IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2022.Doi Number

Deep Reinforcement Learning-Based Computation Offloading in UAV Swarm-Enabled Edge Computing for Surveillance Applications

S. M. Asiful Huda, Sangman Moh, Member, IEEE
Department of Computer Engineering, Chosun University, Gwangju 61452, South Korea

Corresponding author: Sangman Moh (smmoh@chosun.ac.kr).

This research was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) Under Grant NRF-2022R1A2C1009037.

ABSTRACT The rapid development of the Internet of Things and wireless communication has resulted in the emergence of many latency-constrained and computation-intensive applications such as surveillance, virtual reality, and disaster monitoring. To satisfy the computational demand and reduce the prolonged transmission delay to the cloud, mobile edge computing (MEC) has evolved as a potential candidate that can improve task completion efficiency in a reliable fashion. Owing to its high mobile nature and ease of use, as promising candidates, unmanned aerial vehicles (UAVs) can be incorporated with MEC to support such computation-intensive and latency-critical applications. However, determining the ideal offloading decision for the UAV on basis of the task characteristics still remains a crucial challenge. In this paper, we investigate a surveillance application scenario of a hierarchical UAV swarm that includes an UAV-enabled MEC with a team of UAVs surveilling the area to be monitored. To determine the optimal offloading policy, we propose a deep reinforcement learning based computation offloading (DRLCO) scheme using double deep Q-learning, which minimizes the weighted sum cost by jointly considering task execution delay and energy consumption. A performance study shows that the proposed DRLCO technique significantly outperforms conventional schemes in terms of offloading cost, energy consumption, and task execution delay. The better convergence and effectiveness of the proposed method over conventional schemes are also demonstrated.

INDEX TERMS Aerial computing, computation offloading, deep reinforcement learning, double deep Q-learning, mobile edge computing, multi-agent reinforcement learning, unmanned aerial vehicle.

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) are considered a potential solution for performing critical tasks, such as traffic monitoring and surveillance, owing to their dynamic mobility and maneuverability [1]. Nevertheless, in practical scenarios, single-UAV systems often underperform because of their limited energy and payload. Hence, coordination between multiple UAVs (e.g., coalitions) to perform collaborative missions has emerged as a potential solution to this problem. Coalition-based networks can significantly enhance computational capability and ensure successful task completion, outperforming the performance of a single UAV by a large margin [2]. The use of a UAV swarm over a single UAV has several advantages, such as reducing the mission completion time, simultaneously performing multiple missions at different locations, and fault tolerance. This is possible because of the ability to complete the mission cooperatively towards a specific objective [3].

Although it is promising, a cooperative reconnaissance or surveillance task introduces new challenges. Assigning tasks inappropriately may degrade the network performance and increase the information transmission inside the coalition. In addition, tasks related to surveillance demand high computational capacity, such as video capturing, pre-processing, noise removal, and recognition. Owing to the limited payload, communication capacity, and storage, processing computationally intensive tasks such as this becomes challenging for a member UAV in a coalition, resulting in exceeding the local computation

capacity and prolonging the task completion time [4].

To tackle these challenges, mobile edge computing (MEC) is considered an essential solution for enabling computation offloading from UAVs to the edge node with high computational capacity [5]. Offloading the computing-intensive task to a nearby edge server happens via a wireless communication link, which significantly alleviates the problem of prolonged transmission delay to the cloud. However, as the conventional MEC servers located on the ground are constrained by a fixed geographical location, the UAV-assisted MEC architecture that combines MEC with medium and large UAVs with stronger computational capability for performing computationally intensive tasks can be a viable solution to address such a scenario.

Despite being a promising approach, it has been less explored in the existing literature. For UAV-enabled MEC networks, the existing literature emphasizes routing protocol [6], path planning [7], [8], area coverage [9], topology control [10], etc. These studies mostly focused on optimizing the rescue efficiency by minimizing the response delay and enhancing resource utilization. In a UAV-assisted surveillance system, UAVs perform the duty of covering a certain region and execute computationally intensive tasks in which both delay and energy consumption are crucial metrics [5]. In large-scale 3D areas, single-UAV systems often fail to successfully accomplish complex missions because of their limited energy capacity, although they provide sufficient coverage for a specific area. In such cases, a swarm of UAVs comprising many small and low-cost UAVs was observed to be effective in performing missions in large areas [9]. With the availability of a mobile edge server, the offloading computation task minimizes the task execution delay and energy consumption involved in offloading the task from the UAV.

In the UAV-MEC system, one of the most crucial decisions is where the task execution occurs (e.g., local execution, edge server). This may depend on various metrics such as the number of tasks, channel quality, UAV position, and edge nodes [11]. The UAV-enabled MEC system may fail to execute a task when the computational load increases significantly with a limited number of UAVs and insufficient computing resources. A promising solution is to utilize a UAV-enabled MEC system along with a base station (BS)-assisted MEC to enable the UAVs utilize the MEC services provided by the BS [12]. The offloading decision becomes more challenging under dynamic environmental conditions, where the characteristics of the network are highly dynamic. Thus, capturing accurate information to determine an offloading decision becomes difficult. Offloading decision-making has been extensively studied in the existing literature using conventional optimization techniques, such as convex optimization and heuristic techniques [13]–[15].

Although convex optimization methods can find the optimal solution when the problem formulation is simple and the number of nodes are limited, it is difficult to find the global optimal solution without knowing the complete information regarding the environment [16]. For example, global optimization methods require the problem formulation to be as simple as possible to enable it to be decomposed into subproblems. For example, to apply heuristic techniques, the problem formulation requires to be as simple as possible. An extensively used method is convex relaxation which refers to either the relaxation of the integer variables between binary values or the approximation of the binary constraints with quadratic constraints [17], [18]. Nonetheless, the solution obtained from relaxing the constraints is not guaranteed. Most importantly, because the conventional optimization methods statically try to find the optimal solution, the whole procedure needs to be re-run whenever the environment changes. As a result, traditional methods are not able to effectively address the high-dimensional state space as considered in this study, without complete information about the surrounding environment [19]. Thus, the increased number of network parameters and unknown environment characteristics makes the problem formulation even more complicated.

In such context, reinforcement learning (RL) has emerged as the excellent approach in order to address both the uncertainty and the underlying dynamic network characteristics of the environment. Most importantly, RL enables the agents to adaptively change their actions according to the change in the unknown environment, which is impossible in the traditional optimization techniques [20]. The mapping between the large state space and the action space is done, leveraging the impressive learning capability of deep neural networks (DNNs). Incorporating DNNs with RL yields deep reinforcement learning (DRL) algorithms such as a deep Q-learning network (DQN), which can find the optimal action from the large state space without any prior knowledge of the environment [21]–[23].

As discussed above, our work is motivated by the recent developments of RL as RL can handle large state space and action space comfortably. Specifically, in this paper, we focus on the implementation of a UAV swarm-enabled MEC system consisting of a multi-UAV network and a ground BS-enabled MEC that provides computational support to the UAVs in a surveillance mission. The end goal of the mission is to collect the computation task (either the video to be processed or the processed result) captured by UAVs and extract useful information by a specific team or individual who are assigned with the task of analyzing the video data at the ground base station with sufficient computational capacity. After successful processing of the task, certain feedback or outcome is sent back to the UAV for continuing the mission further. In particular, we consider a two-layer UAV-enabled MEC architecture comprising a head UAV (H-UAV), a team of member UAVs (M-UAVs), and a BS-assisted MEC to enhance the task execution efficiency for performing a video surveillance task to avoid unexpected occurrences during any sports event. When the number of tasks increases significantly and the processing capacity of the local edge server is at a maximum, the M-UAVs can offload the task to either the H-

UAV or ground edge server to avoid prolonged task completion delay. The offloading decision of the task depends on various factors, such as task size, task type, and computation capacity at the H-UAV. We aim to minimize the overall energy consumption of the UAV and the task completion delay under stringent delay requirements depending on the location of task execution.

To achieve this goal, we first formulate a weighted cost optimization problem by considering both energy consumption and task transmission delay for offloading the task in either of the three computational nodes in accordance with the offloading decision. The task type (either computation-intensive or delay-sensitive) is taken into the decision consideration along with other task metrics because it may cause task failure and require additional energy. Because the problem involves multiple integer constraints along with dynamic task and network dynamics, minimizing the overall cost using traditional optimization techniques is difficult. Thus, to handle the time varying task and network dynamics, we leverage DRL technique to minimize the overall total cost. Each M-UAV acts as an agent, which runs the DRLCO scheme to make the offloading decision based on the task characteristics and obtains an immediate reward to adaptively update its actions according to the network changes. Our proposed method can decide to offload the task more effectively based on the task requirements to solve the primary problems for UAV-MEC systems such as higher offloading cost, increased delay, and mission failure due to limited battery of the UAV.

A. CONTRIBUTION

In this paper, we propose a solution to address the limitations of existing surveillance and monitoring systems by leveraging coalitional UAV systems integrated with ground base stations (BS) in an aerial-ground cooperative paradigm. The main contributions of this study are summarized as follows:

- First, we introduce the integration of coalitional UAVs with ground base stations (BS) to offer the enhanced computing services. By integrating the coalitional UAVs with the ground BS, we can harness the potential of the aerial-ground cooperative paradigm to enhance computing service and support the increasing computational demand of smart applications. This integration allows for the provision of enhanced computing services, taking advantage of the combined resources and capabilities of both UAVs and the ground infrastructure.
- Second, we consider the heterogeneity of tasks that causes the difference in our objective function. Task-specific delays and energy requirements play a crucial role in determining the success of mission performance. Thus, we consider that M-UAV is generating task that can be classified as either resource (computation)-intensive or delay-sensitive.
- Third, considering the limited energy of the UAV, communication energy, task execution delay, and the task heterogeneity, we design the decision-making problem of computation offloading for the M-UAV as a weighted cost minimization problem considering both the energy consumption of the UAV and the task execution time. To minimize the weighted offloading cost, we formulate the problem as a DRL scheme. The state, action, and reward of the proposed offloading scheme are designed. To maximize the expected cumulative reward (by minimizing the weighted sum cost), we propose a DDQN-based fully decentralized DRLCO scheme, in which each M-UAV is an agent and can make the offloading decision using local observation in absence of any central controller.
- Finally, we conduct a comprehensive numerical simulation to verify the performance of the proposed method and compare our results with those of other conventional schemes (local computing, edge execution, and DQN) with varying parameter configurations. Simulation results show that our approach can outperform the conventional offloading schemes in terms of algorithm convergence, total offloading cost, task execution delay, and energy consumption.

B. OUTLINE OF THE PAPER

The remainder of this paper is organized as follows. In Section II, we review the literature. In Section III, we introduce the preliminaries of the system model, communication model, task computation model, and the RL framework. We present our proposed algorithm in Section IV. Section V presents numerical experiments and discussion. The conclusions of our study are presented in Section VI.

II. RELATED STUDIES

In this section, the background and research environment are discussed by providing an in-depth analysis of the existing literature pertaining to computational offloading in UAV swarm-enabled edge computing. In addition, we present the motivation for this study by describing the potential application scenario of our proposed solution and the significance of this study.

A. COMPUTATION OFFLOADING IN COALITIONAL UAV-MEC

In this subsection, we discuss existing related research on task offloading in coalitional UAV-MEC systems. You and Dong [9] studied a hierarchical UAV swarm comprising a leader UAV and follower UAVs to maximize the area-coverage efficiency in a surveillance scenario. The authors jointly considered the clustering of the UAVs to select the cluster head, transmit power, and relative positions of the UAVs to optimize the area coverage efficiency under delay constraints. In [24], the authors designed a response delay optimization framework in which they used two types of UAVs, denoted as T-UAV and B-UAV, to collect information more flexibly and efficiently. To model the interference between UAVs, the authors used stochastic geometry to derive a successful transmission probability among UAVs. In [22], the authors proposed a similar two-layer UAV architecture in which they attempted to minimize latency by considering both

communication and computational characteristics leveraging a DQN and deep deterministic policy gradient (DDPG) approach for both small and large action spaces. However, the proposed algorithm requires a longer convergence time than the DQN algorithm for a larger state space. Chen et al. [4] proposed a two-layer hierarchical UAV architecture to minimize the energy consumption of a UAV MEC network by jointly considering task offloading, channel allocation, and deployment. The authors used the Stackelberg equilibrium to design the interaction between the leader UAV and follower UAVs in a coalition based MEC network. The authors in [25] proposed a new technique that uses Stackelberg game theory to model the interaction between the edge service provider and mobile users during a disaster. The goal of the technique is to maximize the total utility and provide seamless connectivity and computing services to mobile users when terrestrial infrastructure is affected. Considering resource allocation, a multi-agent DDPG-based cooperative computation offloading approach was presented in [26] where the authors focused on maximizing the expected cumulative reward to reduce the computational cost and optimal resource allocation. They considered both aerial and ground MEC to manage tasks offloaded from the edge device. Because the study used a centralized controller, for a large number of devices, scalability can be a critical problem.

Unlike the aforementioned studies, the authors in [27] combined UAV with high altitude platform station (HAPs) with UAVs to address the limited computational power. This study jointly considered computational resource allocation, task splitting, and reuse of subchannels to formulate the resource allocation problem of the ground power Internet of things devices in a power grid to ensure seamless communication and computing services.

B. RL-BASED SOLUTIONS IN TASK OFFLOADING

RL-enabled solutions have appeared to be promising candidates for capturing the dynamic network characteristics of the UAV-MEC environment, such as data transmission rate, neighboring UAV decision, topology of the network, and the computational load, which are correlated with each other and are dynamic [28]. In particular, the inclusion of deep learning (DL) combined with RL enables the achievement of an optimal policy by utilizing the strong function approximation capability of DL models. Zhang et al. [29] demonstrated a single-UAV-based DRL technique for task offloading in hotspot areas to minimize the latency and energy consumption of the system by considering the offloaded task ratio and UAV trajectory. In [30], the authors presented a Q-learning algorithm in which each UAV interacts with the environment as an agent without communicating with other UAVs to reduce information exchange for providing cost-effective wireless communication via a flying BS. To address the overestimation problem in the DQN algorithm, Tang et al. [23] used a DDQN approach in a space-air-ground integrated network environment to address the highly dynamic state space where the offloading decision is determined based on local and neighboring information. In [31], the authors introduced a novel three-tier cloud-edge computing framework where mobile users need to pay for availing the computation services from the cloud service center. To jointly optimize service caching and resource allocation along with offloading strategy while meeting the delay constraint of the mobile users, the authors proposed a DRL-based asynchronous advantage actor-critic algorithm (DRLCOSCM) to solve the formulated mixed-integer nonlinear programming (MINLP) problem.

C. CHALLENGES OF DYNAMIC ENVIRONMENTS

Task offloading in dynamic environments poses several challenges because the computational tasks are data intensive and the environment characteristics are time varying. With increasing mission requirements (such as surveilling large areas or crowd surveillance), single or even multiple UAVs may not be sufficient to meet the on-demand communication and massive computation requirements. Mathematically formulating this problem using a traditional optimization technique is challenging in the presence of real-life constraints. To capture the maximum network dynamics and reduce the error regarding the offloading decision, devices must be aware of the dynamic properties of the neighboring devices. The system overhead increases even further with an increase in the number of UAVs. These challenges remain dominant in literature and require further study.

D. LIMITATIONS OF EXISITING TASK OFFLOADING SCHEMES

Existing studies on surveillance and monitoring mostly consider single-UAV systems that can provide only a shorter area coverage and service time owing to the limited flying time. When covering a large area, the number of computation tasks would increase significantly, which demands more computational resources. In multi-UAV systems, routing overhead and scalability become a crucial issue and communication becomes crucial with the increased number of UAVs in a large mission area. Additionally, existing studies consider a central controller in the network design that collects all the information and makes the offloading decision centrally for the agents. This increases the amount of information transmission significantly because the controller needs to communicate with each agent, which creates a single point of failure as well. Moreover, with the increased number of UAVs, the central controller cannot effectively handle the increased data and computational load as it deteriorates the system performance significantly. Thus, the scalability of the system is severely degraded. Furthermore, the assumption of a flying MEC server with sufficient computational capacity to execute tasks such as monitoring and surveillance is invalid in larger coverage areas [32]. Coordination between multiple large and small UAVs enables the collaborative surveying of a larger area and provides services for a longer period. Two major advantages of such network models are better line-of-sight links between the UAVs and the reduced communication distance due to the mobile nature of UAVs [12].

An alternative and promising paradigm that shows great potential is the integration of coalitional UAV systems with ground base stations (BS) in an aerial-ground cooperative framework. With increasing mission requirements, leveraging the computation and communication resources of both UAVs and the ground infrastructure (i.e., the BS) becomes an emerging research trend. By integrating coalitional UAVs with the ground BS, we can harness the potential of the aerial-ground cooperative paradigm to enhance computation service and support the increasing data rate demand of data intensive applications. While a few works have investigated the advantages of offloading in such systems, there still remain significant research opportunities in this area.

Motivated by the above shortcomings, in this study, we consider a team of decentralized UAVs and a ground BS-enabled MEC which is considered as an additional computation resource. With the inclusion of a ground MEC server, as the computational load of the M-UAV increases, the tasks can be offloaded to either the H-UAV with better computational resource or the BS-assisted MEC to reduce the computation overhead and ensure better mission performance. Based on the local observations of the agent (M-UAV), the agent can adaptively learn the network dynamics and take offloading decisions in a distributed manner using the proposed DRLCO scheme to minimize the overall offloading cost.

III. PRELIMINARIES

In this section, we present background knowledge consisting of the system model, channel model, and RL framework to provide a better understanding of the proposed model.

A. NETWORK MODEL AND ASSUMPTIONS

We consider a hierarchical UAV swarm-enabled MEC network with N coalitions (i.e., $\{Q_1, Q_2, \dots, Q_N\}$). Each coalition is responsible for its own set of UAVs with one designated coalition head (H-UAV) and multiple coalition members (M-UAVs) as shown in Fig. 1. The H-UAV is supposed to coordinate the activities of the coalition members to ensure successful task execution during the mission. This allows effective collaboration among the UAVs within the coalition with the improvement of task allocation, resource utilization, and overall mission performance. Each UAV in the coalition is equipped with a computing unit to perform computationally intensive tasks. We consider that H-UAV has a high computing performance compared with the coalition members owing to its high performance and sufficient energy. We represent the coalition members and coalition head as $\mathcal{M} = \{1,2, \cdots, M\}$ and $\mathcal{H} = \{1, 2, \cdots, H\}$, respectively, where M and H represent the number of M-UAVs and H-UAVs in a coalition Q_N, respectively. The horizontal coordinates of M-UAV j is denoted as $w_j^M = (x_j, y_j, h)$, where $j \in \mathcal{M}$. The H-UAV flies through a predefined trajectory to minimize the transmission delay. The horizontal coordinate of the H-UAV in time slot t is denoted by $w_t^H = (X_t, Y_t, H)$. Suppose, during a flight period of J, it is divided into T time slots equally. We denote the set of time slot as $\mathcal{F} = \{1, 2, \cdots, t, \cdots, T\}$.

We envision a widely used application scenario dedicated to public venue use (e.g., stadium) based on the ETSI framework [33]. Typically, ETSI considers stadiums as a potential use case requiring MEC services owing to the additional arrangements conducted during large sports events [34]. The M-UAVs are assumed to surveil a particular area and capture videos by using the onboard cameras, in order to process the videos using face recognition algorithms for detecting suspicious activities to avoid any unexpected threats during major sports events.

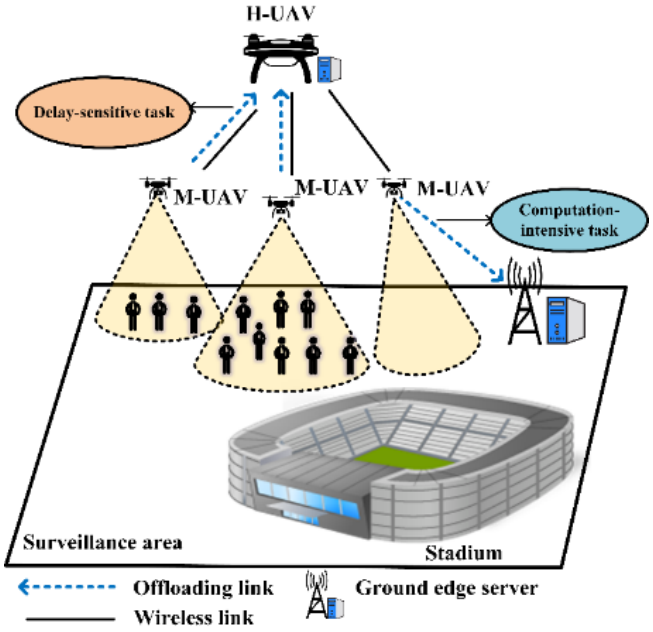


FIGURE 1. UAV-swarm-enabled mobile edge computing system.

When the computing resources are exhausted and computing tasks are prolonged in the M-UAVs, the task can be either offloaded to the H-UAV or to the ground edge server to assist the M-UAVs depending on the task characteristics. The tasks are classified into two types: computationally intensive or delay tolerant task and delay sensitive task.

If the task is delay sensitive and must be computed before a certain time period, the task is offloaded to the H-UAV to avoid transmission delay. Otherwise, if the task is delay tolerant and requires high computational power with more energy, the task is offloaded from the M-UAVs to the ground edge server via a wireless link that has a sufficient computing capacity. Our objective is to minimize the overall computational energy and transmission latency involved in this computation offloading scenario. We assume that the topology of the UAV has been optimized according to the task requirements; therefore, the M-UAVs remain in a quasi-static scenario during offloading the task as in [4]. The M-UAVs and H-UAVs are considered to fly at different altitudes to avoid collisions.

Since computation performance is the focus of this research, for a practical evaluation of our approach, we consider that the propulsion energy of the UAVs are sufficient to perform the mission and wireless communication as stated in recent other studies [4], [22], [35]. Thus, in this work, we focus only on optimizing the computation performance of our proposed method.

B. COMMUNICATION MODEL

We consider that the communication channels between the M-UAVs and H-UAV are characterized by line-of-sight communication, wherein the channel quality heavily relies on the communication distance [36], [37]. The distance between the H-UAV and M-UAV j in time slot t is denoted as

$$d_{j,t} = \sqrt{\left\| w_t^H - w_{j,t}^M \right\|^2}, \forall t \in \mathcal{F}, j \in \mathcal{M}. \quad (1)$$

We assume that the channel gain between the H-UAV and M-UAVs in coalition Q_N follows the free-space path loss model as follows:

$$h_{j,t} = \eta d_{j,t}^{-2} = \frac{\eta}{\left\| w_t^H - w_{j,t}^M \right\|^2}, \forall t \in \mathcal{F}, j \in \mathcal{M}, \quad (2)$$

where η denotes the channel gain, which is located at 1 m and relies on the antenna gain and carrier frequency.

We consider that between the time intervals, all the M-UAVs are served by the H-UAV by following frequency division multiple access (FDMA) [38]. The total bandwidth of system B is partitioned into G subbands without overlapping. In each time slot, each M-UAV is allocated $\frac{B}{G}$ subbands. Following this, as shown in [39], the signal-to-noise ratio is derived as follows:

$$SNR_{j,t} = \frac{\frac{h_{j,t} P_{j,t}}{\mu B/G}}{\frac{\eta P_{j,t}}{\frac{\mu B}{G \left\| w_t^H - w_{t,t}^M \right\|^2}}}, \forall t \in \mathcal{F}, j \in \mathcal{M}, \quad (3)$$

where $P_{j,t}$ denotes the transmit power of the j^{th} M-UAV, and μ indicates the spectrum density of the white Gaussian noise (WGN) in W/Hz at the H-UAV. Following Shannon's formula, the uplink data rate of M-UAV j in time slot t is given by

$$R_{j,t} = \frac{B}{G} log_2 \left(1 + \frac{\rho_0 P_{j,t}}{\mu B/G \left\| w_t^H - w_{j,t}^M \right\|^2} \right), \forall t \in \mathcal{F}, j \in \mathcal{M}. \quad (4)$$

Similarly, when the j^{th} M-UAV offloads the task to the ground edge server, we assume that the location of the edge server is fixed, and the horizontal coordinate of the edge server is denoted as $w_t^{EC} = (x_{EC}, y_{EC}, 0)$. Subsequently, the distance between the j^{th} M-UAV and edge server in time slot t is defined as

$$d_{EC,t} = \left\| \mathrm{w}_{j,t}^M - w_{\mathrm{t}}^{\mathrm{EC}} \right\|. \quad (5)$$

Because the UAV offloads computationally intensive tasks to the ground edge server, the channel gain between the j^{th} M-UAV and edge server in time slot t is denoted by

$$h_{EC,t} = \frac{\mathscr{g}}{[d_{EC,t}]^2}, \quad (6)$$

where $\mathscr{g}$ denotes the power gain and the reference distance is considered to be 1 m. Therefore, the transmission data rate between the j^{th} M-UAV and edge server in time slot t is defined as

$$R_{EC,t} = B_u \log_2 \left(1 + \frac{h_{EC,t} P_{j,t}}{\mu^2} \right), \forall t \in \mathcal{F}, j \in ,. \quad (7)$$

where B_u is the bandwidth pre-assigned to the edge server and μ is the noise power.

C. TASK COMPUTATION MODEL

In this paper, we assume that each M-UAV j in a coalition Q_N has a computation task that can be computed locally in the j^{th} M-UAV or either in the H-UAV or the ground edge server that is co-located with the BS at time slot t, where $t \in \mathcal{F}$. Let a_j^1 define the computation offloading decision of the M-UAV, where $a_j^1 = 1$ indicates that the M-UAV offloads the task, whereas $a_j^1 = 0$ indicates that the task is computed locally. We define each task as $K_{j,t} = (L_{j,t}, s_{j,t})$, where $L_{j,t}$ indicates the CPU cycles required to perform task of the j^{th} M-UAV, and $s_{j,t}$ indicates the data size that must be computed at time slot t. Next, we derive the computation cost in terms of the task execution time and energy consumption for local and edge computing.

1) LOCAL COMPUTING

We denote the computation capability, i.e., the clock frequency of the CPU chip, of M-UAV j for task $K_{j,t}$ as $f_{K_{j,t}}$. The local execution time of task K on M-UAV j is given by

$$T_{K_{j,t}}^{H-UAV,exe} = \frac{L_{j,t}}{f_{h,K}} \quad (8)$$

while the energy consumption of M-UAV j for executing task $K_{j,t}$ is given by

$$E_{K_{j,t}}^{loc,exe} = k L_{j,t} f_{K_{j,t}}^2 \quad (9)$$

where k indicates the switched capacitance of a specific chip architecture of the device. In accordance with previous studies, we consider that $k = 10^{-28}$ [40]. Consequently, the total cost for executing task K_j^t locally is defined by the sum of the local execution time and energy consumption during execution. That is,

$$U_{K_{j,t}}^{local} = \alpha_1^l \frac{T^{loc,exe}}{max\, T^{loc,exe}} + \beta_2^l \frac{E_{K_{j,t}}^{loc,exe}}{max\, E_{K_{j,t}}^{loc,exe}}, \quad (10)$$

where α_1^l and β_2^l are the weights that control the importance of the latency and energy consumption in the local computing phase.

2) TASK OFFLOADING

With an increasing number of tasks, the computation capacity of the M-UAV is exhausted owing to the shortage of computing resources. Thus, the tasks generated after a certain time are continuously dropped. In our paper, we consider that the M-UAV can offload the task to either the H-UAV or the ground edge server, depending on the task characteristics. As discussed earlier, we consider that tasks can be classified into two different categories: delay-sensitive and energy-sensitive tasks. The intuition for such a consideration is that deep-learning-based image recognition techniques involve many phases, which are of various types, with varying sizes and computational requirements. Therefore, for any task $K_{j,t}$, $a_j^2 \in \{0,1\}$ represents the action that the j^{th} M-UAV can perform. When the task is delay-sensitive, it is offloaded to the H-UAV, i.e., $a_j^2 = 1$; hence, the total delay consists of the transmission and computation delays at the H-UAV.

Thus, the transmission delay for offloading the task to the H-UAV is given by

$$T_{K_{j,t}}^{H-UAV,tx} = \frac{s_{j,t}}{R_{j,t}}. \tag{11}$$

Because the H-UAV has a higher computational capacity than the M-UAV, each H-UAV can execute the offloaded task locally using the computing unit onboard. Here, we assume that the H-UAV can run several virtual machines to compute tasks from different M-UAVs [41], [42]. Thus, we consider tasks from different M-UAVs are executed simultaneously. Hence, we ignore the computation capacity allocation in this paper [4], [43]. Therefore, when the H-UAV functions as a server, the computation delay and the energy consumption during transmission are given by

$$T_{K_{j,t}}^{H-UAV,exe} = \frac{L_{j,t}}{f_{h,K}} \tag{12}$$

and

$$E_{K_{j,t}}^{H-UAV,tx} = P_{j,t}\, T_{K_{j,t}}^{H-UAV,exe}, \tag{13}$$

respectively, where $f_{h,K}$ denotes the clock frequency of the H-UAV on task K. Meanwhile, the energy consumption of the H-UAV during task execution can be calculated by

$$E^{H-UAV,exe} = k_H L_{j,t} f_{h,K}^2 \tag{14}$$

where k_H represents the effective switched capacitance of the H-UAV related to the chip architecture; we set $k_H = 10^{-28}$ [40]. The total completion time of the task $K_{j,t}$ is defined by the sum of transmission delay from M-UAV j to the H-UAV and the execution delay at the H-UAV.

$$T_{K_{j,t}}^{H-UAV} = T_{K_{j,t}}^{exe,H-UAV} + T_{K_{j,t}}^{tx,H-UAV}. \tag{15}$$

The total energy consumption for processing the task $K_{j,t}$ in the H-UAV is defined by the sum of energy consumption during transmission and the energy consumption during execution.

$$E^{H-UAV} = E^{H-UAV,exe} + E_{K_{j,t}}^{H-UAV,tx}. \tag{16}$$

As a result, the total cost for executing the task on H-UAV is given by

$$U_{K_{j,t}}^{H-UAV} = \alpha_1^h \frac{T_{K_{j,t}}^{H-UAV}}{T_{K_{j,t}}^{H-UAV}} + \beta_2^h \frac{E^{H-UAV}}{max\, E^{H-UAV}}. \tag{17}$$

However, because deep learning techniques often require extensive computation (e.g., matching face images from existing datasets), tasks can be computationally intensive and require more time, executing such tasks at the H-UAV may degrade the overall network performance. Thus, we consider that the M-UAV can also offload the task to the edge server for edge execution with a strong computation capacity, i.e., $a_j^2 = 0$. In this case, execution occurs in three phases: (i) Task transmission stage, (ii) edge computing stage, and (iii) result transmission stage.

In the first phase, we derive the total cost by considering the energy consumption during the transmission and execution of the task. The transmission time and energy consumption of the task $K_{j,t}$ at the edge server are given by

$$T_{K_{j,t}}^{edge,tx} = \frac{s_{j,t}}{R_{EC,t}} \tag{18}$$

and

$$E_{K_{j,t}}^{edge,tx} = P_{j,t}\, T_{K_{j,t}}^{edge,tx}. \tag{19}$$

For edge execution, the computing task execution time is denoted by

$$T_{K_{j,t}}^{edge,tx} = \frac{s_{j,t}}{f_e}, \tag{20}$$

where f_e denotes the CPU frequency of the ground edge server.

We assume that the frequency remains constant during task execution. Owing to the high computational capacity and sufficient power of the ground edge server, the edge server can easily complete the offloaded task. Corresponding with other studies, e.g., [44] and [45], we omit the result receiving delay because the size of the returned data is exceedingly small. We also ignore the edge energy consumption in the total offloading cost calculation because the energy consumption of the ground edge server is negligible. Thus, the total duration

for completing the execution of the task $K_{j,t}$ in the edge server is given by

$$T_{K_{j,t}}^{edge} = T_{K_{j,t}}^{edge,tx} + T_{K_{j,t}}^{edge,tx}. \quad (21)$$

The total cost of edge execution is given by

$$U_{K_{j,t}}^{edge} = \alpha_1^e \frac{T_{K_{j,t}}^{edge}}{max\ T_{K_{j,t}}^{edge}} + \beta_1^e \frac{E_{K_{j,t}}^{edge,tx}}{max\ E_{K_{j,t}}^{edge,tx}}. \quad (22)$$

Because of the computation capacity among the three computation nodes, based on where the task is being executed, we introduce different weights to enable diversity in the delay and the energy consumption calculation of the three cases. This means α_1^l is different from α_1^h and α_1^e. The same applies for β_2^l, β_2^h, and β_2^e.

D. PROBLEM FORMULATION

In this paper, our aim is to minimize the total weighted cost by considering both task execution delay and energy consumption. When all tasks are locally computed, an M-UAV cannot complete all tasks owing to its limited energy and computational capacity. In addition, latency is another crucial metric in an edge-computing environment that can significantly deteriorate performance. Thus, an optimal task allocation strategy (H-UAV, edge server) is required by considering execution time and energy consumption. Therefore, we jointly consider the energy consumption and execution delay during transmission and computing. Because we have a multi-objective optimization problem, we consider a popular multi-objective optimization method: the linear weighted sum method [38], [46]. This method combines multiple objectives into a single objective function. Because we consider that the computing happens either (i) within the strong head UAV (H-UAV) or (ii) within one of the member UAVs (M-UAVs) or (iii) in the ground edge server, we have three different utility functions for the three computing nodes. The choice of utility function depends on the offloading decision. Thus, our utility function enables modeling the interaction in such architecture.

The execution time and energy consumption are normalized such that they are within the same range. To normalize, we divide the task execution delay and energy consumption by the corresponding maximum delay and maximum energy calculated in each case. Thus, we convert the two different metrics into the same number range. Subsequently, we apply different weights for both energy consumption and delay to configure the video analysis based on task requirements. Based on the above discussion, the objective function for a sequence of tasks $\mathcal{R}$ can be formulated as follows:

$$U_j = \sum_{K=1}^{\mathcal{B}} U_{K_{j,t}} = \sum_{i=0}^{I} a_j^1 a_j^2 U_{K_{j,t}}^{H-UAV} + a_j^1\left(1 - a_j^2\right)U_{K_{j,t}}^{edge} + \left(1 - a_j^1\right)U_{K_{j,t}}^{local}\Big), \quad (23)$$

where $\mathcal{B}$ indicates the total size of set $\mathcal{R}$. Thus, the optimization problem can be formally derived as

$$\min_{A} \sum_j U_j \quad (24)$$

$$\text{s.t.} \quad a_j^1 a_j^2 T_{K_{j,t}}^{H-UAV} + a_j^1\left(1 - a_j^2\right)T_{K_{j,t}}^{edge} + \left(1 - a_j^1\right)T^{loc,exec} \leq T_{K_{j,t}}^{max}, \forall K = 1, \cdots, \mathcal{B} \quad (25)$$

where $A = \left\{ a_j^1, a_j^2 \middle| j \in \mathcal{M}, K \in \mathcal{R} \right\}$. This constraint states that all tasks must be completed by the total completion time, T_K^{max}.

To solve the problem in (25), we need to find the optimal offloading action a_j^1 and a_j^2 under the delay constraint in each time slot. It should be noted that a_j^1 and a_j^2 are integer decision variables that represent the decisions in the formulated problem to reduce the overall cost. The system requires extensive network information (e.g., the task information and remaining computational capacity of the UAVs) to make the optimal offloading decision based on the current state. Here, the UAVs need to determine the offloading decision based on the arriving task information collected within a time slot, which is unknown beforehand because of the time-varying environment. Furthermore, with the increased number of UAVs, the complexity increases exponentially as well. Therefore, the formulated objective function is an MINLP problem which is generally NP-hard [31], [47]. The optimal set of the offloading decision is non-convex, and traditional optimization techniques cannot make intelligent offloading decision considering the changing network characteristics. Thus, we designed a multi-agent reinforcement learning algorithm that can provide a simpler and more efficient solution in less time.

E. RL FRAMEWORK

As formulated above, it is difficult to solve the optimization problem using the traditional optimization techniques due to the following issues:

1. The task characteristics and future computational capability of the UAVs depending on the offloading decision in such a dynamic environment is a subtle problem. As the tasks arrive dynamically and the offloading decision is determined based on the task-specific requirements, the traditional optimization techniques cannot handle such optimization problem.

2. Traditional convex optimization techniques as well as heuristic algorithms can find the optimal solution well when the problem formulation is simple, and the number of UAVs is limited. However, with the increased number of UAVs, the offloading decision becomes challenging as the non-convex problem would become very difficult to solve, resulting in long convergence time. This raises the scalability issue as well.

3. The offloading decision depends on the mission-specific task requirements. Thus, the agents need to adapt to any

change in the environment according to the environment's characteristics. However, using the traditional optimization techniques, it would require redesigning and rerunning the solution again to find the optimal offloading policy whereas, using RL, the agents can simply learn the new task-specific characteristics through trial-and-error method without any prior knowledge of the environment.

4. Furthermore, in a multi-agent setting with the increased number of agents in the system, the state and action space grows exponentially. Utilizing function approximator (e.g., deep neural networks) in DRL can find the underlying pattern and estimate the optimal action from the large state space by finding the mapping between them in a reliable fashion. Thus, using DRL over the traditional techniques can address the scalability issue, improve communication efficiency, and overcome the limitations of complex problem formulation incurred in the traditional optimization techniques.

To address the above shortcomings in the traditional optimization techniques, we propose the use of DRL, a model-free method, where each agent (e.g., M-UAV) aims to make the offloading decision quickly based on the local observation to reduce the amount of information exchange between agents. In traditional RL, the problem is modeled as an MDP. According to the RL framework, we provide the definitions of state space, action space, and the reward function of our stated problem in the next subsection.

1) STATE SPACE $m_{j,t}$

To make the offloading decision, each agent in the network is provided with a set of input metrics that they consider while making the offloading decision. Let $m_{j,t} = \{D, c, f, \text{and } dt\}$ denote the state space. The meanings of these symbols are provided by

- D: size of the task
- C: cycles needed to complete the task
- f: computational capability of the H-UAV
- dt: task type $\in \{0,1\}$ (energy-sensitive or delay-sensitive)

As for the task type, it can be easily determined by the maximum tolerable latency to compute the task. Tasks with less tolerable latency can be denoted as latency-critical tasks whereas tasks with considerable latency, larger size and required CPU cycles can be indicated as computation-intensive tasks. As in the existing literature, we also used the take type in line with in the below references [22], [48], [49] for the same purpose of simplicity. In reality, however, the task type can be determined either based on the task features or from previous similar mission data.

2) ACTION SPACE $a_{j,t}$

Each M-UAV selects a particular action after the tasks are generated in time slot $t \in \mathcal{F}$ in the coalition. The M-UAV can select an action from local information. In this paper, we consider binary offloading, which can either be executed at the H-UAV or offloaded to the ground edge server.

3) REWARD FUNCTION $r_{j,t}$

The aim of each agent is to maximize the total reward. We assume that each agent has the same reward function. Based on the discussion above, the reward function is defined as follows:

$$Z_{j,K} = -a_j^2 a_j^1 \frac{U_{K_{j,t}}^{H-UAV}}{max\, U_{K_{j,t}}^{H-UAV}} - a_j^1\left(1 - a_j^2\right)\frac{U_{K_{j,t}}^{edge}}{max\, U_{K_{j,t}}^{edge}} - (1 - a_j^1)\frac{U_{K_{j,t}}^{local}}{max\, U_{K_{j,t}}^{local}}, \quad (26)$$

where a higher cost results in a smaller reward, and vice versa. Thus, for each agent, the utility is

$$Z_j = \sum_{K=1}^{\mathcal{B}} Z_{j,K}. \quad (27)$$

Each agent aims to maximize this objective that emphasizes the action that produces a better reward value. Because action selection depends on the unique characteristics of the network dynamics in which the agent is interacting, defining the reward function directly from the obtained utility would affect the learning process. The intuition for such a consideration is that an increase in the reward denotes an improvement, such that a particular action should be emphasized [50]. Therefore, the difference between reward values in immediate timesteps are considered as the reward in this paper. Based on the discussion above, the reward value is defined as follows:

$$u_t^j = \begin{cases} p, \ if\ Z_{j,t} - Z_{j,t-1} < 0 \\ q, \ if\ Z_{j,t} - Z_{j,t-1} > 0, \\ 0, \quad otherwise \end{cases} \quad (28)$$

where Z_j^t is the cumulative reward gained at timeslot t. When the reward difference between two immediate timesteps is negative, it is considered an improvement because the reward is formulated as a negative value; thus, p is a positive value, whereas a positive difference between the rewards obtained from immediate timesteps yields q which is a negative reward. For simplicity, we consider that each agent shares the same reward.

IV. DEEP REINFORCEMENT LEARNING BASED COMPUTATION OFFLOADING

Based on the previous discussion, in this section, we propose a decentralized multi-agent computation offloading (DRLCO) algorithm that incorporates a neural network and Q-learning to obtain an optimal offloading decision as shown in Fig. 2. In each time slot, when new tasks are generated to be computed locally or on the MEC server, each agent (M-UAV) runs the DRLCO algorithm locally and learns the offloading policy in a distributed manner to select an action according to its local knowledge and receives an immediate reward for a particular action. By combining the perceptive characteristics of the neural network and decision-making capability of RL, we can determine the optimal offloading decision in a

dynamic environment. Because the tasks to be computed are not known beforehand, the DRLCO algorithm was designed to fall under the category of model-free RL. Algorithm 1 presents the proposed DRLCO algorithm.

Two identical neural networks, whose Q functions are denoted as $Q_j^t(m_{j,t}, a_{j,t}|\varphi_j^e)$ and $Q_j^e(m_{j,t}, a_{j,t}|\varphi_j^e)$, are used to construct the overall structure of the DRLCO, which we call the target network φ_j^t and evaluation network φ_j^e. To enhance the training efficiency and early convergence, we use the experience replay $O_k(t)$ to store past samples. The sample of experiences is defined as $\{m_{j,t}, a_{j,t}, u_{j,t}, m'_{j,t}\}$. The intuition for using experience replay memory is that consecutive samples are highly correlated, which may affect the learning process and result in sample inefficiency. To break this correlation, we use a mini-batch of random samples to train the model, which is stored in the replay memory. As memory fills up, old experiences are removed to create space for newer ones. Each agent (M-UAV) leverages replay memory to determine the optimal mapping between the state and action. We consider that the agent selects a certain action a_j^t using the following ϵ-greedy:

$$a_j^t = \begin{cases} random, & \epsilon \\ arg\ \max\ Q_j^e(s_j^t, a|\varphi_j^e), & 1-\epsilon \end{cases}. \quad (29)$$

In DRLCO, two neural networks aim to determine the optimal action, as stated previously. In a typical DQN, the max operator is used to select and evaluate an action that results in the overestimation of values, causing overoptimistic value estimates [51]. Therefore, to address this challenge, we use two identical networks to separate the task of selecting and evaluating an action, which enables the evaluation of the greedy action taken. Based on this concept in DRLCO, the action that produces the highest Q-value is first selected by the target network. Subsequently, the evaluation network evaluates the selected action by calculating the Q-value of taking that action in that state. Thus, the value of the policy is evaluated evenly using the two neural networks. The expected cumulative reward of the target network can be derived as

$$Rew_j = u_{j,t} + \mu_k Q_j^t(m'_{j,t}, \arg\max Q_j^e(m'_{j,t}, a_{j,t}|\varphi_j^e)|\varphi_j^t). \quad (30)$$

where μ_k denotes the discount factor for controlling further rewards. Therefore, the loss between $Q_j^t(s_j^t, a_j^t|\varphi_j^e)$ and $Q_j^e(s_j^t, a_j^t|\varphi_j^e)$ is calculated as follows:

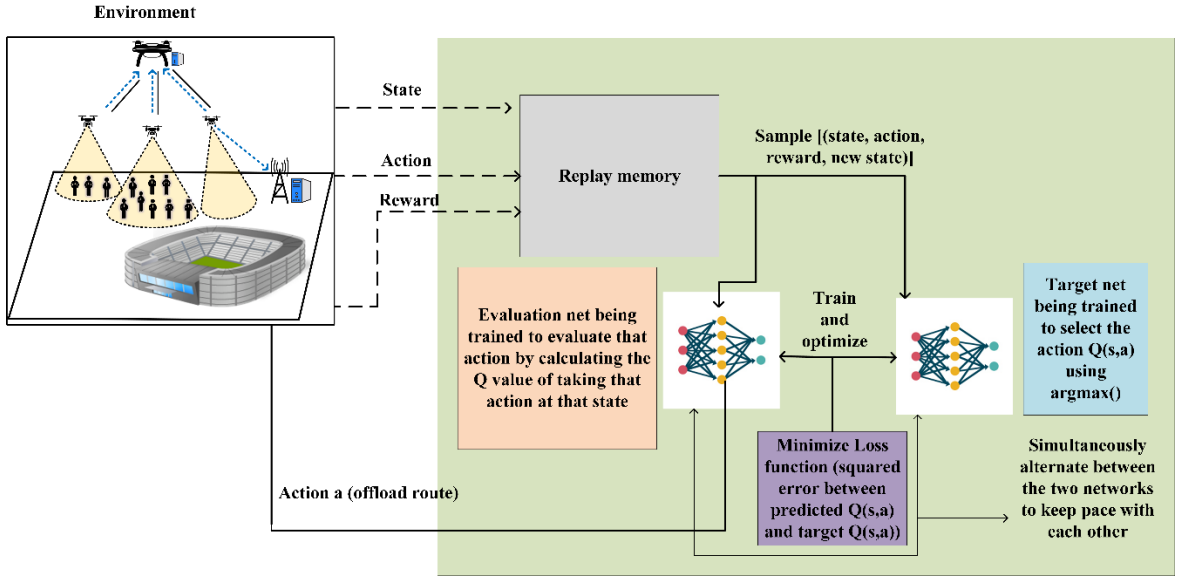


FIGURE 2. **Block diagram of the proposed DRLCO scheme.**

$$Loss_j(Q_j^e, Q_j^t) = \mathrm{E}_{(m_{j,t}, a_{j,t}, u_{j,t}, m'_{j,t})\sim O_k}[Rew_j - Q_j^e(m'_{j,t}, a_{j,t}|\varphi_j^e))]^2. \quad (31)$$

The two networks are alternated simultaneously to ensure stability in the training performance. The target network parameters are updated using the evaluation network until convergence. Thus, each agent can learn the optimal offloading policy in a distributed manner according to its own information. Since each agent can only observe the local information, we incorporate global knowledge by designing identical rewards (average reward) for all the M-UAVs (agents) for better convergence. Thus, all agents (M-UAVs) share the same goal of maximizing the overall reward. The DRLCO algorithm, as provided in Algorithm 1, has two major parts: collecting data samples from the network environment and training based on the collected data. The parameters of the target network and the parameters with initial weights are defined initially along with the replay memory size (lines 1−2). The number of episodes is then defined, and agent begins interacting with the network environment to collect data (lines 3 − 11). In this data-

collection phase, each agent (M-UAV) observes the state, selects an action, receives the reward, and receives a new state (lines 8−10). The replay memory stores the experience gained. Subsequently, in the training phase, a random mini batch from the replay memory is sampled to train the agent and calculate the loss (lines 12−14). Thereafter, based on the calculated loss value, the evaluation network is updated, and parameters are alternated between the two networks to keep the training balanced (lines 15−16). The flowchart of the proposed algorithm is shown in Fig. 3.

Algorithm 1: DRLCO Algorithm
Input: Task feature $\{D, C, f, dt\}$
Output: Optimal offloading location for a given input
1 Initialize parameters of target and evaluation networks for all M-UAV ∈ $\mathcal{M}$;
2 Initialize replay memory χ_j for each agent M-UAV ∈ $\mathcal{M}$;
3 **for** episode = 1 to $episode^{max}$ **do**
4 Reset entire environment for each M-UAV ∈ $\mathcal{M}$;
5 **for** j = 1 to M **do**
6 M-UAV acts on the dynamic environment;
7 **for** t = 1 to T **do**
8 M-UAV observes the state $m_{j,t}$ parameters consisting of task size D, required cycles to execute the task C, computational capability of the H-UAV f, task type dt;
9 M-UAV selects action $a_{j,t}$ regarding offloading the task to the H-UAV or the ground edge server following ϵ-greedy policy;
10 M-UAV obtains the reward u_t^j, next state $m'_{j,t}$;
11 Add sample $\{m_{j,t}, a_{j,t}, u_{j,t}, m'_{j,t}\}$ into replay memory χ_j when replay memory is not full;
if samples are sufficient in $O_k(t)$, **do**
12 Select a mini batch from replay memory $O_k(t)$;
13 Calculate cumulative reward using (30);
14 Calculate loss using (31);
15 Update the evaluation network φ_j^e;
16 Alternate the parameters from evaluation network to target network ($\boldsymbol{\varphi_j^t} \leftarrow \boldsymbol{\varphi_j^e}$);
17 **end if;**
18 **end for**
19 **end for**
20 **end for**
21 **return** offloading location for given input

Because there is only one output layer, and $O(i_w)$ is the complexity of the total number of activation functions, the complexity of these two metrics has been ignored with regard to the overall complexity.

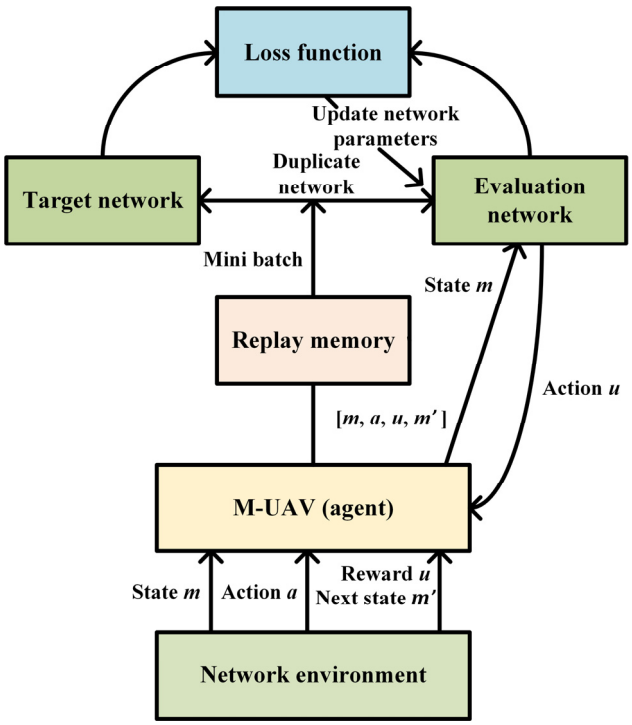


FIGURE 3. **Flowchart of the proposed methodology**

V. PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed DRLCO scheme and compare it with that of conventional schemes through a simulation study.

A. EXPERIMENTAL SETUP

We verified the correctness of our proposed DRLCO using Python and TensorFlow 1.15 on a computer with an AMD Ryzen 5 processor with a processor speed of 3.60 GHz and RAM of 8 GB. We considered five H-UAVs and 15 M-UAVs in our UAV swarm-enabled edge-computing scenario in an area of 1000 m × 1000 m. The task arrival rate at the agent M-UAV follows a uniform distribution. To simulate the DRLCO algorithm, we set that for each M-UAV, the neural network of the DRLCO algorithm consists of an input layer, an output layer, and two FC layers. The sizes of the two hidden layers are 400 and 350. We used ReLU in the hidden layers, as the activation function and AdamOptimizer to optimize the loss

function. For training, we set the maximum number of iterations to 2000. The size of the replay memory is 50000.

To emphasize future rewards, we set the discount factor value to 0.9. Because $\alpha = 0.001$ yielded a higher reward and stable training, in the simulation environment, we set the value of the learning rate as $\alpha = 0.001$. We define equal values for the weights considered in the objective function of our study to have an equivalent emphasis on the total cost evaluation. Depending on the task characteristics (energy-centric or delay-centric), these weights can be adjusted according to the mission specific requirements. As such, the weights of energy consumption and task execution delay are set as $\vartheta_1^l = \vartheta_1^h = \vartheta_1^e = \vartheta_2^l = \vartheta_2^h = \vartheta_2^e = 0.5$ in the experiment. The computational task size varies randomly between 2 Mb and 20 Mb. The computational capacities of M-UAV, H-UAV, and ground edge server are 1.5, 15, and 20 GHz, respectively. The major simulation parameters are listed in Table I.

B. CONVERGENCE ANALYSIS

To verify the convergence of the proposed algorithm, we evaluate the average cumulative reward using different parameters. Because we added a delay constraint in our problem formulation, we introduce a penalty factor of 3 in the reward value when the tasks exceed the deadline to control the learning progress.

TABLE I
SIMULATION PARAMETERS

Parameter	Value
Number of H-UAVs	5
Number of M-UAVs	15
Bandwidth (B)	20 MHz
Bandwidth preassigned to edge (B_u)	1 MHz
Height of H-UAV (H)	1500m
Transmission power ($P_{j,t}$)	20 dBm
Channel power gain (η)	1.42×10^{-4}
Power spectrum density (μ)	-174 dBm
Path loss exponent ($\mathcal{g}$)	-50 dB
CPU cycles required to complete task ($L_{j,t}$)	0~1.5 GHz
Computation task size of each M-UAV ($s_{j,t}$)	2 – 20 Mb
Computation capacity of M-UAV ($f_{j,K}$)	1.5 GHz
Computation capacity of H-UAV ($f_{h,K}$)	15 GHz
Computation capacity of ground edge server (f_e)	20 GHz
Effective switched capacitance of M-UAV and H-UAV (k, k_H)	10^{-28}
Weights($\alpha_1^l = \alpha_1^h = \alpha_1^e = \beta_2^l = \beta_2^h = \beta_2^e$)	0.5
1st hidden layer size	400
2nd hidden layer size	350
Learning rate (α)	1×10^{-3}
Mini batch size	100
Experience replay size (O_k)	50000
Discount factor (μ_k)	0.9
Total episode ($episode^{max}$)	2000
Total timesteps of per episode (T)	100
Deadline for completing the task ($T_{K_{j,t}}^{max}$)	8 time slots
Length of each slot (t)	1 s

With varying network dynamics (e.g., task size, computation capacity, and task type), the reward varies and so does the cumulative reward. Thus, the curves are oscillating at the beginning. We set the values of μ_k as 0.9, $\alpha_1^l = \alpha_1^h = \alpha_1^e = \beta_2^l = \beta_2^h = \beta_2^e = 0.5$ and total timesteps of per episode as 100. We observe that when α = 0.1 or 0.01, the proposed scheme cannot find the optimal policy and takes much time to converge to an optimal value. Thus, with proper hyperparameter tuning, we noticed DRLCO achieves better reward and stable learning when learning rate α = 0.001 as shown in Fig. 4. Therefore, we performed all the experiments in this study at this setting to get better reward during the learning period.

When comparing the DRLCO scheme with the DQN algorithm as shown in Fig. 5, we observed that both the DRLCO and DQN techniques converge after a certain time, and DRLCO reaches a steady state earlier than DQN. Initially, because the M-UAV has no knowledge of the system environment, it takes random offloading actions resulting in lower reward value. Thus, both techniques of DRLCO and DQN fluctuate at a very low value because of the agent's random action selection. Gradually, the M-UAV starts to receive feedback based on the selected offloading actions, begins to collect samples, and trains the network when the replay memory has sufficient samples. Subsequently, the agent can make better decisions based on both the local observation and the neighboring agents' actions through the global reward.

As shown in Fig. 3, we can observe that the curve is flattened after around 1000 episodes that further proves the effectiveness of the proposed technique in choosing the offloading decision. Because the DRLCO scheme solves the overestimation problem of Q-values by separating the estimation into two networks, it can learn valuable states without observing the impact of each action at every state. Thus, we can observe that the reward of the DRLCO scheme is significantly higher than that of the DQN scheme. In contrast, the DQN algorithm consists of a single neural network, namely, a value function network that utilizes a random policy for training. Our DRLCO scheme requires approximately only 250 episodes to reach a stable state and it provides a higher reward compared with the DQN scheme because of the introduction of the target network in DRLCO, which aids in converging faster.

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2023.3292938

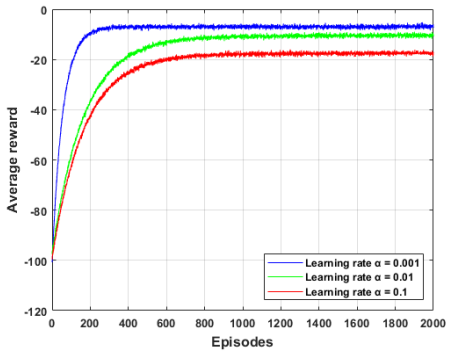


FIGURE 4. **Average cumulative reward under varying learning rate.**

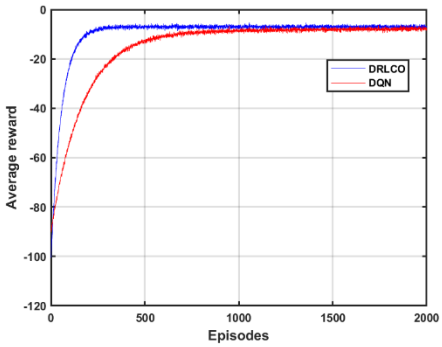


FIGURE 5. **Convergence of DRLCO and DQN**

C. PERFORMANCE METRICS

After completing the training process of the proposed DRLCO, we verified our proposed scheme to validate our model by performing experiments on various performance metrics. Intuitively, the main objective of the proposed multi-agent computation offloading algorithm is to successfully execute the task with the lowest total cost. Because the M-UAV can execute the task locally or through offloading into the H-UAV or the ground edge server, the energy consumption as well as delay involved in the transmission and computation of the task were considered in our performance analysis along with the total offloading cost. This is because both energy consumption and delay are crucial offloading metrics, and determining the optimal offloading policy, which has less delay and consumes less energy, indicates the optimal offloading policy. Before quantitatively verifying the performance of the proposed DRLCO with other benchmarks, we briefly discuss the performance metrics below.

- Average offloading cost: We discuss the average offloading cost in terms of offloading task data size, computation capacity of the H-UAV, and number of M-UAVs. This is the average cost of all the agents (M-UAVs) for performing the task.
- Task execution delay: Delay is another crucial metric for determining the network performance and improving the quality of service of the system. Reducing the delay involved in the offloading and execution of a task can significantly reduce the total system overhead. Thus, to demonstrate the effectiveness of this study, we formulated task execution delay as the sum of both transmission and processing delays and compared it with conventional schemes.
- Energy consumption: In a UAV-enabled edge computing system, the total energy consumption for offloading a task is the most important metric. Inefficient task allocation may result in high overhead and tasks being dropped. The total energy includes both the energy required for the transmission and the execution of the task.

To validate the effectiveness of the proposed DRLCO scheme, we considered the following three conventional methods and compared the performance of our study with these methods.

- Local: In this setup, tasks were executed locally by each M-UAV in each time slot to the maximum computation capacity.
- Edge: All tasks were executed on a ground edge server, which had sufficient computing capacity.
- DQN: We applied the DQN algorithm in our proposed scenario as a benchmark technique because the action space is also discrete in the DQN.

D. SIMULATION RESULTS

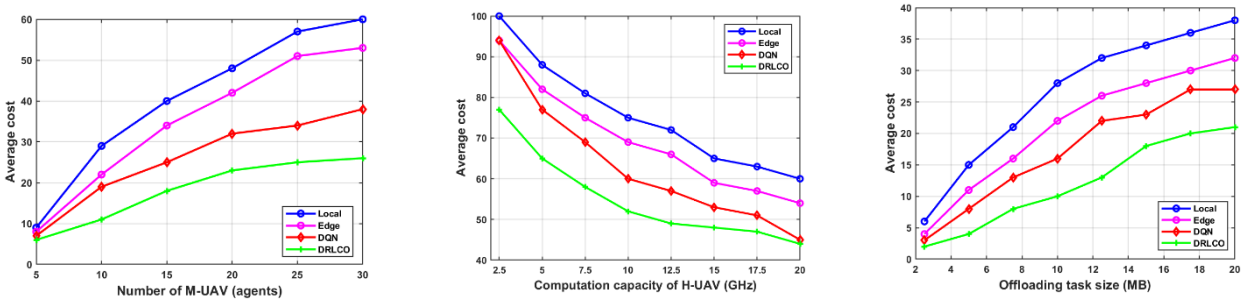
In this section, we present the simulation results of the proposed DRLCO algorithm and compare the with benchmark techniques. First, we analyze the average cost with respect to the number of M-UAVs, computation capacity of the H-UAV, and offloading task size.

To evaluate the scalability of the proposed DRLCO in a large mission area, in Fig. 6(a), we evaluate the impact of increasing the number of agents (M-UAVs) on the average cost for executing the large number of computational tasks. We observed that with the increased number of agents (M-UAVs), more surveillance tasks need to be computed, which eventually increases the overall offloading cost due to the increase in the transmission and execution delay. Aside the number of agents, there can be different possibilities such as the large number of tasks. Although the total average cost increases with the increased number of agents, DRLCO is able to reduce the average cost compared with the local, edge, and DQN, respectively. Thus, we can conclude that, with an increased number of agents, DRLCO can handle the scalability issue outperforming the three benchmarks in terms of the number of agents. However, when the number of tasks is small, it is not appropriate to deploy more UAVs for computation.

Fig. 6(b) shows the impact of the computation capacity on the average cost. The figure shows that increasing the computational capacity of the H-UAV reduces the average cost by a significant margin. This is because, with an increase in the computation capacity, the H-UAV obtains adequate computing resources. To minimize the total cost, therefore, the agents considers offloading the computation-intensive tasks to the H-UAV as the capacity increases, instead of offloading to the ground edge node. This reduces the transmission and processing delays of the task.

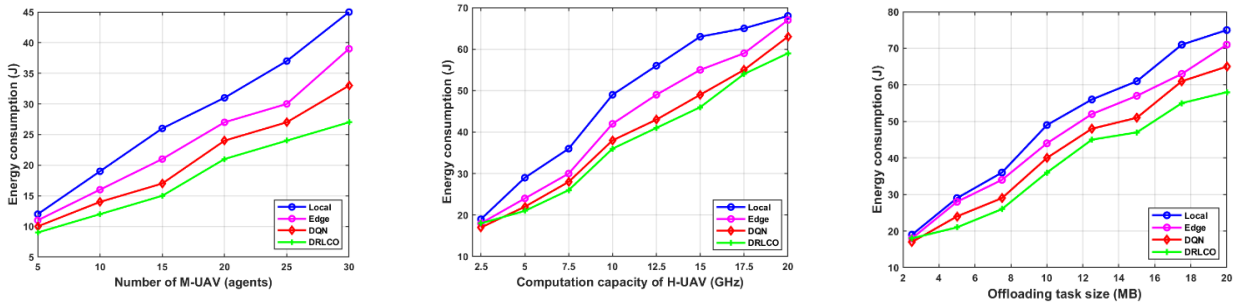
Fig. 6(c) shows the impact of the offloading task size on the average cost. As the task size offloaded from the M-UAV increases, the total cost also increases. This is because the CPU cycles required to execute tasks with large sizes also increase for the H-UAV. Hence, the computational time increases significantly. However, DRLCO can effectively allocate larger tasks to computational node with proper computational capacity according to the other task characteristics. Therefore, the proposed DRLCO scheme can obtain a higher reward to minimize the overall total cost.

Next, we evaluate the two most crucial performance metrics in the network: energy consumption and task execution delay. Using the proposed DRLCO scheme, agents can dynamically select a suitable computational node to minimize the overall cost. To demonstrate the performance of our proposed DRLCO algorithm, Fig. 7 shows the impact of the number of agents, computation capacity, and offloading task size on the energy consumption of the agent.



(a) Varying the number of M-UAVs. (b) Varying computation capacity. (c) Varying offloading task size.

FIGURE 6. Average offloading cost.



(a) Varying the number of M-UAVs. (b) Varying computation capacity. (c) Varying offloading task size.

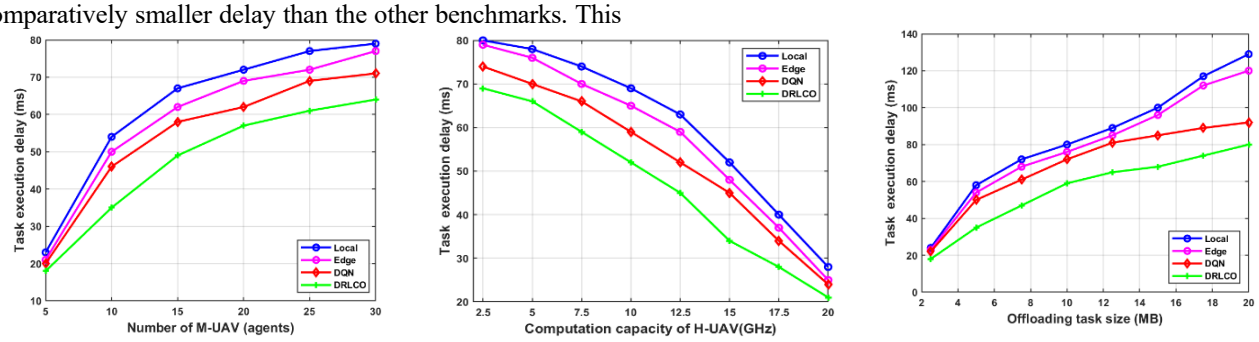
FIGURE 7. Energy consumption.

As shown in Fig. 7(a), the overall energy consumption increases when the number of agents (M-UAVs) increases. This is because an increased number of M-UAVs generate more computational tasks to be executed. Thus, the computation node consumes additional power because the agents show interest in offloading tasks on a suitable computation node to minimize the total cost. However, the local, edge, and DQN algorithms exhibit higher energy consumption than our proposed scheme. Because the proposed scheme dynamically allocates the computation task in terms of task characteristics, the task execution delay is reduced compared to the other conventional approaches.

In Figs. 7(b) and 7(c), we analyze the impact of the computation capacity and task size on the energy consumption. With an increase in the computing capacity of the H-UAV, the task execution time decreases, and the energy consumption also increases as shown in Fig. 7(b). As can be seen in Fig. 7(b), the energy consumption rises notably when the computing node's computation capacity increases. With the increased computational capacity, many agents (M-UAVs) tend to offload more computation intensive tasks to the H-UAV because those tasks can be executed with less transmission delay unlike the case of offloading to the ground MEC server. Subsequently, the agents obtain lower costs, which leads to higher rewards. This leads to significant high-power consumption of the H-UAV which, in turn, affects energy consumption as well because of using the onboard computation unit extensively for executing more tasks. Increasing the task size also increases energy consumption (Fig. 7(c)). This indicates that a larger task requires more CPU cycles to complete, thereby increasing the power consumption of the computation node. Hence, energy consumption also increases. However, the proposed algorithm significantly minimizes the energy consumption compared with the existing techniques, owing to the dynamic allocation of the task in accordance with the task characteristics.

Next, we focus on the delay performance of our proposed DRLCO scheme in terms of the number of agents (M-UAVs), task size, and varying computational capacity with the three other benchmark techniques. Fig. 8(a) shows that increasing the number of agents increases the task execution delay.

Because more M-UAVs generate tasks together, the computation node (H-UAV or edge) requires more CPU cycles and computation time to complete the task. In addition, transmitting a task to the ground edge server incurs a transmission delay. However, the proposed DRLCO obtains a comparatively smaller delay than the other benchmarks. This is because, with effective offloading decision generated by the DRLCO scheme, tasks are executed at the computational node with proper computational capacity, which results in fast execution compared to the other benchmarks.



(a) Varying the number of M-UAVs. (b) Varying computation capacity. (c) Varying offloading task.

FIGURE 8. Task execution delay.

Figs. 8(b) and 8(c) depict the delay performance of the proposed DRLCO in terms of the computation capacity of the H-UAV and varying task size. As shown in Fig. 8(b), we observed that increasing the computational capacity of the H-UAV reduced the task execution delay. Initially, when the computation capacity of the H-UAV was 2.5 GHz, the task execution delay was higher because the computation time was longer owing to the limited capacity. However, with an increase in the computation power, the computation time decreases further and, thus, agents tend to offload more energy-sensitive tasks to the H-UAV, reducing both the transmission and execution delays.

Fig. 8(c) depicts the impact of task size on task execution delay. We can observe that increasing the task size also increases task execution delay. Because a large task requires more CPU cycles to execute, the computation time is longer; thus, the delay also increases. In addition, when the computationally intensive task is offloaded from the M-UAV to the ground edge server, transmission and processing delays are incurred. Therefore, the proposed DRLCO is applicable to a multi-UAV-aided network system to reduce the total offloading cost by reducing energy consumption and task execution delay.

Based on the values, the average total cost, energy consumption, and task execution delay are calculated and shown in Tables II, III, and IV, respectively. As can be seen, DRLCO achieves the best overall performance with overall reduced total cost, energy consumption, and task execution delay in comparison to all the conventional algorithms. This indicates that DRLCO enables the M-UAV to make an optimal offloading decision based on the task size, available computational capacity, and specific task type.

TABLE II
RESULTS OF TOTAL COST

Algorithm	Average cost (Varying the number of UAVs)	Average cost (Varying computational capacity)	Average cost (Varying offloading task size
Local	40.5	70.5	26.25
Edge	35	69.5	21.12
DQN	25.83	63.25	17.37
DRLCO	**18.16**	**55**	**12**

TABLE III
RESULTS OF ENERGY CONSUMPTION

Algorithm	Average energy consumption (in Joules) (Varying the number of UAVs)	Average energy consumption (in Joules) (Varying computational capacity)	Average energy consumption (in Joules) (Varying offloading task size
Local	28.5	48.12	49.5
Edge	22.5	43	45.87
DQN	19.16	39.37	41.87
DRLCO	**18.16**	**37.62**	**38.25**

TABLE IV
RESULTS OF TASK EXECUTION DELAY

Algorithm	Average task execution delay (in *ms*)	Average task execution delay (in *ms*)	Average task execution delay (in *ms*)

	(Varying the number of UAVs)	(Varying computational capacity)	(Varying offloading task size
Local	62	60.5	83.625
Edge	58.5	57.37	79.25
DQN	54.33	53	69
DRLCO	**47.33**	**46.75**	**55.75**

VI. ANALYSIS

In accordance with the simulation results discussed earlier, in this section, we discuss and analyze the performance enhancement of the proposed algorithm over the conventional methods.

To examine the scalability of the proposed scheme, we observed that the total average cost increases with the increased number of agents (M-UAVs). However, the proposed DRLCO scheme can reduce the total offloading cost by 55.13%, 48.08%, and 29.21% in terms of the number of M-UAVs compared with the local, edge, and DQN schemes, respectively. Additionally, in terms of computational capacity and task size, the proposed scheme exhibits at least 15% and 31% less total cost, respectively, compared to the DQN scheme.

We then analyzed the impact of computational capacity and task size on the energy consumption and task execution delay of our DRLCO scheme. In the energy consumption test, DRLCO depicts less energy consumption with at least 5% and 11% reduction in terms of computational capacity and task size compared to the conventional schemes, respectively. Lastly, in the case of task execution delay, it can be noted that DRLCO can achieve less task execution delay with at least 11% and 31% improvement (reduced delay) in terms of computational capacity and task size, respectively, compared to the conventional techniques. Due to the significant improvement in total cost and task execution delay, the reasonable energy consumption is acceptable.

The overall performance enhancement of the proposed method can be attributed to several factors which include (i) the consideration of task execution based on task characteristics using the hierarchical MEC architecture, (ii) the DRLCO reward function aligned with offloading decisions, and (iii) the implementation of the DDQN based scheme to tackle the overestimation problem.

VII. CONCLUSION

In this paper, we investigate the decision-making problem of computation offloading in a UAV swarm-enabled edge-computing system. To support the M-UAV in successfully executing all tasks, the ground edge server provides assistance by enabling the M-UAV to offload computation-intensive tasks. Specifically, we formulate the offloading problem as a weighted sum cost minimization problem by jointly considering energy consumption and task execution delay. We then propose a multi-agent reinforcement learning framework called the DRLCO scheme to reduce the total system cost. Each M-UAV acts as an agent to determine the optimal offloading policy and performs offloading decisions based on the DRLCO scheme. Finally, simulation experiments were performed to validate the performance of the proposed DRLCO scheme. Compared with the local execution, edge execution, and DQN techniques, the proposed method can reduce the total cost by 54.28%, 43%, and 31%, respectively, in terms of offloading task size.

There are some open issues to be addressed in the future works. First, fine-granularity based tasks with interdependency, priority considering the complexity of the practical applications, and mobility of the UAV during offloading the task will be considered in the aerial computing networks. Second, some shadowing and scattering effects need to be considered to model the air-to-ground communication between the UAV and the ground base station. Finally, we aim to explore the more advanced RL techniques (dueling DQN and DDPG with continuous action space) for our offloading scenario to further optimize the performance of our study in a wider range of advanced applications.

ACKNOWLEDGMENT

The authors thank the editor and anonymous referees for their comments, which helped to improve the quality of this paper. This paper is based on the first author S. M. A. Huda's M.E. thesis supervised by the second author S. Moh [52].

REFERENCES

[1] Q. Wu *et al.*, "A Comprehensive Overview on 5G-and-Beyond Networks With UAVs: From Communications to Sensing and Intelligence," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 10, pp. 2912–2945, Oct. 2021, doi: 10.1109/JSAC.2021.3088681.

[2] J. Chen *et al.*, "Joint Task Assignment and Spectrum Allocation in Heterogeneous UAV Communication Networks: A Coalition Formation Game-Theoretic Approach," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 1, pp. 440–452, Jan. 2021, doi: 10.1109/TWC.2020.3025316.

[3] P. Mcenroe, S. Member, S. Wang, and M. Liyanage, "A Survey on the Convergence of Edge Computing and AI for UAVs : Opportunities and Challenges," *IEEE Internet Things J.*, vol. PP, p. 1, 2022, doi: 10.1109/JIOT.2022.3176400.

[4] J. Chen *et al.*, "A Multi-leader Multi-follower Stackelberg Game for Coalition-based UAV MEC Networks," *IEEE Wirel. Commun. Lett.*, vol. 2337, no. c, pp. 1–1, 2021, doi: 10.1109/lwc.2021.3100113.

[5] S. M. A. Huda and S. Moh, "Survey on computation offloading in UAV-Enabled mobile edge computing," *J. Netw. Comput. Appl.*, vol. 201, no. May 2022, p. 103341, 2022, doi: 10.1016/j.jnca.2022.103341.

[6] M. M. Alam and S. Moh, "Q-learning-based routing inspired by adaptive flocking control for collaborative unmanned aerial vehicle swarms," *Veh. Commun.*, vol. 40, p. 100572, Apr. 2023, doi: 10.1016/j.vehcom.2023.100572.

[7] Q. Liu, L. Shi, L. Sun, J. Li, M. Ding, and F. S. Shu, "Path Planning for UAV-Mounted Mobile Edge Computing with Deep Reinforcement Learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5723–5728, 2020, doi: 10.1109/TVT.2020.2982508.

[8] A. M. Raivi, S. M. A. Huda, M. M. Alam, and S. Moh, "Drone Routing for Drone-Based Delivery Systems: A Review of Trajectory Planning, Charging, and Security," *Sensors*, vol. 23, no. 3, 2023, doi: 10.3390/s23031463.

[9] W. You, C. Dong, X. Cheng, X. Zhu, Q. Wu, and G. Chen, "Joint Optimization of Area Coverage and Mobile-Edge Computing with Clustering for FANETs," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 695–707, 2021, doi: 10.1109/JIOT.2020.3006891.

[10] M. M. Alam and S. Moh, "Joint topology control and routing in a UAV swarm for crowd surveillance," *J. Netw. Comput. Appl.*, p. 138954, 2022, doi: https://doi.org/10.1016/j.jnca.2022.103427.

[11] D. Callegaro and M. Levorato, "Optimal Edge Computing for Infrastructure-Assisted UAV Systems," *IEEE Trans. Veh. Technol.*, vol. 70, no. 2, pp. 1782–1792, 2021, doi: 10.1109/TVT.2021.3051378.

[12] B. Dai, J. Niu, T. Ren, Z. Hu, and M. Atiquzzaman, "Towards Energy-Efficient Scheduling of UAV and Base Station Hybrid Enabled Mobile Edge Computing," *IEEE Trans. Veh. Technol.*, vol. 9545, no. c, pp. 1–16, 2021, doi: 10.1109/TVT.2021.3129214.
[13] X. Diao, J. Zheng, Y. Cai, Y. Wu, and A. Anpalagan, "Fair Data Allocation and Trajectory Optimization for UAV-Assisted Mobile Edge Computing," *IEEE Commun. Lett.*, vol. 23, no. 12, pp. 2357–2361, 2019, doi: 10.1109/LCOMM.2019.2943461.
[14] X. Zheng, M. Li, M. Tahir, Y. Chen, and M. Alam, "Stochastic Computation Offloading and Scheduling Based on Mobile Edge Computing," *IEEE Access*, vol. 7, no. 2, pp. 72247–72256, 2019, doi: 10.1109/ACCESS.2019.2919651.
[15] Y. Liu *et al.*, "Joint Communication and Computation Resource Scheduling of a UAV-Assisted Mobile Edge Computing System for Platooning Vehicles," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–16, 2021, doi: 10.1109/TITS.2021.3082539.
[16] N. Zhao, Z. Ye, Y. Pei, Y.-C. Liang, and D. Niyato, "Multi-Agent Deep Reinforcement Learning for Task Offloading in UAV-Assisted Mobile Edge Computing," *IEEE Trans. Wirel. Commun.*, vol. 21, no. 9, pp. 6949–6960, Sep. 2022, doi: 10.1109/TWC.2022.3153316.
[17] 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, 2017, doi: 10.1109/TCOMM.2017.2699660.
[18] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," *Proc. - IEEE INFOCOM*, vol. 2016-July, 2016, doi: 10.1109/INFOCOM.2016.7524497.
[19] N. H. Chu, D. T. Hoang, D. N. Nguyen, N. Van Huynh, and E. Dutkiewicz, "Joint Speed Control and Energy Replenishment Optimization for UAV-assisted IoT Data Collection with Deep Reinforcement Transfer Learning," *IEEE Internet Things J.*, vol. 4662, no. c, 2022, doi: 10.1109/JIOT.2022.3151201.
[20] B. Kar, W. Yahya, Y. D. Lin, and A. Ali, "Offloading using Traditional Optimization and Machine Learning in Federated Cloud-Edge-Fog Systems: A Survey," *IEEE Commun. Surv. Tutorials*, no. c, pp. 1–28, 2023, doi: 10.1109/COMST.2023.3239579.
[21] 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. Mob. Comput.*, vol. 1233, no. c, pp. 1–15, 2021, doi: 10.1109/TMC.2021.3059691.
[22] Y. Liu, J. Yan, and X. Zhao, "Deep Reinforcement Learning based Latency Minimization for Mobile Edge Computing with Virtualization in Maritime UAV Communication Network," *IEEE Trans. Veh. Technol.*, vol. 9545, no. c, pp. 1–1, 2022, doi: 10.1109/tvt.2022.3141799.
[23] F. Tang, H. Hofner, N. Kato, K. Kaneko, Y. Yamashita, and M. Hangai, "A Deep Reinforcement Learning-Based Dynamic Traffic Offloading in Space-Air-Ground Integrated Networks (SAGIN)," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 276–289, 2022, doi: 10.1109/JSAC.2021.3126073.
[24] Q. Zhang, J. Chen, L. Ji, Z. Feng, Z. Han, and Z. Chen, "Response Delay Optimization in Mobile Edge Computing Enabled UAV Swarm," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3280–3295, 2020, doi: 10.1109/TVT.2020.2964821.
[25] H. Zhou, Z. Wang, G. Min, and H. Zhang, "UAV-Aided Computation Offloading in Mobile-Edge Computing Networks: A Stackelberg Game Approach," *IEEE Internet Things J.*, vol. 10, no. 8, pp. 6622–6633, Apr. 2023, doi: 10.1109/JIOT.2022.3197155.
[26] A. M. Seid, G. O. Boateng, J. Lu, B. Mareri, W. Jiang, and G. Sun, "Multi-Agent DRL for Task Offloading and Resource Allocation in Multi-UAV Enabled IoT Edge Network," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 4, pp. 4531–4547, 2021, doi: 10.1109/TNSM.2021.3096673.
[27] P. Qin, Y. Fu, X. Zhao, K. Wu, J. Liu, and M. Wang, "Optimal Task Offloading and Resource Allocation for C-NOMA Heterogeneous Air-Ground Integrated Power Internet of Things Networks," *IEEE Trans. Wirel. Commun.*, pp. 1–17, 2022, doi: 10.1109/TWC.2022.3175472.
[28] M. M. Alam and S. Moh, "Survey on Q-Learning-Based Position-Aware Routing Protocols in Flying Ad Hoc Networks," *Electronics*, vol. 11, no. 7, p. 1099, Mar. 2022, doi: 10.3390/electronics11071099.
[29] L. Zhang *et al.*, "Task Offloading and Trajectory Control for UAV-Assisted Mobile Edge Computing Using Deep Reinforcement Learning," *IEEE Access*, vol. 9, pp. 53708–53719, 2021, doi: 10.1109/ACCESS.2021.3070908.
[30] J. Cui, Y. Liu, and A. Nallanathan, "Multi-Agent Reinforcement Learning-Based Resource Allocation for UAV Networks," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 2, pp. 729–743, 2020, doi: 10.1109/TWC.2019.2935201.
[31] H. Zhou, Z. Wang, H. Zheng, S. He, and M. Dong, "Cost Minimization-Oriented Computation Offloading and Service Caching in Mobile Cloud-Edge Computing: An A3C-Based Approach," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 3, pp. 1326–1338, May 2023, doi: 10.1109/TNSE.2023.3255544.
[32] S. Islam, S. Badsha, I. Khalil, M. Atiquzzaman, and C. Konstantinou, "A Triggerless Backdoor Attack and Defense Mechanism for Intelligent Task Offloading in Multi-UAV Systems," *IEEE Internet Things J.*, vol. 4662, no. c, pp. 1–14, 2022, doi: 10.1109/JIOT.2022.3172936.
[33] E. I. S. Group, "GS MEC 003 - V2.2.1 - Multi-access Edge Computing (MEC); Framework and Reference Architecture," vol. 1, pp. 1–21, 2020.
[34] G. Brown, "Computation offloading game for an UAV network in mobile edge computing," *Juniper White Pap.*, no. July, 2016, [Online]. Available: https://www.juniper.net/assets/us/en/local/pdf/whitepapers/2000642-en.pdf.
[35] J. Chen, Y. Xu, Q. Wu, Y. Zhang, X. Chen, and N. Qi, "Interference-Aware Online Distributed Channel Selection for Multicluster FANET: A Potential Game Approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3792–3804, 2019, doi: 10.1109/TVT.2019.2902177.
[36] T. Yang and X. S. Shen, "Multi-vessel computation offloading in maritime mobile edge computing network," *SpringerBriefs Comput. Sci.*, vol. 6, no. 3, pp. 37–53, 2020, doi: 10.1007/978-981-15-4412-5_4.
[37] M. D. Nguyen, T. M. Ho, L. B. Le, and A. Girard, "UAV Trajectory and Sub-channel Assignment for UAV Based Wireless Networks," *IEEE Wirel. Commun. Netw. Conf. WCNC*, vol. 2020-May, 2020, doi: 10.1109/WCNC45663.2020.9120814.
[38] Z. Yu, Y. Gong, S. Gong, and Y. Guo, "Joint Task Offloading and Resource Allocation in UAV-Enabled Mobile Edge Computing," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3147–3159, Apr. 2020, doi: 10.1109/JIOT.2020.2965898.
[39] J. Zong *et al.*, "Flight time minimization via UAV's trajectory design for ground sensor data collection," *Proc. Int. Symp. Wirel. Commun. Syst.*, vol. 2019-Augus, pp. 255–259, 2019, doi: 10.1109/ISWCS.2019.8877250.
[40] Z. Zhou, Z. Chang, and H. Liao, "Dynamic Computation Offloading Scheme for Fog Computing System with Energy Harvesting Devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 143–161, 2021, doi: 10.1109/JSAC.2016.2611964.
[41] Y. Du, J. Li, L. Shi, T. Liu, F. Shu, and Z. Han, "Two-Tier Matching Game in Small Cell Networks for Mobile Edge Computing," *IEEE Trans. Serv. Comput.*, vol. 15, no. 1, pp. 254–265, 2022, doi: 10.1109/TSC.2019.2937777.
[42] T. Fang, J. Chen, and Y. Zhang, "Content-Aware Multi-Subtask Offloading: A Coalition Formation Game-Theoretic Approach," *IEEE Trans. Serv. Comput.*, vol. 25, no. 8, pp. 2664–2668, 2021.
[43] X. Hu, K. K. Wong, K. Yang, and Z. Zheng, "UAV-Assisted Relaying and Edge Computing: Scheduling and Trajectory Optimization," *IEEE Trans. Wirel. Commun.*, vol. 18, no. 10, pp. 4738–4752, 2019, doi: 10.1109/TWC.2019.2928539.
[44] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wirel. Commun.*, vol. 11, no. 6, pp. 1991–1995, 2012, doi: 10.1109/TWC.2012.041912.110912.
[45] A. Rudenko, P. Reiher, G. J. Popek, and G. H. Kuenning, "Saving portable computer battery power through remote process execution," *ACM SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 2, no. 1, pp. 19–26, 1998, doi: 10.1145/584007.584008.
[46] M. Ayoub Messous, H. Sedjelmaci, N. Houari, and S.-M. Senouci, "Computation offloading game for an UAV network in mobile edge computing," *2017 IEEE Int. Conf. Commun.*, 2014, [Online]. Available: https://doi.org/10.1109/ICC.2017.7996483.
[47] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. M. Leung, "Deep Reinforcement Learning for Energy-Efficient Computation Offloading in Mobile-Edge Computing," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1517–1530, 2022, doi: 10.1109/JIOT.2021.3091142.
[48] Q. Wu *et al.*, "Joint Computation Offloading, Role and Location Selection in Hierarchical Multi-coalition UAV MEC Networks: A Stackelberg Game Learning Approach," *IEEE Internet Things J.*, vol. 4662, no. c, pp. 1–12, 2022, doi: 10.1109/JIOT.2022.3158489.
[49] J. Cai, H. Fu, and Y. Liu, "Multi-task Multi-objective Deep Reinforcement Learning-based Computation Offloading Method for Industrial Internet of Things," *IEEE Internet Things J.*, vol. 10, no. 2, pp. 1848–1859, 2022, doi: 10.1109/JIOT.2022.3209987.
[50] W. Li, F. Zhou, K. R. Chowdhury, and W. M. Meleis, "QTCP: Adaptive Congestion Control with Reinforcement Learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 6, no. 3, pp. 445–458, 2018, doi: 10.1109/TNSE.2018.2835758.
[51] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-Learning," *30th AAAI Conf. Artif. Intell. AAAI 2016*, no. 2, pp. 2094–2100, 2016.
[52] S. M. A. Huda, Reinforcement Learning-Based Offloading in Unmanned Aerial Vehicle-Aided Edge Computing Systems, M.E. Thesis, Dept. of Computer Eng., Chosun Univ., Dec. 2022.

S. M. ASIFUL HUDA received the B.S. degree in Computer Science and Engineering from East West University, Bangladesh, in 2019 and the M.E degree in Computer Engineering from Chosun University, South Korea., in 2023. His main research interests include edge computing, deep learning, and reinforcement learning.

SANGMAN MOH received his Ph.D. degree in Computer Engineering from Korea Advanced Institute of Science and Technology (KAIST), South Korea, in 2002. Since late 2002, he has been a professor at the Dept. of Computer Engineering at Chosun University, Korea. From 2006 to 2007, he was on leave at Cleveland State University, USA. Until 2002, he had been with Electronics and Telecommunications Research Institute (ETRI), Korea, where he served as a project leader after receiving the M.S. degree in computer science from Yonsei University, Korea, in 1991. His research interests include mobile computing and networking, *ad hoc* and sensor networks, cognitive radio networks, unmanned aerial vehicle networks, and mobile edge computing. Dr. Moh is a member of the IEEE, the ACM, the IEICE, the KIISE, the IEIE, the KIPS, the KICS, the KMMS, the IEMEK, the KISM, and the KPEA.