Intelligent Heterogeneous Aerial Edge Computing for Advanced 5G Access

Tri-Hai Nguyen, Thanh Phung Truong, Anh-Tien Tran, Nhu-Ngoc Dao, Laihyuk Park, and Sungrae Cho

***Abstract*—In the context of the Internet of Things (IoT), aerial computing platforms (ACPs) such as unmanned aerial vehicles and high-altitude platforms with edge computing capabilities have the potential to significantly expand coverage, enhance performance, and handle complex computational tasks for IoT devices (IoTDs). Non-orthogonal multiple access (NOMA) has also emerged as a promising multiple access technology for advanced 5G networks. This paper presents a multi-ACP-enabled NOMA edge network, which enables heterogeneous ACPs to provide computational assistance to IoTDs. To minimize delay and energy consumption, we formulate a joint task offloading and resource allocation problem that considers IoTD association, offloading ratio, transmit power, and computational resource allocation variables. To address the complexity of the optimization problem, it is modeled as a multi-agent Markov decision process and solved using a multi-agent deep deterministic policy gradient (MADDPG)-based solution. Extensive simulation results demonstrate that the proposed MADDPG-based framework can remarkably adapt to the dynamic nature of multi-ACP-enabled NOMA edge networks. It consistently outperforms various benchmark schemes regarding energy efficiency and task processing delay across different simulated scenarios.**

***Index Terms*—aerial computing platform, multi-agent deep deterministic policy gradient, non-orthogonal multiple access, resource allocation, task offloading.**

I. Introduction

As the world prepares for advanced 5G wireless communications, expectations are high for capabilities that go further mobile Internet to support advanced Internet of Things (IoT) applications with diverse resource needs and quality of service (QoS) requirements [1]. By offloading computation-intensive tasks to nearby computing servers with multi-access edge computing (MEC) technology, resource-constrained IoT devices (IoTDs) can perform more complex tasks or conserve energy [2], [3]. However, the sole reliance on terrestrial network infrastructure limits the achievement of truly ubiquitous global connectivity. Economic disparities and environmental factors significantly limit the feasibility of sufficient network coverage in remote areas such as deserts, forests, and oceans. Furthermore, terrestrial infrastructure is inherently vulnerable to natural disasters (e.g., earthquakes, floods, and bushfires) and anthropogenic disruptions (e.g., power failures, theft, and sabotage). To address these challenges, aerial access networks leveraging platforms such as unmanned aerial vehicles (UAVs), high-altitude platforms (HAPs), and satellites have emerged as a captivating prospect for augmenting 5G and beyond networks [4], [5]. These aerial platforms offer the capability to provide critical Internet services in regions where terrestrial network coverage remains economically or geographically impractical. Notably, the 3rd Generation Partnership Project (3GPP) is actively exploring the integration of UAVs, HAPs, and satellite access into its standards for advanced 5G access [1], [6], [7]. Utilizing HAPs and UAVs equipped with computational capabilities as aerial computing platforms (ACPs), aerial edge computing can offer seamless communication and computing experiences for IoTDs through the advantages of line-of-sight (LoS) communication and flexible deployment [8], [9]. In particular, HAPs, stationed high in the stratosphere, provide wide-area communication and monitoring. Powered by sustainable energy, HAPs can remain aloft for extended periods. Although UAVs have limited resources and flight time, they can operate at lower altitudes across varied terrains, offering lower latency and deployment costs than HAPs. These ACPs can be applied in diverse use cases and scenarios [8], [9]. For example, in aerial surveillance [10], UAVs perform face identification and offload heavy computations to nearby edge servers, overcoming their limited energy resources. Similarly, UAVs can assist marine operations by hovering as aerial edge servers for unmanned surface vehicles collecting underwater sensor data [11]. HAPs offer high-capacity backhaul links for a wider reach, while UAVs extend connectivity directly to ground users through localized access points [12]. The heterogeneous multi-ACP network leverages the strengths of both HAPs and UAVs, providing access and computational services for remote IoTDs or establishing temporary communications for disaster relief efforts [13]–[15].

A. Motivation

While aerial access infrastructure offers immense potential for expanding connectivity, it also introduces challenges in upcoming 3GPP standard releases [4], [5]. Non-orthogonal multiple access (NOMA) has emerged as the leading access technology for 5G and beyond, garnering substantial attention

This study was supported by the Research Program funded by the SeoulTech (Seoul National University of Science and Technology). Tri-Hai Nguyen and Thanh Phung Truong equally contributed to this study. *(Corresponding authors: Laihyuk Park; Sungrae Cho.)*

Tri-Hai Nguyen and Laihyuk Park are with the Department of Computer Science and Engineering, Seoul National University of Science and Technology, Seoul 01811, Republic of Korea (e-mail: haint93@seoultech.ac.kr; lhpark@seoultech.ac.kr).

Thanh Phung Truong, Anh-Tien Tran, and Sungrae Cho are with the School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, Republic of Korea (e-mail: tptruong@uclab.re.kr; attran@uclab.re.kr; srcho@cau.ac.kr).

Nhu-Ngoc Dao is with the Department of Computer Science and Engineering, Sejong University, Seoul 05006, Republic of Korea (e-mail: nndao@sejong.ac.kr).

in academia and industry [16], [17]. Integrating NOMA into aerial networks with multiple ACPs is expected to provide sufficient resources and satisfy the QoS requirements of IoTDs in different scenarios. To this end, task offloading and resource allocation strategies have to be carefully designed to exploit the spectral and computational resources made available by NOMA and aerial networks effectively while addressing challenges such as user scheduling, latency, and energy efficiency optimization. Furthermore, the inherent dynamics of aerial networks pose difficulties for traditional optimization algorithms, restricting their ability to make real-time decisions in response to rapidly changing environments. Deep reinforcement learning (DRL) has emerged as a suitable technique for addressing complex decision-making problems in dynamic, multi-agent wireless networks [14], [15], [18]–[21]. While Deep Q-network (DQN) [22] remains a foundational DRL algorithm, its discrete optimization capabilities limit its suitability. Deep deterministic policy gradient (DDPG) [23], a policy-based DRL adept at high-dimensional continuous optimization, aligns well with continuous resource allocation in wireless networks. Notably, multi-agent DDPG (MADDPG) [24] extends DDPG to the multi-agent setting through centralized training and decentralized execution, making it particularly suitable for complex aerial network optimization challenges. Motivated by these factors, this paper explores a NOMA-based multi-ACP-enabled edge network and proposes a MADDPG-based solution for jointly optimizing task offloading and resource allocation.

B. Contributions

To the best of our knowledge, the existing literature lacks a comprehensive investigation into the joint task offloading and resource allocation (JTORA) problems in multi-ACP-enabled NOMA edge networks, which serve as the focus of this paper. We propose a multi-agent DRL-based framework employing an enhanced MADDPG, where HAPs and UAVs function as cooperative agents to tackle the JTORA problem. The main contributions of this paper are as follows.

- To meet the growing demands of IoTDs for computation offloading across challenging terrains, we propose a novel heterogeneous multi-ACP-enabled NOMA edge architecture that combines the broad coverage of a HAP with the flexibility and localized coverage of multiple UAVs. In the system, NOMA facilitates efficient spectrum utilization, while the partial offloading scheme allows for concurrent processing of the computational task on both the IoTD and its associated ACP, aiming to minimize the IoTD's computing delay and energy consumption. To achieve this objective, we formulate a JTORA problem that jointly optimizes IoTD association, task offloading ratio, transmission power, and computing resource allocation.
- Due to the complexity of the JTORA problem, we adopt a multi-agent Markov decision process (MAMDP) model and propose a MADDPG-based framework to learn optimal computation offloading and resource allocation policies in the system. In addition, we propose an action pretreatment function, ensuring that the generated actions comply with system constraints and correlations between optimization variables. This enables the agents to learn policies within the specified limits, preventing penalties or undesirable outcomes. The proposed framework, characterized by centralized training and decentralized execution, fosters efficient cooperation among multiple ACPs to maximize the long-term expected reward and optimize the computing overhead.
- To evaluate the effectiveness of the proposed method, we conducted comprehensive simulations. The results demonstrate the superiority of our approach in terms of better convergence and improved performance compared to alternative strategies such as single-agent DDPG, greedy offloading, full offloading, and randomized offloading across diverse scenarios.

The remainder of the paper is organized as follows. A literature review is conducted in Section II. In Section III, we introduce the system model and formulate the JTORA problem. Section IV proposes a multi-agent DRL-based framework to address the formulated optimization problem. In Section V, we present simulation results to evaluate the performance of the proposed framework. Finally, conclusions and future research directions are shown in Section VI.

II. Related Work

Task offloading and resource allocation are vital factors that significantly influence the overall performance of aerial edge computing systems [8], [9]. This section surveys recent, relevant studies in network modeling with UAV and HAP-assisted computation, leveraging DRL-based optimization techniques.

With high mobility and flexibility, UAVs can serve as MEC servers for aiding IoT computing offloading in a space-air-ground integrated network as in [25]. Due to the complexity and dynamics of the system, a policy gradient approach is used to optimize the offloading policies in the large action space, while an actor-critic approach accelerates the learning process. However, the resource allocation issue is not jointly considered, while the single-agent DRL approach cannot capture the interactions between multiple UAVs in the network. Some studies investigated multi-UAV-assisted MEC systems, in which UAVs cooperate to find optimized policies for computation offloading and resource sharing through multi-agent DRL frameworks [19], [20]. In [19], MEC servers deployed at the base station and UAVs collaborate in resource allocation and computation offloading for vehicles using the MADDPG algorithm. However, the binary offloading scheme employed in the computation model lacks flexibility. In [20], MEC-UAVs manage tasks offloaded from ground devices and redirect them to a nearby base station if they cannot handle them. However, the approach lacks consideration for direct offloading from the device to the base station. If a UAV malfunctions, the associated task becomes unprocessed. Furthermore, the assumption of equal allocation of computational resources from the MEC server to each served device is impractical, as each device exhibits varying computational demands that require optimal allocation.

Compared to UAVs, HAPs offer greater computing capabilities and broader network coverage due to their higher flight altitude and larger payload capacity. HAPs are employed to assist vehicular users in computing offloading and caching, aiming to reduce delay and energy consumption using DRL techniques [26]–[29]. In [26], a DQN-based value decomposition networks approach was proposed to optimize caching and computation offloading decisions in a three-layer HAP-road side unit-vehicle network. While the objective is to minimize system delay, it does not consider energy consumption. In [27], vehicles can offload their tasks to HAPs or neighboring vehicles. A double DQN (DDQN) approach, which can overcome the overestimation problem found in DQN, is used to minimize the total computation and communication overhead. In both studies [26], [27], the binary offloading scheme, in which computations are either processed locally or entirely offloaded, restricts the system's flexibility and might not be suitable for scenarios with diverse computational demands. In addition, DQN and DDQN can only apply for discrete optimization. Consequently, the action space needs to be discretized, which reduces the possibility of reaching the optimal solution. As demonstrated in [28], [29], DDPG effectively tackles continuous optimization, proving valuable for HAP-assisted edge computing systems. However, the previous studies [26]–[29] primarily focused on HAP-centric architectures without integrating multi-UAV-enabled edge computing. In [30], HAP serves as a centralized computing platform while UAVs act as aerial users collecting ground data, with the task offloading problem addressed using the DDPG approach. In [21], the authors focused on task and energy offloading in a UAV-based aerial network with a solution based on a multi-agent soft actor-critic model. HAP is responsible for energy sources of UAVs and fog nodes. Similar to [30], UAVs act as task owners, but instead of offloading tasks to HAP, they send tasks to on-ground fog computing nodes. The role of HAP is limited to the centralized training process of multi-agent DRL, and it does not provide direct computing assistance to IoTDs or UAVs. The works [21], [30] demonstrated the potential of HAPs in aerial MEC networks. Still, they also reveal a gap in fully utilizing the complementary strengths of HAPs and UAVs in a collaborative computing framework.

Recent studies have shifted attention towards the cooperation between UAVs and HAPs, recognizing their potential for a more comprehensive and effective computational solution. Task offloading and resource allocation problems in aerial edge computing systems involving a HAP and several UAVs were explored [13]–[15]. In [13], IoTDs offload tasks to UAVs, and if UAVs cannot handle the tasks, they forward them to a HAP. Similar to [19], [26], [27], the work limits the computing model to binary offloading, restricting its flexibility. In [14], UAVs collect tasks from IoTDs and offload a portion to a HAP. A multi-agent proximal policy optimization approach addresses the task offloading problem. However, only UAVs are considered learning agents, while the HAP is not included despite its involvement in the computation model. Due to the hierarchical design in both studies [13], [14], where a HAP can only receive tasks from IoTDs via UAVs, the IoTDs outside UAV coverage cannot be served and the tasks cannot be offloaded in case of UAV failure. In [15], a heterogeneous HAP-UAV computing system was proposed to enhance QoS for IoTDs using the MADDPG algorithm. However, the assumption of equal computing resources for all IoTDs is unrealistic for systems with varying QoS and computing demands.

In previous studies, communication models did not incorporate advanced multiple access techniques such as NOMA, which could enhance system spectrum efficiency. Additionally, computation models assumed equal resource allocation to all IoTDs and relied on a binary offloading scheme, limiting their effectiveness. This motivates the exploration of integrating NOMA into HAP-UAV collaborative computing frameworks with an intelligent task offloading and resource allocation scheme.

TABLE I
KEY NOTATIONS.

Notation	Description
$n, N, \boldsymbol{\mathcal{N}}$	The index, the number, and the set of ACPs ($n = N$ indicates the HAP, otherwise, the UAV)
$k, K, \boldsymbol{\mathcal{K}}$	The index, the number, and the set of IoTDs
$K_n, \boldsymbol{\mathcal{K}}_n$	The number and the set of IoTDs under the coverage of ACP n
$t, T, \boldsymbol{\mathcal{T}}$	The index, the number, and the set of time slots
$\tau_k = (s_k, w_k, d_k)$	Task of IoTD k with task size s_k, required computing resource/bit w_k, and delay limit d_k
$(x_k, y_k, 0)$	Coordinates of IoTD k
(x_n, y_n, z_n)	Coordinates of ACP n
$d_{k,n}$	Distance between IoTD k and ACP n
$p_{\max}$	Maximum transmit power of IoTDs
$g_{k,n}$	Channel gain between IoTD k and ACP n
g_0	Channel gain at a reference distance of 1 m
$r_{k,n}$	Transmission rate between IoTD k and ACP n
κ_k	Energy consumption coefficient of IoTD k
σ_n^2	Noise power at ACP n
W_n	Communication bandwidth of ACP n
F_k, F_n	Computational capacity of IoTD k and ACP n
$T_k, T_k^{\text{local}}, T_{k,n}^{\text{offload}}$	Delay cost, IoTD k's delay, and ACP n's delay for processing τ_k
$E_k, E_k^{\text{local}}, E_{k,n}^{\text{offload}}$	Energy cost, IoTD k's energy, and ACP n's energy for processing τ_k
ω_k	Objective weight coefficient for IoTD k
C_k	Overall computing overhead for processing τ_k
Decision variables	
$a_{k,n}$	Association between IoTD k and ACP n
o_k	Offloading ratio for task τ_k
p_k	Transmit power of IoTD k
$f_{k,n}$	Computing resources of ACP n allocated to IoTD k

III. System Model and Problem Formulation

This section details the system model and problem formulation. Key notations used in the article are presented in Table I.

A. Network Scenario

We examine a heterogeneous multi-ACP-enabled NOMA edge network with a HAP, multiple UAVs, and various terrestrial IoTDs within the HAP's coverage, as shown in Fig. 1. In remote areas, the proposed heterogeneous multi-ACP system promises significant advantages regarding availability and reliability. The HAP offers wide-area coverage and redundancy for consistent connectivity to a large number of

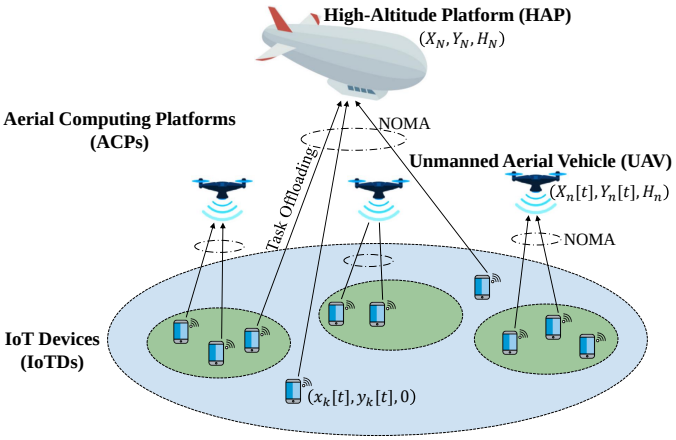


Fig. 1. A multi-ACP-enabled NOMA edge network.

IoTDs, while UAVs provide targeted support in specific areas. Additionally, the HAP is less susceptible to environmental factors due to its high altitude operation, improving reliability compared to UAVs. Finally, the system can dynamically allocate resources based on real-time demand, with UAVs assisting areas experiencing high computational load, while HAPs serve as the backbone for overall connectivity and processing power. It is also assumed that the UAVs fly constantly along predetermined paths, and harvesting techniques replenish their energy [15], [19], [25]. The set of IoTDs is denoted as $k \in \boldsymbol{\mathcal{K}} = \{1, \ldots, K\}$, and the HAP and UAVs equipped with MEC servers form a set of ACPs, represented as $n \in \boldsymbol{\mathcal{N}} = \{1, \ldots, N\}$, where $n = N$ represents the HAP, otherwise, the UAV. Each ACP serves as a computing node for the IoTDs under its coverage. The set of IoTDs covered by the ACP n is denoted as $\boldsymbol{\mathcal{K}}_n$ with the number of IoTDs K_n accordingly.

The network operates throughout T discrete time slots of equal duration, with each slot being small enough to assume that the network is static during that time. The set of time slots is specified as $t \in \boldsymbol{\mathcal{T}} = \{1, \ldots, T\}$. The network coordinates are represented in a three-dimensional Euclidean coordinate system. The IoTDs' positions at time slot t are specified as $(x_k[t], y_k[t], 0), \forall k$. The UAVs' positions at time slot t are denoted as $(x_n[t], y_n[t], z_n), \forall n \neq N$, with a constant altitude of z_n. The HAP is deployed at a fixed point over the region of interest at an altitude of z_N in the stratosphere [31], and its position is denoted as (x_N, y_N, z_N). Hence, at time slot t, the distance between IoTD k and UAV n, referred to as $d_{k,n}[t]$, and the distance between IoTD k and the HAP N, referred to as $d_{k,N}[t]$, are computed by

$$d_{k,n}[t] = \sqrt{(x_n[t] - x_k[t])^2 + (y_n[t] - y_k[t])^2 + z_n^2}, \quad (1)$$

$$d_{k,N}[t] = \sqrt{(x_n - x_k[t])^2 + (y_N - y_k[t])^2 + z_N^2}. \quad (2)$$

Each IoTD k generates a computation-intensive and time-sensitive task τ_k in the network. At time slot t, the task generated by IoTD k is represented by $\tau_k[t] = (s_k[t], w_k[t], d_k[t])$. Here, s_k is the task size (bits), w_k is the required computational resources to process a data bit (CPU cycles/bit), and d_k is the limited delay. A partial offloading model is applied to provide flexibility, allowing for a portion of the task to be offloaded and the remainder to be processed locally. This is more flexible than a binary offloading model [2], [32]. Hence, for each task $\tau_k[t]$, we introduce an offloading rate variable $o_k[t] \in [0, 1]$ as the portion of the task offloaded to the ACP, while the remaining portion $(1 - o_k[t])$ is processed locally.

Since the IoTDs can be under the coverage of multiple ACPs, we introduce an IoTD association variable, $a_{k,n}[t] \in \{0, 1\}, \forall k, n, t$, to represent the association between the IoTD k and the ACP n. Hence, $a_{k,n}[t] = 1$ indicates that the IoTD k associate with the ACP n, otherwise, $a_{k,n}[t] = 0$. At each time slot, each IoTD is only served by at most one ACP represented by

$$\sum_{n \in \boldsymbol{\mathcal{N}}} a_{k,n}[t] \leq 1, \forall k, t. \quad (3)$$

B. Communication Model

NOMA is used to connect IoTDs and ACPs as it can provide better spectrum utilization than traditional orthogonal multiple access (OMA) techniques. In NOMA, multiple users can be supported in the same time-frequency resource block, while each user has their time-frequency resource block in OMA. To ensure reliable data acquisition, we assume each ACP can obtain accurate channel state information (CSI) and perform perfect successive interference cancellation (SIC). With the perfect SIC, the ACP can completely subtract the signals of IoTDs with stronger channels from those with weaker channels, enabling significantly improved decoding accuracy for all IoTD messages [12], [17].

In contrast to densely populated urban environments, the relative absence of physical obstructions in remote areas facilitates a dominance of LoS communication for connecting IoTDs with ACPs [11]–[13], [15], [20]. In addition, the system leverages the deployment flexibility, mobility, and versatility of ACPs to readily establish and maintain LoS conditions. Hence, the contribution of non-LoS (NLoS) propagation can be ignored. Therefore, the channel gain between IoTD $k \in \boldsymbol{\mathcal{K}}_n$ and the associated ACP n at time slot t can be calculated by

$$g_{k,n}[t] = g_0 d_{k,n}^{-\alpha}[t], \quad (4)$$

where $\alpha \geq 2$ denotes a path loss exponent, and g_0 denotes the channel gain at a reference distance of 1 m. From the principle of SIC, the signals are decoded in the decreasing order of channel gains, i.e., $g_{1,n} \geq g_{2,n} \geq \cdots \geq g_{K_n,n}$. Thus, at time slot t, the transmission rate (bits/s) of IoTD k to ACP n can be estimated as

$$r_{k,n}[t] = W_n \log_2 \left(1 + \frac{a_{k,n}[t] p_k[t] g_{k,n}[t]}{\sum_{j=k+1}^{K_n} a_{j,n}[t] p_j[t] g_{j,n}[t] + \sigma_n^2}\right), \quad (5)$$

where the communication bandwidth of ACP n is represented as W_n, $p_k[t]$ represents the transmit power of IoTD k, and σ_n^2 denotes the noise power at ACP n. It is worth noting that the transmit power of IoTD k is constrained by a maximum transmit power $p_{\max}$, i.e., $0 \leq p_k[t] \leq p_{\max}, \forall k, t$.

C. Computation Model

By leveraging IoTD associations, tasks can be concurrently offloaded to the corresponding ACPs and processed locally. This work considers the computing overhead regarding task processing delay and energy consumption of IoTDs. Similar to previous studies [2], [15], [32], the overhead associated with transmitting the results is negligible due to their typically small size.

1) Local Computation: The local processing delay of $(1 - o_k[t])$ part of the task $\tau_k[t]$ of the IoTD k is computed by

$$T_k^{\text{local}}[t] = \frac{(1 - o_k[t])s_k[t]w_k[t]}{F_k}, \tag{6}$$

where F_k denotes the computational capacity of IoTD k (CPU cycles/s). As in [15], [32], the energy consumed by the IoTD for processing the task can be calculated as

$$E_k^{\text{local}}[t] = \kappa_k F_k^2 (1 - o_k[t]) s_k[t] w_k[t], \tag{7}$$

where $\kappa_k \geq 0$ represents the energy consumption factor for IoTD k.

2) ACP-Assisted Computation: If IoTD k delegates $o_k[t]$ portion of task τ_k to the connected ACP n, then the processing latency comprises two phases: the offloading duration from IoTD k to ACP n, and the execution duration at ACP n. Hence, during time slot t, the latency for handling $o_k[t]$ fraction of task τ_k through offloading is estimated as

$$T_{k,n}^{\text{offload}}[t] = a_{k,n}[t] \left(\frac{o_k[t]s_k[t]}{r_{k,n}[t]} + \frac{o_k[t]s_k[t]w_k[t]}{f_{k,n}[t]} \right), \tag{8}$$

where $f_{k,n}[t]$ denotes the computing resources of the ACP n allocated to IoTD k. The allocated computing resources cannot exceed the available computing capacity F_n of the ACP n, which should satisfy $\sum_{k \in \mathcal{K}} a_{k,n}[t] f_{k,n}[t] \leq F_n, \forall n, t$.

Thus, the energy utilization of IoTD k for delegating the $o_k[t]$ part of task τ_k to ACP n is determined by

$$E_{k,n}^{\text{offload}}[t] = \frac{a_{k,n}[t]p_k[t]o_k[t]s_k[t]}{r_{k,n}[t]}. \tag{9}$$

Offloading a task (τ_k) to an ACP requires careful resource allocation to ensure successful execution. This means when offloading part of the task ($o_k > 0$), we must set non-zero values for three key variables: $a_{k,n}$, p_k, and $f_{k,n}$. In other words, the chosen ACP n, identified by $a_{k,n}$, must dedicate processing power $f_{k,n}$ to handle the offloaded part of the task o_k sent from the IoTD k with transmit power p_k. This ensures the offloaded part of the task is fully executed at the ACP and not dropped or incomplete. Hence, we impose a constraint as

$$\sum_{n \in \mathcal{N}} \lceil a_{k,n}[t]p_k[t]f_{k,n}[t] \rceil \geq \lceil o_k[t] \rceil, \forall k, t, \tag{10}$$

where $\lceil \cdot \rceil$ represents the ceiling function, which returns to the least integer greater than or equal to the input number.

3) Computing Overhead: Under the partial offloading scheme, an IoT device's task can be processed locally and through offloading simultaneously [2], [32]. Hence, the time taken to complete IoTD k's task is determined by

$$T_k[t] = \max \left\{ T_k^{\text{local}}[t], T_{k,n}^{\text{offload}}[t] \right\}. \tag{11}$$

The IoTD k consumes energy for local processing and offloading the task to the ACP, represented by

$$E_k[t] = E_k^{\text{local}}[t] + E_{k,n}^{\text{offload}}[t]. \tag{12}$$

Overall, the computation overhead of the task $\tau_k[t]$ for IoTD k is determined by

$$C_k[t] = \omega_k T_k[t] + (1 - \omega_k) E_k[t], \tag{13}$$

where ω_k is a coefficient that represents IoTD k's preference for task execution latency and energy consumption. To ensure the QoS, the task has to be completed within a maximum delay $d_k[t]$, expressed as

$$T_k[t] \leq d_k[t], \forall k, t. \tag{14}$$

D. Joint Optimization Problem

We formulate the JTORA problem in minimizing the consumed energy of the IoTDs and the delay in processing their tasks, subject to the system constraints. Specifically, the decision variables to be optimized include IoTD association $\{a_{k,n}[t], \forall k, n, t\}$, offloading ratio $\{o_k[t], \forall k, t\}$, transmit power of IoTDs $\{p_k[t], \forall k, t\}$, and computing resource allocation of the HAP and UAVs $\{f_{k,n}[t], \forall k, n, t\}$. The JTORA problem is formulated as

$$\text{(P1):} \min_{\{a_{k,n}[t], o_k[t], p_k[t], f_{k,n}[t], \forall k, n, t\}} \sum_{k \in \mathcal{K}} C_k[t], \tag{15a}$$

$$\text{s.t. } a_{k,n}[t] \in \{0, 1\}, \forall k, n, t, \tag{15b}$$

$$\sum_{n \in \mathcal{N}} a_{k,n}[t] \leq 1, \forall k, t, \tag{15c}$$

$$o_k[t] \in [0, 1], \forall k, t, \tag{15d}$$

$$0 \leq p_k[t] \leq p_{\max}, \forall k, t, \tag{15e}$$

$$\sum_{k \in \mathcal{K}} a_{k,n}[t] f_{k,n}[t] \leq F_n, \forall n, t, \tag{15f}$$

$$\sum_{n \in \mathcal{N}} \lceil a_{k,n}[t]p_k[t]f_{k,n}[t] \rceil \geq \lceil o_k[t] \rceil, \forall k, t, \tag{15g}$$

$$T_k[t] \leq d_k[t], \forall k, t, \tag{15h}$$

where (15b) and (15c) ensure that each IoTD is assigned to one ACP at a time, (15d) restricts the offloading ratio to a valid range, (15e) limits the IoTD's transmit power not to exceed the maximum transmit power, (15f) ensures that the allocated computing resources for all associated IoTDs do not exceed the ACP's available computing resources, (15g) guarantees that the tasks are executed completely, and (15h) limits the task's execution delay not to exceed the delay limit, ensuring the system's QoS. However, the problem (P1) is a mixed-integer programming problem, which is non-convex and NP-hard due to the discrete integer variable (i.e., IoTD association) and continuous variables (i.e., offloading ratio, transmission power, and computation resource allocation). In addition, system states are temporally correlated among adjacent time slots, and behaviors of ACPs and IoTDs do not change arbitrarily. Solving the problem using conventional optimization techniques is challenging. Hence, we turn the problem into a MAMDP model and leverage a MADDPG algorithm to handle complex problem dynamics and achieve efficient policies for task offloading and resource allocation.

IV. Multi-Agent DRL Framework for Multi-ACP-enabled NOMA Edge Computing

In this section, we undertake a normalization process to eliminate the impact of the diversity of optimization variables. Then, the MAMDP model is constructed for the normalized problem and solved using the MADDPG-based framework.

A. Problem Normalization

To eliminate the effect of the diversity of the decision variables, we normalize them into a continuous range suitable for activation functions in the multi-agent DRL framework. First, we utilize power variable $p'_k[t]$, and resource variable $f'_{k,n}[t] \in [0,1]$ as the normalized variables of $p_k[t]$, and $f_{k,n}[t]$, respectively. Then, the values of $p_k[t]$ and $f_{k,n}[t]$ are computed by

$$\begin{aligned} p_k[t] &= p'_k[t] p_{\max}, \\ f_{k,n}[t] &= f'_{k,n}[t] F_n. \end{aligned} \tag{16}$$

Furthermore, we relax the binary association variables $a_{k,n}$ to the continuous variables $a'_{k,n}$ with the range $[0,1]$. Then, the binary association variable $a_{k,n}$ is represented by

$$a_{k,n} = \begin{cases} 1, & \text{if } n = \arg\max\limits_{n} \{a'_{k,n}, \forall n\}, \\ 0, & \text{otherwise}, \end{cases} \tag{17}$$

where the $\arg\max$ function returns the maximum index in the considered list. By using the relaxation variables $a'_{k,n}$ and the $\arg\max$ function, the constraints (15b) and (15c) in the problem (P1) are naturally satisfied. Besides, the ACPs' computing resources are assumed to be fully utilized to optimize system performance regarding cost function minimization. Consequently, the constraints (15e), (15f), (15g) in the problem (P1) are rewritten with normalized variables as

$$p'_k[t], f'_{k,n}[t] \in [0,1], \tag{18}$$

$$\sum_{k \in \boldsymbol{\mathcal{K}}} a_{k,n}[t] f'_{k,n}[t] = \begin{cases} 1, & \text{if} \sum_{k \in \boldsymbol{\mathcal{K}}} a_{k,n}[t] \geq 1, \\ 0, & \text{otherwise}, \end{cases} \tag{19}$$

$$\sum_{n \in \boldsymbol{\mathcal{N}}} \lceil a_{k,n}[t] p'_k[t] f'_{k,n}[t] \rceil = \lceil o_k[t] \rceil. \tag{20}$$

After normalizing the variables, the original problem (P1) can be reformulated as

$$\begin{aligned} &\text{(P2):} \min_{\{a'_{k,n}[t], o_k[t], p'_k[t], f'_{k,n}[t], \forall k,n,t\}} \sum_{k \in \boldsymbol{\mathcal{K}}} C_k[t], \\ &\text{s.t. } a'_{k,n}[t], o_k[t] \in [0,1], (14), (18), (19), (20), \forall k, n, t. \end{aligned} \tag{21}$$

B. Multi-Agent Markov Decision Process Transformation

In this part, the problem (P2) is transformed into a MAMDP model, where ACPs act as agents that sense their environment and jointly optimize task offloading and resource allocation. The set of ACP agents is denoted by $n \in \boldsymbol{\mathcal{N}} = \{1, \ldots, N\}$, where $n = N$ indicates the HAP agent; otherwise, the UAV agent. We define a tuple $(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{R}})$ to describe the interaction process within MAMDP. Here, $\mathcal{S}$ denotes the global state space, $\mathcal{O}_n \in \boldsymbol{\mathcal{O}} = \{\mathcal{O}_1, \ldots, \mathcal{O}_N\}$ represents the set of observations, $\mathcal{A}_n \in \boldsymbol{\mathcal{A}} = \{\mathcal{A}_1, \ldots, \mathcal{A}_N\}$ indicates the set of actions, and $\mathcal{R}_n \in \boldsymbol{\mathcal{R}} = \{\mathcal{R}_1, \ldots, \mathcal{R}_N\}$ represents the set of rewards. In a given state $\mathbf{s}[t] \in \boldsymbol{\mathcal{S}}$, each agent n receives a local observation $\mathbf{o}_n[t] \in \mathcal{O}_n$, takes an action $\mathbf{a}_n[t] \in \mathcal{A}_n$, and obtains an individual reward $\mathbf{r}_n[t] \in \mathcal{R}_n$. Furthermore, we propose an action pretreatment function that ensures the action is feasible before interacting with the environment to handle constraints in the problem (P2).

1) State: The current system state that includes the ACP-IoTD channel information and computational tasks is denoted as the state $\mathbf{s}[t] \in \mathcal{S}$. Formally, the state $\mathbf{s}[t]$ during time slot t can be expressed by

$$\mathbf{s}[t] = \{g_{1,n}[t], \ldots, g_{K_n,n}[t], \tau_1[t], \ldots, \tau_K[t], \forall n \in \boldsymbol{\mathcal{N}}\}. \tag{22}$$

2) Observation: The observation of each ACP agent n is a partial view of the state available to each agent, including its channel information with the associated IoTDs and corresponding tasks. Hence, the observation $\mathbf{o}_n[t] \in \mathcal{O}_n$ of ACP agent n during time slot t is described as

$$\mathbf{o}_n[t] = \{g_{1,n}[t], \ldots, g_{K_n,n}[t], \tau_1[t], \ldots, \tau_{K_n}[t]\}. \tag{23}$$

3) Action: The action is the decision made by each agent based on its observation. Accordingly, the ACP agents make decisions on the device association, the ratio of task offloading, the transmit power, and the computing resource allocation. According to the problem (P2), the action $\mathbf{a}_n[t] \in \mathcal{A}_n$ of ACP agent n at time slot t is defined as

$$\mathbf{a}_n[t] = \{a'_{k,n}[t], o_k[t], p'_k[t], f'_{k,n}[t], \forall k \in \boldsymbol{\mathcal{K}}_n\}. \tag{24}$$

4) Action Pretreatment: An action pretreatment function ensures that the agent's actions adhere to the environmental constraints. This enables the agent to learn policies within the specified limits, preventing penalties or undesirable outcomes. Initially, a *sigmoid* function, defined as $\textit{sigmoid}(x) = \frac{1}{1+e^{-x}}$, is applied as the activation function for the actor network's output. This scales the action values within the $[0,1]$ range. To address the remaining constraints in the problem (P2), a multi-stage mapping function (Algorithm 1), denoted as $\mathcal{M}(\cdot)$, is introduced. It operates as follows.

- Stage 1: If the offloading ratio variable for the IoTD k is 0, indicating no intention to offload, the corresponding mapping IoTD association variable is also set to 0. This signifies that no ACP is designated for that particular IoTD.
- Stage 2: The binary IoTD association variables $a_{k,n}$ in (17) are recalculated based on the updated mapping IoTD association variables obtained from Stage 1.
- Stage 3: When no ACP is available to execute the task's offloaded portion o_k, the offloading ratio variable is set to 0. This ensures that no offloaded task is left unassigned, satisfying constraint (20).
- Stage 4: Computing resources are allocated exclusively to IoTDs intended to offload to a specific ACP n, as determined by the mapping IoTD association variables. The vector of computing resource allocation variables for ACP n, after filtering out non-associated IoTDs, is denoted as $\{f_{k,n}^{\text{filter}}, \forall k \in \boldsymbol{\mathcal{K}}_n\}$.

Algorithm 1: Multi-stage mapping function $\mathcal{M}(\cdot)$

1: **Input:** Action $\mathbf{a}_n$

2: **Stage 1:** Map association variables $a_{k,n}^{\text{map}}$ observing offloading ratio variables $o_k, \forall n, k \in \boldsymbol{\mathcal{K}}_n$

$$a_{k,n}^{\text{map}} = \begin{cases} 0, & \text{if } o_k = 0, \\ a'_{k,n}, & \text{otherwise.} \end{cases}$$

3: **Stage 2:** Re-calculate $a_{k,n}, \forall n, k \in \boldsymbol{\mathcal{K}}_n$

$$a_{k,n} = \begin{cases} 1, & \text{if } n = \arg\max\limits_{n} \{a_{k,n}^{\text{map}}, \forall n\} \text{ and } a_{k,n}^{\text{map}} > 0, \\ 0, & \text{otherwise.} \end{cases}$$

4: **Stage 3:** Map offloading ratio variables o_k^{map} observing remaining variables $a_{k,n}, p'_k, f'_{k,n}, \forall n, k \in \boldsymbol{\mathcal{K}}_n$

$$o_k^{\text{map}} = \begin{cases} 0, & \text{if } \sum_{n \in \boldsymbol{\mathcal{N}}} a_{k,n} p'_k f'_{k,n} = 0, \\ o_k, & \text{otherwise.} \end{cases}$$

5: **Stage 4:** Filter out non-associated IoTDs for the ACPs

$$f_{k,n}^{\text{filter}} = a_{k,n} f'_{k,n}, \forall n, k \in \boldsymbol{\mathcal{K}}_n$$

6: **Stage 5:** Map computing resource allocation variables $f_{k,n}^{\text{map}}$

$$f_{k,n}^{\text{map}} = \textit{softmax}(f_{k,n}^{\text{filter}}) = \frac{e^{f_{k,n}^{\text{filter}}}}{\sum_{j=1}^{K_n} e^{f_{j,n}^{\text{filter}}}}, \forall n, k \in \boldsymbol{\mathcal{K}}_n$$

7: **return** $a_{k,n}^{map}, o_k^{map}, p'_k, f_{k,n}^{map}$

- Stage 5: We apply the *softmax* function to normalize the vector of filtered computing resource allocation variables $\{f_{k,n}^{\text{filter}}, \forall k \in \boldsymbol{\mathcal{K}}_n\}$ into a probability distribution, represented by $\{f_{k,n}^{\text{map}}, \forall k \in \boldsymbol{\mathcal{K}}_n\}$. This normalization ensures that the available computing resources of ACP n are fully utilized, as stated in constraint (19).

With the implementation of the action pretreatment function, only the QoS constraint (14) remains unaddressed. This constraint, along with the delay limit value, aids in determining whether a task is successfully executed, contributing to the calculation of the task success rate, a key performance evaluation metric.

5) Reward: The reward evaluates the effect of an agent's action on the current state. In this paper, it is defined as the negative of the computing overhead of tasks with a penalty factor. The reward of each ACP agent n is defined by

$$\mathbf{r}_n[t] = \sum_{k \in \boldsymbol{\mathcal{K}}_n} -(C_k[t] + \lambda_k[t]\Delta), \tag{25}$$

where Δ is the value of the penalty and $\lambda_k[t]$ denotes the binary penalty variable, which is expressed by

$$\lambda_k[t] = \begin{cases} 1, & \text{if } T_k[t] > d_k[t], \\ 0, & \text{otherwise.} \end{cases} \tag{26}$$

Each ACP agent n aims to maximize its long-term expected reward as

$$\mathbf{R}_n = \max_{\mathbf{a}_n[t]} \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} \mathbf{r}_n[t]\right], \tag{27}$$

where $\gamma \in [0, 1]$ is a discount factor that determines how much importance should be given to future rewards and $\mathbb{E}[\cdot]$ returns the expected value.

C. Multi-Agent DRL Design for JTORA Optimization

MADDPG [24] is a potent algorithm for addressing continuous optimization challenges in multi-agent systems within the realm of multi-agent DRL. MADDPG expands on the DDPG algorithm [23] by employing a centralized training and decentralized execution framework. Each agent's policy is represented by a DNN, which is trained using the DDPG algorithm. During training, agents leverage information from other agents' observations and actions to refine their policies, fostering cooperation. During execution, each agent relies solely on its observations to determine its actions.

We propose a MADDPG-based framework to solve the described MAMDP. For simplicity, we have omitted the time slot index in the following. In the centralized training phase, we consider a set of agents denoted by $\boldsymbol{\mathcal{N}}$. Each agent n has its own set of parameters denoted by $\theta_n = \{\theta_n^{\mu}, \theta_n^{Q}, \theta_n^{\mu'}, \theta_n^{Q'}\}$, where θ_n^{μ} and θ_n^{Q} are the parameters of its actor network μ_n and critic network Q_n, respectively. Additionally, $\theta_n^{\mu'}$ and $\theta_n^{Q'}$ are the parameters of the target actor network μ'_n and target critic network Q'_n, respectively. The actor network generates actions based on the agent's local observation, while the critic network evaluates the quality of the chosen action. Meanwhile, the target networks enhance the stability of the training process by periodically updating them with the parameters from the main networks [23]. Each agent n aims to maximize its long-term expected reward through an objective function, $\mathcal{J}(\theta_n^{\mu}) = \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} \mathbf{r}_n[t]\right]$, and can choose an action based on its local observation, $\mathbf{a}_n = \mu_n(\mathbf{o}_n \mid \theta_n^{\mu})$. The agent n can calculate the gradient of the deterministic policy μ_n by

$$\begin{aligned} \nabla_{\theta_n^{\mu}} \mathcal{J} = \ & \mathbb{E}_{\mathbf{s},\mathbf{a} \sim \mathcal{D}} [\nabla_{\theta_n^{\mu}} \mu_n(\mathbf{o}_n \mid \theta_n^{\mu}) \\ & \nabla_{\mathbf{a}_n} Q_n\left(\mathbf{s}, \mathbf{a} \mid \theta_n^{Q}\right) |_{\mathbf{a}_n = \mu_n(\mathbf{o}_n \mid \theta_n^{\mu})}], \end{aligned} \tag{28}$$

where $\mathcal{D}$ denotes the experience replay buffer recording all agents' experiences, which contains a series of tuples $(\mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}')$. Here, $\mathbf{s}$, $\mathbf{a} = \{\mathbf{a}_n, \forall n\}$, $\mathbf{r} = \{\mathbf{r}_n, \forall n\}$, and $\mathbf{s}'$ represent the states, actions, rewards, and next states for all agents, respectively. $Q_n\left(\mathbf{s}, \mathbf{a} \mid \theta_n^{Q}\right) |_{\mathbf{a}_n = \mu_n(\mathbf{o}_n \mid \theta_n^{\mu})}$ is a centralized critic function, also known as an action-value function. In the training procedure, the critic network Q_n is updated by minimizing a loss function parameterized by θ_n^{Q} as

$$\begin{aligned} & \mathcal{L}\left(\theta_n^{Q}\right) = \mathbb{E}_{\mathbf{s},\mathbf{a},\mathbf{r},\mathbf{s}'} \left[\left(Q_n\left(\mathbf{s}, \mathbf{a} \mid \theta_n^{Q}\right) - Y_n\right)^2\right], \\ & Y_n = \mathbf{r}_n + \gamma Q'_n\left(\mathbf{s}', \mathbf{a}' \mid \theta_n^{Q'}\right)\Big|_{\mathbf{a}'_n = \mu'_n\left(\mathbf{o}'_n \mid \theta_n^{\mu'}\right)}, \end{aligned} \tag{29}$$

where Y_n is the target state-action value estimated by the target networks, $\mathbf{a}' = \{\mathbf{a}'_n, \forall n\}$, and $\mathbf{a}'_n$ denotes the action of the agent n given its next observation $\mathbf{o}'_n$ in the next state $\mathbf{s}'$. Then,

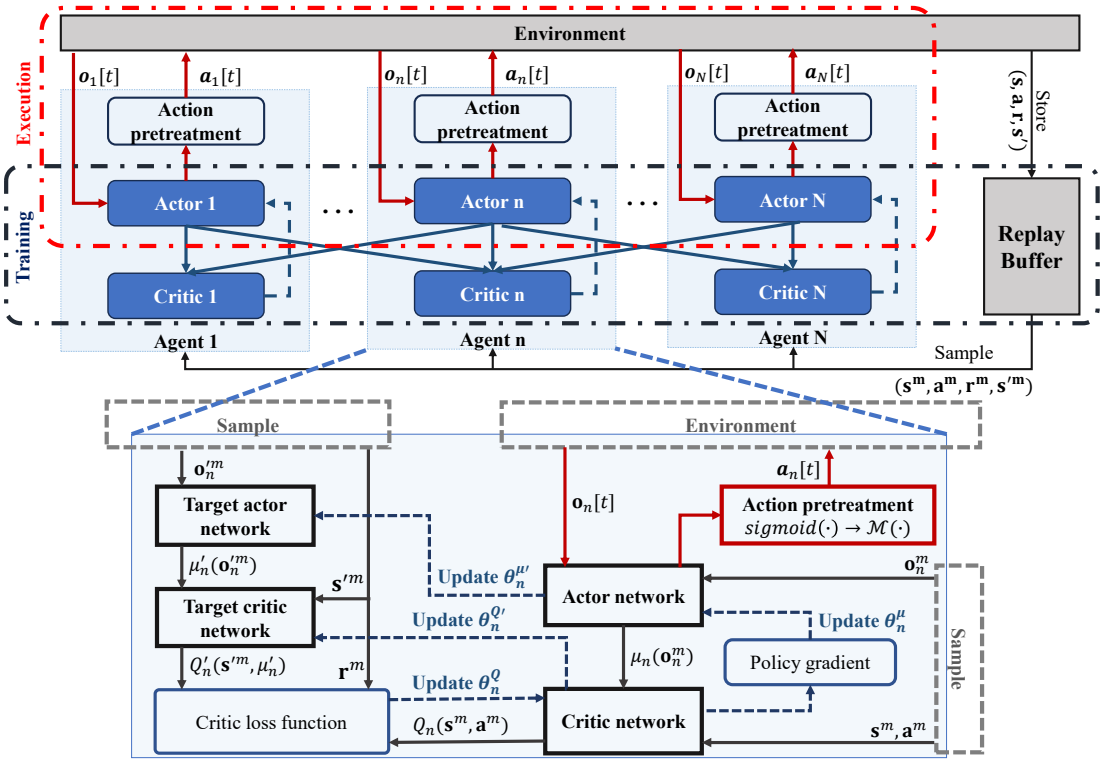


Fig. 2. MADDPG-based optimization framework for multi-ACP-enabled NOMA edge network.

the parameters of the n-th actor network can be updated based on the critic network using the policy gradient given by

$$\nabla_{\theta_n^\mu} \mathcal{J} \approx \frac{1}{M} \sum_{m}^{M} [\nabla_{\theta_n^\mu} \mu_n(\mathbf{o}_n^m \mid \theta_n^\mu) \nabla_{\mathbf{a}_n} Q_n(\mathbf{s}, \mathbf{a} \mid \theta_n^Q)|_{\mathbf{a}_n = \mu_n(\mathbf{o}_n^m \mid \theta_n^\mu)}], \tag{30}$$

where M is the size of the mini-batch randomly sampled from the relay buffer $\mathcal{D}$. Finally, the parameter update process for the target networks is performed by each agent n to enhance learning stability as

$$\begin{aligned} \theta_n^{Q'} &\leftarrow \varsigma \theta_n^Q + (1-\varsigma)\theta_n^{Q'}, \\ \theta_n^{\mu'} &\leftarrow \varsigma \theta_n^\mu + (1-\varsigma)\theta_n^{\mu'}, \end{aligned} \tag{31}$$

where $\varsigma \ll 1$ is the soft update coefficient.

The proposed MADDPG framework is illustrated in Fig. 2, which is the centralized training and decentralized execution as considered in many previous works [18]–[20], [24]. In the centralized training stage, besides local observation, each agent is trained using extra information, including other agents' observations and actions. In the decentralized execution stage, each agent n decides its action a_n according to the local observation o_n and applies the proposed action pretreatment function to satisfy the action constraints. We detail the proposed framework in Algorithm 2. Initialization of networks, hyperparameters, and the network environment is performed (lines 1–4). Each episode involves observing the initial state and observations (line 6). The workflow comprises data collection and training. At time slot t, each agent determines an action based on its observation using its actor network (line 10). To enhance agent performance and encourage action exploration, we add noise to the actor policy, resulting in the action $\mathbf{a}_n = \mu_n(\mathbf{o}_n) + \mathcal{OU}$, where $\mathcal{OU}$ is augmented Ornstein-Uhlenbeck noise. A pretreatment function handles constraints (line 11). Each agent interacts independently with the environment, obtaining reward $\mathbf{r}_n$ and storing $(\mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}')$ in replay buffer $\mathcal{D}$ (lines 13–14). Training is enforced through information sharing. In each training step, a batch of experience samples $(\mathbf{s}^m, \mathbf{a}^m, \mathbf{r}^m, \mathbf{s}'^m)$ is randomly drawn from $\mathcal{D}$ (line 18). Agent n obtains its sample observation $\mathbf{o}_n^m$ from sample state $\mathbf{s}^m$ (line 19). Each agent updates its critic and actor networks accordingly (lines 20–21). Finally, the soft update rule updates the target networks (line 23). The algorithm repeats until convergence or the desired number of episodes is reached, yielding optimal actor network parameters $\{\theta_n^\mu, \forall n\}$ (line 26). After training, each UAV downloads the trained action networks from the HAP and performs actions in a decentralized manner.

D. Computational Complexity

We examine the computational complexity of the proposed MADDPG-based framework. The DNNs in the framework have an input layer, two hidden layers, and an output layer, which are fully connected. The computational complexity is mainly determined by the number of matrix multiplications in DNNs [33]. Hence, in each training step, the computational complexity of each agent is computed as

$$O\left(\sum_{l^a=0}^{L^a-1} X_{l^a}^a X_{l^a+1}^a + \sum_{l_c=0}^{L^c-1} X_{l^c}^c X_{l^c+1}^c\right), \tag{32}$$

Algorithm 2: MADDPG-based framework for joint task offloading and resource allocation.

Initialize actor and critic networks with random weights
Initialize target networks with the same weights as corresponding networks
Initialize hyperparameters: number of episodes Z, number of time steps T, learning rates (lr_a, lr_c), relay buffer $\mathcal{D}$, mini-batch size M, discount factor γ
Initialize network environment
for $z = 1, \ldots, Z$ **do**
  Reset environment, observe initial state $\mathbf{s}[1]$ and observations $\{\mathbf{o}_n[1], \forall n\}$
  for $t = 1, \ldots, T$ **do**
    # Collecting observed data
    for $n = 1, \ldots, N$ **do**
      Select action $\mathbf{a}_n = \mu_n(\mathbf{o}_n) + \mathcal{OU}$
      Perform action pretreatment $\mathcal{M}(\mathbf{a}_n)$
    end for
    Execute joint actions $\mathbf{a}$, obtain rewards $\mathbf{r}$, and observe new state $\mathbf{s}'$
    Add experience tuple $(\mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}')$ to $\mathcal{D}$
    Set $\mathbf{s} \leftarrow \mathbf{s}'$
    # Training process
    for $n = 1, \ldots, N$ **do**
      Randomly sample a mini-batch of M samples $(\mathbf{s}^m, \mathbf{a}^m, \mathbf{r}^m, \mathbf{s}'^m)$ from $\mathcal{D}$
      Obtain observations $\{\mathbf{o}_n^m, \forall n\}$ from state $\mathbf{s}^m$
      Update critic network by (29)
      Update actor network by (30)
    end for
    Update target networks for each agent by (31)
  end for
end for
return $\{\theta_n^\mu, \forall n\}$

where L^a and L^c are the number of layers in the actor and critic networks, respectively, and $X_{l^a}^a$ and $X_{l^c}^c$ are the number of nodes in the l^a-th actor layer and the l^c-th critic layer, respectively. The overall computational complexity of the MADDPG-based training framework is given by

$$O\left(NTZM\left(\sum_{l^a=0}^{L^a-1} X_{l^a}^a X_{l^a+1}^a + \sum_{l_c=0}^{L^c-1} X_{l^c}^c X_{l^c+1}^c\right)\right), \quad (33)$$

where N is the number of agents, T is the number of training steps, Z is the number of training episodes, and M is the mini-batch size. The training stage can be done in the HAP, which has a powerful computation capability [18], [21]. During the decentralized execution phase, each ACP agent simply executes its policy based on the trained actor network with the computational complexity given as $O\left(\sum_{l^a=0}^{L^a-1} X_{l^a}^a X_{l^a+1}^a\right)$, which is favorable and suitable for real-time inference.

V. Performance Evaluation

In this section, we evaluate the convergence and performance of the proposed MADDPG-based framework in a simulated multi-ACP-enabled NOMA edge network.

TABLE II
Simulation parameters.

Parameter	Value
Environment parameters	
Time slot duration	0.5 s
Noise power, σ_n^2	-170 dBm/Hz
Reference channel gain, g_0	$1.42e^{-4}$
Path loss exponent, α	2
HAP bandwidth, W_N	200 MHz
HAP computing resource, F_N	100 GHz
UAV bandwidth, W_n	20 MHz
UAV computing resource, F_n	20 GHz
Number of IoTDs, K	20
IoTDs computing resource, F_k	1 GHz
IoTDs energy coefficient, κ_k	$1e^{-28}$
IoTDs maximum transmit power, $p_{\max}$	20 dBm
Task size, s_k	[0.8, 1.2] Mbits
Required computation per task, w_k	[700, 900] CPU cycles/bit
Task delay constraint, d_k	[0.4, 0.5] s
Weight coefficient, w_k	0.5
Algorithm parameters	
Penalty value, Δ	10000
Actor hidden layer 1	512 nodes
Actor hidden layer 2	128 nodes
Critic hidden layer 1	1024 nodes
Critic hidden layer 2	256 nodes
Replay buffer size	$5e^4$
Soft update coefficient, ς	0.01
Number of episodes for training	4000
Number of episodes for testing	100
Number of time steps per episode	200

A. Simulation Configurations

We use Python 3.10 and Pytorch 1.13.1 to simulate and train agents in an IoT edge network environment with a 1000m $\times$ 1000m area. The HAP is positioned at $(500, 500)$ (m) at a 20 km altitude, covering a 500 m radius. Twenty IoTDs ($K = 20$) are randomly distributed. The UAVs fly along predetermined paths at a velocity of 10 m/s and an altitude of 100 m with a coverage radius of 100 m. We evaluate two network environments: *3ACPs-env* with one HAP and two UAVs ($N = 3$), and *5ACPs-env* with one HAP and four UAVs ($N = 5$). Unless otherwise specified, the simulation parameters are given in Table II as in [13]–[15]. Given the lack of a standard, directly comparable algorithm for our specific optimization problem, we evaluate the performance of the proposed framework against several customized benchmarks below.

- *Proposed MADDPG-based Task Offloading and Resource Allocation (MADDPG)*: The proposed MADDPG framework jointly optimizes IoTD association, offloading ratio, transmit power, and computing resource allocation in a multi-agent setting, where all ACPs act as cooperative agents.
- *DDPG-based Task Offloading and Resource Allocation (DDPG)* [2], [30]: Via the DDPG algorithm, the HAP as a single agent observes the network environment and specifies the IoTD association, offloading ratio, transmit power, and computing resource allocation decisions for all IoTDs.
- *Greedy-based Task Offloading and Resource Allocation (GREEDY)* [29]: This scheme involves discretizing ac-

tions into discrete spaces, each comprising six value levels. A local search algorithm based on a greedy strategy determines the action at each step.
- *Full Task Offloading (FULL)* [15], [27]: All IoTD tasks are fully offloaded to the associated HAP or UAV.
- *Random Task Offloading (RANDOM)* [15], [27]: The IoTD tasks are processed with random decision variables.

The total reward obtained during each training episode is the evaluation metric for convergence analysis. For performance analysis, the following metrics are employed.

- *Episode cost*: It quantifies the average cost (or computing overhead) incurred per episode during the testing phase, which comprises 100 episodes. A lower episode cost indicates a more efficient and cost-effective approach.
- *Energy consumption* (Joule): It measures the overall energy consumed by the IoTDs during task execution.
- *Tasks success rate* (%): It represents the proportion of completed tasks within the delay limit. It is calculated as the number of successful tasks divided by the total number of tasks.

B. Convergence Analysis

To optimize the MADDPG-based framework, we perform Monte Carlo simulations by testing and analyzing the impact of four hyperparameters, i.e., learning rate, batch size, and discount factor, on the convergence of the proposed framework in *3ACPs-env*. The simulation results are displayed in Fig. 3. The same hyperparameters are used for all agents.

Firstly, we assess the convergence of the proposed training algorithm by testing three predetermined values for the learning rates, $(lr_a, lr_c) = \{(2e^{-3}, 5e^{-3}), (1e^{-3}, 5e^{-3}), (1e^{-3}, 3e^{-3})\}$, where lr_a refers to the learning rate of the actor network and lr_c refers to the learning rate of the critic network. The batch size used for training is 32, and the discount factor is 0.9. As shown in Fig. 3a, the learning rate set of $(1e^{-3}, 5e^{-3})$ achieves the best performance. The algorithm starts to converge after around 500 episodes with this learning rate set. Meantime, the learning rate set of $(1e^{-3}, 3e^{-3})$ starts to converge after approximately 1500 episodes. On the other hand, the set of $(2e^{-3}, 5e^{-3})$ fails to reach convergence. As a result, the set of learning rate $(1e^{-3}, 5e^{-3})$ is set as defaults in the following evaluations.

Secondly, we vary the batch size $M = \{16, 32, 64\}$ to investigate its impact on the training process. The discount factor is set to 0.9. As depicted in Fig. 3b, the proposed algorithm performs best when $M = 32$, while a smaller value of M (i.e., $M = 16$) results in slower convergence, and a larger value of M (i.e., $M = 64$) results in instability during training due to the noise introduced by the larger batch size. Hence, the batch size $M = 32$ is fixed during the following evaluations.

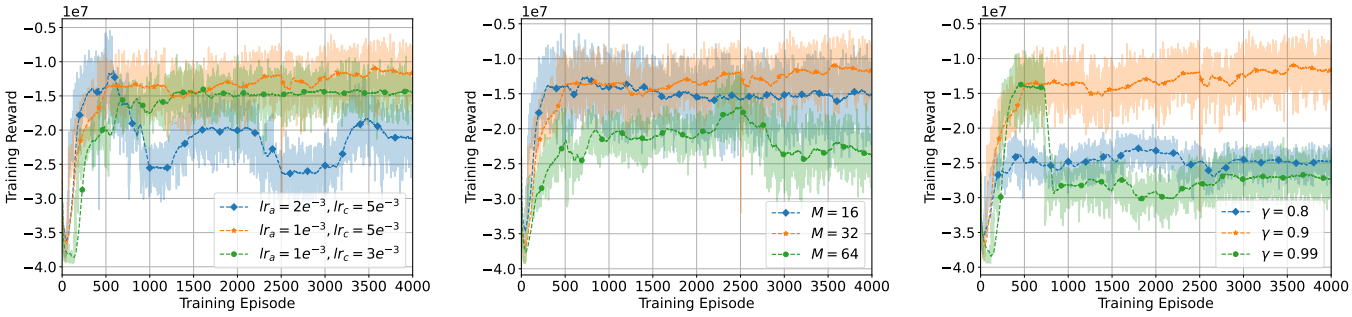
Thirdly, we analyze how changing the discount factor γ affects the convergence of the proposed MADDPG-based algorithm. The discount factor is a crucial hyperparameter that significantly impacts how the agent values future rewards. If the discount factor is set to 0, then only the immediate reward is considered, while a value of 1 would mean that all future rewards are considered. If the agent focuses on short-term rewards within a dynamic environment, the reward may fail to accurately capture the effectiveness of action over an extended period, especially when the environment undergoes changes; otherwise, convergence becomes unattainable when using a large discount factor, as it deviates significantly from the fundamental nature of the environment. As shown in Fig. 3c, a small discount factor ($\gamma = 0.8$) cannot correctly affect the nature of the dynamic environment, which makes the agent converge to a bad sub-optimal point. Also, a significant big discount factor ($\gamma = 0.99$) fails to capture the essence of the environment because of the significant focus on future rewards, which causes a big drop in the chaotic environment when the environment state in the training sample is large enough. The results indicate that a discount factor of $\gamma = 0.9$ performs the best in our system. Then, this discount factor value is used throughout the remaining simulations.

Finally, we assess the improvement of the MADDPG over the single-agent DDPG by modeling performance in two scenarios: *3ACPs-env* and *5ACPs-env*. For a fair comparison, MADDPG and single-agent DDPG use the same set of optimal hyperparameters evaluated above. Fig. 4a depicts the training performances of 3-agent DDPG and 1-agent DDPG in the *3ACPs-env*, whereas Fig. 4b depicts the training performances of 5-agent DDPG and 1-agent DDPG in the *5ACPs-env*. The findings in Fig. 4 indicate that MADDPG performs better than single-agent DDPG in both cases. The 3-agent DDPG performs approximately 35.33% better than the 1-agent DDPG in the first scenario. The 5-agent DDPG performs approximately 59% better than the 1-agent DDPG in the second scenario. MADDPG involves a centralized training phase followed by decentralized execution of the trained policies in the individual agents. MADDPG differs from single-agent DDPG by allowing agents to interact and coordinate their actions, enabling better collaboration and collective decision-making for improved system performance [15], [20], [21].

C. Performance Analysis

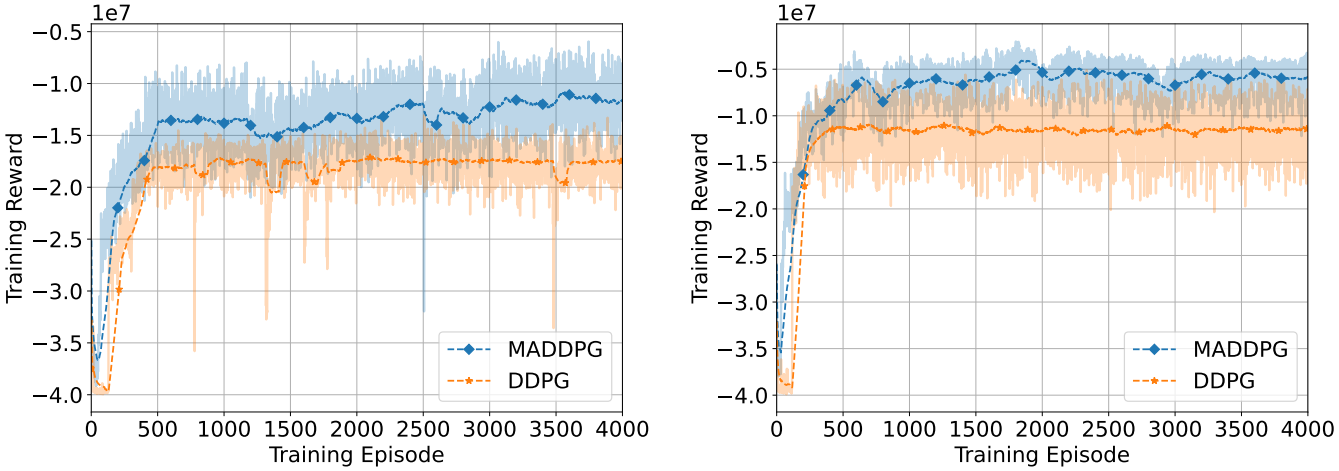
In this part, we assess the proposed MADDPG framework performance compared with other benchmarks in two scenarios, i.e., *3ACPs-env* and *5ACPs-env*.

We evaluate the system performance in the *3ACPs-env* scenario by changing the task size from 0.8 to 1.2 Mbits, as shown in Fig. 5. Increasing the task size leads to a rise in system cost due to the increased resource demands (Fig. 5a). In particular, the cost in our proposed approach grows by about 66.2% when increasing the size by 0.1 Mbits. In addition, a larger task size makes the environment more challenging, reducing the task success rate (Fig. 5b). Our MADDPG framework had a high task success rate of approximately 95% in the case of 0.8 Mbits and about 70% in the case of 1.2 Mbits. Similarly, the system energy consumption increases with the rise in task size as the number of bits to execute and offload increases (Fig. 5c). Our proposed framework demonstrates superior performance compared to other approaches. Notably, the episode cost is reduced by approximately 30% compared



(a) Learning rate. (b) Batch size. (c) Discount factor.

Fig. 3. Model convergence concerning training hyperparameters.



(a) Learning rate. (b) Batch size.

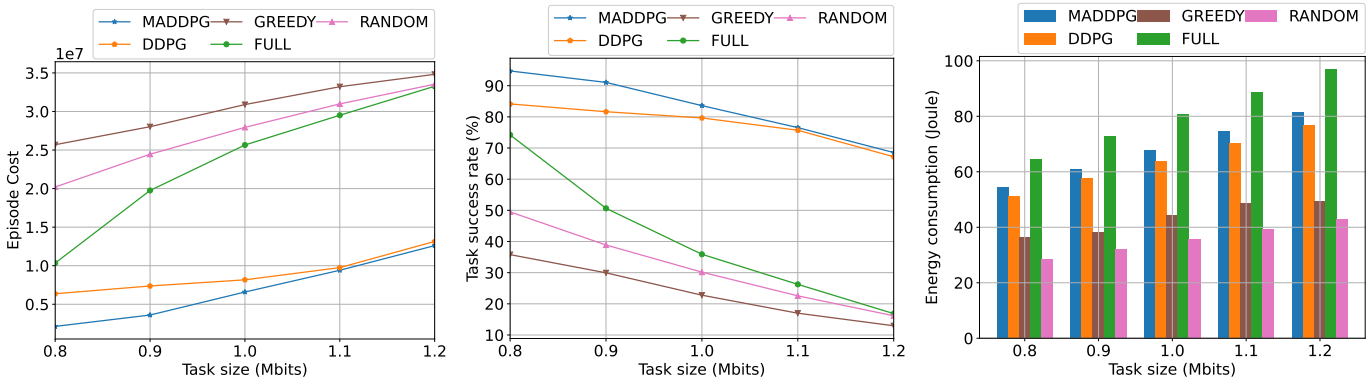
Fig. 4. Convergence comparison between MADDPG and single-agent DDPG.

to DDPG and by factors of 2.4, 2.99, and 3.44 compared to FULL, RANDOM, and GREEDY schemes, respectively. Moreover, in a highly complex problem, using GREEDY with discretized variables resulted in poor performance. Discrete spaces can contain many sub-optimal points, and the greedy-based searching approach can get stuck at a sub-optimal point, leading to possibly low performance.

We also assess the system performance in the *5ACPs-env* scenario with the varying task size, as shown in Fig. 6. Similar to the previous scenario, increasing the task size decreases the system performance (i.e., increases the episode cost, reduces the task success rate, and increases the energy consumption). Our proposed MADDPG framework outperforms the remaining benchmarks in this scenario with the highest task success rate and lowest episode cost. The *5ACPs-env* scenario also performs better than the *3ACPs-env* scenario, with an overall reduction in episode cost of about 1.28, 1.48, and 1.59 times in the MADDPG, DDPG, and FULL schemes, respectively, and approximate reductions of 11% and 9% in the RANDOM and GREEDY schemes, respectively. This is because more ACPs provide more computing and communication resources, increasing the system's capacity. In particular, the task success rate in this scenario is about 79.4% in the strictest case (i.e., task size 1.2 Mbits), while it is about 70% in the *3ACPs-env* scenario. Therefore, environmental requirements should be carefully considered when deploying a multi-ACP-enabled edge system to provide suitable resources and ensure system performance.

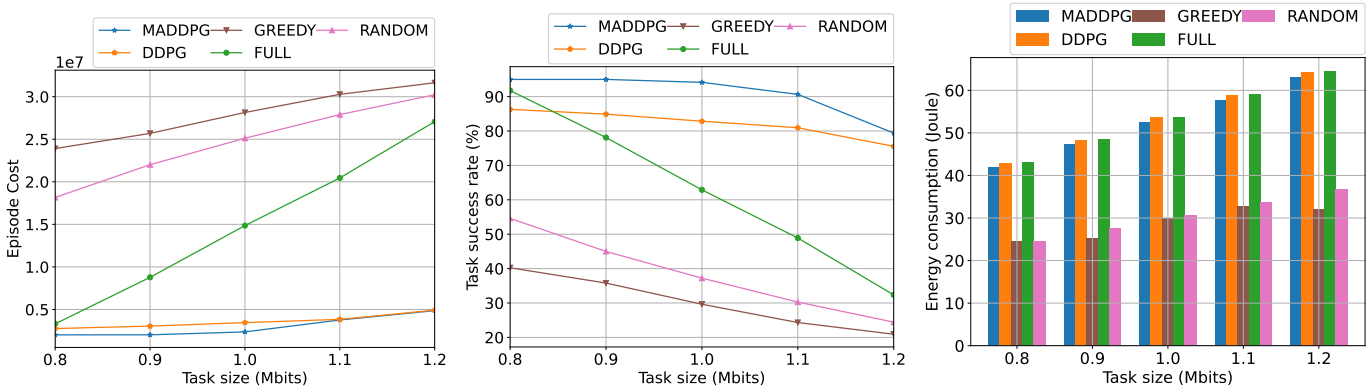
VI. Conclusion

This paper investigated a multi-ACP-enabled NOMA edge system with one HAP and several UAVs that assist in the computational tasks offloaded by IoTDs. Our goal was to minimize task processing latency and energy consumption of IoTDs by formulating a JTORA problem. This formulation considered device association, offloading ratio, transmit power, and computational resource allocation. The problem is transformed into the MAMDP, which is then addressed using the proposed MADDPG-based framework with the action pretreatment function. Our numerical results show that, compared to single-agent DDPG and other benchmark schemes, the proposed MADDPG-based approach can determine the efficient offloading and resource allocation policies while significantly reducing task computation overhead. In future works, we will



(a) Episode cost. (b) Task success rate. (c) Energy consumption.

Fig. 5. System performance with respect to task sizes in *3ACPs-env*.



(a) Episode cost. (b) Task success rate. (c) Energy consumption.

Fig. 6. System performance with respect to task sizes in *5ACPs-env*.

delve into alternative prominent multi-agent DRL optimization algorithms and other emerging communication technologies.

REFERENCES

[1] W. Chen, X. Lin, J. Lee, A. Toskala, S. Sun, C. F. Chiasserini, and L. Liu, "5G-Advanced toward 6G: Past, present, and future," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 6, pp. 1592–1619, jun 2023.

[2] V. D. Tuong, T. P. Truong, T.-V. Nguyen, W. Noh, and S. Cho, "Partial computation offloading in NOMA-assisted mobile-edge computing systems using deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 8, no. 17, pp. 13 196–13 208, 2021.

[3] C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu, and L. Guo, "Computation offloading in mobile edge computing networks: A survey," *Journal of Network and Computer Applications*, vol. 202, p. 103366, 2022.

[4] N.-N. Dao, Q.-V. Pham, N. H. Tu, T. T. Thanh, V. N. Q. Bao, D. S. Lakew, and S. Cho, "Survey on aerial radio access networks: Toward a comprehensive 6G access infrastructure," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1193–1225, 2021.

[5] M. M. Azari, S. Solanki, S. Chatzinotas, O. Kodheli, H. Sallouha, A. Colpaert, J. F. M. Montoya, S. Pollin, A. Haqiqatnejad, A. Mostaani, E. Lagunas, and B. Ottersten, "Evolution of non-terrestrial networks from 5G to 6G: A survey," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2633–2672, 2022.

[6] X. Lin, S. Rommer, S. Euler, E. A. Yavuz, and R. S. Karlsson, "5G from space: An overview of 3GPP non-terrestrial networks," *IEEE Communications Standards Magazine*, vol. 5, no. 4, pp. 147–153, 2021.

[7] X. Lin, "An overview of 5G advanced evolution in 3GPP release 18," *IEEE Communications Standards Magazine*, vol. 6, no. 3, pp. 77–83, 2022.

[8] Q. Zhang, Y. Luo, H. Jiang, and K. Zhang, "Aerial edge computing: A survey," *IEEE Internet of Things Journal*, vol. 10, no. 16, pp. 14 357–14 374, 2023.

[9] Q.-V. Pham, R. Ruby, F. Fang, D. C. Nguyen, Z. Yang, M. Le, Z. Ding, and W.-J. Hwang, "Aerial computing: A new computing paradigm, applications, and challenges," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8339–8363, 2022.

[10] S. Jung, C. Park, M. Levorato, J.-H. Kim, and J. Kim, "Two-stage self-adaptive task outsourcing decision making for edge-assisted multi-uav networks," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 11, pp. 14 889–14 905, 2023.

[11] M. Dai, Y. Wu, L. Qian, Z. Su, B. Lin, and N. Chen, "UAV-assisted multi-access computation offloading via hybrid NOMA and FDMA in marine networks," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 1, pp. 113–127, Jan. 2023.

[12] P. Qin, X. Wu, Z. Cai, X. Zhao, Y. Fu, M. Wang, and S. Geng, "Joint trajectory plan and resource allocation for uav-enabled c-noma in air-ground integrated 6g heterogeneous network," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 6, pp. 3421–3434, 2023.

[13] 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, 2023.

[14] 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*, vol. 10, no. 12, pp. 10 497–10 509, 2023.

[15] D. S. Lakew, A.-T. Tran, N.-N. Dao, and S. Cho, "Intelligent offloading and resource allocation in heterogeneous aerial access IoT networks," *IEEE Internet of Things Journal*, vol. 10, no. 7, pp. 5704–5718, 2023.

[16] Y. Yuan, Z. Yuan, and L. Tian, "5G non-orthogonal multiple access study in 3GPP," *IEEE Communications Magazine*, vol. 58, no. 7, pp. 90–96, 2020.

[17] Y. Liu, S. Zhang, X. Mu, Z. Ding, R. Schober, N. Al-Dhahir, E. Hossain, and X. Shen, "Evolution of NOMA toward next generation multiple access (NGMA) for 6G," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 4, pp. 1037–1071, 2022.

[18] Y. Zhang, Z. Mou, F. Gao, J. Jiang, R. Ding, and Z. Han, "UAV-enabled secure communications by multi-agent deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 11 599–11 611, 2020.

[19] H. Peng and X. Shen, "Multi-agent reinforcement learning based resource management in MEC- and UAV-assisted vehicular networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 1, pp. 131–141, 2021.

[20] N. Zhao, Z. Ye, Y. Pei, Y.-C. Liang, and D. Niyato, "Multi-agent deep reinforcement learning for task offloading in UAV-assisted mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 6949–6960, 2022.

[21] Z. Cheng, M. Liwang, N. Chen, L. Huang, X. Du, and M. Guizani, "Deep reinforcement learning-based joint task and energy offloading in UAV-aided 6G intelligent edge networks," *Computer Communications*, vol. 192, pp. 234–244, 2022.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[23] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016.

[24] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6382–6393.

[25] X. Cheng, F. Lyu, W. Quan, C. Zhou, H. He, W. Shi, and X. Shen, "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1117–1129, 2019.

[26] Q. Ren, O. Abbasi, G. K. Kurt, H. Yanikomeroglu, and J. Chen, "Caching and computation offloading in high altitude platform station (HAPS) assisted intelligent transportation systems," *IEEE Transactions on Wireless Communications*, vol. 21, no. 11, pp. 9010–9024, 2022.

[27] N. Waqar, S. A. Hassan, A. Mahmood, K. Dev, D.-T. Do, and M. Gidlund, "Computation offloading and resource allocation in MEC-enabled integrated aerial-terrestrial vehicular networks: A reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 21 478–21 491, 2022.

[28] T.-H. Nguyen, T. P. Truong, N.-N. Dao, W. Na, H. Park, and L. Park, "Deep reinforcement learning-based partial task offloading in high altitude platform-aided vehicular networks," in *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2022.

[29] T.-H. Nguyen and L. Park, "HAP-assisted RSMA-enabled vehicular edge computing: A DRL-based optimization framework," *Mathematics*, vol. 11, no. 10, p. 2376, 2023.

[30] T. P. Truong, N.-N. Dao, and S. Cho, "HAMEC-RSMA: Enhanced aerial computing systems with rate splitting multiple access," *IEEE Access*, vol. 10, pp. 52 398–52 409, 2022.

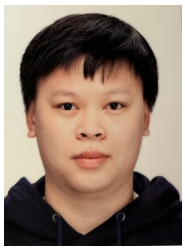
[31] G. K. Kurt, M. G. Khoshkholgh, S. Alfattani, A. Ibrahim, T. S. J. Darwish, M. S. Alam, H. Yanikomeroglu, and A. Yongacoglu, "A vision and framework for the high altitude platform station (HAPS) networks of the future," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 729–779, 2021.

[32] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1930–1941, 2019.

[33] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press: Cambridge, MA, USA, 2016, http://www.deeplearningbook.org.

Tri-Hai Nguyen is a Research Professor at the Department of Computer Science and Engineering, Seoul National University of Science and Technology (SeoulTech), Seoul, Korea. He received his Ph.D. degree in Computer Science and Engineering from Chung-Ang University, Seoul, Korea, in 2022; his M.Eng. degree in Information and Communication Technology from Soongsil University, Seoul, Korea, in 2017; and his B.S. degree (honor program) in Computer Science from the University of Information Technology, VNU-HCM, Vietnam, in 2015. His research interests include Intelligence of Things, aerial computing, and beyond 5G/6G networks.

Thanh Phung Truong received his B.S. degree in Electronics-Telecommunications Engineering from Ho Chi Minh City University of Technology, Vietnam, in 2018, and M.S. degree in Computer Science and Engineering from Chung-Ang University, Korea, in 2022. He is pursuing his Ph.D. in Computer Science and Engineering at Chung-Ang University, Korea. His research interests include machine learning, mobile edge computing, and wireless communication.



Anh-Tien Tran received his B.S. degree in Electronics and Telecommunications from Da Nang University of Science and Technology, Da Nang, Vietnam, in 2018. He is pursuing his Ph.D. in Computer Science and Engineering at Chung-Ang University, Seoul, Korea. His research interests include wireless network communication, video streaming, and machine learning.

Nhu-Ngoc Dao is an Assistant Professor at the Department of Computer Science and Engineering, Sejong University, Seoul, Republic of Korea. He received his M.S. and Ph.D. degrees in Computer Science at the School of Computer Science and Engineering, Chung-Ang University, Seoul, Republic of Korea, in 2016 and 2019, respectively. He received his B.S. degree in Electronics and Telecommunications from the Posts and Telecommunications Institute of Technology, Hanoi, Vietnam, in 2009. Prior to joining Sejong University, he was a visiting researcher at the University of Newcastle, NSW, Australia, in 2019 and a postdoc researcher at the Institute of Computer Science, University of Bern, Switzerland, from 2019 to 2020. He is currently an Editor of the *Scientific Reports* and *PLOS ONE* journals. His research interests include network softwarization, mobile cloudization, intelligent systems, and the Intelligence of Things. Dr. Dao is a Senior Member of IEEE and a Professional Member of ACM.

Laihyuk Park received the B.S., M.S., and Ph.D. degrees in computer science and engineering from Chung-Ang University, Seoul, Korea, in 2008, 2010, and 2017, respectively. From 2011 to 2016, he was a Research Engineer with Innowireless, Bundang, Korea. From 2018 to 2019, he held the position of Assistant Professor at Chung-Ang University. He is an Assistant Professor at the Department of Computer Science and Engineering, Seoul National University of Science and Technology (SeoulTech), Seoul, Korea. His research interests include demand response, smart grids, and the Internet of Things.

Sungrae Cho received the B.S. and M.S. degrees in Electronics Engineering from Korea University, Seoul, Korea, in 1992 and 1994, respectively, and the Ph.D. degree in Electrical and Computer Engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2002. He is a Professor at the School of Computer Science and Engineering, Chung-Ang University (CAU), Seoul, Korea. Prior to joining CAU, he was an Assistant Professor with the Department of Computer Sciences, Georgia Southern University, Statesboro, GA, USA, from 2003 to 2006, and a Senior Member of Technical Staff with the Samsung Advanced Institute of Technology (SAIT), Kiheung, Korea, in 2003. From 1994 to 1996, he was a Research Staff Member with the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea. From 2012 to 2013, he held a Visiting Professorship with the National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA. His research interests include wireless networking, ubiquitous computing, and ICT convergence. He has been a Subject Editor of *Electronics Letter* (IET) since 2018 and was an Area Editor of *Ad Hoc Networks* (Elsevier) from 2012 to 2017. He has served numerous international conferences as an Organizing Committee Chair, such as IEEE SECON, ICOIN, ICTC, ICUFN, TridentCom, and IEEE MASS, and as a Program Committee Member, such as IEEE ICC, GLOBECOM, VTC, MobiApps, SENSORNETS, and WINSYS.