JDACO: Joint Data Aggregation and Computation Offloading in UAV-Enabled Internet of Things for Post-Disaster Scenarios

Asif Mahmud Raivi, Sangman Moh, *Member, IEEE*

***Abstract*—Owing to high flexibility and rapid deployment, unmanned aerial vehicles (UAVs) can offer network coverage for Internet of things (IoT) devices in post-disaster scenarios. UAV-aided mobile edge computing (MEC) provides computational support and facilitates optimal decision-making processes for ground-based IoT devices. However, existing literature has separately examined both data aggregation and computational offloading. In this paper, we introduce a Joint Data Aggregation and Computational Offloading (JDACO) scheme for UAV-enabled IoT systems in post-disaster scenarios. JDACO's primary objective is to minimize the overall energy consumption and latency in the aggregation and computation processes. It achieves this by employing UAVs as MEC servers and deploying multiple UAVs. We initially design an objective function to assess the costs associated with the aggregation and offloading processes. Subsequently, we frame the optimization problem as a Markov model and employ a multi-agent deep reinforcement learning algorithm. This approach utilizes value decomposition with the double deep Q-Network algorithm to optimize data aggregation and enable a cost-effective offloading process through cooperative learning. Our experimental results demonstrate that our proposed JDACO scheme surpasses existing methods in terms of training time reduction, processed data volume, energy efficiency, and mission duration by 20%, 11.4%, 5.6%, and 11.2%, respectively, compared to the conventional schemes while serving up to 98% of IoT devices.**

***Index Terms*—Computation offloading, data aggregation, Internet of things, unmanned aerial vehicle, mobile edge computing, multi-agent reinforcement learning.**

I. INTRODUCTION

Rapid advent in wireless communication networks and Internet of things (IoT) has made terrestrial communication possible [1]. Unmanned aerial vehicles (UAVs) have opened new avenues for communication technology [2]. UAVs will soon become an integral part of existing communication systems owing to their easy and rapid deployment. The use of UAVs is increasing from military missions to industrial and commercial applications [3], [4]. Recently, UAVs have been proposed for restoring communications in post-disaster scenarios [5]. Therefore, UAVs are expected to become powerful and important entities for shaping communication systems in the near future.

The implementation of these new technologies poses various challenges. UAVs have limited battery capacity, resulting in limited flight time and need to be replenished before the next deployment. Therefore, to ensure smooth operation during the mission, the UAV flight trajectory should be carefully designed. Additionally, IoT devices installed for environmental monitoring are resource-constrained with limited computational capabilities and are often installed in hard-to-reach areas with the expectation of a long service life. Thus, any disruption in the existing communication systems can defeat the entire purpose of installing IoT devices. Moreover, because of their limited energy, IoT devices cannot communicate over long distances. Therefore, a well-planned strategy is required to maintain a stable connectivity between IoT devices and base stations (BS).

Owing to the rapid deployment capability of UAVs with extended battery life resulting from recent technological advancements, they can perform aggregation missions and edge units to support data-driven IoT applications [6]. There are several approaches in the literature in which UAV collect data from ground IoT devices [7]. These studies primarily focused on the optimal point of data gathering, trajectory design for the UAV, devising an energy-saving scheme, resource allocation, and reducing the data collection period while ensuring the quality of service (QoS), data freshness, maximum data collection, and reduced loss of aggregated data.

Similarly, considering UAV as edge servers, the existing literature focuses on minimizing the task execution delay and energy requirements while ensuring maximum throughput and computation capability within the available edge server resources [8]. If the computation requirement is beyond the processing power, all or some of the computations are offloaded to another server with a higher computation capability, such as the BS. Consequently, IoT nodes can eliminate the burden of computation and perform for long periods of time. UAVs are perfect suitors for solving communication and computational issues resulting from both natural and man-made disasters.

The use of UAV as data aggregators has attracted considerable attention in recent years. Existing studies focus

This research was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korean Government (MSIT) (No. 2022R1A2C1009037).

Asif Mahmud Raivi and Sangman Moh are with the Department of Computer Engineering, Chosun University, Gwangju 61452, South Korea.

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

on finding the optimal hovering location or cluster head selection for aggregating data from ground IoT nodes, as designing optimal path planning is essential for UAVs while ensuring minimal travel time and energy efficiency [9]. Similarly, UAVs are considered as an edge unit for offloading the computation-incentive tasks of IoT nodes, in which either binary or partial offloading is exploited in existing works [10]. Thus, IoT nodes are protected from heavy computation and long service times. Existing studies primarily focus on latency and energy minimization for the offloading process, while ensuring maximum data computation and throughput maximization. Although many studies have considered static and single-UAV scenarios, recent studies have focused on UAV mobility and multi-UAV deployment [10].

In the existing literature, it can be observed that data aggregation and computational offloading were studied separately. Recognizing the future prospects of UAVs for data-driven applications, especially for post-disaster scenarios where existing communication infrastructure is disrupted or no longer available, we propose a joint data aggregation and computation offloading scheme and introduce the two problems under the same umbrella, rather than considering them as separate problems. A more detailed study of the existing literature will be discussed in Section II.

To address the aforementioned discussions and limitations, we propose a joint multi-UAV-based data aggregation and computation offloading scheme to mitigate the overall system cost of the process. More explicitly, multiple UAVs are deployed, where each UAV is responsible for data aggregation and computation, as well as offloading some computation to another UAV or BS with higher resources and computational capability. The introduction of a multiagent paradigm brings the action and decision-making of each UAV into unison, as all agents share their experiences with each other. The key contributions of this study are as follows:

- We study a joint scenario of data aggregation and computation offloading from a data-driven aerial computing perspective, which has not yet been explored together for UAV-enabled services.
- We develop a joint data aggregation and computational offloading (JDACO) scheme mathematically for a multi-UAV scenario. Our proposed optimization problem primarily focuses on minimizing the total cost of energy consumption and delay for the aggregation and offloading processes, while ensuring maximum IoT device coverage.
- To address the joint optimization problem, we propose a multi-agent deep reinforcement learning (MA-DRL)-based algorithm in which we adopt a dueling double deep Q network (D3QN) for the discrete action space and a decision maker for each UAV. We employ a value decomposition network (VDN) algorithm for cooperative learning among the UAVs. By combining D3QN and VDN, we propose value decomposition dueling double deep Q-network (VD3QN), which is an off-policy approach to solve our optimization problem.
- We evaluate our algorithm using two other off-policy learning algorithms and one non-learning algorithm in terms of key performance metrics. Simulation results demonstrate the superiority of the proposed algorithm over other benchmarks.

The remainder of this paper is organized as follows: We first explore the relevant studies that have been conducted thus far in the respective fields of data aggregation and task offloading in Section II. We present our system model in Section III, and formulate the optimization problem in Section IV. In Section V, the formulated optimization problem is transformed into a Markov game model. In Section VI, the performance of the proposed JDACO algorithm is demonstrated and compared with that of other benchmarks. Finally, we conclude our study in Section VII.

II. Related Works

Most studies on UAV-aided data-driven applications can be categorized into two main classes. The first focuses on utilizing UAV as relays or base stations (BS) to provide a backbone for data-gathering applications. In such cases, the UAV is considered a data aggregator, where the trajectory of the UAVs is designed based on the communication schedule [11], [12]. In the second class, UAVs act as mobile edge computing (MEC) units to support the computational capabilities of a given network [13].

A. UAV as Data Aggregator

While considering aerial data aggregation scenarios, the existing studies have primarily focused on designing optimal trajectories by finding the optimal hovering point, minimizing mission energy, and covering the maximum number of IoT devices for aggregation. In [11], the authors studied mission cost minimization while covering the maximum number of IoT devices using multiple UAV as aggregators. A heuristic approach was used to solve the proposed problem. This study considers an IoT device activation model for an intruding probability scenario of communication between a UAV and IoT nodes.

In [14], a UAV was employed as a data aggregator, and the aggregated data were relayed to a base station (BS). In addition to the impact of the UAV altitude on the aggregation rate, the data-to-overhead ratio was studied to measure the effectiveness of data aggregation using UAV. The study mentions the possibility of further utilization of UAV as an edge unit to minimize end-to-end delay but does not explore that option.

The study in [15] aimed to minimize the hovering and travelling time for data aggregation by a UAV visiting each node using a decoupled heuristic approach. This study demonstrated that a clear trade-off between hovering and travelling times is necessary for optimal data aggregation.

In [16], the struggle between the trajectory and data aggregation based on device activation in a multi-UAV scenario was studied. The concept of shared observation among UAV used a long short-term memory (LSTM) deep

deterministic policy gradient (DDPG) approach. The scheme addressed the pressing issues of data loss owing to buffer overflow and communication failures that may occur at ground IoT nodes.

B. UAV as Edge Server for Computation Offloading

Computation of the sensing data is necessary because IoT nodes are reconstructed and have very limited computation capability. Edge units or devices are often introduced to address the computation problem and reduce the overall latency and energy consumption of the offloading process [17].

The study in [10] studied UAVs as MEC edge units, where the offloading process was classified into two categories: binary and partial offloading. The main idea behind utilizing UAVs as edge units was to reduce the overall delay and energy consumption for IoT devices with resource-intensive tasks. Moreover, in hierarchical aerial networks such as the space-air-ground integrated network (SAGIN), every component of each layer was considered an MEC unit and was able to perform resource-intensive tasks.

In [18], a partial offloading mechanism was proposed for a hierarchical network, where the IoT and UAV game-theoretic-based offloading decision method was suggested. Additionally, a heuristic approach was proposed to make offloading decisions between the UAV and a high-altitude platform (HAP). To utilize all the layers to the fullest extent possible, an adjustment algorithm was introduced. However, the mobility of the UAV and HAP was static, and the detailed mechanism of task collection had not been studied.

In [19], another partial offloading mechanism was studied, in which UAV mobility was considered. The proposed method aimed to minimize the overall delay and energy requirements of the offloading process, while maximizing the number of arrival tasks. This study considered the local processing and task queuing delays in the total processing delay calculation. The algorithmic approach utilized the multi-objective reinforcement learning (MORL) to obtain the optimal solution. However, their proposed problem was demonstrated using only a single UAV.

In [20], a binary offloading problem was proposed, which was solved by a multi-agent actor-critic approach that aimed to solve multiple objectives such as offloading decisions, flight direction, and distance. The proposed model had a relatively simple sensing model that is unrealistic considering the real environment.

Considering all the matters at hand, we propose the JDACO scheme for maximum IoT device coverage with minimal energy and time expenditure for both data aggregation and computation offloading processes. Table I presents a relative summary of the existing works.

III. System Model

In this section, we present our system model of a multi-UAV-aided MEC for UAV-enabled IoT. After introducing the application scenario, mobility, communication, data aggregation, local computation, and offloading computation models were formally addressed.

A. Application Scenario

We consider a post-disaster region where multiple UAVs are deployed to aggregate data from live homogeneous IoT nodes with various sensors on the ground, as existing communication infrastructure such as base stations (BS) are no longer available. The deployed IoT nodes are responsible for monitoring environmental conditions, and low-tier UAVs (LT-UAV) are responsible for aggregating and offloading data based on the task size. Because the existing communication network has been disrupted, a UAV with a longer flight time and computation power, called a high-tier UAV (HT-UAV), hovers at a fixed altitude (which is higher than the altitudes of LT-UAVs) such that all the LT-UAVs are under the communication coverage of the HT-UAV.

TABLE I
COMPARATIVE SUMMARY OF EXISTING WORKS AND OURS

Ref.	Major focus		Algorithm		Number of UAVs	Optimization objective	Considered factors			
	Data aggregation	Offloading	DRL	Non-DRL			Trajectory and hovering location	Energy	Delay	Max number of IoT devices
[11]	✓	–	–	✓	Multiple	Max device coverage with min travel time	✓	✓	–	✓
[14]	✓	–	–	✓	Single	Min aggregation delay	–	–	✓	–
[15]	✓	–	–	✓	Multiple	Min aggregation delay	✓	–	✓	✓
[16]	✓	–	✓	–	Multiple	Min travel time with device scheduling	✓	–	✓	–
[17]	–	✓	✓	–	Multiple	Energy and delay minimization	–	✓	✓	–
[18]	–	✓	–	✓	Multiple	Max data computation	–	✓	✓	✓
[19]	–	✓	–	✓	Single	Min energy and delay and max task computation	–	✓	✓	✓
[20]	–	✓	✓	–	Multiple	Min energy and delay for task computation	✓	✓	✓	–
Ours	✓	✓	✓	–	Multiple	Max device coverage, task with min energy and delay	✓	✓	✓	✓

Note: ✓ Studied; – Not studied

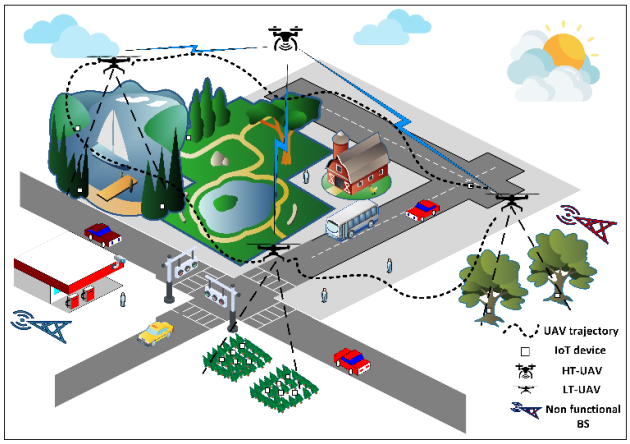


Fig. 1. Application scenario of typical network configuration.

Fig.1 shows a typical example of the network configuration in our application scenario. To aggregate data from ground IoT devices, LT-UAVs must hover over several hovering locations where data can be collected from the maximum number of IoT devices. As LT-UAVs aggregate data from IoT devices, each LT-UAV flies for the maximum travel time of $T_{\max}$ before the maximum energy $E_{\max}$ of the UAV is depleted, and then lands on the ground. Based on the aggregated data from the IoT devices, LT-UAVs begin to process the data. LT-UAVs are equipped with single-core CPU, which means that they can execute or handle one task simultaneously. Based on the task size of the received data, the LT-UAV offloads the data to the HT-UAV, where further processing occurs.

We assume that, during the hovering mode, each LT-UAV flies at a pre-defined average velocity of V_{avg} and $V = 0$ m/s. The ground node locations are known beforehand and are distributed statically over the area of interest. Node locations can be expressed as $i = [x_i, y_i]$. To avoid collisions with other UAVs or foreign objects, each UAV has object detection capabilities, thereby ensuring a safe flight plan. IoT devices are static and randomly distributed across geographical areas. Each UAV maintains a considerable altitude to ensure strong line-of-sight (LoS) communication. Additionally, the HT-UAV ensures the synchronized trajectory of other LT-UAVs for both non-overlapping aggregation locations and the estimation of the number of active IoT devices in the area of interest. Table II lists the key notations with respective definition used in this formulation.

B. LT-UAV Mobility Model

To ensure a strong LoS and avoid obstacles in the vicinity we assume that the LT-UAVs fly at a considerable altitude of h_j. The horizontal direction and distance travelled by the LT-UAV at time slot t is denoted as $\emptyset(t)$ and $d(t)$, respectively, provided the following conditions are satisfied:

$$0 \le \emptyset(t) \le 2\pi, \;\; 0 \le d(t) \le d_{\max}, \tag{1}$$

where d_{max} is the maximum flying distance of the LT-UAV owing to its limited battery capacity.

TABLE II
KEY NOTATIONS

Notation	Definition
$P_{\text{LoS}}^{i,j}, P_{\text{NLoS}}^{i,j}$	Line-of-sight (LoS) and non-line-of-sight (NLoS) probability between IoT node *i* and LT-UAV *j*.
$L_{\text{LoS}}^{i,j}, L_{\text{NLoS}}^{i,j}$	Path loss for LoS and NLoS condition
$\Lambda_{i,j}$	Average path loss between IoT node *i* and LT-UAV *j*.
$\mathbb{S}_{i,j}$	Signal-to-interference-plus-noise-ratio (SINR) between IoT node *i* and LT-UAV *j*.
$R_{i,j}$	Expected data rate between IoT node *i* and LT-UAV *j*.
$G_{j,k}$	Channel gain between LT-UAV *j* and HT-UAV *k*.
$\mathbb{S}_{j,k}$	Signal-to-interference-plus-noise-ratio (SINR) between LT-UAV *j* and HT-UAV *k*.
$R_{j,k}$	Uplink data rate between LT-UAV *j* and HT-UAV *k*.
N_i	The number of active IoT nodes ready for transmitting data.
$\mathring{A}_{i,j}, \tilde{A}_{i,j}$	Indicator function for establishing communication and multiple transmissions between IoT node *i* and LT-UAV *j*.
$\Phi_{j,k}$	Binary decision variable for local computation and offloading between LT-UAV *j* and HT-UAV *k*.

We adopted a conventional Cartesian coordinate system to represent the mobility of the UAV. Let $\mathcal{U}(t) = [x_j(t), y_j(t)]$ represent the LT-UAV's location at time slot t. Thus, based on the $\emptyset(t)$ and $d(t)$, the coordinate of the LT-UAV at the next time slot $t+1$ can be expressed as

$$\begin{cases} x_j(t+1) = x_j(t) + d(t) \cdot \cos(\emptyset(t)) \\ y_j(t+1) = y_j(t) + d(t) \cdot \sin(\emptyset(t)) \end{cases}. \tag{2}$$

The LT-UAV was assumed to travel within an enclosed rectangular region with side lengths are $x_{\max}$ and $y_{\max}$. We have

$$0 \le x_j(t) \le x_{\max}, \;\; 0 \le y_j(t) \le y_{\max}. \tag{3}$$

Similar to previous studies [19], [21], we adopted the propulsion power requirement of a rotary-wing UAV to define its power consumption, which is given by

$$P(v(t)) = P_1\left(1 + \frac{3v(t)^2}{U_{\text{tip}}^2}\right) + P_2\left(\sqrt{1 + \frac{v(t)^4}{4v_0^4}} - \frac{v(t)^2}{2v_0^2}\right)^{1/2} + \frac{1}{2}d_0\rho g A v(t)^3. \tag{4}$$

The given equation comprises three components: blade profile, induced power, and parasitic power. P_1 is the blade profile power in the hovering state, and P_2 is the induced power. U_{tip} refers to the speed of the rotor blade tip and v_0 is the average induced rotor velocity during the hovering state. The power of the parasite was also contained. d_0, ρ, g, and A which are fuselage drag ratio, density of air, solidity of the rotor, and disk area, respectively. Under hovering conditions, the power consumption of the UAV is an aggregation of P_1 and P_2. The overall energy requirement of the UAV during its flight duration T is given by

$$E_j^{\text{fly}}(t) = \sum_0^T P\big(v(t)\big)\Delta t. \tag{5}$$

C. Communication Model

We formulated our communication model into two different segments: communication between the IoT node and LT-UAV

and communication between the LT-UAV and HT-UAV.

1) *Downlink Communication model*

As stated previously, the LT-UAV maintains a considerable altitude to maintain a strong LoS. Therefore, the LoS probability between the ground IoT node i and LT-UAV j can be expressed as

$$P_{\mathrm{LoS}}^{i,j} = \frac{1}{1+\alpha e^{-\beta(\theta_{i,j}-\alpha)}}, \quad (6)$$

where α and β are the environmental constant values and the elevation angle, respectively, and $\theta_{i,j} = (\frac{180}{\pi}) \sin^{-1}(\frac{h_j}{d_{i,j}})$, where $d_{i,j}$ denotes the distance between IoT node i and LT-UAV j and can be expressed as $d_{i,j} = \sqrt{((x_j(t)-x_i)^2 + (y_j(t)-y_i)^2 + h_j^2))}$. As expected, non-line-of-sight (NLoS) probability is $P_{\mathrm{NLoS}}^{i,j} = 1 - P_{\mathrm{LoS}}^{i,j}$. The path-loss expression for both LoS and NLoS is expressed as

$$L_{\mathrm{LoS}}^{i,j} = \eta_{\mathrm{LoS}}(\frac{4\pi f_c}{c} d_{i,j})^{\xi} \quad (7)$$

and

$$L_{\mathrm{NLoS}}^{i,j} = \eta_{\mathrm{NLoS}}(\frac{4\pi f_c}{c} d_{i,j})^{\xi}, \quad (8)$$

respectively, where η_{LoS} and η_{NLoS} are the attenuation factor for LoS and NLoS state, respectively, f_c is the carrier frequency, c is the speed of light, and ξ is the path loss component. Thus, the average path loss $\Lambda_{i,j}$ between IoT node i and LT-UAV j can be found as

$$\Lambda_{i,j} = P_{\mathrm{LoS}}^{i,j} \times L_{\mathrm{LoS}}^{i,j} + P_{\mathrm{NLoS}}^{i,j} \times L_{\mathrm{NLoS}}^{i,j}. \quad (9)$$

Therefore, average channel gain at time instant t is $G_{i,j}(t) = {\Lambda_{i,j}}^{-1}$ as studied in [11]. It is assumed that each IoT node i has a transmit power $P_i(t)$ at time instant t and the IoT devices communicate with the LT-UAV via a time-division multiple access (TDMA) scheme. Employing the TDMA scheme eliminates intracluster interference. However, neighboring UAV may cause interference. Considering these factors, the signal-to-interference-plus-noise-ratio (SINR) between IoT node i and LT-UAV j at time instant t is expressed as

$$\mathbb{S}_{i,j}(t) = \frac{P_i(t)\, G_{i,j}(t)}{P_n(t) G_{m,n}(t) + \sigma^2} \; n=1, m\neq i, n\neq j, \quad (10)$$

where σ^2 denotes the Gaussian noise variance. Using Shannon's theorem, we calculated the approximate data rate between IoT node i and LT-UAV j, which is denoted as

$$R_{i,j}(t) = \mathcal{B}_1 \, log_2(1 + \mathbb{S}_{i,j}(t)). \quad (11)$$

where $\mathcal{B}_1$ is the channel bandwidth for the downlink communication.

2) *Uplink Communication Model*

Because the existing communication infrastructure, such as the BS, is no longer available in the post-disaster scenario, LT-UAVs are resource-constrained and need to offload the aggregated data to the HT-UAV, which has higher processing power and computation capacity. Assuming that the wireless link between the LT-UAV and HT-UAV maintains clear LoS characteristics, the channel quality depends on the instantaneous distance between them [22]. Let $\mathcal{v} = [x_{\mathrm{k}}, y_{\mathrm{k}}]$ denote HT-UAV coordinates. Then, the instantaneous distance between LT-UAV j and HT-UAV k is given as $d_{\mathrm{j,k}} = \sqrt{||\mathcal{v} - \mathcal{U}(\mathrm{t})||^2}$. Therefore, the channel power gain between the LT and HT-UAV, following the path loss model in free space at time instant t, can be expressed as

$$G_{j,k}(t) = \mathcal{P}_0 d_{j,k}^{-2} = \frac{\mathcal{P}_0}{||\mathcal{v} - \mathcal{U}(t)||^2}, \quad (12)$$

where $\mathcal{P}_0$ is the power gain of the channel at 1 m distance and is subjected to the antenna gain and carrier frequency.

Because we intend to maintain communication between LT-UAVs and HT-UAV continuously, we exploit the benefit of the frequency division multiple access (FDMA) scheme. The uplink bandwidth $\mathcal{B}_2$ is divided into J non-overlapping sub-bands of J LT-UAVs. Thus, in each time slot, each LTUAV uplink was allotted a subband of $\frac{\mathcal{B}_2}{J}$. Then, the SINR can be formulated as

$$\mathbb{S}_{j,k}(t) = \frac{P_j(t) G_{j,k}(t)}{\frac{\mathcal{B}_2}{J}\sigma_0^2} = \frac{P_j(t)\mathcal{P}_0}{\frac{\mathcal{B}_2}{J}||\mathcal{v} - \mathcal{U}(t)||^2 \sigma_0^2}, \quad (13)$$

where $P_j(\mathrm{t})$ is the transmission power of the LT-UAV and σ_0^2 is the spectrum power density of white Gaussian noise (WGN) at the HT-UAV. Similar to Equation (10), we can calculate the uplink data rate using Shannon's theorem:

$$R_{j,k}(t) = \frac{\mathcal{B}_2}{J} log_2(1 + \mathbb{S}_{j,k}(t)). \quad (14)$$

D. Data Aggregation Model

For data aggregation from ground IoT nodes using a UAV, a definitive IoT device activation pattern is essential for designing appropriate waypoints and optimal hovering location [15].

1) *IoT Device Activation Model*

Monitoring IoT sensors such as smart metering are usually accompanied by periodic activation, whereas event-driven IoT sensors such as wildfire monitoring follow random activation scenarios [11]. The central server has prior information regarding the periodic activation conditions. Thus, the periodic IoT device activation model for the number of active IoT devices, $\mathcal{N}_{\mathrm{act}}$, over period $[0, T]$ can be expressed as

$$\mathcal{N}_{\mathrm{act}} = \frac{T}{\tau_i}, \quad (15)$$

where τ_i is the period during which the IoT device is active. In the case of randomly activated IoT devices that are often subjected to bursty traffic and short activation intervals, we incorporate the probability density function of the random activation model, $\mathcal{N}_{\mathrm{act}}$, as studied in [11] over the time period $[0, T]$, which is defined as

$$D(t) = \frac{t^{a-1}(T-t)^{\mathscr{b}-1}}{T^{a+\mathscr{b}-1}B(a,\mathscr{b})}, \quad (16)$$

where $B(a,\mathscr{b}) = \int_0^1 t^{a-1}(1-t)^{\mathscr{b}-1}dt$ is the beta function with parameters denoted as a and $\mathscr{b}$ known as shape parameters $(a,\mathscr{b} \geq 0)$. Using both periodic and random activation models is a crucial design consideration because we aim to determine the optimal hovering location for maximal data aggregation.

2) *Aggregation Cost Calculation*

The selection of an appropriate aggregation location is a prerequisite for energy-saving. In our work, we aim to find the optimal hovering location, where the maximum number of IoT devices can be served based on the received SINR at LT-UAV j. The number of active IoT devices at any given time can be expressed as $|N_i| = N_i^{\text{per}} + N_i^{\text{rand}}$. A more elaboration is

$$N_i(t) = \sum_{i=1}^{W_i^{\text{per}}} \varphi_i^{\text{per}}(t) + \sum_{i=1}^{W_i^{\text{rand}}} \varphi_i^{\text{rand}}(t), \quad (17)$$

where $\varphi_i^{\text{per}}(t)$ and $\varphi_i^{\text{rand}}(t)$ are binary functions and defined as

$$\varphi_i^{\text{per}}(t) = \begin{cases} 1, & \text{if } i \text{ is active at } t_{\text{p}} \\ 0, & \text{otherwise} \end{cases}. \quad (18a)$$

and

$$\varphi_i^{\text{rand}}(t) = \begin{cases} 1, & \text{if } D(t) \geq D(t_{\text{th}}) \\ 0, & \text{otherwise} \end{cases}. \quad (18b)$$

respectively.

Furthermore, if the SINR value reaches certain threshold, $\mathbb{S}_{\text{i,j}}^{\text{th}}$, then the IoT node establishes communication with LT-UAV and $\mathbb{S}_{i,j}(t) \geq \mathbb{S}_{i,j}^{\text{th}}$. Therefore, the indicator function can be defined as:

$$\mathring{\text{A}}_{i,j}(t) = \begin{cases} 1, \text{if } \mathbb{S}_{i,j}(t) \geq \mathbb{S}_{i,j}^{\text{th}} \\ 0, \quad \text{otherwise} \end{cases}. \quad (19)$$

To avoid multiple communications and ensure that one IoT node communicates with a particular LT-UAV simultaneously, we introduce another indicator function:

$$\tilde{A}_{i,j}(t) = \begin{cases} 1, \text{if } \mathring{\text{A}}_{i,j}(t) = 1 \text{ and } \varphi_i = 1 \\ 0, \quad \text{otherwise} \end{cases}. \quad (20)$$

Therefore, the modified expression for the data rate in Equation (10) is transformed into:

$$R_{i,j}(t) = \mathring{\text{A}}_{i,j}(t)\tilde{A}_{i,j}(t)\mathcal{B}_1 \, log_2(1 + \mathbb{S}_{i,j}(t)). \quad (21)$$

Finally, the time period for aggregating data at a particular hovering point of the LT-UAV can be expressed as

$$T_j^{\text{agg}} = \sum_{i=1}^{W_i^{\text{per}}} \varphi_i^{\text{per}} \left(\frac{S_i}{R_{i,j(t)}}\right) + \sum_{i=1}^{W_i^{\text{rand}}} \varphi_i^{\text{rand}} \left(\frac{S_i}{R_{i,j(t)}}\right). \quad (22)$$

where S_i is the data or task size collected from the IoT devices, and the energy during hovering can be expressed as

$$E_j^{\text{agg}}(t) = \sum_0^{T_j^{\text{agg}}} P\big(v(t)\big)\Delta t. \quad (23)$$

E. Local Computation Model

After all data from the IoT nodes are aggregated by the LT-UAV, the onboard processing unit starts processing the data locally. Similar to the local processing model in [20], the internal unit of each LT-UAV was equipped with a single-core CPU. Thus, the LT-UAV can execute only one task at a time, and the remainder is offloaded to the HT-UAV for further processing. Thus, the queuing delay was not considered in the proposed model. Because IoT nodes are homogenous, the task size is uniform, and the number of arriving tasks is the same as the number of active IoT nodes at a particular time, $N_i(t)$. The duration of the local computing can be expressed as

$$T_j^{\text{loc}}(\text{t}) = \sum_{j=1}^{\text{J}}(1 - \Phi_{j,k}) \frac{S_i V_j}{f_j^{\text{loc}}}, \quad (24)$$

where f_j^{loc} is the local CPU frequency of the LT-UAV, and V_j is the task processing density (in CPU cycles/bit) to complete the task. $\Phi_{j,k} \in [0,1]$ is the binary decision variable for either local execution or offloading to the HT-UAV, and can be represented as

$$\Phi_{j,k} = \begin{cases} 1, & \text{if offloaded} \\ 0, & \text{locally computed} \end{cases}. \quad (25)$$

We can compute the energy expanded for the local computation as

$$E_j^{\text{loc}}(t) = P_j^{\text{loc}} \times T_j^{\text{loc}}(t) = P_j^{\text{loc}} \sum_{j=1}^{J}(1 - \Phi_{j,k}) \frac{S_i V_{\text{j}}}{f_{\text{j}}^{\text{loc}}}, \quad (26)$$

where P_j^{loc} is the power requirement for local computation and is proportional to the cubic power of the local frequency f_j^{loc} of the LT-UAV [23]. The resulting equation is as follows:

$$E_j^{\text{loc}}(t) = \mu(f_{\text{j}}^{\text{loc}})^2 \sum_{j=1}^{J}(1 - \Phi_{j,k}) S_i V_j, \quad (27)$$

where μ is the LT-UAV's effective capacitance factor subjected to the CPU chip architecture.

F. Offloading Computation Model

Considering the limited resources and computational capacity of LT-UAVs, tasks are offloaded to HT-UAV. In our study, the processing power of the HT-UAV is limited but sufficient enough to process the tasks come from a given number of LT-UAVs as in [24]. Regarding the resource allocation in the HT-UAV, interesting readers may refer to [24] for more details. Therefore, the transmission time from an LT-UAV j to an HT-UAV k can be written as

$$T_j^{\text{tran}}(t) = \sum_{\text{j=1}}^{\text{J}} \sum_{n=0}^{N_i - 1} (\Phi_{j,k}) \frac{S_i V_k}{R_{j,k}(t)}. \quad (28)$$

Similarly, the energy requirement for the data transmission is expressed as

$$E_j^{\text{tran}}(t) = P_j^{\text{tran}} \times T_j^{\text{tran}}(t), \quad (29)$$

where P_j^{tran} is the transmission energy required for offloading. Similar to the previously defined duration of the local computation, we define the offloading computation time as

$$T_j^{\mathrm{Off}}(\mathrm{t}) = \sum_{j=1}^{\mathrm{J}} \sum_{n=0}^{Ni-1} (\Phi_{j,k}) \frac{S_i V_k}{f_k^{\mathrm{off}}}, \quad (30)$$

where f_k^{off} denotes the computation capacity (cycles/sec) of the HT-UAV edge unit which is allocated to the LT-UAV j at time slot t [25] and V_k is the task processing density (in CPU cycles/bit). Similarly, the energy expanded for offloading can be calculated as follows:

$$E_k^{\mathrm{off}}(t) = P_j^{\mathrm{off}} \times T_j^{\mathrm{Off}} = P_j^{\mathrm{off}} \sum_{j=1}^{\mathrm{J}} \sum_{n=0}^{Ni-1} (\Phi_{j,k}) \frac{S_i V_k}{f_k^{\mathrm{off}}}, \quad (31)$$

where P_j^{off} denotes the processing power required for the HT-UAV and μ is the effective capacitance factor subjected to the CPU chip architecture. As mentioned previously, the resulting equation becomes

$$E_k^{\mathrm{off}}(t) = \mu (f_k^{\mathrm{off}})^2 \sum_{j=1}^{J} \sum_{n=0}^{Ni-1} (\Phi_{j,k}) S_i V_k, \quad (32)$$

G. Energy and Delay Cost Calculation

According to all the defined equations, we can obtain the total cost of energy and delay associated with the proposed problem. Therefore, the overall energy cost associated with the joint data aggregation and offloading processes can be expressed as:

$$E_{\mathrm{tot}}(t) = E_j^{\mathrm{fly}}(t) + E_j^{\mathrm{agg}}(t) + E_j^{\mathrm{loc}}(t) + E_j^{\mathrm{tran}}(t) + E_k^{\mathrm{off}}(t). \quad (33)$$

Similarly, the total delay can be expressed as

$$T_{\mathrm{tot}}(t) = T_j^{\mathrm{agg}}(t) + T_j^{\mathrm{loc}}(t) + T_j^{\mathrm{tran}}(t) + T_k^{\mathrm{off}}(t). \quad (34)$$

IV. Problem Formulation

Our objective is to design an optimized algorithm for the overall data aggregation and offloading processes while serving the maximum number of IoT nodes. Based on an earlier discussion, our goal is to minimize both the total energy consumption of LT-UAVs and the total aggregation and task execution time. Task execution time is the elapsed time for local execution and offloading. The total aggregation and task execution time is the time elapsed from the beginning of the data aggregation (i.e., the first transmission of the data to be aggregated from the IoT devices) to the end of the computation offloading (i.e., the last reception of the task to be offloaded). To define the optimization problem, we normalized E_{tot} and T_{tot} as $E_n = E_{\mathrm{tot}}/E_{\max}$ and $T_n = T_{\mathrm{tot}}/T_{\max}$, respectively, where $E_{\max}$ and $T_{\max}$ are the maximum values of E_{tot} and T_{tot}, respectively. The maximum values of E_{tot} and T_{tot} are dynamically updated at each time step. The optimization problem can be formulated as follows:

$$\mathrm{P}_1: \quad \min_{N_{\mathrm{i}}, H} \omega_1 E_{\mathrm{n}} + \omega_2 T_{\mathrm{n}} \quad (35)$$

$$s.t. \; E_{\mathrm{tot}} \le E_{\mathrm{th}}, \quad (\mathrm{C1})$$
$$0 \le \emptyset(t) \le 2\pi, \quad (\mathrm{C2})$$
$$0 \le d(t) \le d_{\max}, \quad (\mathrm{C3})$$
$$\mathbb{S}_{i,j}(t) \ge \mathbb{S}_{i,j}^{\mathrm{th}}, \quad (\mathrm{C4})$$
$$0 \le \mathring{\mathrm{A}}_{i,j}(t) \le 1, \quad (\mathrm{C5})$$
$$0 \le \tilde{A}_{i,j}(t) \le 1, \quad (\mathrm{C6})$$
$$S_i \le S_{i(\max)} \quad (\mathrm{C7})$$
$$0 \le \Phi_{j,k} \le 1, \quad (\mathrm{C8})$$

where ω_1 and ω_2 are the weight parameters for the total energy requirement of LT-UAVs and the total aggregation and task execution time, respectively, and $\omega_1 + \omega_2 = 1$. Based on the mission requirement, the two parameters ω_1 and ω_2 can be adjusted. Constraint C1 ensures that each UAV does not exceed the maximum threshold energy available for the duration of the mission. Constraints C2 and C3 are UAV movement constraints. Constraint C4 ensures the optimal hovering location selection based on the received SINR between the UAV and IoT nodes. where C5 is the indicator constraint for C4, Constraint C6 ensures that each IoT node can be connected simultaneously to a particular LT-UAV. Constraint C7 ensures that the UAV buffer memory is not overflowed by incoming data. C8 is the offloading constraint between LT-UAVs and HT-UAV, which depends on the computational capability of LT-UAVs.

To select the optimal hovering location $H = \|h_{\mathrm{e}} - h_{\mathrm{f}}\|$ for data aggregation, we introduce several constraints on the UAV mobility to minimize the aggregation energy and the hovering time [12]. Therefore, we introduce another optimization problem for solving the UAV trajectory problem, which can be expressed as

$$\mathrm{P}_2: \quad \min_{H} \sum_{\mathrm{u}=1}^{U} \sum_{\mathrm{e}=0}^{E} \sum_{\substack{\mathrm{f}=0 \\ \mathrm{f} \ne \mathrm{e}}}^{F} \|h_{\mathrm{e}} - h_{\mathrm{f}}\| \, x_{\mathrm{ef}}^{\mathrm{U}} \quad (36)$$

$$s.t. \; \sum_{\mathrm{u}=1}^{\mathrm{U}} \sum_{\mathrm{e}=0}^{\mathrm{E}} x_{\mathrm{ef}}^{\mathrm{U}} = 1 \; \forall_f = 1, \dots, F, \; \mathrm{f} \ne \mathrm{i}, \quad (\mathrm{C9})$$
$$\sum_{\mathrm{e}=1}^{\mathrm{E}} x_{\mathrm{eg}}^{\mathrm{U}} - \sum_{\mathrm{f}=1}^{\mathrm{E}} x_{\mathrm{gf}}^{\mathrm{U}} = 0 \; \forall_{\mathrm{g}} = 1, \dots, G, \quad (\mathrm{C10})$$
$$\sum_{\mathrm{f}=1}^{\mathrm{F}} x_{0\mathrm{f}}^{\mathrm{U}} = 1 \; \forall_{\mathrm{f}} = 1, \dots, \mathrm{F}, \quad (\mathrm{C11})$$
$$\sum_{\mathrm{f}}^{\mathrm{F}} x_{\mathrm{f}0}^{\mathrm{U}} = 1 \; \forall_{\mathrm{f}} = \mathrm{F}, \quad (\mathrm{C12})$$
$$\vartheta_{h_{\mathrm{e}}} - \vartheta_{h_{\mathrm{f}}} + E \sum_{\mathrm{u}=1}^{\mathrm{U}} x_{\mathrm{ef}}^{\mathrm{U}} \le E - 1, \quad (\mathrm{C13})$$
$$2 \le \mathrm{e} \ne \mathrm{f} \le E, \quad (\mathrm{C14})$$

where, $x_{\mathrm{ef}}^{\mathrm{U}} \in \{0,1\}$ is a binary variable indicating LT-UAV's movement for points e to f. Constraint C9 ensures that each hovering location is visited by UAV at least once. C10 ensures that each LT UAV leaves the same hovering point after aggregation. C11 and C12 indicate that the LT-UAV started its mission from the designated initial position and returned to the initial point after the completion of the mission. C13 and C14 are known as Miller-Tucker-Zemlin constraints [11], [26], which eliminate the subtour of LT-UAVs.

V. Joint Data Aggregation and Computation Offloading

In this section, we introduce an MA-DRL-based approach to address proposed optimization problem P1 for joint data aggregation and computation offloading. We first model our original optimization problem as a Markov game, and then use the VD3QN [27], which is an off-policy-based approach to solve the optimization problem.

A. Markov Game Formation

Because we deployed multiple UAV for joint data aggregation and offloading, each UAV's action was affected by the collaborative action of the other UAVs. Therefore, the proposed optimization problem can be transformed into a Markov game framework. The Markov game is an extension of the Markov decision process (MDP) for multi-agent scenarios [28]. A Markov game with Nnumber agents can be designated as tuple (S, A, R, P), where S, A, R, P denote the state, action, reward function, and state-transition probability, respectively. At each time step, the agent η takes an action $a_\eta \in A$ based on a certain policy after observing the current environment state $s_\eta \in S$. A next state s' is chosen according to the state transition probability $P(s'_\eta | s_\eta, a_1, a_2, a_3, \dots, a_\eta)$. By selecting the next state in an environment, reward r is obtained based on the reward function R. In terms of machine learning, a reward is simply a quantitative value that demonstrates the amount of an agent's action that has an impact on the agent's learning or objective.

We next discuss each component's definition for the Markov game formulation.

- Agent: Each LT-UAV is considered an agent as it begins interacting with the given environment and other LT-UAVs to maximize collaborative non-overlapping rewards and exchange information with each other. Therefore, the environment becomes fully observable, and each observation can be considered a state. Because each LT-UAV (agent) is deployed from the depot for data aggregation and makes local computations or offloading decisions, each UAV performs an appropriate action based on its respective policy. As each action is performed, a reward is generated from the environment and forwarded to the subsequent state. When each agent reaches its optimal goal, it stops receiving an additional reward from the environment or moves to the next state. Our approach is an off-policy which allows the agent to learn from a mixture of data generated by different policies. The key idea is that they separate the policy used to explore the environment from the policy being learned.
- State S: Because we deployed multiple LT-UAV in our simulation environment, our optimization problem can be described as a multi-agent Markov game. Every agent has its own state and acts independently of others. For the agent η, the state space, s_η, can be defined as

$$s_\eta = \{O_\eta, O_{-\eta}\}. \tag{37}$$

The state space has two components, the first component O_η is the self-observations of LT-UAV, whereas, the second component $O_{-\eta}$ is the observation of the other LT-UAVs. The self-observations, O_η, can be defined as $O_\eta = \{b_\eta, E_\eta, i, \mathcal{U}_\eta, \mathbb{S}_\eta^{i,j}, \tilde{A}_{i,j(\eta)}, N_i\}$ where b_η is the network identifier of each LT-UAV as we utilize the network-sharing method [29] and represented by one-hot vector, E_η is the remaining energy of the LT-UAV, i is the location information of IoT nodes, $\mathcal{U}_\eta$ is the location information, $\mathbb{S}_\eta^{i,j}$ is the SINR between i and j, $\tilde{A}_{i,j(\eta)}$ is the one-hot vector indexing indicating the UAV and IoT association and N_i is the task aggregated by LT-UAV to process. Similarly, $O_{-\eta}$ is the shared observation resulting from the other agents in the environment and can be described as $O_{-\eta} = \{i, \mathcal{U}_{-\eta}, \tilde{A}_{i,j(-\eta)}\}$ where $\mathcal{U}_{-\eta}$ is the location information of other LT-UAVs and $\tilde{A}_{i,j(-\eta)}$ is the device association parameter.

- Action A: Each agent requires an appropriate action in every time slot based on the current self and shared observations. The combined action of the agents can be expressed as $a_\eta = \{a_M, \mathring{A}_{i,j(\eta)}, \tilde{A}_{i,j(\eta)}, x_{ef}^U, \Phi_{j(\eta),k}\}$ where all the actions are taken in discrete action place and $\mathring{A}_{i,j(\eta)}, \tilde{A}_{i,j(\eta)}, x_{ef}^U, \Phi_{j(\eta),k} \in \aleph$, $\aleph$ being the number of possible actions of $\mathring{A}_{i,j(\eta)}, \tilde{A}_{i,j(\eta)}, x_{ef}^U, \Phi_{j(\eta),k}$ and are all binary variables. By integrating the LT-UAV mobility in horizontal direction, $\emptyset$ in discrete action space, the total number of possible actions for the LT-UAV is $2^\aleph \times \emptyset$.
- State Transition Probability P: The state of each agent or LT-UAV depends on its present location. Using equation (2) we can define the deterministic environment for the LT-UAV's position where the state transition probability for the next state of the agent is $P(s'_\eta | s_\eta, a_1, a_2, a_3, \dots, a_\eta) = 1$.
- Reward function R: As discussed earlier, the reward is the quantitative value received by an agent after interacting with the given environment, which numerically demonstrates how well the optimization objective has been achieved. The reward for the discrete time step can be defined as:

$$r_t = r_c + r_e + r_p\,. \tag{38}$$

As indicated in (34), the reward equation comprises the following three parts: The first part r_c is awarded to successfully complete the overall mission and is a positive number. The second part r_e is a violation constraint owing to the energy of the agent and the negative number. The final term is the penalty term r_p, which is also a negative number. The penalty term is $r_p = r_{SINR} + r_{ass} + r_{mov} + r_{off}$, where r_{SINR} is the SINR constraint violation term, r_{ass} is the device association violation term, r_{mov} is the movement constraint violation term, and r_{off} is the offloading constraint violation term.

For each episode of time step τ, minimizing the overall energy and delay for the aggregation and offloading process, our proposed problem (P1) turns into maximizing the cumulative reward $G = \sum_{t=1}^{T} \sum_{\eta=1}^{N} r_t^\eta$. Therefore, the proposed Markov game formulation was an episodic task [30]. In every episode, the agent begins its journey from the initial state and ends in the terminal state by returning to its initial deployment position.

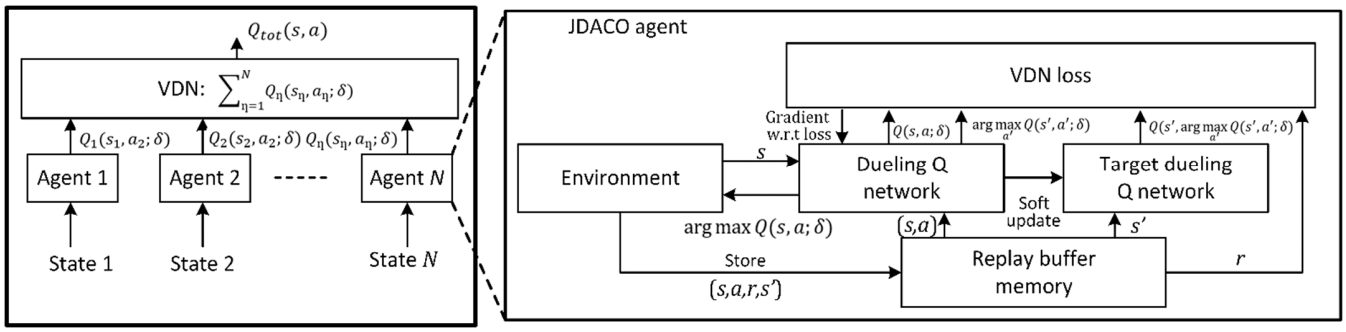


Fig. 2. JDACO architecture and workflow.

B. VD3QN Based Solution Approach

To solve our modified formulated problem, we adopted a learning-based algorithm called VD3QN [27], which is a combination of VDN [29] and a Dueling Double Deep Q Network (D3QN) [31] as shown in **Fig. 2**. We modified the existing VD3QN algorithm to our advantage and modeled the proposed problem accordingly. The D3QN act as decision maker for each agent using the local action values $Q(s_\eta, a_\eta)$, whereas the VDN generates the global action value $Q_{\text{tot}}(s, a)$. Therefore, sequential optimal actions were achieved by achieving a common objective for an individual agent or LT-UAV. In the following section, the D3QN and VDN architectures are studied to solve the formulated Markov game.

1) *D3QN***:** To obtain an optimal policy for an action-value function, the D3QN can be utilized as a value-based reinforcement learning technique [31]. Unlike standard DQNs [32], D3QN approximates the state value and state-dependent information first for each action taken and then perform aggregation function of the layers to obtain estimated action value function Q.

Each agent acquires observation of the environmental state s and utilizing the parameterized deep neural network (DNN) to produce action-value function, $Q(s, a; \delta)$ which is an approximal value of the original action-value function, $Q(s, a)$. Moreover, utilizing the dueling architecture, the D3QN can quickly identify the Q value, which ultimately helps in a faster training process by choosing the appropriate action.

To learn the parameters of the neural network (NN), storing the state transitions (s, a, r, s') in experience replay buffer $\boldsymbol{B}$ plays a significant role, where s' is the next state after action a is performed and reward r is received in return. As the training phase continued, a minibatch of state transitions was randomly chosen from the replay memory buffer. Then, the parameters are brought up to date each time by reducing the square of the temporal difference (TD) error, which is given by

$$L(\delta) = \boldsymbol{E}_{s,a,r,s'}[\left(\boldsymbol{y}^{D3QN} - Q(s, a; \delta)\right)^2]. \quad (39)$$

To address the overestimation problem of original Q-learning, we utilized the double Q-learning architecture [33], which is given by

$$\boldsymbol{y}^{\text{D3QN}} = r + \varsigma Q\,(\text{s}', \arg\min_{\text{a}'} Q(\text{s}', \text{a}'\,; \delta); \delta^{\text{t}}). \quad (40)$$

where ς is the discount factor and δ^{t} is the target parameters of the target neural network. It is to be noted that the architecture of the target network is same as action-value NN which obtains value from δ to ensure stable learning [34].

2) *VDN Architecture***:** In the proposed model, each agent works for the common objective of maximizing the number of devices served while minimizing the overall energy and time. Value decomposition divides the value function of a multi-agent problem into separate value functions for each agent. This allows agents to learn to cooperate with each other because they do not compete for the same resources. Therefore, all agents work independently and share their current state and observations cooperatively to find the global solution. Thus, we adopted the VDN [29] approach to find the global action-value function, which is denoted by Q_{tot}.VDN calculates the joint action-value function using the value-decomposition layer. Then, the summation of the action-value functions is calculated from the other agents, which is defined as

$$Q_{tot}(s, a) = \sum_{\eta=1}^{N} Q_\eta(s_\eta, a_\eta; \delta), \quad (41)$$

where s_η and a_η are each agent's state and actions respectively. By utilizing the value-decomposition layer, each agent can learn a better joint action in a noncompeting cooperative manner.

Algorithm 1 describes the proposed JDACO algorithm. In the training mode, every episode is defined by events where all agents start from the initial position, carry out aggregation, local computing, and offloading procedures, and then return to the initial position based on the remaining battery level. For each agent, each episode begins with $\tau = 0$ with initial state and reset all other parameters as defined in line 3. In line 4, we impose the maximum number of allowable steps to prevent an agent wandering around, and an energy condition is imposed to ensure that the agent does not fall off while wandering around. This is necessary as at the early stage of the training phase, agents have very little knowledge with a high probability of exploration, $\boldsymbol{\epsilon}$. Therefore, a new episode is initiated if an agent meets the desired target, or if a selected number of steps is reached. As the training phase continues, each agent encounters local state s_η (line 6). From line 7 to 8, based on the observed state, a random action is chosen from the action space using $\boldsymbol{\epsilon}$-greedy policy or using action-value function $Q(a_\eta, s_\eta; \boldsymbol{\delta})$ and then agent's location, energy information and other binary parameters are updated. As stated in line 10, the environment then generates reward r_η based on the prescribed reward formulation. As agents work in a cooperative manner, the sum of all the agent's reward is

Algorithm 1: JDACO for energy and delay minimization

Input: Maximum episode number $\sqcap_{\mathbf{eps}}$, maximum step number per episode $\sqcap_{\mathbf{stp}}$, exploration at start $\boldsymbol{\epsilon_0}$, decay rate $\boldsymbol{\varepsilon}$, achieving objective reward $\boldsymbol{r_o}$, defilement reward $\boldsymbol{r_d}$, replay buffer $\boldsymbol{B}$, batch size $\sqcap_{\mathbf{b}}$, Number of agent η, initial LT-UAV position $\boldsymbol{\mathcal{U}_0}$, LT-UAV maximum energy $\boldsymbol{E}_{\max}$ IoT node location $\boldsymbol{i}$, Number of tasks $\boldsymbol{N_i}$, initial parameters for NN $\boldsymbol{\delta}$, target parameters for NN $\boldsymbol{\delta^t}$, rate of learning $\boldsymbol{\alpha}$, rate of soft update $\boldsymbol{\beta}$.
Output: Trained parameter $\boldsymbol{\delta}$.

1 Initialize $\boldsymbol{\delta}, \boldsymbol{\epsilon} = \boldsymbol{\varepsilon_0}, \boldsymbol{\delta^t} = \boldsymbol{\delta}$;
2 **for** $n_{\text{eps}} = 1,2,3, \dots, \sqcap_{\text{eps}}$ **do**
3 Set timestep $\tau = 0$ and reset agent's position and other parameters
$\mathcal{U}_\eta(\text{t}) = \mathcal{U}_\eta^{\text{init}}$ for each LT-UAV η;
4 **while** $\mathcal{U}_\eta(\text{t}) \neq \mathcal{U}_\eta^{\text{fin}}$ and $E_\eta \leq \boldsymbol{E}_{\max}$ and $\tau \leq \sqcap_{\text{stp}}$ **do**
5 **for** agent $\eta = 1,2,3, \dots, N$ **do**
6 Get state s_η, based on agent's position;
7 Selection of action a_η from the defined action-space A using $\boldsymbol{\epsilon}$ -greedy exploration policy, as $a_\eta = \begin{cases} \text{random action}, & \text{probabilistic } \boldsymbol{\epsilon} \\ \arg\max_{a_\eta \in A} Q(a_\eta, s_\eta; \boldsymbol{\delta}), & \text{otherwise;} \end{cases}$
8 Perform action a_η, update agent's position $\mathcal{U}_\eta(\text{t})$, agent's energy and other binary parameters
9 **end**
10 Calculate reward r_t using equation (38) and obtain the cumulative reward G.
11 Collect combined action a, current state s and following state s';
12 Record the state transition information $(s, a, s', r_{\text{total}})$ in replay buffer $\boldsymbol{B}$;
13 Update new timestep $\tau \rightarrow \tau + 1$ and new exploration rate $\boldsymbol{\epsilon} \rightarrow \boldsymbol{\varepsilon} \times \boldsymbol{\epsilon}$
14 **end**
15 Sample $\sqcap_{\mathbf{b}}$ episodes of minibatch from replay buffer $\boldsymbol{B}$;
16 Obtain the loss function using equation (37)
17 Using gradient descent optimizer, update $\boldsymbol{\delta}$
$\boldsymbol{\delta} \rightarrow \boldsymbol{\delta} - \alpha \nabla_{\boldsymbol{\delta}} \big(y_{\text{tot}} - Q_{\text{tot}}(s, a; \boldsymbol{\delta})\big)$;
18 After every W episodes, update target parameter $\boldsymbol{\delta^t}$ using soft update mechanism using,
$\boldsymbol{\delta^t} = (1 - \boldsymbol{\beta})\boldsymbol{\delta^t} + \boldsymbol{\beta}\boldsymbol{\delta}$;
19 **end**

calculated as $r_{\text{total}} = \sum_\eta r_\eta$ (line 10). At this stage, the combined action a, current state s, following state s', and total reward r_{total} are recorded in the replay memory buffer $\boldsymbol{B}$. After that, time step τ and exploration rate $\boldsymbol{\epsilon}$ are updated as stated in line 13. Using the stored transition at the end, the DNN of the agent is trained, as stated in lines 15–18. More importantly, the $\boldsymbol{\delta}$ parameter is updated as loss function is minimized and denoted as:

$$L(\delta) = \frac{1}{\sqcap_{\mathbf{b}}} \sum_{\sqcap_{\mathbf{b}}} \left[\left(y_{\text{tot}} - Q_{\text{tot}}(s, a; \delta)\right)^2\right], \quad (42)$$

with

$$y_{\text{tot}} = r_{\text{total}} + \varsigma Q_{\text{tot}}\left(s', \arg\min_{a'} Q_{\text{tot}}(s', a'; \delta); \delta^{\text{t}}\right), \quad (43)$$

where $\sqcap_{\mathbf{b}}$ is the sampled episode number from replay buffer and $\boldsymbol{\delta^t}$ are the parameters of the target NN. To stabilize the training process, $\boldsymbol{\delta^t}$ are soft updated after every W episodes, as mentioned in line 18. Notably the aggregation operation is performed to sum the respective Q values and is not included in the parameters of the NN.

To analyze the complexity of our proposed scheme, we studied the time and space complexities of the DDQN training and VDN aggregation separately, and then studied the time and space complexities of our proposed JDACO algorithm based on the modified VD3QN. The time complexity of the DDQN architecture for experience collection is expressed by $O(\sqcap \times \sqcap_{\mathbf{stp}})$, where $\sqcap$ is the number of episodes during training phase and $\sqcap_{\mathbf{stp}}$ is the timesteps per episode. To update the replay buffer during training phase, the time complexity is denoted as $O(\sqcap_{\mathbf{b}} \times M)$, where $\sqcap_{\mathbf{b}}$ is the batch size and M is the number of iterations per episode. For the VDN aggregation, the time complexity can be given as $O(\eta \times A)$, where η being the number of agents and A being the cardinality of the individual agent's action space. Thus, the time complexity of JDACO can be given as $O(\sqcap_{\mathbf{stp}} \times S \times A^2)$, where S being the number of states and A **is** the number of possible actions from the action space. This is because JDACO must explore all possible combinations of decision variables for all agents at each time step.

The space complexity of the DDQN is $O(B + P)$, where B and P are the replay buffer size and number of parameters in the network used for storing experiences and neural parameters, respectively. As for the VDN space complexity, it can be defined as $O(A \times \eta)$, The space complexity of JDACO can be expressed as $O(S \times A)$ as JDACO needs to store the Q-table, for every state and action.

VI. Performance Evaluation

In this section, the performance of the proposed JDACO is evaluated by simulation results and compared to conventional schemes using TensorFlow framework version 1.15 on a desktop computer equipped with two 1070Ti processors with a total of 16GB of memory. To demonstrate the effectiveness of the proposed algorithm, we select two learning-based approaches as our benchmarks: Q-learning with mixing (Qmix) [35] and counterfactual multi-agent policy gradients (COMA) [36]. As our proposed scheme is an off-policy based algorithm, our benchmark selection procedure includes Qmix and COMA which are also off-policy based algorithms. Alongside, we also selected the heuristic greedy approach (HGA) as the non-learning based approach.

Similar to the VDN approach, Qmix is a value-based approach that utilizes centralized training decentralize execution method. It addresses the challenge of coordinating multiple agents to achieve a common goal by combining individual agent policies into a joint action-value function. We replace the gated recurrent unit (GRU) with a D3QN unit to adopt Qmix in our problem. On the other hand, COMA is a deep reinforcement learning algorithm that uses counterfactual reasoning to assign credits to individual agents in cooperative multi-agent systems. It combines a centralized critic to evaluate the joint action value with individual actor networks that select actions for each agent based on local observations. To incorporate COMA into our formulated MDP, we replace the GRU with multi-layer perceptron (MPL) NN. For the non-

learning-based approach, we formulate the HGA as a binary integer programming problem by utilizing only the binary variables that are solved by the Python library called PuLP [37].

A. Simulation Setup

We perform the primary simulation by deploying multiple LT-UAVs from the initial coordinates (0, 0). The IoT nodes are randomly deployed over an area of (10 × 10 Km^2). The coordinate of the HT-UAV is (5, 5). As we aim to minimize both the energy and delay for the aggregation and computational offloading processes simultaneously, we choose the two weight parameters as equal (i.e., $\omega_1 = \omega_2 = 0.5$). The simulation parameters with respective values are listed in Table III.

TABLE III
SIMULATION PARAMETERS

Parameter	Value
Simulation area	10×10 Km^2
η_{LoS} and η_{NLoS} values	1.6 dBm and 23 dBm
Carrier frequency,	2 GHz
Environment constant α and β	10.39, 0.05
Weight parameters ω_1 and ω_2	0.5 ($\omega_1 = \omega_2$)
UAV altitude, h_j	100 m
Number of IoT nodes	[20, 40, 60 80, 100 (default),]
Number of UAVs	[3, 5(default), 7]
Blade profile power, P_1	79.8563 W
Induced power, P_2	88.6279 W
Rotor blade tip speed, U_{tip}	120 m/s
Average induced velocity of rotor during hovering state, v_0	4.03 m/s
Fuselage drag ratio, d_0	0.6
Density of air, ρ	1.225 Kg/m^3
Solidity of the rotor, g	0.05
Disk area, A	0.503 m^2
Effective capacitance factor, μ	10^{-28}
Task size	[2.5, 5 (default), 7.5, 10] Mbit
Channel power, $\mathcal{P}_0$	1.42×10^{-4}
Local CPU frequency, f_j^{loc}	10^9 cycles/s
CPU cycle to finish the task, V_j	270 cycles/bit
CPU frequency of the HT-UAV, f_k^{off}	5×10^{10} cycles/s
Maximum episodes, $\sqcap_{\mathbf{eps}}$	1200
Maximum steps, $\sqcap_{\mathbf{stp}}$	60
Exploration probability at beginning	1
Mission completion reward, r_c	130
Energy violation reward, r_e	-20
Batch size, $\sqcap_{\mathbf{b}}$	64
Soft update, β	0.01
Learning rate, $\boldsymbol{\alpha}$	0.001
Discount factor, ٦	0.99
Buffer size $\boldsymbol{B}$	500000

In the proposed JDACO architecture, we utilize a feed-forward, fully connected neural network with three hidden layers containing 256, 512, and 128 neurons. The neuron in the final layer corresponds to all possible actions that the agents can take. For simplicity, we consider three degrees of freedom (forward, left, and right) for each LT-UAV. The simulation values for the propulsion-power calculation of the LT-UAV are adopted from [38]. It is important to note that different factors, such as the number of agents and violation constraints, can have an impact on algorithm convergence.

In our simulation, the following performance metrics are evaluated: A brief description of each metric is provided below.

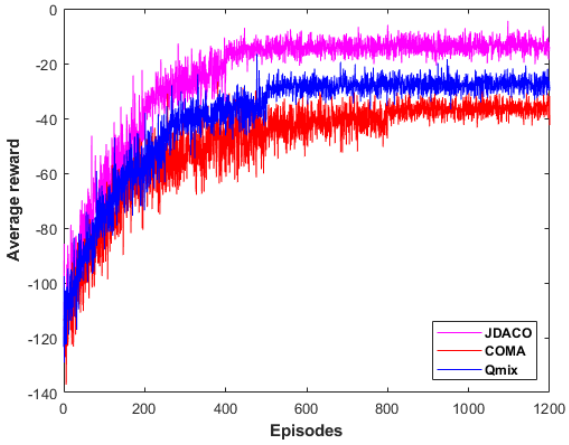
- Average reward: The performance indication of an agent over time helps to visualize the agent's learning interactions from the environment. It comprises the cumulative reward over time by improving the decision-making policy. An average reward curve or learning curve illustrates the fluctuating rewards as an agent explores different strategies to achieve convergence or stability of the learning process.
- Total number of IoT nodes in service: This performance metric indicates the active IoT nodes among all deployed IoT nodes that have successfully transmitted data to the LT-UAV for computation. Usually, a higher number suggests that the proposed scheme can achieve large amounts of data without missing any IoT nodes that are ready to upload the data.
- Total amount of computed data: This indicates the total amount of data that an LT-UAV can gather for processing. A higher amount of data computation indicates that the LT-UAV was able to aggregate data from IoT nodes for computation or offloading without missing any of the nodes, which might result in data loss owing to the overflow of the buffer memory of IoT nodes.
- Mission time: This indicates the average time required for LT-UAVs to complete their journey, starting from deployment from the initial position, data aggregation time, data computation time, offloading time, and finally returning to the initial deployment position. The shorter time required for the predefined energy of the LT-UAV demonstrates the effectiveness of the proposed scheme.
- Total energy consumption: The total energy required for the LT-UAV to complete its mission, which includes travelling to the optimal hovering location, data aggregation, computation, and energy consumption offloading. Overall, lower energy consumption is an indication of an energy-efficient scheme.
- Total aggregation and task execution time: This refers to the time required to process data starting from the aggregation point when the UAV is hovering. In this state, the LT-UAV aggregates data from the ground nodes until no other nodes are ready to transmit the data. The execution time refers to the combination of local computation by the LT-UAV, transmission from the LT-UAV to the HT-UAV and offloading by the HT-UAV. Because the delay for each data point is variable, we calculate the average delay for the overall aggregation, offloading, and computation processes. In our proposed scheme, we ignore the queuing delay because the LT-UAV processes only a single task at a time, and the HT-UAV has sufficient processing power, making the queuing delay insignificant. The aggregation time for the LT-UAV is higher than the task execution time, as the LT-UAV collects and aggregates data from IoT nodes and is usually expressed in seconds. On the other hand, task execution usually takes a shorter time frame of approximately a few milliseconds.

B. Simulation Results and Discussion

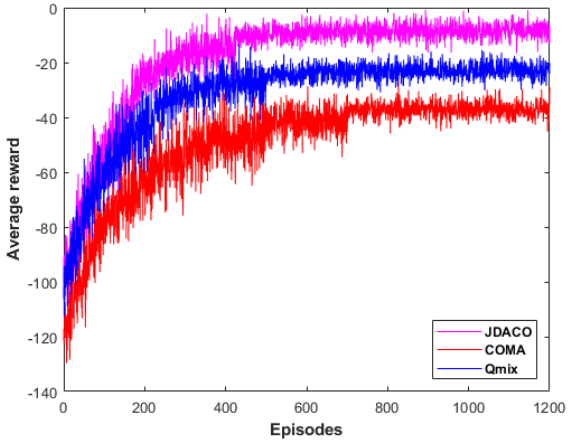
First, we compare the performances of the training processes of the DRL-based approaches, as illustrated in **Fig. 3**. The simulation results for the training process involve two instances with three LT-UAVs and five HT-UAVs for all DRL approaches. The results demonstrate the convergence of the algorithms for all instances. However, among learning-based approaches, COMA performs the worst. COMA operates under the principle of a counterfactual baseline mechanism, which inhibits the exploratory ability of the centralized critic. This renders COMA unsuitable for the proposed JDACO scheme. However, JDACO and Qmix show similarities in performance because both provide value-factorization-based solutions. The performance of the Qmix network can be improved by combining it with a more complex NN architecture and a global state with an action value. This is still unlikely to outshine the performance of JDACO because the local state of an agent has full observation of all other agents, and further improvement is not guaranteed. Our JDACO algorithm takes leverage of both VDN and D3QN architecture to reach faster convergence and better stability in learning. Additionally, JDACO leverages the dueling architecture, which helps to identify the Q value quickly, which ultimately helps a faster training process by choosing the appropriate action Overall, JDACO reaches convergence at a faster rate than other baseline algorithms, reducing the training time by 20% compared to the Qmix approach, which is the second fastest one among the baselines.

Fig. 4 shows the simulation scenario of our scheme with respect to an example deployment of HT-UAV, LT-UAVs, and IoT nodes. The IoT node distribution, LT-UAV coverage, HT-UAV coverage, and respective trajectories of the LT-UAVs are graphically shown. A simulated environment is generated using three LT-UAVs for 100 IoT nodes. The trajectory of each LT-UAV is demonstrated by different colors. The coordinate (0, 0) indicates the deployment points of the LT-UAVs, and coordinate (5, 5) indicates the position of the HT-UAVs. Note that the negative distance is a vector representation of the simulation area.

In **Fig. 5**, we explore the performance of all benchmarks for IoT devices. Compared to all the other benchmarks, HGA exhibits the poorest performance with 64% node coverage. This is understandable because a greedy approach aims to find the shortest way to finish the mission without prioritizing the number of IoT nodes in service and fulfilling the other constraints. On the other hand, among the learning-based benchmarks, JDACO and Qmix show similar performances, as both provide value-factorization-based solutions. The improvement of Qmix network is still not guaranteed even after combining it with a more complex NN architecture whereas JDACO benefits from the local state of an agent with full observation of all other agents. JDACO performs with 98% served IoT nodes which is superior to Qmix and COMA with 94% and 88%, respectively.



(a)



(b)

Fig. 3. Average reward with (a) 3 LT-UAVs and (b) 5 LT-UAVs.

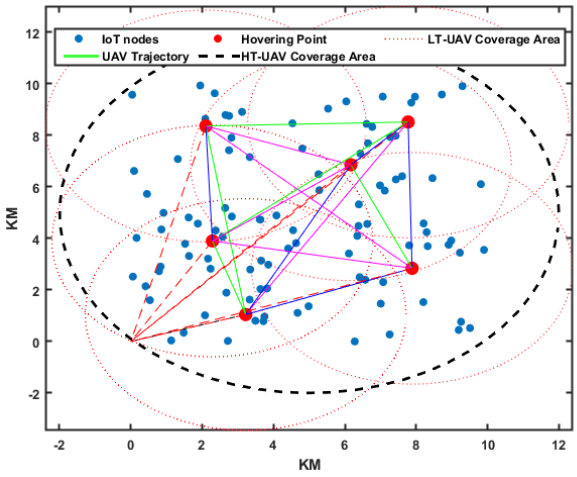


Fig. 4. An example deployment of HT-UAV, LT-UAVs, and IoT nodes.

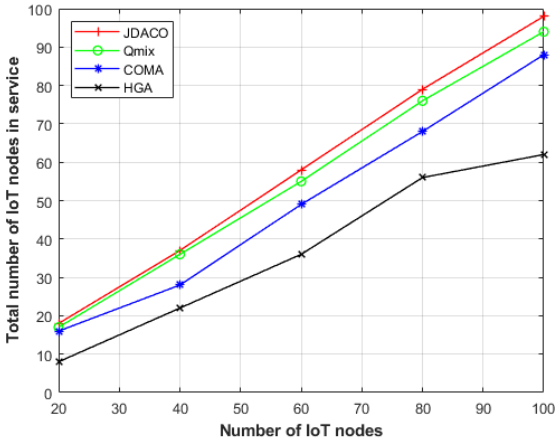


Fig. 5. Total number of IoT nodes in service.

We further compare the amount of computed data among the different baseline algorithms, as shown in **Fig. 6**. For simplicity, we assume that the deployed nodes are sensory in nature, and each aggregated datum per sensor contains approximately 5 Mbits of data. It can be observed that JDACO computes more data than the other baseline approaches. Our proposed JDACO scheme computes 4.9 Gbits of data for 100 IoT nodes whereas Qmix and COMA compute 4.4 Gbits and 4.1 Gbits of data, respectively, showing an increased computation volume of around 11.4%. The value decomposition architecture of JDACO divides the value function of a multi-agent problem into separate value functions for each agent. This allows agents to learn to cooperate with each other because they do not compete for the same resources. This indicates that less data loss is ensured by the proposed JDACO scheme, whereas the other schemes fail to compute a portion of their data. This result also indicates the linear scalability of our proposed scheme compared to other baseline algorithms.

We compare the mission times for the different schemes while varying the number of LT-UAVs deployed, as illustrated in **Fig. 7.** It is not surprising that the HGA scheme requires the longest time to complete the aggregation and offloading mission, as it must satisfy all conditions for aggregation and computation constraints. While it is true that increasing the number of UAVs reduces the overall average mission time for all baseline schemes, JDACO requires the shortest average mission time for all three use cases (i.e., for different numbers of LT-UAVs). In case of 5 LT-UAVs deployment scenario, JDACO takes only 206s on average whereas the other two learning-based approaches take 232s and 256s, respectively, and the non-learning-based approach takes 298s. Therefore, our scheme demonstrates an 11.2% reduced average mission time when compared to the baseline schemes. This is because the sequential optimal actions of our JDACO algorithm are achieved by going through with a common objective for an individual agent or LT-UAV.

We study the energy consumed by the LT-UAVs for different baseline schemes. We calculate the average energy expenditures for different CPU cycles for an LT-UAV with 100 IoT nodes. As shown in **Fig. 8**, the proposed JDACO algorithm consumes less average energy than the other learning-based algorithms. We also observe the impact of computational capability on the energy requirements. By proposing an energy-saving scheme, the UAVs can perform missions for a longer time in JDACO, which extends the scalability of the proposed scheme.

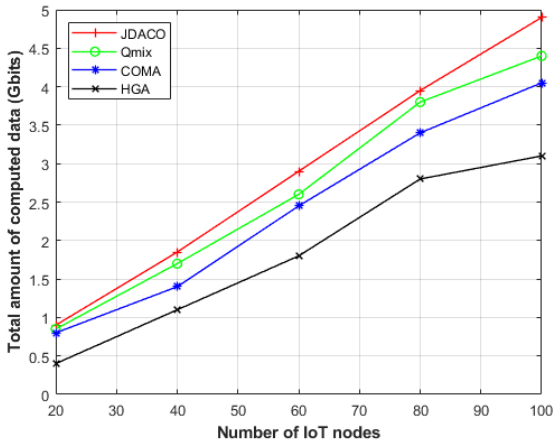


Fig. 6. Total amount of computed data.

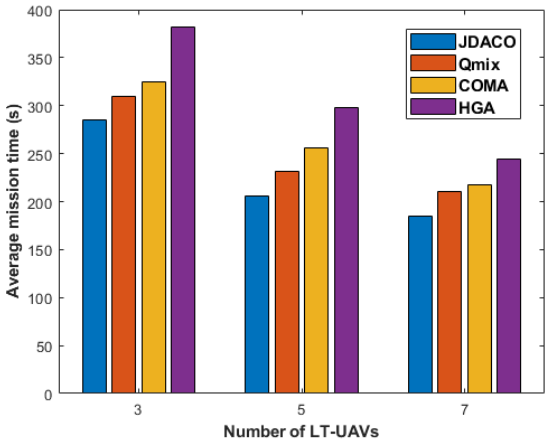


Fig. 7. Average mission time for different number of LT-UAVs.

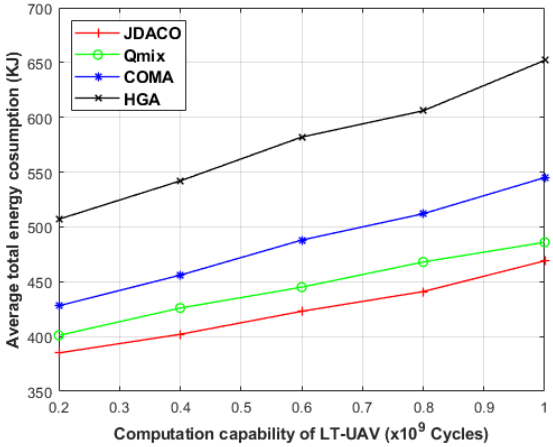
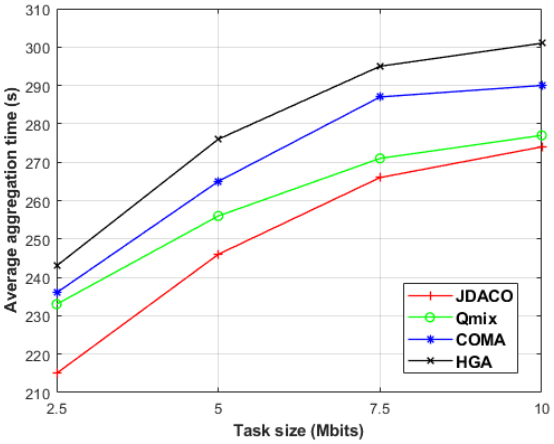
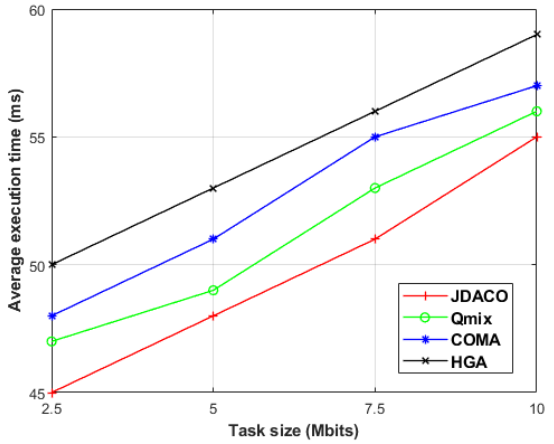


Fig. 8. Average total energy consumption of LT-UAVs for different computation power of LT-UAV processor.

We also evaluate the average aggregation and execution times for different task sizes and compared them with those of other benchmarks. We explore the average aggregation and offloading times for each benchmark because each LT-UAV has its own respective time based on observations from the environment. As seen from **Fig. 9,** our proposed JDACO scheme has overall less average aggregation and execution time when matched with other benchmarks. As for **Fig. 9(a)**, for the default 5 Mbits of task size, our proposed JDACO scheme has an average aggregation time of 245s which is 11s lower than the Qmix and 20s lower than the COMA approach. The HGA has the highest aggregation time of 276s for the same task size. As for the average execution time shown in **Fig. 9(b)**, our approach only requires 47ms whereas the other three benchmarks of Qmix, COMA and HGA require 48ms, 52ms and 55ms, respectively. The HGA has a higher time requirement for both instances of aggregation and execution time. This shows a clear distinction of reaching global optima effectively for the proposed scheme by utilizing the decomposition layer architecture. Although Qmix demonstrates a performance similar to that of our proposed scheme, COMA requires longer aggregation and task execution times in both cases.



(a)



(b)

Fig. 9. Average (a) aggregation time and (b) execution time for different task sizes.

We study the impact of task size on performance. In other words, the total energy consumption and mission execution time are observed by varying the task size. First, we study the average energy consumed by the UAVs for different task sizes. **Fig. 10** shows the energy consumed by the different benchmarks. It should be noted that increasing task size influences the overall energy consumption for the process (i.e., the task) to be completed. For the default 5 Mbits of data size, JDACO demonstrates an average reduction of 24 KJ which is around 5.6% of energy reduction. JDACO has a requirement of 402KJ whereas Qmix, COMA and HGA had an average energy requirement of 426KJ, 456KJ and 542KJ, respectively. Compared to the Qmix approach, our proposed scheme saves 24KJ of energy. This is expected as both JDACO and Qmix show similar performances since both provide value-factorization-based solutions. Still the proposed JDACO algorithm consumes the least amount of energy among all other benchmarks for the aggregation and computational offloading processes.

Similar to the energy consumption, we also examine the average mission time by varying the task size. Increasing the task size increases the overall mission time, because additional time is required to aggregate and compute the data for different task sizes. **Fig. 11** illustrates the different mission times required for the data aggregation and computation

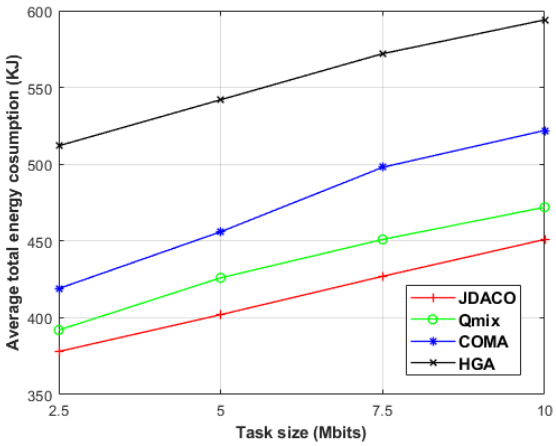


Fig. 10. Average total energy consumption of LT-UAVs for different task sizes.

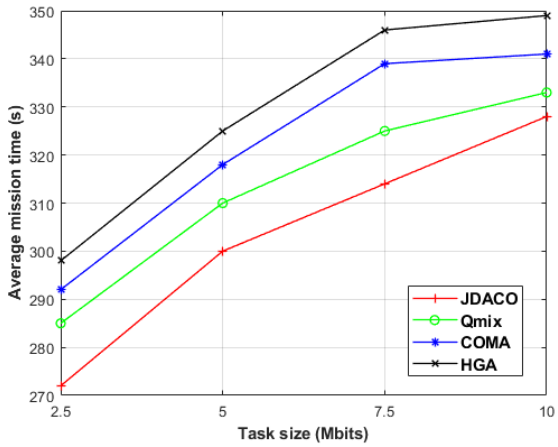


Fig. 11. Average mission time for different task sizes.

offloading processes for different schemes. Our proposed JDACO scheme requires the least mission time compared to all the other benchmarks. This is because COMA operates under the principle of a counterfactual baseline mechanism, which inhibits the exploratory ability of the centralized critic. For the task size of 5 Mbit, JDACO took a duration of 300s which is than 10s less than the Qmix algorithm.

VII. Conclusion

In this study, we have presented a joint data aggregation and computation offloading scheme for post-disaster scenarios that minimizes the total cost of energy consumption and delays the aggregation and offloading processes. Our work addresses the need for efficient and adaptable solutions in scenarios where timely data aggregation and processing are of utmost importance. The joint optimization problem has been defined and formulated as an MDP. We have then solved the formulated MDP problem by proposing an MA-DRL-based JDACO algorithm to perform discrete cooperative action. Our simulation study shows that the proposed JDACO algorithm performs superiorly compared to other benchmarks in terms of mission execution time (i.e., delay) and energy consumption, while ensuring the maximum number of IoT devices in service. In our future work, we will not only consider mobile ground nodes, but also incorporate the object detection ability of individual UAVs as an extension of this work. Additionally, we would like to further extend our work with the heterogeneous IoT nodes.

Acknowledgment

The authors thank the editor and anonymous referees for their comments, which helped to improve the quality of this research.

References

[1] W. Saad, M. Bennis, and M. Chen, "A Vision of 6G Wireless Systems: Applications, Trends, Technologies, and Open Research Problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, 2020, doi: 10.1109/MNET.001.1900287.

[2] G. Geraci *et al.*, "What Will the Future of UAV Cellular Communications Be? A Flight from 5G to 6G," *IEEE Commun. Surv. Tutorials*, vol. 24, no. 3, pp. 1304–1335, 2022, doi: 10.1109/COMST.2022.3171135.

[3] R. Akter, M. Golam, V.-S. Doan, J.-M. Lee, and D.-S. Kim, "IoMT-Net: Blockchain-Integrated Unauthorized UAV Localization Using Lightweight Convolution Neural Network for Internet of Military Things," *IEEE Internet Things J.*, vol. 10, no. 8, pp. 6634–6651, Apr. 2023, doi: 10.1109/JIOT.2022.3176310.

[4] 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, p. 1463, Jan. 2023, doi: 10.3390/s23031463.

[5] M. Dai, T. H. Luan, Z. Su, N. Zhang, Q. Xu, and R. Li, "Joint Channel Allocation and Data Delivery for UAV-Assisted Cooperative Transportation Communications in Post-Disaster Networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 16676–16689, 2022, doi: 10.1109/TITS.2022.3178789.

[6] B. Alzahrani, O. S. Oubbati, A. Barnawi, M. Atiquzzaman, and D. Alghazzawi, "UAV assistance paradigm: State-of-the-art in applications and challenges," *J. Netw. Comput. Appl.*, vol. 166, no. February, p. 102706, Sep. 2020, doi: 10.1016/j.jnca.2020.102706.

[7] K. Messaoudi, O. S. Oubbati, A. Rachedi, A. Lakas, T. Bendouma, and N. Chaib, "A survey of UAV-based data collection: Challenges, solutions and future perspectives," *J. Netw. Comput. Appl.*, vol. 216, no. January, p. 103670, Jul. 2023, doi: 10.1016/j.jnca.2023.103670.

[8] P. McEnroe, 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. 9, no. 17, pp. 15435–15459, 2022, doi: 10.1109/JIOT.2022.3176400.

[9] K. Li, W. Ni, A. Noor, and M. Guizani, "Employing Intelligent Aerial Data Aggregators for the Internet of Things: Challenges and Solutions," *IEEE Internet Things Mag.*, vol. 5, no. 1, pp. 136–141, 2022, doi: 10.1109/iotm.001.2100161.

[10] S. M. A. Huda and S. Moh, "Survey on computation offloading in UAV-Enabled mobile edge computing," *J. Netw. Comput. Appl.*, vol. 201, no. October 2021, p. 103341, 2022, doi: 10.1016/j.jnca.2022.103341.

[11] A. Bera, S. Misra, C. Chatterjee, and S. Mao, "CEDAN: Cost-Effective Data Aggregation for UAV-Enabled IoT Networks," *IEEE Trans. Mob. Comput.*, vol. 1233, no. c, pp. 1–1, 2022, doi: 10.1109/TMC.2022.3172444.

[12] A. M. Raivi and S. Moh, "A comprehensive survey on data aggregation techniques in UAV-enabled Internet of things," *Comput. Sci. Rev.*, vol. 50, no. 2, p. 100599, Nov. 2023, doi: 10.1016/j.cosrev.2023.100599.

[13] A. A. Baktayan and I. A. Al-Baltah, "A survey on intelligent computation offloading and pricing strategy in UAV-Enabled MEC network: Challenges and research directions," *Sustain. Eng. Innov.*, vol. 4, no. 2, pp. 156–190, Dec. 2022, doi: 10.37868/sei.v4i2.id179.

[14] L. Bai, J. Liu, J. Wang, R. Han, and J. Choi, "Data Aggregation in UAV-Aided Random Access for Internet of Vehicles," *IEEE Internet Things J.*, vol. 9, no. 8, pp. 5755–5764, 2022, doi: 10.1109/JIOT.2021.3063734.

[15] O. M. Bushnaq, A. Celik, H. Elsawy, M. S. Alouini, and T. Y. Al-Naffouri, "Aeronautical Data Aggregation and Field Estimation in IoT Networks: Hovering and Traveling Time Dilemma of UAVs," *IEEE Trans. Wirel. Commun.*, vol. 18, no. 10, pp. 4620–4635, 2019, doi: 10.1109/TWC.2019.2921955.

[16] K. Li, W. Ni, Y. Emami, and F. Dressler, "Data-Driven Flight Control of Internet-of-Drones for Sensor Data Aggregation Using Multi-Agent Deep Reinforcement Learning," *IEEE Wirel. Commun.*, vol. 29, no. 4, pp. 18–23, Aug. 2022, doi: 10.1109/MWC.002.2100681.

[17] S. M. A. Huda and S. Moh, "Deep Reinforcement Learning-Based Computation Offloading in UAV Swarm-Enabled Edge Computing for Surveillance Applications," *IEEE Access*, vol. 11, no. June, pp. 68269–68285, 2023, doi: 10.1109/ACCESS.2023.3292938.

[18] Z. Jia, Q. Wu, S. Member, and C. Dong, "Hierarchical Aerial Computing for Internet of Things via Cooperation of HAPs and UAVs," *IEEE Internet Things J.*, vol. 10, no. 7, pp. 5676–5688, 2023, doi: 10.1109/JIOT.2022.3151639.

[19] F. Song *et al.*, "Evolutionary Multi-Objective Reinforcement Learning Based Trajectory Control and Task Offloading in UAV-Assisted Mobile Edge Computing," *IEEE Trans. Mob. Comput.*, pp. 1–18, 2022, doi: 10.1109/TMC.2022.3208457.

[20] T. Cai *et al.*, "Cooperative Data Sensing and Computation Offloading in UAV-Assisted Crowdsensing With Multi-Agent Deep Reinforcement Learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3197–3211, Sep. 2022, doi: 10.1109/TNSE.2021.3121690.

[21] Y. Yu, J. Tang, J. Huang, X. Zhang, D. K. C. So, and K.-K. Wong, "Multi-Objective Optimization for UAV-Assisted Wireless Powered IoT Networks Based on Extended DDPG Algorithm," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 6361–6374, Sep. 2021, doi: 10.1109/TCOMM.2021.3089476.

[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. 71, no. 4, pp. 4225–4236, Apr. 2022, doi: 10.1109/TVT.2022.3141799.

[23] Q. Liu, H. Liang, R. Luo, and Q. Liu, "Energy-Efficiency Computation Offloading Strategy in UAV Aided V2X Network With Integrated Sensing and Communication," *IEEE Open J. Commun. Soc.*, vol. 3, no. June, pp. 1337–1346, 2022, doi: 10.1109/OJCOMS.2022.3195703.

[24] H. Kang, X. Chang, J. Mišić, V. B. Mišić, J. Fan, and Y. Liu, "Cooperative UAV Resource Allocation and Task Offloading in Hierarchical Aerial Computing Systems: A MAPPO-Based Approach," *IEEE Internet Things J.*, vol. 10, no. 12, pp. 10497–10509, Jun. 2023, doi: 10.1109/JIOT.2023.3240173.

[25] Y. K. Tun, Y. M. Park, N. H. Tran, W. Saad, S. R. Pandey, and C. S. Hong, "Energy-Efficient Resource Management in UAV-Assisted Mobile Edge Computing," *IEEE Commun. Lett.*, vol. 25, no. 1, pp. 249–253, Jan. 2021, doi: 10.1109/LCOMM.2020.3026033.

[26] C. E. Miller, R. A. Zemlin, and A. W. Tucker, "Integer Programming Formulation of Traveling Salesman Problems," *J. ACM*, vol. 7, no. 4, pp. 326–329, 1960, doi: 10.1145/321043.321046.
[27] R. Han, H. Li, E. J. Knoblock, M. R. Gasper, and R. D. Apaza, "Joint Velocity and Spectrum Optimization in Urban Air Transportation System via Multi-agent Deep Reinforcement Learning," *IEEE Trans. Veh. Technol.*, pp. 1–13, 2023, doi: 10.1109/TVT.2023.3256067.
[28] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings 1994*, vol. 120, no. 1, Elsevier, 1994, pp. 157–163. doi: 10.1016/B978-1-55860-335-6.50027-1.
[29] P. Sunehag *et al.*, "Value-Decomposition Networks For Cooperative Multi-Agent Learning," *Proc. Int. Jt. Conf. Auton. Agents Multiagent Syst. AAMAS*, vol. 3, pp. 2085–2087, Jun. 2017, [Online]. Available: http://arxiv.org/abs/1706.05296
[30] A. G. B. Richard S. Sutton, *Reinforcement Learning: An Introduction*, Second Edi. A Bradford Book, 1998.
[31] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Frcitas, "Dueling Network Architectures for Deep Reinforcement Learning," *33rd Int. Conf. Mach. Learn. ICML 2016*, vol. 4, no. 9, pp. 2939–2947, 2016.
[32] B. Jang, M. Kim, G. Harerimana, and J. W. Kim, "Q-Learning Algorithms: A Comprehensive Classification and Applications," *IEEE Access*, vol. 7, pp. 133653–133667, 2019, doi: 10.1109/ACCESS.2019.2941229.
[33] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-Learning," *30th AAAI Conf. Artif. Intell. AAAI 2016*, pp. 2094–2100, 2016, doi: 10.1609/aaai.v30i1.10295.
[34] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015, doi: 10.1038/nature14236.
[35] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning," *35th Int. Conf. Mach. Learn. ICML 2018*, vol. 10, pp. 4295–4304, Mar. 2018, [Online]. Available: http://arxiv.org/abs/2003.08839
[36] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," *32nd AAAI Conf. Artif. Intell. AAAI 2018*, pp. 2974–2982, 2018, doi: 10.1609/aaai.v32i1.11794.
[37] S. Mitchell, M. O'Sullivan, and I. Dunning, "PuLP: A Linear Programming Toolkit for Python," *Univ. Auckland, Auckland, New Zeal.*, p. 65, 2011, [Online]. Available: http://www.optimization-online.org/DB_FILE/2011/09/3178.pdf
[38] Y. Wang *et al.*, "Trajectory Design for UAV-Based Internet of Things Data Collection: A Deep Reinforcement Learning Approach," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3899–3912, 2022, doi: 10.1109/JIOT.2021.3102185.

Asif Mahmud Raivi received his B.S. in electrical, electronic, and communication engineering from Military Institute of Science and Technology, Dhaka, Bangladesh, in 2016. From 2017 to 2022, he joined the Bangladesh Computer Council (BCC) as an assistant network engineer and participated in IT infrastructure development projects. He is now pursuing graduate studies in computer engineering at Chosun University, South Korea. His research interests include unmanned aerial vehicle networks, mobile edge computing, artificial intelligence, machine learning, aerial data aggregation, and the Internet of things.

Sangman Moh received the 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, South Korea. From 2006 to 2007, he was on leave at Cleveland State University, USA. Until 2002, he had been with the Electronics and Telecommunications Research Institute (ETRI), South Korea, where he served as a project leader after receiving his M.S. degree in computer science from Yonsei University, South 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.