Adaptive Energy and Latency Optimization in Multi-Tier Networks via DDPG-Based Multi-Objective Control

Vitou That, Vanda Yorn
Department of Intelligent Energy Industry Convergence
Chung-Ang University
Seoul, South Korea
vitou1707@cau.ac.kr, vandayorn@cau.ac.kr

Jung-Ryun Lee *(Senior Member IEEE)*
School of Electrical and Electronics Engineering
Department of Intelligent Energy and Industry
Chung-Ang University
Seoul, South Korea
jrlee@cau.ac.kr

***Abstract*—With the increasing computational demands of Internet of Things (IoT) applications, air-ground integrated networks (AGIN), leveraging the capabilities of Unmanned Aerial Vehicles (UAVs) and High-Altitude Platform (HAP), provides an essential solution to these challenges. In this paper, we propose a framework that facilitates local computing at IoT devices and offers the flexibility to offload tasks to aerial platforms when necessary. Specifically, we formulate a multi-objective optimization model aiming at simultaneously minimizing energy consumption and reducing task latency by adjusting control variables such as transmit power, offloading decisions, and UAV placement in a distributed network of IoT devices. Our proposed framework employs Deep Deterministic Policy Gradient (DDPG) techniques to dynamically optimize network operations, allowing for efficient real-time adjustments to network conditions and task demands. The performance of the proposed algorithm is compared to traditional algorithms, including the Whale Optimization Algorithm (WOA), Gradient Search with Barrier, and Bayesian Optimization (BO). Simulation results show that this approach significantly minimizes energy consumption and latency, outperforming conventional optimization methods. Additionally, scalability tests confirm that our framework can efficiently integrate an increasing number of IoT devices and UAVs.**

***Index Terms*—Unmanned Aerial Vehicles, deep reinforcement learning, multi-tier networks, multi-objective functions, multi-parameters control.**

I. Introduction

In many disaster-affected areas, the absence of traditional communication infrastructure complicates the deployment of conventional emergency response strategies [1]. In these scenarios, the deployment of Internet of Things (IoT) technologies is invaluable due to their quick deployability across various environments. Nonetheless, these devices often face significant limitations in computational power and battery life, which can delay critical emergency responses [2].

This work was supported in part by the Ministry of Science and ICT (MSIT), South Korea, through the Information Technology Research Center (ITRC) Support Program, supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP), under Grant IITP-2020-2018-0-01799, and in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MEST) under Grant NRF-2020R1A2C1010929.

On the other hand, Unmanned Aerial Vehicles (UAVs) are recognized for their agility, low cost, and ease of deployment, making them suitable for on-demand mobile network applications. Moreover, the high-altitude and flexible positioning of the High-Altitude Platform (HAP) allows them to handle high data traffic and maintain continuous coverage [3]. By integrating UAVs and HAP, edge computing can be dynamically adjusted to meet user demands and optimize resource utilization efficiently. Additionally, incorporating Non-Orthogonal Multiple Access (NOMA) communication enhances uplink capabilities from IoT devices to both UAVs and HAP, increasing spectral efficiency [4].

In the realm of IoT, two critical performance metrics are energy consumption and task latency. Optimizing these metrics is important not only for enhancing the operational efficiency of IoT devices but also for extending their functional lifespan and reliability. Lower energy consumption leads to longer battery life while minimizing latency is equally important. Mobile Edge Computing (MEC) has emerged as a key solution to support such computation-intensive tasks. However, relying solely on terrestrial infrastructure may not always ensure robust performance [5]. To address these challenges, integrating HAP and UAVs as part of the MEC and IoT system is seen as a valuable approach to extending the capabilities of terrestrial networks.

To maximize the network's effectiveness, a well-coordinated and integrated design between ground and aerial networks is essential. This includes optimizing transmit power, offloading decisions, and UAV placements.

This paper addresses the challenges of optimizing IoT systems that incorporate both UAV-based and HAP-based edge computing networks, focusing specifically on minimizing energy consumption and task latency. The major contributions of our work are summarized as follows:

- We propose a joint optimization framework that simultaneously integrates multi-parameter such as UAV placement, mode selection, offloading ratio, and transmit power, specifically for multi-objective optimization in a multi-tier network. This framework is designed to address

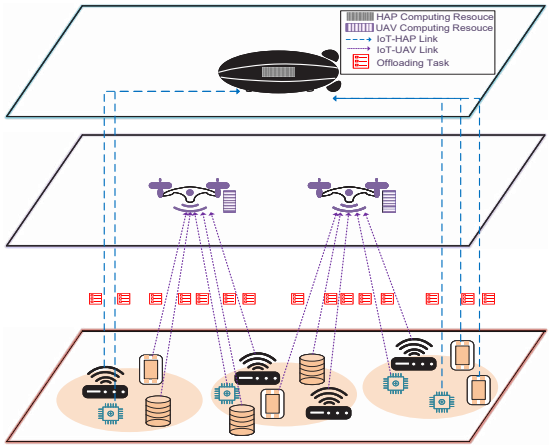

Figure 1. System Model

the dual objectives of minimizing energy consumption and latency through a weighted sum cost function while considering Quality of Service (QoS) constraints and the added complexity from real-time dynamics of wireless channel gains, interference.

- We employ a Deep Deterministic Policy Gradient (DDPG) algorithm to effectively solve our complex, multi-objective optimization problem. Our work combines the optimization of multi-parameters into a comprehensive framework, handling both continuous and discrete parameters. Recognizing the mixed integer and continuous nature of the optimization variables, we apply state-action normalization to effectively handle these integer and discrete decision variables that can refine the input and output of the DDPG algorithm.
- Through extensive simulations, our framework demonstrates high performance in reducing both energy consumption and latency, outperforming traditional optimization methods such as Whale Optimization Algorithm (WOA), Gradient Search with Barrier (GS), and Bayesian Optimization (BO). These results highlight the effectiveness of our approach in practical IoT scenarios, offering a significant improvement over existing methods.

II. System Model

As illustrated in Fig. 1, we consider an AGIN system that consists of a set of terrestrial IoT devices with fixed position, denoted by $\mathcal{I} = \{1, 2, ..., I\}$, a set of UAVs, denoted by $\mathcal{U} = \{1, 2, ..., U\}$, and a HAP $\mathcal{H} = \{0\}$. For simplicity, let $j \in J = \mathcal{U} \cup \mathcal{H}$ denote the set of edge server. In our model, both UAVs and the HAP serve as robust aerial base stations and are equipped with edge servers. We define the operational time frame of the system in discrete time slots, indexed by $\mathcal{T} = \{1, 2, \ldots, t\}$. In each time slot, each IoT device has computational tasks, expressed as $M[t] = \{m_1[t], m_2[t], ..., m_i[t]\}$, and they can be processed either locally or offloaded to the UAV or the HAP for remote computing. However, each device can select only one offloading option per time slot.

A. Communication Model

1) Channel Model: Let $g_{i,u}$ denote the channel gain from IoT device i to UAV u. We assume that the communication involves direct line-of-sight (LoS) link and the quality of the channel therefore depends on the distance between communication devices. The channel gain $g_{i,u}$ can be modeled using free-space path loss formula

$$g_{i,u} = \left(\frac{\lambda}{4\pi d_{i,u}}\right)^2, \tag{1}$$

where $d_{i,u}$ denotes the distance from IoT device i to UAV u and $\lambda = c/f_c$ is the wavelength with c, f_c are the speed of light and carrier frequency respectively.

Let $g_{i,h}$ denote the channel gain from IoT device i to HAP h. In emergency scenarios, such as in dense forests or conflict zones, the signal from the IoT devices to HAP encounters various forms of attenuation. The channel gain is given as

$$g_{i,h} = \left(\frac{\lambda}{4\pi d_{i,h}}\right)^2 \frac{1}{g^{\text{clutter}}} \frac{1}{g^{\text{atm}}}, \tag{2}$$

2) Signal Transmission: For network connectivity, each IoT device has several connectivity options to the edge servers, either through a HAP or one of multiple UAVs. These options are encoded as $j = \{0, 1, \ldots, u\}$, where $j = \{0\}$ links the data to the HAP and $j = \{1, \ldots, u\}$ represents the link to different UAV, indicated by a unique index. Without loss of generality, the channel gains of I devices are ordered as $|g_{1,j}| \leq |g_{2,j}| \leq \cdots \leq |g_{i,j}|$.

The effectiveness of signal transmission in this sytem is quantified by the signal-to-interference-plus-noise (SINR). For an IoT device i transmitting to edge server link j at time t, the SINR is obtained as

$$\text{SINR}_{i,j}[t] = \frac{p_i[t]\, g_{i,j}}{\sum_{v=i+1}^{I} p_v[t]\, g_{v,j} + \sigma^2}, \tag{3}$$

where p_i is the transmit power of device i and σ^2 is the noise power density.

The achieved data rate for an IoT i at edge server link j can then be calculated as follows:

$$R_{i,j}[t] = B_{i,j} log_2(1 + \text{SINR}_{i,j}[t]), \tag{4}$$

where $B_{i,j}$ represents the bandwidth allocated to the communication link between device i at edge server link j.

B. Energy Consumption Model

The selection of the edge server link j for transmitting data is dictated by $\psi_i = [\psi_{i,0}, \psi_{i,1}, \ldots, \psi_{i,j}]$, which specifies the mode selection for IoT device i selecting edge server link j. It is a binary indicator vector where each element corresponds to the edge server link, therefore ensuring only one connection at any time t. For example, $\psi_i(t) = [0, 1, 0, \ldots, 0]$ indicates the selected edge server link $j = \{1\}$.

The energy consumption for transmitting data by device i at time t is calculated as follows [2]

$$E_{i,j}[t] = \psi_i[t] p_i[t] \left(\frac{\lambda_i[t] m_i[t]}{R_{i,j}[t]} \right), \tag{5}$$

where λ_i is the computational task ratio that IoT device i offloads to edge server j at time t.

The processing speed of the device is denoted as ω^i, which is measured in CPU cycles per second. The variable ϕ^i represents the number of CPU cycles required to process one bit of data. The data processing rate of an IoT device as $\chi^i = \frac{\omega^i}{\phi^i}$. The power consumed per CPU cycle is presented by $\kappa(\omega^i)^2$, where κ is the constant coefficient represents as the energy efficiency of computing. The energy required for local processing is expressed as

$$E_{i,loc}\left[t\right] = \kappa(\omega^i)^3 \left(\frac{(1-\lambda_i\left[t\right]) m_i\left[t\right]}{\chi^i} \right). \tag{6}$$

The total energy consumption for an IoT device at given time t is the sum of the energy used for local processing and transmission, which is given by

$$E_i\left[t\right] = E_{i,loc}\left[t\right] + E_{i,j}\left[t\right]. \tag{7}$$

C. Computation Model

The model specifically considers the latency of computing on the device, data transmission latency, computational latency at the edge server, and transmission latency for feedback. Compared to computation latency and data offloading latency, the transmission latency for feedback is relatively minor and can be disregarded [6].

1) Local Computing Latency: Local computing on an IoT device involves processing a portion of the task after offloading part of it to the edge. The latency taken for the process, during a specific interval t, is defined by

$$T_{i,loc}\left[t\right] = \frac{\left(1-\lambda_i\left[t\right]\right) m_i\left[t\right]}{\chi^i/\phi^i} = \frac{\phi^i\left(1-\lambda_i\left[t\right]\right) m_i\left[t\right]}{\chi^i}. \tag{8}$$

2) Edge Computing Latency: The task, not processed locally, is offloaded to an edge server (UAV or HAP), causing transmission latency. The transmission latency for offloading data from an IoT device i to edge server link j is written by

$$T_{i,j,trans}\left[t\right] = \psi_i[t] \frac{\lambda_i\left[t\right] m_i\left[t\right]}{R_{i,j}\left[t\right]}. \tag{9}$$

We consider that offload tasks are processed simultaneously. The delay for each task is determined by the proportion of computation resources allocated to it. Let $\Omega^j = \frac{c^j}{\rho^j}$ represent the total processing capacity of edge server link j. The processing capacity is allocated fairly based on the size of the task, which can be simplified as

$$k^{i,j}\left[t\right] = \Omega^j \frac{\lambda_i\left[t\right] m_i\left[t\right]}{\sum_{v=1}^{I} \psi_v\left[t\right] \lambda_v\left[t\right] m_v\left[t\right]}. \tag{10}$$

Therefore, the latency for each individual task computation at edge server link j is expressed as

$$T_{i,j,compute}\left[t\right] = \psi_i\left[t\right] \frac{\lambda_i\left[t\right] m_i\left[t\right]}{k^{i,j}\left[t\right]}. \tag{11}$$

The total latency for each task at edge server link j is thus the sum of transmission and computation latency, given as

$$T_{i,j}\left[t\right] = T_{i,j,trans}\left[t\right] + T_{i,j,compute}\left[t\right]. \tag{12}$$

Task latency considers the longest latency experienced by each task, whether it is processed locally or at the edge.

$$T_i\left[t\right] = \max(T_{i,loc}\left[t\right], T_{i,j}\left[t\right]). \tag{13}$$

III. Problem Formulation

The primary aim of this work is to minimize total energy consumption and total task latency simultaneously in AGIN. The objective function is expressed in the following

$$\mathcal{P}: \min_{p_i,\lambda_i,\psi_i,x_u,y_u} \sum_{t=1}^{\mathcal{T}} \sum_{i=1}^{I} E_i[t] \quad \text{and} \quad \min_{p_i,\lambda_i,\psi_i,x_u,y_u} \sum_{t=1}^{\mathcal{T}} \sum_{i=1}^{I} T_i[t]. \tag{14}$$

By formulating the problem as a multi-objective optimization, we highlight our approach of using a weighted sum method to effectively balance and optimize these two critical performance metrics [7]. The optimization problem $\mathcal{P}$ in our work is transformed into a single objective optimization problem $\mathcal{P}0$ [8] as follows.

$$\mathcal{P}0: \min_{p_i,\lambda_i,\psi_i,x_u,y_u} \sum_{t=1}^{\mathcal{T}} \sum_{i=1}^{I} \left(\alpha_e E_i[t] + \alpha_t T_i[t]\right) \tag{15a}$$

$$\text{subject to } \mathrm{C1}: 0 \le \lambda_i \le 1 \tag{15b}$$

$$\mathrm{C2}: p_i^{\min} \le p_i \le p_i^{\max} \tag{15c}$$

$$\mathrm{C3}: \psi_{i,j} \in \{0,1\} \tag{15d}$$

$$\mathrm{C4}: \sum_{j=1}^{u} \psi_{i,j} = 1 \tag{15e}$$

$$\mathrm{C5}: x_u^{\min} \le x_u \le x_u^{\max} \tag{15f}$$

$$\mathrm{C6}: y_u^{\min} \le y_u \le y_u^{\max} \tag{15g}$$

$$\mathrm{C7}: E_i \le E_i^{\max} \tag{15h}$$

$$\mathrm{C8}: T_i \le T_i^{\max} \tag{15i}$$

where $\alpha_e = \frac{1}{E_i^{\max}}$ and $\alpha_t = \frac{1}{T_i^{\max}}$ are the normalization factors to balance energy consumption and task latency on a comparable scale [7]. Constraint C1 ensures that the offloading ratio for each IoT device remains within a practical range from 0 to 1. Constraint C2 governs the transmit power of each device that lies within a operational range specified by minimum and maximum limits. Constraints C3 and C4 indicate that each IoT device operates with a single edge server link at any given time t. Constraints C5 and C6 dictate the allowable positions of UAVs along the x and y axes, respectively, ensuring UAVs are positioned within strategic locations to maximize coverage and connectivity. Constraint C7 sets an upper limit on the energy consumption for each

Algorithm 1 DDPG-Based Optimization Algorithm

input : discount factor γ, soft update rate τ, learning rates for actor $\alpha_{\mathcal{A}}$ and critic $\alpha_{\mathcal{C}}$
output : Optimized policy parameters $\theta^{\mathcal{A}*}$, $\theta^{\mathcal{C}*}$

1: Randomly initialize actor network weights $\theta^{\mathcal{A}}$ and critic network weights $\theta^{\mathcal{C}}$
2: Initialize target networks weights $\theta^{\bar{\mathcal{A}}} \leftarrow \theta^{\mathcal{A}}, \theta^{\bar{\mathcal{C}}} \leftarrow \theta^{\mathcal{C}}$
3: Initialize replay buffer memory B, batch size b, number of episodes K and timesteps T
4: **for** $k = 1 : K$ **do**
5: Initialize OU-noise N
6: Get initial observation state $s[0]$
7: **for** $t = 1 : T$ **do**
8: Select $a[t] = \mu^{\theta^{\mathcal{A}}}(s[t])$
9: Normalize $a[t]$ to $\bar{a}[t]$, execute action $\bar{a}[t]$
10: Observe reward $R[t]$ and new state $s[t+1]$
11: **store** $(s[t], a[t], R[t], s[t+1])$ in B
12: **if** $B_{\text{size}} \geq b_{\text{size}}$ **then**
13: **sample** mini-batch $b = \{(s[t], a[t], R[t], s[t+1])\}$ from B
14: **calculate** y^{target} by equation (16)
15: **calculate** $\nabla_{\theta^{\mathcal{C}}} L\left(\theta^{\mathcal{C}}\right)$ by equation (18)
16: **update** $\theta^{\mathcal{C}}$ by equation (17)
17: **calculate** $\nabla_{\theta^{\mathcal{A}}} J$ by equation (20)
18: **update** $\theta^{\mathcal{A}}$ by equation (19)
19: **update** $\theta^{\bar{\mathcal{A}}}$, $\theta^{\bar{\mathcal{C}}}$ by equations (21), (22) respectively

task. Lastly, constraint C8 addresses task latency, ensuring that processing times remain within acceptable thresholds.

IV. Proposed Algorithms

Our optimization problem $\mathcal{P}0$, is an integer nonlinear optimization problem. In this scenario, traditional optimization methods struggle to achieve ideal outcomes due to the complexity introduced by the problem's nonlinearity and the mixed nature of its variables. Additionally, the dynamic and unpredictable nature of the environment significantly impacts the performance metrics of our system, such as channel gains.

A. DDPG Algorithm

To address these complexities, we employ the DDPG algorithm. As detailed in Algorithm 1, the DDPG algorithm starts by setting up the actor network with weights denoted as $\theta^{\mathcal{A}}$ and the critic network with weights $\theta^{\mathcal{C}}$. The target networks, represented as $\theta^{\bar{\mathcal{A}}}$ and $\theta^{\bar{\mathcal{C}}}$, are created with the same weights as their corresponding actor and critic networks. A replay buffer B is used to store transitions collected during training, including states, actions, rewards, and subsequent states in each entry. This approach allows algorithm to learn from past experiences.

Each training episode employs an Ornstein-Uhlenbeck (OU) noise process for action exploration to mitigate the risk of local minima and balance the exploration of new strategies against the exploitation of known ones. From line 8 to 11, the actions, derived from the actor network and normalized, are executed in the environment to obtain rewards and new states which are stored in B. Once B contains sufficient data, a mini-batch b is sampled for network updates. Target values y^{target} is computed as

$$y^{\text{target}} = R[t] + \gamma \mathbb{Q}^{\theta^{\bar{\mathcal{C}}}}\left(\mathrm{s[t+1]}, \mu^{\theta^{\bar{\mathcal{A}}}}(\mathrm{s[t]})\right), \tag{16}$$

using the rewards and the discounted Q-values from the target critic. The update of the critic network's weights is guided in a way to minimize the loss and is expressed as

$$\theta^{\mathcal{C}} \leftarrow \theta^{\mathcal{C}} - \alpha_{\mathcal{C}} \nabla_{\theta^{\mathcal{C}}} L(\theta^{\mathcal{C}}), \tag{17}$$

where the $\nabla_{\theta^{\mathcal{C}}} L(\theta^{\mathcal{C}})$ is gradient loss function of $\theta^{\mathcal{C}}$ and is formed as

$$\nabla_{\theta^{\mathcal{C}}} L\left(\theta^{\mathcal{C}}\right) = \nabla_{\theta^{\mathcal{C}}} \mathbb{E}\left[\left(\mathrm{y}^{\text{target}} - \mathbb{Q}^{\theta^{\mathcal{C}}}(\mathrm{s[t]}, \mathrm{a[t]})\right)^2\right]. \tag{18}$$

Concurrently, the actor network's parameters are refined via a policy gradient method to maximize expected rewards

$$\theta^{\mathcal{A}} \leftarrow \theta^{\mathcal{A}} + \alpha_{\mathcal{A}} \nabla_{\theta^{\mathcal{A}}} J, \tag{19}$$

where the policy gradient for the actor network is approximated as

$$\nabla_{\theta^{\mathcal{A}}} J \approx \mathbb{E}_{(s,a)\sim B}\left[\nabla_a Q^{\theta^{\mathcal{C}}}(s, \mu^{\theta^{\mathcal{A}}}(s)) \cdot \nabla_{\theta^{\mathcal{A}}} \mu^{\theta^{\mathcal{A}}}(s)\right], \tag{20}$$

while both target networks softly are updated to integrate changes gradually by

$$\theta^{\bar{\mathcal{A}}} \leftarrow \tau\theta^{\mathcal{A}} + (1-\tau)\theta^{\bar{\mathcal{A}}} \tag{21}$$

$$\theta^{\bar{\mathcal{C}}} \leftarrow \tau\theta^{\mathcal{C}} + (1-\tau)\theta^{\bar{\mathcal{C}}}. \tag{22}$$

To effectively apply the DDPG algorithm for handling continuous action spaces and dynamic environments, we first transform our multi-objective optimization problem into a Markov Decision Process (MDP). In this section, we will outline the process of defining the states, actions, and reward function of our MDP.

MDP Transformation

We denote $\mathbf{S}$ and $\mathbf{A}$ as sets of states and actions, respectively. In the DDPG algorithm, an action is selected from the given state space according to a policy $\pi : \mathbf{S} \rightarrow \mathbf{A}$, which maps states to their corresponding actions.

Environment State: At each time slot t, the environment state $s_i[t] \in \mathbf{S}$ of IoT device i includes all the necessary information for decision-making. Specifically, the state consists of the current task size $m_i[t]$, the fixed positional coordinates the IoT device $[x_i, y_i]$, and the dynamic channel state matrix from IoT device i to edge server j, $[g_{i,0}[t], g_{i,1}[t], \ldots, g_{i,u}[t]]$, which can be defined as:

$$s_i[t] = \{m_i[t], [x_i, y_i], [g_{i,0}[t], g_{i,1}[t], \ldots, g_{i,u}[t]]\}. \tag{23}$$

Therefore, the current state of the AGIN at each time slot t is expressed as

$$S[t] = \{s_1[t], s_2[t], \ldots, s_i[t]\}. \tag{24}$$

Action: In our MDP model, the action $a[t]$ at each time slot t represents the set of operational decisions that respond to the current state $s[t]$.

For each IoT device, the action $a_i[t]$ includes transmit power, offloading ratio, and mode selection. The action is denoted as

$$a_i\left[t\right] = \left\{p_i\left[t\right], \lambda_i\left[t\right], \psi_i\left[t\right]\right\} \tag{25}$$

Similarly, for UAVs, the action $a_u[t]$ includes the coordinates of UAV u:

$$a_u\left[t\right] = \left\{x_u\left[t\right], y_u\left[t\right]\right\}. \tag{26}$$

As shown from (25) to (26), the complete action combines the actions for IoT device i and actions for UAV u, written as:

$$A\left[t\right] = \left\{\left\{a_i\left[t\right] | i \in \mathcal{I}\right\} \cup \left\{a_u\left[t\right] | u \in \mathcal{U}\right\}\right\}. \tag{27}$$

Reward: In our optimization problem, the objective function aims to minimize the total energy consumption and task latency. To adapt this with the DDPG framework, which is designed to maximize cumulative rewards, we define our reward as the negative of the weighted sum of energy consumption $E_i\left[t\right]$ and task latency $T_i\left[t\right]$. This is expressed as follows:

$$R\left[t\right] = -\sum_{i=1}^{I}\left(\alpha_e E_i\left[t\right] + \alpha_t T_i\left[t\right] + \xi\right), \tag{28}$$

where ξ is penalty rewards incurred from violating constraints, ensuring the solution remains feasible within the defined limits.

V. Simulation and Results

A. Simulation Settings

In our simulation, IoT devices are randomly deployed across a network area of 1000 m $\times$ 1000 m. The UAVs operate at a height of 300 m, while the HAP is positioned in the center of network area with a height of 20000 m. Task sizes assigned to devices are uniformly distributed, ranging between $[1.0 - 2.0]$ MB. The bandwidth is set to 10 MHz with a noise spectrum density of -174 dBm/Hz. Table I provides the other simulation parameters used in our simulation runs.

The DDPG algorithm used in our work is configured with 4 fully connected layers including an input layer, an output layer, and 2 hidden layers. The hidden layers consist of 400, and 300 neurons, respectively. The Tanh activation function is used to normalize the output between -1 and 1. The training process involves over 1000 episodes, each consisting of 500 steps.

B. Results

Table II shows the relationship between energy consumption and task latency across different strategies in a network with 30 IoT devices. Our proposed method, combining local processing with offloading to UAVs and HAP, is compared against scenarios without these options and models with full or no offloading. The results indicate that UAVs help reduce energy consumption due to their proximity and efficiency, while the HAP is essential for minimizing task latency. Full offloading to either UAVs or HAP leads to the highest energy consumption and latency.

Table I
Simulation Parameters

Parameter	Value
Speed of Light (m/s)	3×10^8
Carrier Frequency (GHz)	2.4
Transmit Power (dBm)	$20 - 26$
IoT Constant Coefficient	1×10^{-27}
IoT CPU Cycles(cycles/bit)	500
IoT Computing Speed (GHz)	0.5
UAV CPU Cycles(cycles/bit)	270
UAV Computing Speed (GHz)	1
HAP CPU Cycles(cycles/bit)	1100
HAP Computing Speed (GHz)	40
UAV X-coordinate Range	$0 - 1000$
UAV Y-coordinate Range	$0 - 1000$
Maximum Task Latency (s)	2
Buffer Size	1×10^5
Batch Size	300
Discount Factor	0.99
Soft Update Rate	0.1
Actor Learning Rate	1×10^{-3}
Critic Learning Rate	1×10^{-2}

Table II
Evaluation of Optimization Method on Energy Consumption and Task Latency

Method	Task Latency (s)	Energy Consumption (J)
Proposed Method	0.950	0.152
Without UAV	1.20	0.225
Without HAP	1.54	0.195
Full Offloading	1.74	0.241
Without Offloading	1.70	0.212

Fig. 2 compares the performance of four optimization techniques: DDPG, WOA, BO, and GS applied in a scenario of 2 UAVs and 1 HAP, differentiated by the number of IoT devices ranging from 10 to 30. Fig. 2(a) highlights the objective values under each algorithms, derived from the final converged value of each algorithm. Fig. 2(b) and (c) illustrate the energy consumption and task latency, respectively. It can be seen that as the number of device increases, both energy consumption and task latency also increase for each method.

Fig. 3 shows the comparative performance of the four algorithms with different numbers of UAVs: 2, 3, and 4, in a scenario involving 30 IoT devices and 1 HAP. Fig. 3(a) reveal the objectives value achieved by each method, demonstrating how each algorithms improves as the number of UAVs increases, while Figure 5(b) and (c) further explore the energy consumption and task latency, respectively. It can be seen that despite the growing number of UAVs, DDPG maintains lower energy consumption and reduced task latency compared to other methods.

Fig. 4 shows the strategic deployment of UAVs and system associated in a 2D plane based on our proposed algorithm. The figure captures the dynamic assignment of IoT devices to either UAVs or the HAP depending on their geographical distribution.

VI. Conclusion

This paper presented a comprehensive study on optimizing energy consumption and latency in an air-ground integrated

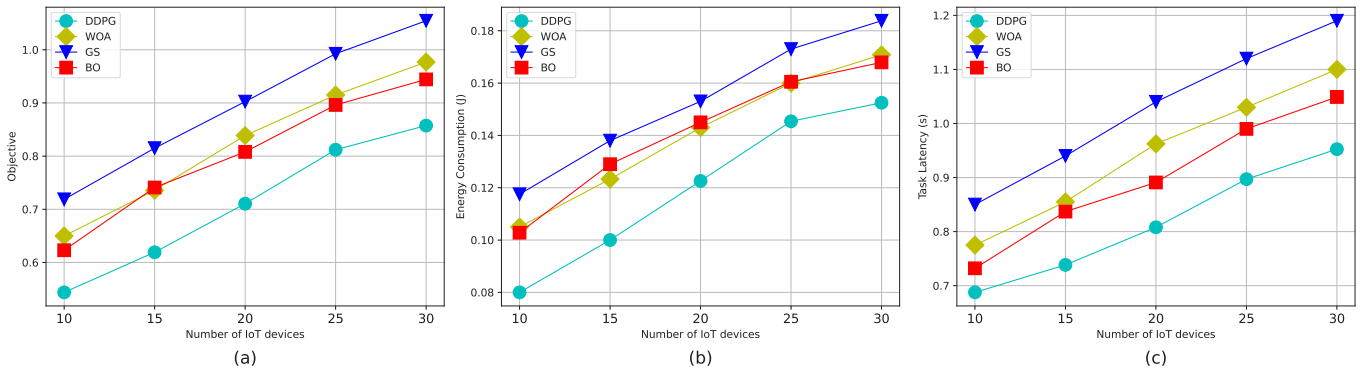

Figure 2. Comparative performance of IoT devices. (a) Objective optimization. (b) IoT energy computation. (c) Task latency.

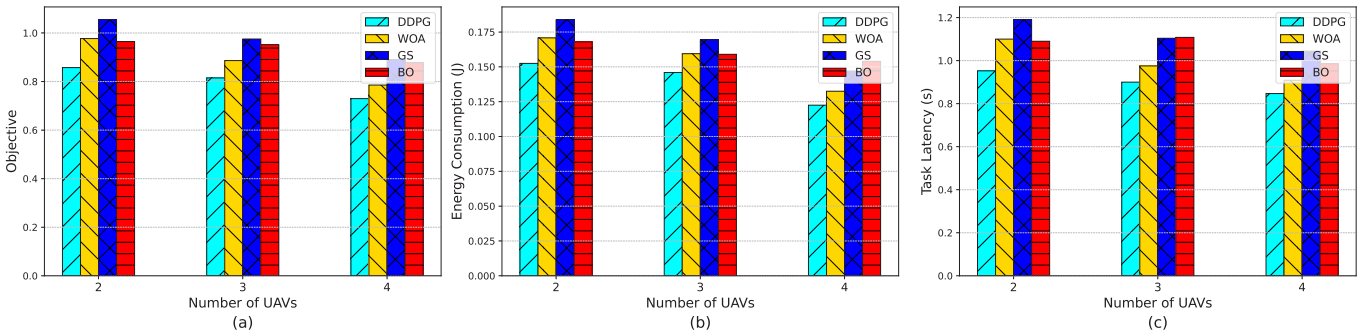

Figure 3. Comparative performance of UAVs. (a) Objective optimization. (b) IoT energy computation. (c) Task latency.

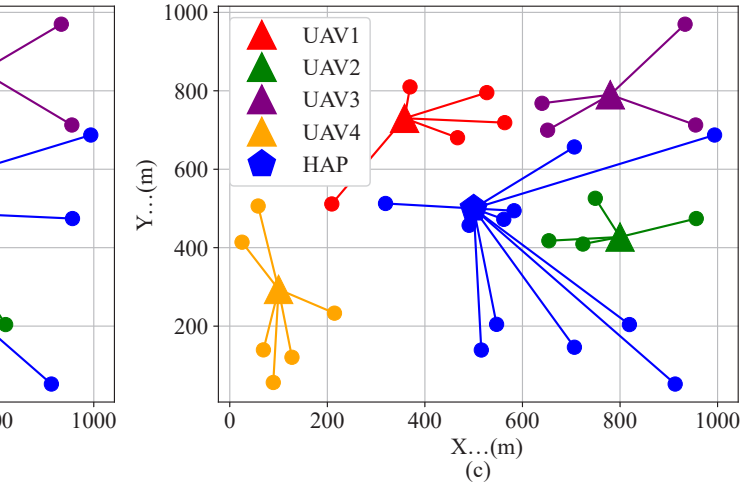

Figure 4. Strategic Deployment and Connectivity Patterns of UAVs and HAP in an IoT Network

network by using a DDPG-based algorithm. We proposed a novel framework that efficiently manages UAV placement, task offloading, and power optimization within a multi-UAV and single HAP environment. Through the simulation results, it can be seen that integrating UAVs and a HAP allows for dynamic resource utilization, which is crucial for maintaining robust network performance under varying conditions.

For future work, we plan to apply federated learning to our proposed framework. Additionally, we will investigate the integration of multi-agent deep reinforcement learning to further enhance system responsiveness and efficiency.

References

[1] 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 of Things Journal*, 2023.

[2] Z. Jia, Q. Wu, C. Dong, C. Yuen, and Z. Han, "Hierarchical aerial computing for internet of things via cooperation of HAPs and UAVs," *IEEE Internet of Things Journal*, vol. 10, no. 7, pp. 5676–5688, 2022, publisher: IEEE.

[3] K. An, Y. Sun, Z. Lin, Y. Zhu, W. Ni, N. Al-Dhahir, K.-K. Wong, and D. Niyato, "Exploiting multi-layer refracting ris-assisted receiver for hap-swipt networks," *IEEE Transactions on Wireless Communications*, 2024.

[4] F. Fang, Y. Xu, Z. Ding, C. Shen, M. Peng, and G. K. Karagiannidis, "Optimal resource allocation for delay minimization in NOMA-MEC networks," *IEEE Transactions on Communications*, vol. 68, no. 12, pp. 7867–7881, 2020, publisher: IEEE.

[5] C. Zhou, W. Wu, H. He, P. Yang, F. Lyu, N. Cheng, and X. Shen, "Deep reinforcement learning for delay-oriented iot task scheduling in sagin," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 911–925, 2020.

[6] Z. Ding, J. Xu, O. A. Dobre, and H. V. Poor, "Joint power and time allocation for noma–mec offloading," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 6207–6211, 2019.

[7] H. Wu, H. Lu, F. Wu, and C. W. Chen, "Energy and delay optimization for cache-enabled dense small cell networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7663–7678, 2020.

[8] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and multidisciplinary optimization*, vol. 26, pp. 369–395, 2004.