CISTER
Research Centre in
Real-Time & Embedded
Computing Systems

Conference Paper

Exploring Graph Neural Networks for Joint Cruise Control and Task Offloading in UAV-enabled Mobile Edge Computing

Kai Li*
Wei Ni
Xin Yuan
Alam Noor*
Abbas Jamalipour

*CISTER Research Centre

CISTER-TR-230403

2023/06/20

Exploring Graph Neural Networks for Joint Cruise Control and Task Offloading in UAV-enabled Mobile Edge Computing

Kai Li*, Wei Ni, Xin Yuan, Alam Noor*, Abbas Jamalipour

*CISTER Research Centre

Polytechnic Institute of Porto (ISEP P.Porto)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: kai@isep.ipp.pt, Wei.Ni@data61.csiro.au, xin.yuan@data61.csiro.au, alamn@isep.ipp.pt

https://www.cister-labs.pt

Abstract

Unmanned aerial vehicles (UAVs) have been increasingly considered as aerial servers in mobile edge computing (MEC) to assist mission-critical computation tasks of edge ground nodes. The tasks are buffered at the ground node, while the task offloading is scheduled by the UAV. When one ground node in MEC is scheduled to offload its tasks, other unselected ground nodes' tasks could expire and be cancelled. To maximize the offloaded tasks to the UAV, this paper proposes a new joint optimization of cruise control and task offloading scheduling, which synthetically takes into account the computation capacity and battery energy of the ground nodes, and the speed limit of the UAV. Given a large and unknown network state and action space, a new deep reinforcement learning (DRL) framework based on graph neural networks (GNN) is developed to train online the continuous cruise control of the UAV and the task offloading schedule. Particularly, GNN explores feature correlations of network states to supervise the action training of the UAV in DRL. We implement the proposed GNN-DRL framework on Google Tensorflow. Extensive numerical results show that GNN-DRL improves the task offloading rate by 43%, compared to the DRL solution without GNN.

Exploring Graph Neural Networks for Joint Cruise Control and Task Offloading in UAV-enabled Mobile Edge Computing

Kai Li*, Wei Ni†, Xin Yuan†, Alam Noor*, and Abbas Jamalipour‡
*CISTER, Porto, Portugal
Email: kaili@ieee.org, alamn@isep.ipp.pt
†CSIRO, Sydney, Australia
Email: {wei.ni,xin.yuan}@data61.csiro.au
‡School of Electrical and Information Engineering, The University of Sydney, Australia
Email: a.jamalipour@ieee.org

***Abstract*—Unmanned aerial vehicles (UAVs) have been increasingly considered as aerial servers in mobile edge computing (MEC) to assist mission-critical computation tasks of edge ground nodes. The tasks are buffered at the ground node, while the task offloading is scheduled by the UAV. When one ground node in MEC is scheduled to offload its tasks, other unselected ground nodes' tasks could expire and be cancelled. To maximize the offloaded tasks to the UAV, this paper proposes a new joint optimization of cruise control and task offloading scheduling, which synthetically takes into account the computation capacity and battery energy of the ground nodes, and the speed limit of the UAV. Given a large and unknown network state and action space, a new deep reinforcement learning (DRL) framework based on graph neural networks (GNN) is developed to train online the continuous cruise control of the UAV and the task offloading schedule. Particularly, GNN explores feature correlations of network states to supervise the action training of the UAV in DRL. We implement the proposed GNN-DRL framework on Google Tensorflow. Extensive numerical results show that GNN-DRL improves the task offloading rate by 43%, compared to the DRL solution without GNN.**

***Index Terms*—Unmanned aerial vehicle, Mobile edge computing, Graph neural network, Deep reinforcement learning, Task offloading rate**

I. Introduction

Mobile devices operate increasingly in human-unfriendly, harsh areas, such as remote farmlands, or rural vineyards, where computation-intensive operations, e.g., monitoring crops and water-efficient irrigation control, are carried out [1], [2]. Since conventional terrestrial communication networks are distributed sparsely, reliable connections for mobile edge computing (MEC) systems are difficult to be guaranteed. Thanks to high mobility, low cost, and on-demand deployment, unmanned aerial vehicles (UAVs) can be used to expand the cloud computing service coverage in the UAV-enabled MEC system [3]. In Figure 1, the UAV as an aerial edge server patrols along its flight trajectory over the target area [4]. The edge nodes in MEC systems can handle lightweight tasks that require small computing resources, but for big and complex tasks that exceed the edge node's computing capabilities, the tasks need to be buffered in a task container and offloaded to the UAV, which is equipped with more powerful CPUs and GPUs. This is necessary due to the edge node's limited computing capability and local resources. Moreover, dividing an atomic task into sub-tasks that are partially processed on different devices can lead to inconsistencies, as the sub-tasks rely on shared data or resources. This results in incomplete operations or incorrect outcomes, undermining the fundamental principles of atomicity and integrity. Therefore, by buffering the tasks, the edge nodes can avoid overloading their resources and ensure that the tasks are processed efficiently and integrally by the UAV, reducing the overall processing time and system latency. The tasks in the container can be offloaded when the UAV is around and schedules the edge node. A line-of-sight (LoS) communication link between the UAV and the edge node improves the channel quality and enables a high-speed task offloading rate at the edge node [5].

When the UAV schedules an edge node to offload its tasks, the buffered tasks of the other unselected edge nodes can expire and be cancelled due to the task container overflow. This gives rise to a large number of dropped computation tasks [6]. Moreover, offloading a task from a scheduled edge node can experience a poor channel quality since the UAV's movement results in time-varying channel dynamics [7]. The offloading errors, in turn, overflow the task container of the unscheduled edge nodes. In practice, occupancy of the task container and channel conditions between the UAV and the edge node are unknown to the UAV. Hence, scheduling the task offloading online in the presence of the joint cruise control (i.e., speed and heading) is crucial to maximizing the task offloading rate of the edge nodes.

It is difficult for a UAV to optimize its cruise control and task offloading scheduling when the solution space is large and the current network states are unknown. The UAV needs to learn a thorough understanding of the environment

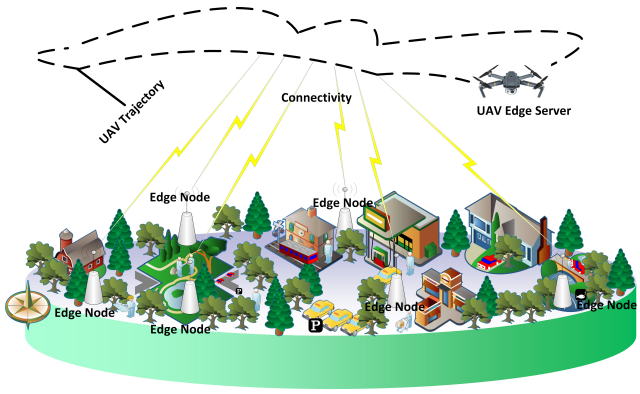


Fig. 1: The UAV, as an aerial MEC server, patrols over the target area along its flight trajectory. When the UAV is around, the edge nodes offload their computation tasks to the UAV for processing.

and the time-varying factors that affect its actions in order to make effective decisions. In this paper, we propose a graph neural network (GNN)-assisted deep reinforcement learning (GNN-DRL) framework to maximize the number of computation tasks offloaded to the UAV, subject to the computation capacity and battery energy of multiple edge nodes, and the speed limit of the UAV. Network states consist of the buffered tasks in the container, battery levels, and channel qualities between the edge nodes and the UAV. Since multiple edge nodes are deployed in the same target field, the network state can be affected by the same environmental conditions. For example, the edge nodes close to each other may experience similar channel quality or energy harvesting, which leads to a correlated battery status, the channel quality variation, and the number of generated tasks. The GNN-DRL carried out at the UAV aims to explore the hidden representations resulting from the network feature correlation, which can be used to supervise the training of UAV's actions in terms of the flight speed, heading, and the offloading schedule of the edge nodes. To validate the proposed GNN-assisted DRL framework, we implement GNN-DRL in Python 3.9 based on Google Tensorflow. A Linux workstation with 64-bit Ubuntu 18.04 is used as the hardware platform. Numerical results show that the GNN-DRL significantly improves the task offloading rate while effectively controlling the flight cruise to process the computation tasks.

The key contributions of this paper are summarized as follows:

- In the context of UAV-enabled MEC systems, we develop the task offloading scheduling optimization to maximize the rate of task offloading that is cancelled or fails due to offloading errors. This is done by carefully controlling the optimal flight speed and heading for the UAV and selecting the edge nodes, taking into account factors such as the buffered tasks in the container, battery levels, and channel qualities.
- The proposed GNN-DRL framework is designed to tackle the task offloading scheduling optimization in UAV-enabled MEC systems. The training of the UAV's real-time continuous actions is supervised by the GNN [8], allowing the UAV to explore the hidden representations of the network based on feature correlations. This helps the UAV to make better decisions about its speed and heading in order to maximize the task offloading rate. The use of GNN-DRL allows the UAV to effectively navigate the large solution space and make optimal decisions even in the face of uncertainty about the current network conditions.

This paper is organized as follows. Section II presents the literature on the GNN and DRL solutions in UAV-enabled MEC. In Section III, the system model of the UAV-enabled MEC is formulated. In Section IV, we investigate the proposed GNN-DRL framework to jointly optimize the cruise control and task offloading. The implementation and performance of the GNN-DRL are presented in Section V. Finally, Section VI concludes the paper.

II. Related Work

Graph neural networks (GNNs) construct a graph consisting of vertices (i.e., network states) linked by edges (representing dependencies) [9]. In GNNs, hidden representations of vertices are passed iteratively between adjacent vertices. These hidden representations are propagated throughout the graph using multiple iterations until a fixed vertex is found. The final hidden representation is then used to predict future network states with respect to vertices. Therefore, GNNs can be used to exploit the topological dependencies of resource-constrained edge devices for efficient inference. Given an unknown MEC environment, DRL can be used to design the trajectory of a UAV to serve edge nodes [9]. According to the quality of service requirements of ground equipment, the energy efficiency of the UAV can be improved. In [10], deep Q-network (DQN) and DRL were exploited to learn the offloading task ratio and the UAV trajectory to reduce the energy consumption of the UAV and the edge nodes. In wireless powered sensor networks [11], [12], deep deterministic policy gradient (DDPG)-based or DQN-based strategies were studied to schedule the UAV's trajectory and resource allocation to minimize the buffer overflow of the sensors.

The existing literature uses DRL to shorten the mission time or improve the energy efficiency of the UAV, such as [9]–[11], where the correlation of network states in MEC is not considered. In some studies, DRL combined with GNN has been used to design task offloading policies for the MEC systems. For example, in [13], a communication framework based on DRL was studied to offload tasks to a static server, where the MEC is modeled as an acyclic graph, and the offloading policy is modeled by graph state migration. The framework combines the GNN with actor-critic networks to train offloading policies without labels, where the task offloading policies are adapted to improving

the service coverage and energy efficiency. However, this work considers a fixed number of offloaded tasks while the GNN is trained without the mobility optimization.

Different from the existing literature, this paper investigates a new challenge in UAV-enabled MEC, where poor cruise control of the UAV results in an unsuccessful task offloading of the selected edge node while the tasks in the buffer at unselected edge nodes are outdated and canceled. In addition, a new GNN-DRL structure is developed, where the GNN extracts the correlated features of the network states to supervise the training of the UAV's actions.

III. System Model

In this section, we present the system model of the UAV-enabled MEC system. Table I lists the notations used in this paper.

As shown in Figure 1, N edge nodes can offload computing tasks to the UAV edge server. Each edge node i can partially offload its computation load through a wireless link [14]. The UAV is equipped with a portable edge server, and patrols over the target field, while the edge nodes can be scheduled to offload the task. The battery level of node $i \in [1, N]$ at time t is $e_i(t)$. Given the limited edge node's battery capacity, $e_i(t) \leq E$, where E (in Joules) is the battery capacity of the edge node. The tasks that are buffered at the task container of the edge node are offloaded to the UAV following the first-come-first-serve discipline. In particular, edge node i can generate $m_i(t)$ tasks at t, where $m_i(t) \leq M$ and M is the size of the task container. The newly generated tasks have to be dropped if $m_i(t) > M$, i.e., the overflows at the task container.

The position of the UAV at time t can be denoted by (x_t, y_t, z), where the UAV maintains its altitude at z meters [15]. For safety considerations, the instantaneous flight speed of the UAV, denoted by v_t, has to be no larger than the maximum speed V_{max} [16]. Thus, we have $v_t \leq V_{max}$. Let W_t denote the time when the UAV moves from (x_t, y_t, z) to (x_{t+1}, y_{t+1}, z). Given the speed difference $\hat{v_t} = v_{t+1} - v_t$, the acceleration at (x_t, y_t, z) is

$$\hat{v_t}/W_t = (v_{t+1} - v_t)/W_t, \tag{1}$$

where

$$0 \leq \hat{v_t}/W_t \leq V_{max}/W_t. \tag{2}$$

Let e_t^{UAV} denote the battery level of the UAV at time t. Suppose that the UAV can harvest energy to recharge its battery. The harvested energy at time t can be [17]

$$\hat{e}_t^{\mathrm{UAV}} = a_{\mathrm{solar}} b_{\mathrm{solar}} c \exp\Big(\frac{-\kappa}{\cos a_t^{\mathrm{solar}}}(1 - 2.2556 \times 10^{-5} z)^{5.2561}\Big), \tag{3}$$

where $c \in (0, 1)$ and a_{solar} denote the charging efficiency and size of the solar panel, respectively [18]. b_{solar} is the constant power intensity of the solar beams. $\kappa > 0$ is the sum atmospheric extinction. a_t^{solar} denotes the solar zenith angle at time t. z is the altitude of the UAV.

TABLE I: Notation and definition

Notation	Definition
N	number of edge nodes
$e_i(t)$	battery level of node i at t
$m_i(t)$	tasks generated at node i
M	size of the task container
E	battery capacity of the edge node
v_t	instantaneous flight speed of the UAV
V_{max}	the maximum allowable speed of the UAV
e_t^{UAV}	battery level of the UAV
$\hat{e}_t^{\mathrm{UAV}}$	harvested energy of the UAV
$e_t^{\mathrm{UAV}\prime}$	propulsion energy consumption of the UAV
$h_i(t)$	channel gain between edge node i and the UAV
$p_i(t)$	number of tasks of node i being offloaded to the UAV till time t
S_α	network state
A_{S_α}	action carried out by the UAV at S_α
$\mathcal{G}(\mathcal{V}, \mathcal{E}; \mathbf{w})$	a graph built with $\mathcal{V}$, $\mathcal{E}$, and $\mathbf{w}$
$\eta_{\mathcal{G}}^k$	graph generation loss
$\mathbf{f}_{\mathcal{V}}^k$	hidden state of $\mathcal{V}$
$R\{S_\beta \vert S_\alpha^{\mathcal{G}}, A_{S_\alpha^{\mathcal{G}}}\}$	reward of the DRL training
$L_{i_{S_\alpha^{\mathcal{G}}}}(t)$	a regression model of the offloaded tasks

The propulsion energy of the UAV is

$$e_t^{\mathrm{UAV}\prime} = P_0\Big(1 + \frac{3v_t^2}{\omega_t^2}\Big) + P_0'\Big(\sqrt{1 + \frac{v_t^4}{4v_0^4}} - \frac{v_t^2}{2v_0^2}\Big)^{1/2} + \frac{1}{2}\xi_{\mathrm{drag}}\rho_{\mathrm{air}}\xi_{\mathrm{rotor}}\xi'_{\mathrm{rotor}}v_t^3, \tag{4}$$

where P_0 and P_0' are two constants [19]. ω_t is the tip speed of the rotor blade. v_0 is the mean rotor induced velocity in hover. ξ_{drag} and ξ_{rotor} denote the fuselage drag ratio and rotor solidity, respectively. ρ_{air} and ξ'_{rotor} denote the air density and rotor disc area, respectively. Thus, we have

$$e_t^{\mathrm{UAV}} + \hat{e}_t^{\mathrm{UAV}} - e_t^{\mathrm{UAV}\prime} \geq e_0^{\mathrm{UAV}}, \tag{5}$$

where e_0^{UAV} is the minimum safe battery level of the UAV.

IV. GNN-assisted Deep Reinforcement Learning

In this section, we develop the GNN-assisted DRL, where GNN extracts network state features and hidden connections to supervise the training of the UAV's actions.

A. Feature correlation with the graph

The network state, denoted by S_α, consists of the battery levels of the edge nodes and the UAV at time t, i.e., $e_i(t)$ and e_t^{UAV}, the number of generated tasks at the edge node $m_i(t)$, the channel gain $h_i(t)$, the location of the UAV, and the number of tasks of node i being offloaded to the UAV till time t, i.e., $p_i(t)$. Therefore, we have

$$S_\alpha = \{e_i, m_i, h_i, p_i, e^{\mathrm{UAV}}, (x, y, z)\}, \quad \forall i \in [1, N]. \tag{6}$$

The UAV takes actions along the trajectory to control its heading and speed, while scheduling the edge nodes to offload their computation tasks. The action is given by

$$A_{S_\alpha} = \Big\{\Big(x(S_\beta), y(S_\beta), z\Big), v_{S_\alpha}, i_{S_\alpha}\Big\}, \tag{7}$$

where $(x(S_\beta), y(S_\beta), z)$ is the next location of the UAV. v_{S_α} is the speed of the UAV at the state S_α. i_{S_α} indicates the selected edge node at state S_α.

A graph $\mathcal{G}(\mathcal{V}, \mathcal{E}; \mathbf{w})$ can be used to describe the correlation between the network states. Specifically, a vertex $\mathcal{V}$ indicates one network state S_α and the vertex correlation is encoded by an edge $\mathcal{E}$. $\mathcal{V}_\mathcal{G}$ represents the vertex space, and the weights of the edges are denoted as $\mathbf{w}$. We denote K as the total number of layers in $\mathcal{G}(\mathcal{V}, \mathcal{E}; \mathbf{w})$. At the k-th layer, a learnable vector of the weights of $\mathcal{V}$'s edges is $w_\mathcal{V}^k$, where $w_\mathcal{V}^k \subset \mathbf{w}$. Thus, the hidden state of the vertex $\mathcal{V}$ is

$$\mathbf{f}_\mathcal{V}^k = \Phi^k\Big(\mathbf{f}_\mathcal{V}^{k-1} \oplus agt^k\Big(\{\mathbf{f}_{\mathcal{V}',\mathcal{E}}^{k-1} : (\mathcal{V}, \mathcal{V}') \in \mathcal{E}^k\}_{\mathcal{E}^k \in \mathbb{R}^\mathcal{E}}\Big); w_\mathcal{V}^k\Big), \tag{8}$$

where $\oplus$ represents the embedding summation operation; $\Phi^k(\cdot)$ is a nonlinear activation function (e.g., $\tanh(\cdot)$ or $ReLU(\cdot)$); $\mathbf{f}_\mathcal{V}$, $\mathbf{f}_\mathcal{E}$ and $\mathbf{f}_{\mathcal{V}'}$ are the representations of the vertex $\mathcal{V}$, edge $\mathcal{E}$, and neighbors of $\mathcal{V}$, respectively; $\mathcal{E}^k$ collects the edges at the k-th layer; $\mathbb{R}^\mathcal{E}$ denotes the hidden state dimension; and $agt^k(\cdot)$ is the aggregation function at the k-th layer, which maps the neighborhood information from different relations into a vector, e.g., mean aggregation and attention aggregation. $\mathbf{f}_\mathcal{V}^k$ can be initialized by $\mathbf{f}_\mathcal{V}^0 = \mathcal{V}$.

Based on (8), we can obtain the graph generation loss, as given by,

$$\eta_\mathcal{G}^k = \sum_{\mathcal{V} \in \mathcal{V}_\mathcal{G}} -\log(\Phi^k(\tanh(\mathbf{f}_\mathcal{V}^k))), \tag{9}$$

where $\tanh(\cdot)$ is a multilayer perceptron. The input of $\tanh$ at the k-th layer is the node embedding at the previous layer. The output is a scalar which is then fed into a nonlinear activation function $\Phi^k(\cdot)$.

B. The GNN-DRL algorithm

Since the proposed GNN-DRL algorithm aims to maximize the task offloading rate, the reward of the DRL training, denoted by $R\{S_\beta | S_\alpha^\mathcal{G}, A_{S_\alpha^\mathcal{G}}\}$, is formulated as the number of offloaded tasks to the UAV, as given by

$$R\{S_\beta | S_\alpha^\mathcal{G}, A_{S_\alpha^\mathcal{G}}\} = \min_{\forall i_{S_\alpha^\mathcal{G}} \in [1,N]} L_{i_{S_\alpha^\mathcal{G}}}(S_\alpha^\mathcal{G}), \tag{10}$$

where $S_\alpha^\mathcal{G}$ denotes the network state trained by the GNN. An episode defines a sequence of interactions between the UAV and the training environment. During an episode, the action $A_{S_\alpha^\mathcal{G}}$ is carried out by the UAV and the network state transits from $S_\alpha^\mathcal{G}$ to S_β, providing a reward $R\{S_\beta | S_\alpha^\mathcal{G}, A_{S_\alpha^\mathcal{G}}\}$ to the UAV. GNN aims to supervise the training of the UAV's actions in DRL, i.e., A_{S_α} in (7). a regression model can map the SNR to the number of computation tasks successfully collected by the UAV edge server. $L_{i_{S_\alpha^\mathcal{G}}}$ is a regression model of the offloaded tasks, as given by

$$L_{i_{S_\alpha^\mathcal{G}}}(t) = (1 - \frac{1}{2} e^{\beta_1 - \beta_0 H_{i_{S_\alpha^\mathcal{G}}}(t)})^{8(2f-l)}, \tag{11}$$

where β_0 and β_1 are two constants in the regression model. β_0 controls the shape of the regression curve and β_1 induces horizontal shifts of the curve. f and l denote the frame size and preamble size of a computation task offloaded to the UAV, respectively. We have $f > l$, as a frame must be longer than the preamble.

Algorithm 1 GNN-DRL for joint cruise control and task offloading

1: **1. Input**: $\mathcal{G}(\mathcal{V}, \mathcal{E}; \mathbf{w})$, $\mathcal{V}_\mathcal{G}$, S_α, A_{S_α}, t_{learning}, $\mathcal{M}_{\text{replay}}$.
2: **for** episode 1 to ep **do**
3: **for** Vertex $\mathcal{V} \in \mathcal{V}_\mathcal{G}$ **do**
4: **for** k = 1 to K **do**
5: $\mathbf{f}_\mathcal{V}^k \leftarrow$ Eq. (8). $\eta_\mathcal{G}^k \leftarrow$ Eq. (9).
6: $\eta_\mathcal{G}^k$ is minimized.
7: **end for**
8: **end for**
9: $< S_\alpha^\mathcal{G}, S_\beta, A_{S_\alpha}, R\{S_\beta | S_\alpha^\mathcal{G}, A_{S_\alpha}\} >$ is stored in $\mathcal{M}_{\text{replay}}$.
10: Minibatches in $\mathcal{M}_{\text{replay}}$ are randomly sampled.
11: $S_\alpha^\mathcal{G} \rightarrow$ the critic neural network, and the state-value function $\rightarrow V(S_\alpha^\mathcal{G})$.
12: The critic $\leftarrow R\{S_\beta | S_\alpha^\mathcal{G}, A_{S_\alpha^\mathcal{G}}\}$.
13: $A(S_\alpha^\mathcal{G}, A_{S_\alpha^\mathcal{G}})$ is updated by (13).
14: $S_\alpha^\mathcal{G}$ and $S_\beta \rightarrow$ the actor neural network.
15: (12) updates the policy gradient $\nabla\Gamma(\vartheta)$, and $A_{S_\alpha^\mathcal{G}}^* = \arg\min \Gamma_{\text{loss}}^c$ is carried out by the UAV in the environment.
16: **end for**

Algorithm 1 presents the proposed GNN-DRL that is carried out at the UAV for joint cruise control and scheduling the task offloading. Specifically, S_α is observed by the UAV and mapped to $\mathcal{V}$ in $\mathcal{G}(\mathcal{V}, \mathcal{E}; \mathbf{w})$. The actor neural network approximates cruise control and task offloading rules, and prioritizes data streams at the network state. The actor neural network can be updated by the policy gradient, i.e.,

$$\nabla\Gamma(\vartheta) = \mathbb{E}_\vartheta[\nabla_\vartheta \log \pi\{S_\beta | S_\alpha^\mathcal{G}, A_{S_\alpha^\mathcal{G}}; \vartheta\} A(S_\alpha^\mathcal{G}, A_{S_\alpha^\mathcal{G}})], \tag{12}$$

where $A(S_\alpha^\mathcal{G}, A_{S_\alpha^\mathcal{G}})$ denotes the advantage function. The advantage function measures the difference between the action-value function $Q\{S_\beta | S_\alpha^\mathcal{G}, A_{S_\alpha^\mathcal{G}}\}$ and the state-value function $V(S_\alpha^\mathcal{G})$ [20]. The advantage function can be estimated by the temporal difference (TD) error, as given by

$$\begin{aligned} A(S_\alpha^\mathcal{G}, A_{S_\alpha^\mathcal{G}}) &= Q\{S_\beta | S_\alpha^\mathcal{G}, A_{S_\alpha^\mathcal{G}}\} - V(S_\alpha^\mathcal{G}) \\ &\approx R\{S_\beta | S_\alpha^\mathcal{G}, A_{S_\alpha^\mathcal{G}}\} + \mu V(S_\beta | S_\alpha^\mathcal{G}, A_{S_\alpha^\mathcal{G}}) - V(S_\alpha^\mathcal{G}), \end{aligned} \tag{13}$$

where μ is a discount value ($0 < \mu < 1$).

The complexity of the proposed GNN-DRL algorithm can be given as $O(\mathcal{V}_\mathcal{G} \mathbb{R}^\mathcal{E} t_{\text{learning}} K \mathcal{M}_{\text{replay}})$, which can be performed at most of commercialized UAVs that are equipped with powerful onboard computation and data processing units.

V. Performance Evaluation

The proposed GNN-DRL is implemented in Python 3.9 with Google Tensorflow, setting up on a Linux workstation with 64-bit Ubuntu 18.04. N edge nodes are randomly deployed in a target field with a size of 1000 $\times$ 1000 meters. Initially, the location of the UAV is set to (600, 500) m. V_{max} is 15 m/s, and E of the edge node is 800 Joules. In addition, we compare the task offloading rate achieved by the proposed GNN-DRL framework with a DRL-based trajectory planning strategy without GNN in [21].

Figure 2 shows that the task offloading rate of the proposed GNN-DRL framework gradually grows with the growing number of training episodes. This is because the GNN module extracts the feature correlation of the network states and predicts the hidden network states, enriching the training environment of the DRL module. Hence, the action of the UAV is sufficiently trained by more state observations S_α with the growing number of episodes. This leads to the fast convergence of the proposed GNN-DRL framework, where the task offloading rate converges within about 150 episodes. Given 100 nodes and that 10% of the edge nodes generate new tasks in each time step (i.e., $\Delta m(t) = 0.1$), the task offloading rate drops from 96% to 81%. This is due to the limited cruising speed of the UAV. Consequently, the edge nodes far away from the UAV may not be served in time.

Figure 3 shows the task offloading rate in terms of different network scalability. We compare the proposed GNN-DRL framework with the DQN-based trajectory planning strategy [21] and the communication-aware trajectory planning (CATP) heuristic [22]. The task offloading rate generally decreases with the number of edge nodes. When N = 600, GNN-DRL outperforms the DQN and CATP solutions by about 43% and 62%, respectively. This is because the traditional DRL solution barely explores the hidden states, resulting in insufficient action training. The CATP heuristic hardly adapts the cruise control of the UAV to the unknown network state dynamics. Different from the DRL and CATP solutions, the proposed GNN-DRL framework extracts the features of the network state observation, and the hidden states are predicted to guide the actions in DRL to be sufficiently trained. Furthermore, GNN-DRL trains the actions of the UAV in a continuous domain, which optimally controls the real-time flight speed and heading of the UAV.

Figure 4 presents the flight trajectories of the UAV controlled by the proposed GNN-DRL framework, given the different learning iterations. Specifically, Figures 4(a) and 4(b) show the flight trajectories of the UAV when the training iterations of the GNN-DRL increase from 1500 to 3000. It is well-known that increasing the training iterations can sufficiently train GNN and DRL. Figure 4 validates that the GNN-DRL framework is able to effectively learn how to control the UAV's speed and heading, as well as how to allocate tasks for offloading, in order to maximize the

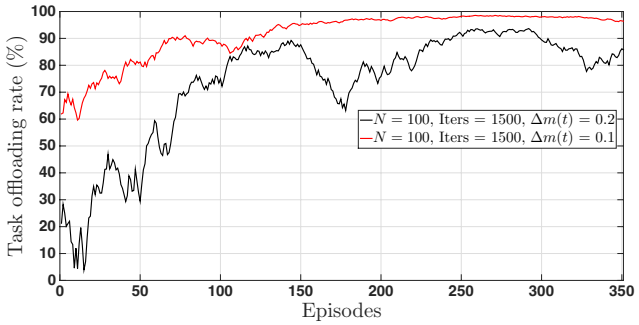


Fig. 2: Task offloading rate in terms of the training episodes.

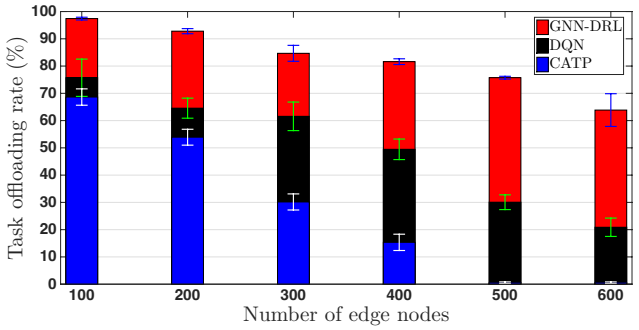


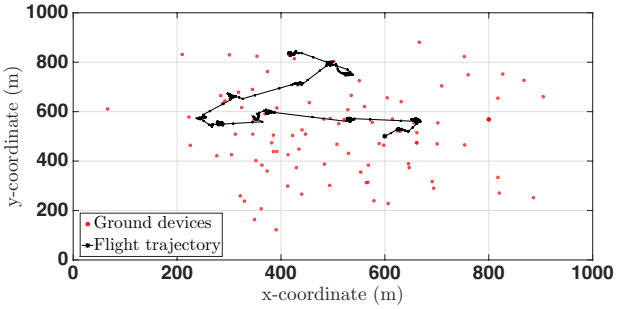
Fig. 3: The task offloading rate versus the number of edge nodes N, where M = 100. Each error bar is calculated with the standard deviation over 10 experiments.

task offloading rate. The use of GNN-DRL allows the UAV to adapt to changing network conditions and make optimal decisions in real-time.

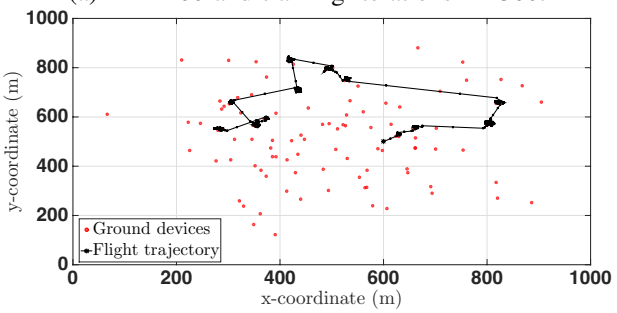
In Table II, we measure the runtime of the GNN-DRL and the DQN. It can be observed that GNN-DRL is about 6.8 $\sim$ 8.5 ms faster than DQN. This is because DQN-based strategy trains the action with a random process for the action exploration, which takes extra time. In contrast, the GNN-DRL predicts the future network states for training the UAV's actions in DRL at each episode, saving time on the random action exploration.

VI. Conclusion

In this paper, a new joint optimization of the UAV's cruise control and task offloading allocation was proposed to maximize the number of offloaded computation tasks. Since the optimization contains a large solution space while the instantaneous network states are unknown to the UAV, a new GNN-based DRL (GNN-DRL) was developed to take advantage of the GNN to supervise the training of UAV's actions in DRL, which explores the hidden network states according to the network feature correlation. The proposed GNN-DRL framework was implemented on Google Tensorflow. Numerical studies showed that the GNN-DRL significantly enhances the task offloading rate in the UAV-enabled MEC system while effectively controlling the flight cruise to process the computation tasks.



(a) N = 100 and training iterations = 1500.



(b) N = 100 and training iterations = 3000.

Fig. 4: Flight trajectories of the UAV, which is controlled by the proposed GNN-DRL framework given different training iterations.

TABLE II: Runtime of the GNN-DRL and DQN (ms).

ep	DQN	GNN-DRL
50	13.7	6.9
100	13.1	5.8
150	14.2	5.7
200	13.4	5.1
250	13.9	5.7
300	13.2	5.6
350	14.1	5.5

ACKNOWLEDGEMENTS

This work was supported by the CISTER Research Unit (UIDP/UIDB/04234/2020) and by project ADANET (PTDC/EEI-COM/3362/2021), financed by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology); also by National Funds through the FCT, under CMU Portugal partnership, within project CMU/TIC/0022/2019 (CRUAV).

REFERENCES

[1] Y. Wang, Y. Niu, H. Wu, S. Mao, B. Ai, Z. Zhong, and N. Wang, "Scheduling of uav-assisted millimeter wave communications for high-speed railway," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 8, pp. 8756–8767, 2022.

[2] K. Li, W. Ni, X. Wang, R. P. Liu, S. S. Kanhere, and S. Jha, "Energy-efficient cooperative relaying for unmanned aerial vehicles," *IEEE Transactions on Mobile Computing*, vol. 15, no. 6, pp. 1377–1386, 2015.

[3] P. Zhang, C. Wang, C. Jiang, and A. Benslimane, "UAV-assisted multi-access edge computing: Technologies and challenges," *IEEE Internet of Things Magazine*, vol. 4, no. 4, pp. 12–17, 2021.

[4] H. Kurunathan, H. Huang, K. Li, W. Ni, and E. Hossain, "Machine learning-aided operations and communications of unmanned aerial vehicles: A contemporary survey," *arXiv preprint arXiv:2211.04324*, 2022.

[5] K. Li, W. Ni, E. Tovar, and A. Jamalipour, "Online velocity control and data capture of drones for the internet of things: An onboard deep reinforcement learning approach," *IEEE Vehicular Technology Magazine*, vol. 16, no. 1, pp. 49–56, 2020.

[6] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: A survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2131–2165, 2021.

[7] B. P. L. Lau, B. J. Y. Ong, L. K. Y. Loh, R. Liu, C. Yuen, G. S. Soh, and U.-X. Tan, "Multi-AGV's temporal memory-based RRT exploration in unknown environment," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9256–9263, 2022.

[8] J. Suárez-Varela, P. Almasan, M. Ferriol-Galmés, K. Rusek, F. Geyer, X. Cheng, X. Shi, S. Xiao, F. Scarselli, A. Cabellos-Aparicio *et al.*, "Graph neural networks for communication networks: Context, use cases and opportunities," *IEEE Network*, 2022.

[9] Q. Liu, L. Shi, L. Sun, J. Li, M. Ding, and F. Shu, "Path planning for UAV-mounted mobile edge computing with deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5723–5728, 2020.

[10] L. Zhang, Z.-Y. Zhang, L. Min, C. Tang, H.-Y. Zhang, Y.-H. Wang, and P. Cai, "Task offloading and trajectory control for UAV-assisted mobile edge computing using deep reinforcement learning," *IEEE Access*, vol. 9, pp. 53 708–53 719, 2021.

[11] K. Li, W. Ni, and F. Dressler, "LSTM-characterized deep reinforcement learning for continuous flight control and resource allocation in UAV-assisted sensor network," *IEEE Internet of Things Journal*, 2021.

[12] K. Li, W. Ni, E. Tovar, and A. Jamalipour, "On-board deep q-network for UAV-assisted online power transfer and data collection," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 12 215–12 226, 2019.

[13] Z. Sun, Y. Mo, and C. Yu, "Graph reinforcement learning based task offloading for multi-access edge computing," *IEEE Internet of Things Journal*, 2021.

[14] J. Zheng, K. Li, N. Mhaisen, W. Ni, E. Tovar, and M. Guizani, "Exploring deep reinforcement learning-assisted federated learning for online resource allocation in privacy-preserving edgeiot," *IEEE Internet of Things Journal*, 2022.

[15] K. Li, R. C. Voicu, S. S. Kanhere, W. Ni, and E. Tovar, "Energy efficient legitimate wireless surveillance of UAV communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2283–2293, 2019.

[16] Y. Emami, B. Wei, K. Li, W. Ni, and E. Tovar, "Joint communication scheduling and velocity control in multi-UAV-assisted sensor networks: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 10, pp. 10 986–10 998, 2021.

[17] H. Huang, A. V. Savkin, and W. Ni, "Energy-efficient 3D navigation of a solar-powered UAV for secure communication in the presence of eavesdroppers and no-fly zones," *Energies*, vol. 13, no. 6, p. 1445, 2020.

[18] S. Hu, W. Ni, X. Wang, and A. Jamalipour, "Disguised tailing and video surveillance with solar-powered fixed-wing unmanned aerial vehicle," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 5507–5518, 2022.

[19] Y. Ma, Y. Niu, Z. Han, B. Ai, K. Li, Z. Zhong, and N. Wang, "Robust transmission scheduling for UAV-assisted millimeter-wave train-ground communication system," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 11, pp. 11 741–11 755, 2022.

[20] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[21] H. Kurunathan, K. Li, W. Ni, E. Tovar, and F. Dressler, "Deep reinforcement learning for persistent cruise control in UAV-aided data collection," in *IEEE Conference on Local Computer Networks (LCN)*. IEEE, 2021, pp. 347–350.

[22] A. Mardani, M. Chiaberge, and P. Giaccone, "Communication-aware UAV path planning," *IEEE Access*, vol. 7, pp. 52 609–52 621, 2019.