UE,n}$	$0.5 \sim 1$ GHz	M_{fly}	9.65 kg
k	10^{-27}	L	400 second
$E_{Battery}$	500 kJ		

The following module of FC_1 and FC_2 is a two-layer R-GCN [27] with powerful graph data information extraction capability. The actor and critic networks are constructed with two FCs each, where the dimension of the actor's output is $N+2$ and the dimension of the critic's output is 1. To train the network parameters, the Adam optimizer is employed, with a learning rate of 0.001 assigned to both the actor and critic networks. To ensure efficient training, the experience buffer capacity is set to 10,000, and a mini-batch size of 64 is used. Additionally, the soft update factor τ is set to 0.001, which helps stabilize the training process.

B. Baseline Methods

In order to evaluate the GCP-DDPG algorithm more comprehensively, we tested three different graph pooling approaches with the same network structure, which can extract the most informative nodes and connections and capture higher-level relationships in the graph. Specifically:

- **GCP(C)-DDPG**: It is the GCP-DDPG algorithm that employs the concatenate pooling method proposed in [42], which is computed by Eq. (19).
- **GCP(M)-DDPG**: It is the GCP-DDPG algorithm that utilizes the mean pooling method, which is computed by Eq. (20).
- **GCP(A)-DDPG**: It is the GCP-DDPG algorithm that adopts the add pooling method, which is computed by Eq. (21).

By comparing the performance of these three methods, we aim to determine which approach results in the most effective and efficient AEC system.

We establish the same baseline algorithms as described in [21] and [31] to evaluate the performance of the GCP-DDPG algorithm.

- **OLNA**: All tasks of all UEs are executed locally, therefore in this scheme, no offloading action occurs.
- **EFOA**: The UAV offers computation offloading services to UEs at its initial position, with all UEs offloading their tasks to the nano-server carried by the UAV.
- **Random**: The UAV is positioned at a fixed location, and the offloading strategy for each task is selected randomly.
- **DDPG** [21]: To ensure a fair evaluation of the effectiveness of our GCP-DDPG algorithm, we exclusively utilize the DDPG algorithm for evaluation purposes. In detail, we employ only two FCs in both the actor and critic networks.

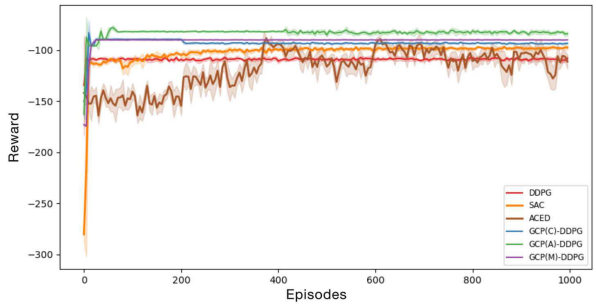


Fig. 4. Reward comparison between GCP-DDPG-based methods and baselines.

- **ACED** [31]: The ACED algorithm utilizes a Multilayer Perceptron (MLP) to extract information from IoT nodes within the MEC system and employs GNNs to extract task-related information. Subsequently, the Actor-Critic algorithm [17] is employed for task offloading. It is noteworthy that the proposed AEC system does not involve the modeling of tasks as directed acyclic graphs. As a result, the GNN module in the ACED algorithm is neither utilized nor included in the proposed system.
- **SAC** [20]: The SAC algorithm, relying on the Actor-Critic architecture, integrates entropy regularization to promote exploration and prevent convergence to local optima. It demonstrates proficiency in learning policies within continuous action spaces and showcases exceptional real-time performance and adaptability in contexts characterized by limited resources, such as AEC systems.

C. Reward Comparison

To conduct the evaluation, we performed 1000 episodes of training using the DRL algorithm, as well as three different variants of the GCP-DDPG algorithm. The neural networks in these algorithms were configured with different learning rates for plotting and averaging to minimize simulation errors. The learning rates ranged from 0.01 to 0.001. The performance comparison among various algorithms is illustrated in Fig. 4.

As depicted in Fig. 4, it is noticeable that with the increase in the number of episodes, all algorithms exhibit convergence within 1000 episodes. Importantly, it should be noted that the GCP-DDPG algorithm demonstrates faster convergence speed, improved performance, and greater stability when compared to other DRL algorithms like DPPG, ACED, and SAC. As a result, our experimental results indicate that the GCP-DDPG algorithm outperforms the others.

D. Impact of Different Computing Frequencies on Nano-MEC Server

To further validate the powerful performance of the proposed algorithm, we conducted performance testing of different algorithms at various computing frequencies of the nano-MEC server.

As depicted in Fig. 5, an augmentation in the computing capacity of the nano-MEC server leads to the ability to allocate a greater amount of computational resources to UEs. However, the OLNA algorithm exclusively relies on local

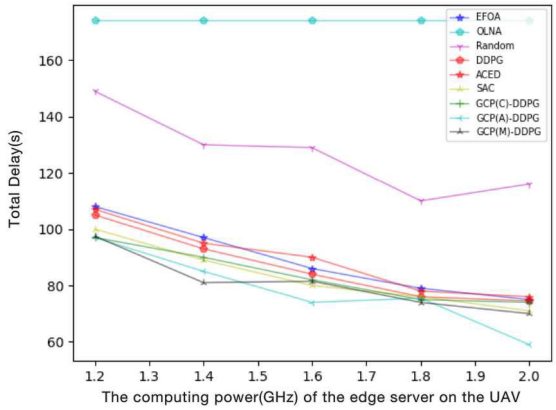


Fig. 5. Latency comparison between GCP-DDPG-based methods and baselines under different nano-MEC server computing frequencies.

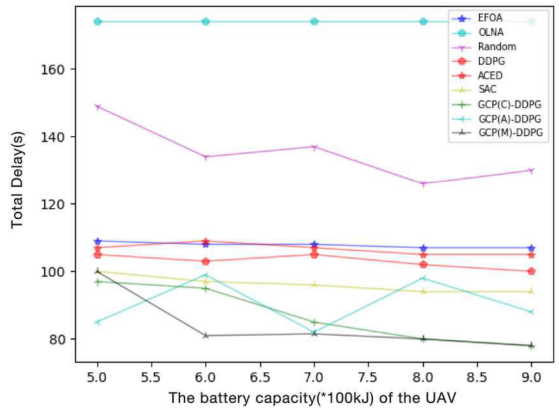


Fig. 6. Latency comparison of GCP-DDPG-based methods and baselines under different UAV battery capacities.

computational resources and neglects those on the nano-MEC server where the UAV operates. As a result, the OLNA algorithm suffers from high processing latency. Conversely, the EFOA algorithm offloads all tasks to the nano-MEC server and disregards the local resources, resulting in resource wastage and high processing latency. The Random algorithm displays high uncertainty, with no discernible optimal performance. As popular deep reinforcement learning (DRL) algorithms, DDPG, ACED, and SAC algorithms outperform the OLNA, EFOA, and Random algorithms, providing better solutions. Furthermore, the proposed GCP-DDPG algorithm not only preserves the advantages of the DDPG algorithm but also leverages the efficient processing capability of GNNs on graph data, leading to superior performance over the baselines and lower processing delay. The GCP-DDPG algorithm exhibits a performance improvement of up to 194.4% compared to OLNA, EFOA, and Random algorithms, and up to 26.7% compared to DDPG, ACED and SAC algorithms under varying computing frequencies of UAVs.

E. Impact of Different UAV Battery Capacities

Fig. 6 presents the performance comparison of each algorithm under different UAV battery capacities.

With the increase of the battery capacity of the UAV, the UAV gains greater autonomy and flexibility in controlling its trajectory, which in turn allows it to provide better computing services for the UEs. Since OLNA, EFOA, and Random algorithms do not optimize the trajectory of the UAV, the increase in the battery power of the UAV does not have a significant impact on the algorithm results. Furthermore, both our proposed GCP-DDPG algorithm and the DDPG algorithm significantly reduce the processing delay as the battery power of the UAV increases, with the variants of the GCP-DDPG algorithm exhibiting optimal performance. Under different UAV battery capacities, the GCP-DDPG algorithm we proposed demonstrates a performance improvement of up to 123.0% compared to OLNA, EFOA, and Random algorithms, and a performance improvement of up to 34.6% compared to DDPG, ACED and SAC algorithms.

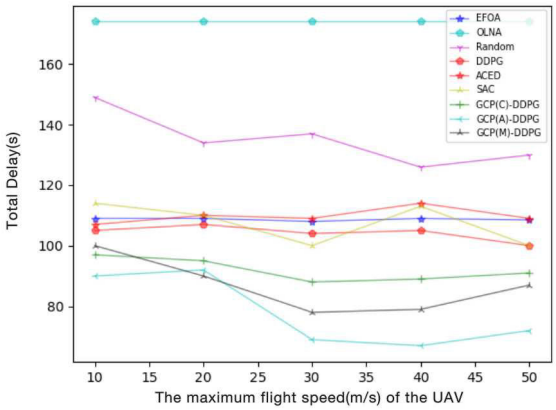


Fig. 7. Latency comparison of GCP-DDPG-based methods and baselines under maximum flight speed of different UAVs.

F. Impact of Maximum Flight Speed of Different UAVs

Fig. 7 presents the performance comparison of each algorithm at the maximum flight speed of the UAV.

Similar to the previous analysis, OLNA, EFOA, and Random algorithms are not impacted by the change in the maximum flight speed of the UAV. On the other hand, our proposed GCP-DDPG algorithm achieves optimal performance when compared to the baselines. At different flying speeds of the UAV, the proposed GCP-DDPG algorithm shows a performance improvement of up to 93.3% when compared to OLNA, EFOA, and Random algorithms. Furthermore, the proposed GCP-DDPG algorithm outperforms the current popular DDPG algorithm, achieving a performance improvement of up to 16.7%.

G. Impact of Different Task Data Size Generated by UEs

The algorithm's performance was also evaluated using different task data sizes generated by UEs. In particular, the tasks were divided into time slots of $0 \sim 0.5$ Mbits, $0.5 \sim 1$ Mbits, $1 \sim 1.5$ Mbits, $1.5 \sim 2$ Mbits, $2 \sim 2.5$ Mbits, and $2.5 \sim 3$ Mbits, and the algorithm was tested against each range to assess its effectiveness.

As shown in Fig. 8, with the increase of the task data generated by UEs, the task processing delays of all UEs increase accordingly under any algorithm, while our proposed

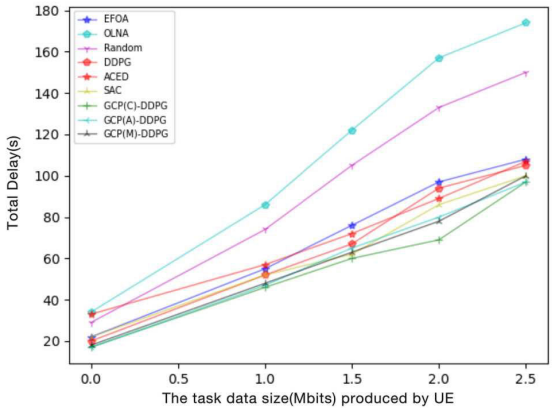


Fig. 8. Latency comparison between GCP-DDPG-based methods and baselines under different task data sizes generated by UEs.

GCP-DDPG algorithm exhibits significantly lower delays compared to the baselines, indicating its superior performance. Under varying task sizes, the proposed algorithm demonstrates significant performance improvements compared to other algorithms. Specifically, it achieves an improvement of up to 127.5% compared to OLNA, EFOA, and Random algorithms. Furthermore, it outperforms state-of-the-art DRL algorithms such as DDPG, ACED, and SAC with a performance improvement of up to 36.3%.

VI. Conclusion and Future Work

In this paper, we establish a UAV-assisted MEC system as a multi-relational graph data structure, wherein UAVs play a crucial role in efficiently serving UEs and providing computational support in temporary hotspot areas. The primary objective is to minimize the processing latency experienced by all UEs by jointly optimizing the UAV trajectory, flight speed, and task offloading ratio. To tackle this challenge, we propose the GCP-DDPG algorithm, which incorporates GNNs to enhance the solution of the task offloading problem based on DDPG in MEC environments. The superiority of the proposed GCP-DDPG algorithm is validated through extensive simulation experiments. Furthermore, the proposed method exhibits ease of deployment in real-world MEC environments, accommodating various communication conditions.

In future work, we plan to leverage the capabilities of GNNs to investigate the task offloading problem in other MEC scenarios. We aim to explore scenarios where multiple UAVs collaborate to accomplish task offloading, with each UAV assigned to fly in distinct regions to assist ground UEs. We will specifically focus on developing algorithms based on multi-agent DRL and convolutional forms in heterogeneous graphs with multiple types of nodes and edges.

References

[1] M. K. Afzal, Y. B. Zikria, S. Mumtaz, A. Rayes, A. Al-Dulaimi, and M. Guizani, "Unlocking 5G spectrum potential for intelligent IoT: Opportunities, challenges, and solutions," *IEEE Commun. Mag.*, vol. 56, no. 10, pp. 92–93, Oct. 2018.

[2] R. Gupta, S. Tanwar, S. Tyagi, and N. Kumar, "Tactile internet and its applications in 5G era: A comprehensive review," *Int. J. Commun. Syst.*, vol. 32, no. 14, p. e3981, Sep. 2019.

[3] P. Sun et al., "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2446–2454.

[4] H. Tang, H. Wu, G. Qu, and R. Li, "Double deep Q-network based dynamic framing offloading in vehicular edge computing," *IEEE Trans. Network Sci. Eng.*, vol. 10, no. 3, pp. 1297–1310, May/Jun. 2023.

[5] R. Bashshur, C. R. Doarn, J. M. Frenk, J. C. Kvedar, and J. O. Woolliscroft, "Telemedicine and the COVID-19 pandemic, lessons for the future," *Telemedicine e-Health*, vol. 26, no. 5, pp. 571–573, May 2020.

[6] H. Wu, J. Chen, T. Nguyen, and H. Tang, "Lyapunov-guided delay-aware energy efficient offloading in IIoT-MEC systems," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 2117–2128, Feb. 2023.

[7] X. Zhou et al., "Edge computation offloading with content caching in 6G-enabled IoV," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 3, pp. 2733–2747, Jul. 2024.

[8] M. Huang, W. Liu, T. Wang, A. Liu, and S. Zhang, "A cloud–MEC collaborative task offloading scheme with service orchestration," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5792–5805, Jul. 2019.

[9] N. Mohamed, J. Al-Jaroodi, I. Jawhar, H. Noura, and S. Mahmoud, "UAVFog: A UAV-based fog computing for Internet of Things," in *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Computed, Scalable Comput. Commun., Cloud Big Data Comput., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Aug. 2017, pp. 1–8.

[10] W. Z. Khan, M. Y. Aalsalem, M. K. Khan, M. S. Hossain, and M. Atiquzzaman, "A reliable Internet of Things based architecture for oil and gas industry," in *Proc. 19th Int. Conf. Adv. Commun. Technol. (ICACT)*, 2017, pp. 705–710.

[11] T. Zhang, Y. Xu, J. Loo, D. Yang, and L. Xiao, "Joint computation and communication design for UAV-assisted mobile edge computing in IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5505–5516, Aug. 2020.

[12] J. Zhang et al., "Computation-efficient offloading and trajectory scheduling for multi-UAV assisted mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2114–2125, Feb. 2019.

[13] P. Tong, M. Li, M. Li, J. Huang, and X. Hua, "Large-scale vehicle trajectory reconstruction with camera sensing network," in *Proc. 27th Annu. Int. Conf. Mobile Comput. Netw.*, 2021, pp. 188–200.

[14] P.-Q. Huang, Y. Wang, and K.-Z. Wang, "Energy-efficient trajectory planning for a multi-UAV-assisted mobile edge computing system," *Frontiers Inf. Technol. Electron. Eng.*, vol. 21, no. 12, pp. 1713–1725, Dec. 2020.

[15] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, Aug. 2017.

[16] Y. Xu, T. Zhang, D. Yang, Y. Liu, and M. Tao, "Joint resource and trajectory optimization for security in UAV-assisted MEC systems," *IEEE Trans. Commun.*, vol. 69, no. 1, pp. 573–588, Jan. 2021.

[17] N. Cheng et al., "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.

[18] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proc. ACM Special Interest Group Data Commun.*, Aug. 2019, pp. 270–288.

[19] Y. Hou, L. Liu, Q. Wei, X. Xu, and C. Chen, "A novel DDPG method with prioritized experience replay," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2017, pp. 316–321.

[20] S. Li, X. Hu, and Y. Du, "Deep reinforcement learning for computation offloading and resource allocation in unmanned-aerial-vehicle assisted edge computing," *Sensors*, vol. 21, no. 19, p. 6499, 2021.

[21] Y. Wang, W. Fang, Y. Ding, and N. Xiong, "Computation offloading optimization for UAV-assisted mobile edge computing: A deep deterministic policy gradient approach," *Wireless Netw.*, vol. 27, no. 4, pp. 2991–3006, May 2021.

[22] J. Zhou et al., "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Sep. 2020.

[23] T. Zhao, X. Zhang, and S. Wang, "GraphSMOTE: Imbalanced node classification on graphs with graph neural networks," in *Proc. 14th ACM Int. Conf. Web Search Data Mining*, Mar. 2021, pp. 833–841.

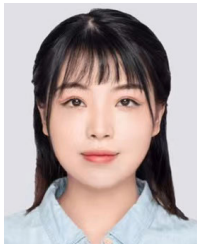
[24] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: a survey," *ACM Comput. Surveys*, vol. 55, no. 5, pp. 1–37, 2022.

[25] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

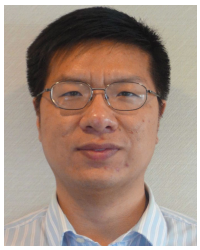
[26] P. Veličković et al., "Graph attention networks," 2017, *arXiv:1710.10903*.
[27] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. Eur. Semantic Web Conf.*, Heraklion, Crete, Greece. Berlin, Germany: Springer-Verlag, 2018, pp. 593–607.
[28] 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 Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 73–84, Mar. 2020.
[29] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and A. Nallanathan, "Deep reinforcement learning based dynamic trajectory control for UAV-assisted mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 10, pp. 3536–3550, Oct. 2022.
[30] A. Chowdhury, G. Verma, C. Rao, A. Swami, and S. Segarra, "Unfolding WMMSE using graph neural networks for efficient power allocation," *IEEE Trans. Wireless Commun.*, vol. 20, no. 9, pp. 6004–6017, Sep. 2021.
[31] J. Chen, Y. Yang, C. Wang, H. Zhang, C. Qiu, and X. Wang, "Multitask offloading strategy optimization based on directed acyclic graphs for edge computing," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9367–9378, Jun. 2022.
[32] Y. Shen, J. Zhang, S. H. Song, and K. B. Letaief, "Graph neural networks for wireless communications: From theory to practice," *IEEE Trans. Wireless Commun.*, vol. 22, no. 5, pp. 3554–3569, May 2023.
[33] Y. Li, J. Li, and J. Pang, "A graph attention mechanism-based multiagent reinforcement-learning method for task scheduling in edge computing," *Electronics*, vol. 11, no. 9, p. 1357, Apr. 2022.
[34] Y. Huang and Y. Wang, "The application of graph neural network based on edge computing in English teaching mode reform," *Wireless Commun. Mobile Comput.*, vol. 2022, pp. 1–12, Mar. 2022.
[35] T. Pamuklu, A. Syed, W. S. Kennedy, and M. Erol-Kantarci, "Heterogeneous GNN-RL based task offloading for UAV-aided smart agriculture," *IEEE Netw. Lett.*, vol. 5, no. 4, pp. 213–217, Jun. 2023.
[36] K. Li, W. Ni, X. Yuan, A. Noor, and A. Jamalipour, "Deep-graph-based reinforcement learning for joint cruise control and task offloading for aerial edge Internet of Things (EdgeIoT)," *IEEE Internet Things J.*, vol. 9, no. 21, pp. 21676–21686, Nov. 2022.
[37] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, "Energy-efficient UAV-assisted mobile edge computing: Resource allocation and trajectory optimization," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3424–3438, Mar. 2020.
[38] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, San Juan, Puerto Rico, May 2016, pp. 1–14.
[39] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Mar. 2020.
[40] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.
[41] V. Mnih, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.
[42] F. Hu, Y. Zhu, S. Wu, L. Wang, and T. Tan, "Hierarchical graph convolutional networks for semi-supervised node classification," 2019, *arXiv:1902.06667*.

Huaming Wu (Senior Member, IEEE) received the B.E. and M.S. degrees from Harbin Institute of Technology, China, in 2009 and 2011, respectively, both in electrical engineering, and the Ph.D. degree (Hons.) in computer science from Freie Universität Berlin, Germany, in 2015. He is currently a Professor with the Center for Applied Mathematics, Tianjin University, China. His research interests include mobile cloud computing, edge computing, the Internet of Things, deep learning, complex networks, and DNA storage.

Lei Tian received the B.Sc. degree from Yanshan University, China, in 2020, and the M.S. degree from the Center for Applied Mathematics, Tianjin University, China, in 2023. His research interests include the Internet of Things, mobile edge computing, and deep learning.



Huijun Tang (Member, IEEE) received the B.Sc. degree from Jinan University, China, in 2016, and the M.S. and Ph.D. degrees from Tianjin University, China, in 2018 and 2022, respectively. She is currently a Lecturer with the School of Cyberspace, Hangzhou Dianzi University. Her research interests include the Internet of Things, mobile edge computing, and deep learning.



Ruidong Li (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science from the University of Tsukuba in 2005 and 2008, respectively. He is currently an Associate Professor with Kanazawa University, Japan. Before joining Kanazawa University, he was a Senior Researcher with the National Institute of Information and Communications Technology (NICT), Japan. His research interests include future networks, big data, intelligent Internet edge, the Internet of Things, network security, information-centric networks, artificial intelligence, quantum internet, cyber-physical systems, and wireless networks. He is a member of IEICE. He served as the Chair for several conferences and workshops, such as the General Co-Chair for IEEE MSN 2021, AIVR2019, and IEEE INFOCOM 2019/2020/2021 ICCN Workshop, and the TPC Co-Chair for IWQoS 2021, IEEE MSN 2020, BRAINS 2020, IEEE ICDCS 2019/2020 NMIC Workshop, and ICCSSE 2019. He serves as the Secretary for the IEEE ComSoc Internet Technical Committee (ITC) and the Founder and the Chair for the IEEE SIG on Big Data Intelligent Networking and IEEE SIG on Intelligent Internet Edge. He is an Associate Editor of IEEE INTERNET OF THINGS JOURNAL and also served as the Guest Editor for a set of prestigious magazines, transactions, and journals, such as *IEEE Communications Magazine*, IEEE NETWORK, and IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING.

Pengfei Jiao (Member, IEEE) received the Ph.D. degree in computer science from Tianjin University, Tianjin, China, in 2018. From 2018 to 2021, he was a Lecturer with the Center of Biosafety Research and Strategy, Tianjin University. He is currently a Professor with the School of Cyberspace, Hangzhou Dianzi University, Hangzhou, China. His current research interests include complex network analysis and its applications.