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

Digital Object Identifier 10.1109/ACCESS.2022.DOI

HAMEC-RSMA: Enhanced Aerial Computing Systems with Rate Splitting Multiple Access

THANH PHUNG TRUONG[1], **NHU-NGOC DAO**[2], **AND SUNGRAE CHO.**[3]

[1]School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, South Korea (e-mail: tptruong@uclab.re.kr)
[2]Department of Computer Science and Engineering, Sejong University, Seoul 05006, South Korea (e-mail: nndao@sejong.ac.kr)
[3]School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, South Korea (e-mail: srcho@cau.ac.kr)

Corresponding author: Sungrae Cho (e-mail: srcho@cau.ac.kr).

This research was supported by the Chung-Ang University Young Scientist Scholarship in 2020.

ABSTRACT Aerial networks have been widely considered a crucial component for ubiquitous coverage in the next-generation mobile networks. In this scenario, mobile edge computing (MEC) and rate splitting multiple access (RSMA) are potential technologies, which are enabled at aerial platforms for computation and communication enhancements, respectively. Motivated from this vision, we proposed a high altitude platform-mounted MEC (HAMEC) system in such an RSMA environment, where aerial users (e.g., unmanned aerial vehicles) can efficiently offload their tasks to the HAMEC for external computing acquisition. To this end, a joint configuration of key parameters in HAMEC and RSMA (referred to as HAMEC-RSMA) such as offloading decision, splitting ratio, transmit power, and decoding order was optimally designed for a processing cost minimization in terms of response latency and energy consumption. Subsequently, the optimization problem was transformed into a reinforcement learning model, which is solvable using the deep deterministic policy gradient (DDPG) method. To improve the training exploration of the algorithm, we employed parameter noises to the DDPG algorithm to enhance training performance. Simulation results demonstrated the efficiency of the HAMEC-RSMA system with superior performances compared to benchmark schemes.

INDEX TERMS 6G, rate splitting multiple access, edge computing, unmanned aerial vehicle, deep reinforcement learning.

I. INTRODUCTION

Mobile network evolution has brought the Internet of things (IoT) toward a new revolution in diverse application scenarios, where spatial limits, geographical location, the microworld, and the biological environment can be efficiently supported by sustainable access infrastructure [1]. To support an ubiquitous coverage, aerial radio access networks (ARANs) appear to be a potential strategy for enhancing existing terrestrial communication infrastructures, which is able to provide services in underserved areas [2], [3]. Particularly, owing to the high mobility, coverage capacity, and the ability to reach places inaccessible to humans, airborne platforms such as aircraft and unmanned aerial vehicles (UAVs) can be successfully exploited in a variety of professional applications, including agriculture, mission-critical services, search and rescue missions, and surveillance systems [4], [5].

To assist the aforementioned scenarios, mobile cloudification paradigms have changed considerably in tandem with the rise of IoT, particularly in mobile edge computing (MEC). MEC enhances the cloud computing effectively by bringing cloud services to the network edge, where they are available nearby and have low latency to users [6]. Consequently, an integration of MEC into aerial networks consisting of high altitude platforms (HAPs) and UAVs provides additional resources that significantly enhance system capabilities and performances [7]. Due to unique characteristics of the aerial networks, airborne platforms face several challenges such as battery limitations and flying formation to work stably on the air. In this study, we considered a high altitude platform-mounted MEC (HAMEC) system, where an edge computing server equipped at a HAP to enhance the performance of an aerial network. In such a HAMEC system, aerial users (AUs) flying in a 3D space coverage harvest information from terrestrial devices to assist professional applications in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3173125, IEEE Access

agriculture, surveillance, etc. In these scenarios, AUs can partially offload their tasks to the HAMEC with high computation resources for processing to reduce user costs in the network.

On the other hand, to increase the robustness in task transmission, rate splitting multiple access (RSMA) techniques are assumed for communication between AUs and the HAMEC. RSMA is a powerful emerging multiplexing strategy for spectral efficiency improvement by integrating space-division multiple access and non-orthogonal multiple access techniques [8]. Regarding our investigated scenarios, where uplink RSMA is exploited to transmit information from AUs to HAMEC, each transmit signal is split into multiple sub-signals, and the successive interference cancellation (SIC) technique is applied at the receiver to decode all the sub-signals. To the best of our knowledge, a combination of the HAMEC model and the RSMA technique is still an open research direction, where multiple system parameter configuration such as offloading decision, splitting ratio, transmit power, and decoding order should be optimized for a processing cost minimization in terms of response latency and energy consumption. Motivated by this observation, our paper dedicates to resolve the mentioned problem. Accordingly, the main contributions of this research are as follows:

- First, we investigated a HAMEC system in a uplink RSMA-enabled aerial network, where the AUs serve in a 3D space coverage. In this system model, we proposed an optimization problem of offloading decision, splitting ratio, transmit power, and decoding order to minimize the processing costs in terms of total latency and energy consumption.
- Second, we transformed the problem into a reinforcement learning model that can be solved by applying the deep deterministic policy gradient (DDPG) algorithm, referred to as `HAMEC-RSMA`. To cope with that the action space noise for training exploration in DDPG may violate the variable value range constraints in the optimization problem, we added parameter noises to the DDPG algorithm during training to ensure that the exploration meets all the problem constraints.
- Third, we simulated the environment scenario and trained the model. Numerical results demonstrated the efficiency of the HAMEC system in a uplink RSMA-enabled aerial network, where the proposed `HAMEC-RSMA` algorithm outperforms existing benchmark schemes.

The rest of this research is organized as follows. We summarized some related works in Section I. Then, we introduced the system model and formulated the optimization problem in Section II and Section III, respectively. Next, we proposed the `HAMEC-RSMA` algorithm in Section IV. Simulation results were presented in Section V. Finally, we concluded the study in Section VI.

RELATED WORK

Mobile edge computing (MEC) has been recently studied in a vast number of studies due to the rapid expansion of the mobile network. Especially, minimizing the cost of task processing is typically an attractive challenge [9]–[14]. For instance, the authors in [9] proposed a task offloading model in an industrial IoT (IIoT) scenario, where they optimized the user equipment's resource allocation and binary offloading decisions to reduce the system cost function. Then, they designed a reinforcement learning model that apply Q-learning algorithm to address the problem. The study in [10] aimed to minimize the total task delay in a multi-MEC system by optimizing the binary offloading decision, resource allocation, and cooperative mode selection. The authors presented an iterative technique based on Lagrangian dual decomposition, the monotonic optimization method, and the ShengJin Formula method to address the problem. In [11], a stabilized green crosshaul orchestration framework is proposed utilizing a Lyapunov-theory-based drift-plus-penalty policy to optimize the offloaded data for an energy consumption minimization problem. In addition, the integration of the UAV and MEC has also been considered in [13]. The authors introduced a UAV-assisted MEC system intending to minimize the total cost of IoT devices. They devised the AA-CAP algorithm to optimize binary computation offloading, computation resource allocation, spectrum resource allocation, and UAV placement for this purpose. The authors in [14] investigated a scenario where the UAV established wireless communication between the mobile users (MUs) and edge clouds. Based on the successive convex approximation method, they developed an algorithm to optimize the UAV placement, communication and computing resource allocation, and task partition variables for minimizing the cost in terms of total service delay and MUs energy consumption.

Along with MEC, multiple access has recently been an attractive research topic. The non-orthogonal multiple access (NOMA) technique was examined to improve the system performance in many scenarios, such as ultra-reliable low-latency communications systems [15], and blind signal classification and detection systems [16]. In terms of RSMA, several recent studies have looked into various problems related to the adoption of RSMA in wireless networks, which considered both downlink [17]–[19] and uplink [20], [21] transmissions. For instance, the work in [17] considered two schemes of multiple access: NOMA and RSMA. The authors examined the energy efficiency in a millimeter-wave downlink system, and the results showed the outperformance of the RSMA to NOMA in this scenario. In [18], RSMA was applied in a downlink network to maximize user sum rates by optimizing rate allocation and power control at the base station (BS). The authors in [19] considered the combination of RSMA and UAV networks. They present the integration of RSMA with a UAV-BS downlink transmission scenario, where a low-complexity iterative algorithm was proposed to optimize the UAV placement, RSMA precoding, and rate splitting decision to maximize the weighted sum rate of users.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2022.3173125, IEEE Access

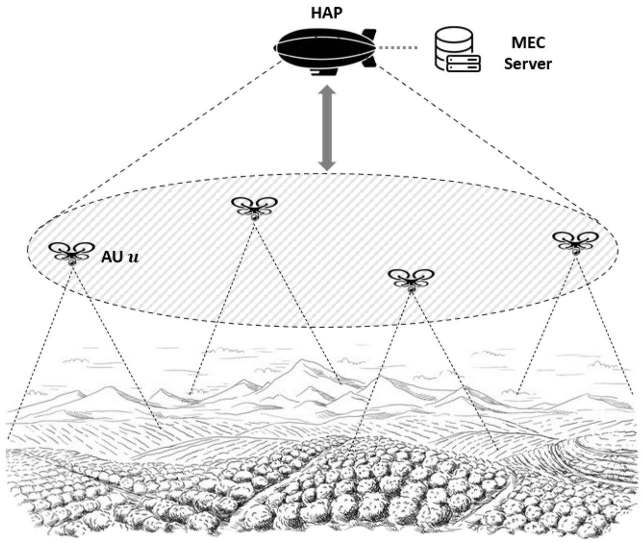


FIGURE 1: HAMEC-enhanced aerial networking scenario.

Different from downlink, only a few research has focused on RSMA in uplink systems. The paper [20] investigated the performance of a two-source uplink RSMA network in terms of outage probability and throughput. In [21], the authors considered the uplink of a wireless network using RSMA. They aim to maximize the user sum-rates by optimizing the decoding order and power allocation at users, which is solved by the difference of two convex function method and an exhaustive search method.

Although the number of studies on MEC and RSMA has grown over the past years, to the best of our knowledge, an integration of RSMA and MEC remains an open and attractive issue. Hence, this observation motivates us to conduct this study.

II. SYSTEM MODEL

A. NETWORK SCENARIO

As illustrated in Fig. 1, we consider a HAMEC system in RSMA-enabled aerial networks consisting of an AU set denoted by $\mathcal{U} = \{1, ..., u, ..., U\}$ and its cardinality U. In this model, AUs fly in a 3D space coverage to serve a certain terrestrial area, where the terrestrial base station is unavailable. To improve the quality of services, a HAP equipped with a MEC server hovers in the air covering all the AUs.

In this study, the time is divided into discrete time slots. At each time slot t, each AU u is assumed to have a computation task to execute, denoted as $\tau_u[t] = (w_u[t], c_u[t])$, where $w_u[t]$ and $c_u[t]$ are the size and required computation resource in bits of the task, respectively. Each AU can process its task locally or offload the task to HAMEC for processing and sending back the task result. To ensure generality, a partial offloading scheme is investigated for each AU u at time slot t, where the offloading rate is denoted as $o_u[t] \in [0, 1]$. The rate determines that $o_u[t]$ percentage of task τ_u is offloaded to HAP and $(1 - o_u[t])$ part of task τ_u is processed locally.

B. COMMUNICATION MODEL

In this scenario, the light-of-sight (LoS) channel is considered for all communication links since the aerial network has almost no obstacles. Hence, we adopt free space path loss model for the channel gain between HAP and AU u, which is calculated as

$$g_u[t] = \frac{\beta_0}{d_u[t]^\alpha}, \tag{1}$$

where α is the path loss exponent, β_0 is the channel power gain at reference distance 1m, and $d_u[t]$ is the distance between HAP and AU u at time slot t. We denote $l_u[t] = (x_u[t], y_u[t], z_u[t])$, and $l_h = (x_h, y_h, z_h)$ are the location of the AU u at time slot t, and fixed location of HAP, respectively. Then, the distance $d_u[t]$ is calculated as

$$d_u[t] = \sqrt{(x_h - x_u[t])^2 + (y_h - y_u[t])^2 + (z_h - z_u[t])^2} \tag{2}$$

To enhance the spectrum efficiency, RSMA technique is assumed for communications between HAP and the AUs. In this scheme, the transmitted signal $s_u[t]$ of each AU u at time slot t is split into K sub-signals. Without loss of generality, we choose $K = 2$ as in [21], and denote $\mathcal{K}_u[t] = \{s_{u1}[t], s_{u2}[t]\}$ is the set of sub-signals of $s_u[t]$, the transmitted signal is then expressed as

$$s_u[t] = \sum_{k=1}^{2} \sqrt{p_{uk}[t]} s_{uk}[t], \tag{3}$$

where $p_{uk}[t] \geq 0$ is the transmit power of sub-signal $s_{uk}[t]$ at time slot t. Then, the total offloaded signal received at HAP is represented as

$$\begin{aligned} s_h[t] &= \sum_{u=1}^{U} \sqrt{g_u[t]} s_u[t] + n_0 \\ &= \sum_{u=1}^{U} \sum_{u=1}^{2} \sqrt{g_u[t] p_{uk}[t]} s_{uk}[t] + n_0, \end{aligned} \tag{4}$$

where n_0 is the additive white Gaussian noise with power spectral density σ^2.

As RSMA, we denote $\Delta_u^o[t] = \{\delta_{u1}^o[t], \delta_{u2}^o[t]\}$ as the splitting ratio set with two variables of the offloaded part of task $\tau_u[t]$. In case that the offloading rate is zero, the splitting ratio variables should be zero due to no offloaded data. Then, the splitting ratio should follow the constraints as

$$\begin{aligned} &\delta_{uk}^o[t] \in [0, \lceil o_u[t] \rceil] , \\ &\sum_{k=1}^{2} \delta_{uk}^o[t] = \lceil o_u[t] \rceil, \end{aligned} \tag{5}$$

where $\delta_{uk}^o[t]$ is the ratio of the k sub-offloaded task of AU u at time slot t, and $\lceil o_u[t] \rceil$ is the ceiling function of $o_u[t]$, which is equal to 1 when offloading and equal to 0 in otherwise.

Then, each sub-offloaded task k of each AU u is denoted as $\tau_{uk}^o[t] = (w_{uk}^o[t], c_u[t])$, where $w_{uk}^o[t] = \delta_{uk}^o[t] o_u[t] w_u[t]$ is the size in bits of the sub-offloaded task k. When $\delta^o_{uk}[t] > 0$, the corresponding power $p_{uk}[t]$ also needs to be greater than 0 to ensure all split parts are offloaded to HAP, otherwise, the transmit power is set to zero because of no offloaded data. Thus, we define a RSMA offloading power constraint as

$$\begin{cases} p_{uk}[t] > 0 & \text{, if } \delta^o_{uk}[t] > 0, \\ p_{uk}[t] = 0 & \text{, otherwise.} \end{cases} \tag{6}$$

At HAP, the successive interference cancellation (SIC) technique is applied for decoding all sub-signals $s_{uk}[t]$ from the received signal $s_h[t]$. We denote $\Phi[t] = \{\phi_{11}[t], ..., \phi_{uk}[t], ..., \phi_{U2}[t]\}$ is the set of decoding order of all sub-signals, where $\phi_{uk}[t] \in \{1, 2, ..., 2U\}$ is the decoding order of sub-signal k of AU u at time slot t. Then, according to the ascending order, the uplink rate of sub-signal $s_{uk}[t]$ on RSMA can be calculated as

$$r_{uk}[t] = \mathcal{B}\log_2\left(1 + \frac{g_u[t]p_{uk}[t]}{\sum_{\phi_{vj}[t]>\phi_{uk}[t]} g_v[t]p_{vj}[t] + \sigma^2}\right), \tag{7}$$

where $\mathcal{B}$ is the communication bandwidth of HAP, and $\phi_{vj}[t]$ is the decoding order of sub-signal j of AU v at time slot t.

C. COMPUTATION MODEL

For the task $\tau_u[t]$ with offloading rate $o_u[t]$, the local execution time at the AU u is calculated as

$$T^l_u[t] = \frac{(1 - o_u[t])w_u[t]c_u[t]}{f^l_u}, \tag{8}$$

and the execution time at the HAP is calculated as

$$T^c_u[t] = \frac{o_u[t]w_u[t]c_u[t]}{f^c_{hu}}, \tag{9}$$

where f^l_u and f^c_{hu} are the computation resource of AU u and the computation resource that HAP allocates to AU u, respectively. The local execution energy consumption of AU u is then calculated as [22]

$$E^l_u[t] = (1 - o_u[t])w_u[t]\kappa_u \left(f^l_u\right)^2, \tag{10}$$

where κ_u is the energy coefficient of the AU u depended on the hardware architecture.

D. OFFLOADING MODEL

When offloading a sub-task, it must be uploaded to HAP before execution. The time for uploading sub-offloaded task $\tau^o_{uk}[t]$ is calculated as

$$T^{\text{up}}_{uk}[t] = \frac{w^o_{uk}[t]}{r_{uk}[t]}, \tag{11}$$

and the energy consumption of AU u for uploading sub-offloaded task $\tau^o_{uk}[t]$ is calculated as

$$E^{\text{up}}_{uk}[t] = p_{uk}[t]T^{\text{up}}_{uk}[t] = \frac{p_{uk}[t]w^o_{uk}[t]}{r_{uk}[t]}. \tag{12}$$

Consequently, the total time for uploading the offloaded part of task $\tau_u[t]$ is calculated as

$$\begin{aligned} T^{\text{up}}_u[t] &= \sum_{k=1}^{2} T^{\text{up}}_{uk}[t] = \sum_{k=1}^{2} \frac{w^o_{uk}[t]}{r_{uk}[t]} \\ &= \sum_{k=1}^{2} \frac{\delta^o_{uk}[t]o_u[t]w_u[t]}{r_{uk}[t]}. \end{aligned} \tag{13}$$

The computation tasks in this study are considered with small size results enough that the downloading results time can be neglected. Accordingly, the total latency for processing the offloaded part of task $\tau_u[t]$ consists of uploading time $T^{\text{up}}_u[t]$ and execution time $T^c_u[t]$, which can be calculated as

$$\begin{aligned} T^{\text{off}}_u[t] &= T^{\text{up}}_u[t] + T^c_u[t] \\ &= \sum_{k=1}^{2} \frac{\delta^o_{uk}[t]o_u[t]w_u[t]}{r_{uk}[t]} + \frac{o_u[t]w_u[t]c_u[t]}{f^c_{hu}} \\ &= o_u[t]w_u[t]\left(\frac{c_u[t]}{f^c_{hu}} + \sum_{k=1}^{2} \frac{\delta^o_{uk}[t]}{r_{uk}[t]}\right), \end{aligned} \tag{14}$$

and the energy consumption of AU u for completing the offloaded part of task $\tau_u[t]$ is the energy consumption for uploading, which is calculated as

$$\begin{aligned} E^{\text{off}}_u[t] &= \sum_{k=1}^{2} E^{\text{up}}_{uk}[t] = \sum_{k=1}^{2} \frac{p_{uk}[t]w^o_{uk}[t]}{r_{uk}[t]} \\ &= \sum_{k=1}^{2} \frac{p_{uk}[t]\delta^o_{uk}[t]o_u[t]w_u[t]}{r_{uk}[t]}. \end{aligned} \tag{15}$$

III. PROBLEM FORMULATION

A. COST FUNCTION

According to the partial offloading scheme, the total task latency of task $\tau_u[t]$, $T_u[t]$, is determined as the longest time between the local execution time and offloaded time, which is provided as

$$T_u[t] = \max\left(T^l_u[t], T^{\text{off}}_u[t]\right), \tag{16}$$

and the energy consumption $E_u[t]$ of AU u for processing the task $\tau_u[t]$ is computed as the total energy consumption for local computing and offloading, which is provided as

$$E_u[t] = E^l_u[t] + E^{\text{off}}_u[t]. \tag{17}$$

In addition, we examine a task processing cost function of all AUs in terms of energy consumption and the total latency, which is calculated by the sum of energy consumption of all AUs and the total task latency with weight parameters. The cost function at each time slot t is given as

$$\mathcal{C}[t] = \sum_{u\in\mathcal{U}} \left(\eta_e E_u[t] + \eta_t T_u[t]\right), \tag{18}$$

where η_e, $\eta_t \in [0, 1]$ are the weight parameter of the energy consumption and latency, respectively, which satisfy $\eta_t + \eta_e = 1$.

B. PROBLEM FORMULATION

The goal of this study is to minimize the cost function by optimizing the offloading rates, transmission powers, and splitting ratios of AUs, as well as the decoding order at HAP. We denote $p_u[t] = \{p_{u1}[t], p_{u2}[t]\}$ is the set of transmit power of each AU u, the optimization problem can be formulated as

$$(P1): \min_{\substack{\Phi[t] \\ \{o_u[t],\Delta_u^o[t],p_u[t]\},\, u\in\mathcal{U}}} \quad \mathcal{C}[t], \tag{19a}$$

$$\text{s.t.} \quad (6), \quad p_{uk}[t] \in p_u[t], u \in \mathcal{U}, \tag{19b}$$

$$T_u[t] \leq \zeta_t, \quad u \in \mathcal{U}, \tag{19c}$$

$$\phi_{uk}[t] \in \{1, 2, ..., 2U\}, \quad \phi_{uk}[t] \in \Phi[t], \tag{19d}$$

$$o_u[t] \in [0, 1], \quad u \in \mathcal{U}, \tag{19e}$$

$$\delta_{uk}^o[t] \in [0, \lceil o_u[t] \rceil], \quad \delta_{uk}^o[t] \in \Delta_u^o[t], u \in \mathcal{U}, \tag{19f}$$

$$\sum_{k=1}^{2} \delta_{uk}^o[t] = \lceil o_u[t] \rceil, \quad u \in \mathcal{U}, \tag{19g}$$

$$\sum_{k=1}^{2} p_{uk}[t] \leq P_u^{\max}, \quad u \in \mathcal{U}, \tag{19h}$$

where (19d) and (19e) are the value range constraints of the decoding order and the offloading rate, respectively; constraints (19f) and (19g) specify the value range of splitting ratio; (19b) and (19h) indicate the value constraints of the transmit power of each sub-signal, where $P_u^{\max}$ is the maximum power of each AU u, and the constraint (19b) is to make sure that all split parts are offloaded; the tasks at time slot t need to be completely executed before the next tasks to ensure system capacity, hence, we set a constraint in (19c), where ζ_t is the time slot duration.

The problem (19) is non-convex due to the combination of continuous variables and the decoding order. In addition, the dynamic environment yields a large number of possible model observations in real-time. Therefore, we design a reinforcement learning framework named `HAMEC-RSMA` for solving the problem, which trains the agent using the DDPG algorithm.

IV. PROPOSED DEEP REINFORCEMENT LEARNING FRAMEWORK

A. `HAMEC-RSMA` FRAMEWORK

Before transforming the problem to the RL model, we first change the decoding order variables to continuous variables so that all the actions have continuous values. We denote $\Pi[t] = \{\pi_{11}[t], ..., \pi_{uk}[t], ..., \pi_{U2}[t]\}$ is the set of decoding priority of all sub-signals, where $\pi_{uk}[t] \in [0, 1]$ is the decoding priority of sub-signal k of AU u at time slot t. Then, according to the ascending order of the decoding priority, the uplink rate of sub-signal $s_{uk}[t]$ is calculated as

$$r_{uk}[t] = \mathcal{B}\log_2\left(1 + \frac{g_u[t]p_{uk}[t]}{\sum_{\pi_{vj}[t]>\pi_{uk}[t]} g_v[t]p_{vj}[t] + \sigma^2}\right), \tag{20}$$

where $\pi_{vj}[t]$ is the decoding priority of sub-signal j of AU v at time slot t.

To simplify the optimization variables, we reform the splitting ratio set of offloaded task $\tau_u[t]$ to $\varepsilon_u^o[t]$, where $\varepsilon_u^o[t] \in [0, 1]$ denotes the splitting ratio variable of each task $\tau_u[t]$. As a result, the values the splitting ratios of each task $\tau_u[t]$ are calculated as

$$\begin{aligned} \delta_{u1}^o[t] &= \varepsilon_u^o[t]\lceil o_u[t] \rceil, \\ \delta_{u2}^o[t] &= (1 - \varepsilon_u^o[t])\lceil o_u[t] \rceil. \end{aligned} \tag{21}$$

Besides, with the transmission power, given the value of $p_{u1}[t]$, according to (19h), the value of $p_{u2}[t]$ must satisfy

$$p_{u2}[t] \leq P_u^{\max} - p_{u1}[t]. \tag{22}$$

We denote $\varepsilon_{u1}^p[t]$, $\varepsilon_{u2}^p[t] \in [0, 1]$ are the power variables of $p_{u1}[t]$, $p_{u2}[t]$, respectively. According to (22), the values of $p_{u1}[t]$ and $p_{u2}[t]$ are then calculated as

$$\begin{aligned} p_{u1}[t] &= \varepsilon_{u1}^p[t]P_u^{\max}, \\ p_{u2}[t] &= \varepsilon_{u2}^p[t](P_u^{\max} - p_{u1}[t]). \end{aligned} \tag{23}$$

Hence, according to the constraints (19b) and (19h), the sub-signal transmit powers $p_{u1}[t]$ and $p_{u2}[t]$ are calculated as

$$\begin{aligned} p_{u1}[t] &= \lceil \delta_{u1}^o[t] \rceil \varepsilon_{u1}^p[t]P_u^{\max}, \\ p_{u2}[t] &= \lceil \delta_{u2}^o[t] \rceil \varepsilon_{u2}^p[t](P_u^{\max} - p_{u1}[t]). \end{aligned} \tag{24}$$

In summary, the decoding order $\Phi[t]$ is changed to decoding priority $\Pi[t]$, the splitting ratio $\Delta_u^o[t]$ is reduced to $\varepsilon_u^o[t]$, and the transmission powers of each AU u at time slot t are changed to $\varepsilon_{u1}^p[t]$ and $\varepsilon_{u2}^p[t]$. Thus, the optimization problem is rewritten as

$$(P2): \min_{\left\{\substack{\pi_{u1}[t],\pi_{u2}[t],o_u[t], \\ \varepsilon_u^o[t],\varepsilon_{u1}^p[t],\varepsilon_{u2}^p[t]}\right\},\, u\in\mathcal{U}} \quad \mathcal{C}[t], \tag{25a}$$

$$\text{s.t.} \quad (19\text{b}), (19\text{c}). \tag{25b}$$

$$\pi_{uk}[t] \in [0, 1], \quad \pi_{uk}[t] \in \Pi[t], \tag{25c}$$

$$o_u[t], \varepsilon_u^o[t], \varepsilon_{u1}^p[t], \varepsilon_{u2}^p[t] \in [0, 1], \quad u \in \mathcal{U}, \tag{25d}$$

We transform the problem into an RL model, in which the agent is the HAP with high energy and computational resources, and the environment is the whole system. At each time slot t, based on the system state $s[t]$, the agent decides the action $a[t]$ interacting with the environment, then receives back the reward $r[t]$ and the next state $s[t+1]$. The state space, action space, and reward function are defined as follows.

1) State Space

The state space determines dynamic parameters of the environment that affect the reward of the RL model. In this environment, the state space consists of the location and the task information of all AUs, which is represented as

$$s[t] = \left\{ \begin{matrix} x_1[t], ..., x_u[t], ..., x_U[t], \\ y_1[t], ..., y_u[t], ..., y_U[t], \\ w_1[t], ..., w_u[t], ..., w_U[t], \\ c_1[t], ..., c_u[t], ..., c_U[t] \end{matrix} \right\}. \tag{26}$$

The number of location indexes, task sizes, and task required computation resources are U. Then, the total number of entries in state space is $4U$.

2) Action Space

The action space includes all the optimization variables, which are the decoding priorities, offloading rates, splitting variables, and transmit power variables. At each time slot t, the action space is represented as

$$a[t] = \left\{ \begin{matrix} \pi_{11}[t], ..., \pi_{u1}[t], ..., \pi_{U1}[t], \\ \pi_{12}[t], ..., \pi_{u2}[t], ..., \pi_{U2}[t], \\ o_1[t], ..., o_u[t], ..., o_U[t], \\ \varepsilon_1^o[t], ..., \varepsilon_u^o[t], ..., \varepsilon_U^o[t], \\ \varepsilon_{11}^p[t], ..., \varepsilon_{u1}^p[t], ..., \varepsilon_{U1}^p[t], \\ \varepsilon_{12}^p[t], ..., \varepsilon_{u2}^p[t], ..., \varepsilon_{U2}^p[t] \end{matrix} \right\}, \tag{27}$$

where all the constraints in (25) need to be satisfied. The number of decoding priorities, offloading rates, splitting variables, and transmit power variables are $2U$, U, U, and $2U$, respectively. Therefore, the total number of entries in action space is $6U$. All actions are designed to have a value range of $[0, 1]$, which satisfies the value ranges in (25).

3) Reward Function

The reward determines the effect of action $a[t]$ on the environment at state $s[t]$. Since this study aims to minimize the task processing cost function, a negative cost value is added to the reward function. In addition, if the constraint (25b) is violated, the task $\tau_u[t]$ will not be successfully executed. Thus, we add a penalty reward to penalize the action violating the constraints in (25b). In summary, the reward function at time slot t can be represented as

$$r[t] = \lambda[t]\nabla - (1 - \lambda[t])\mathcal{C}[t], \tag{28}$$

where ∇ and $\lambda[t]$ are the negative penalty value and the corresponding binary variable, which is given by

$$\lambda[t] = \begin{cases} 1 & \text{, if constraints (25b) are not satisfied,} \\ 0 & \text{, otherwise.} \end{cases} \tag{29}$$

B. HAMEC-RSMA ALGORITHM

1) Applied DDPG algorithm

The DDPG algorithm proposed in [23] is an actor-critic algorithm that handles the continuous action domain in reinforcement learning. It includes an actor-network, $\mu(s|\theta^{\mu})$ with the parameter θ^{μ}, defining the policy for the agent to decide the action a at each time step according to the observed state s from the environment. The algorithm also includes a critic-network, $Q(s, a|\theta^Q)$ with the parameter θ^Q, measuring each action a at each state s to determine how effective it is in this corresponding state. In updating the networks, the DDPG algorithm employs a target actor-network, $\mu'(s|\theta^{\mu'})$ with the parameter $\theta^{\mu'}$, and a target critic-network, $Q'(s, a|\theta^{Q'})$ with the parameter $\theta^{Q'}$, to improve model training stability.

At each training step, using batch learning, the parameter of the main critic-network is updated by minimizing the loss (L) between the action-value function, $Q(s_i, a_i|\theta^Q)$, and the target value y_i, which is calculated as

$$L = \frac{1}{B}\sum_{i=1}^{B}\left(Q(s_i, a_i|\theta^Q) - y_i\right)^2, \tag{30}$$

where B is the mini-batch size, s_i and a_i are the state and action of sample i in the mini-batch, respectively. The target value of sample i, y_i, is calculated as

$$y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}\right), \tag{31}$$

where r_i and s_{i+1} are the reward and next state of the sample i, respectively, and γ is the discount factor. Besides, the parameter of the main actor-network is updated using the policy gradient as

$$\nabla_{\theta^{\mu}} J = \frac{1}{B}\sum_{i=1}^{B}\left(\nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)}\nabla_{\theta^{\mu}}\mu(s_i|\theta^{\mu})\right). \tag{32}$$

Then, the target network parameters are updated using the soft update with a small constant, τ, which are calculated as

$$\begin{aligned} \theta^{\mu'} &\leftarrow \tau\theta^{\mu} + (1 - \tau)\theta^{\mu'}, \\ \theta^{Q'} &\leftarrow \tau\theta^{Q} + (1 - \tau)\theta^{Q'}. \end{aligned} \tag{33}$$

Owing to ensure exploration in the training samples, the DDPG adds noise samples to the actor policy to produce action interacting with the environment. This method can be called action space noise. In training, the produced action at time slot t can be given as

$$a[t] = \mu\left(s[t]|\theta^{\mu}\right) + \mathcal{N}[t], \tag{34}$$

where $\mathcal{N}[t]$ can be chosen from Ornstein-Uhlenbeck process.

Remark. The decided actions have to satisfy the value range of $[0, 1]$ as mentioned in subsection IV-A2. Therefore, the decided action from the policy have to be in the range of $[0, 1]$, which can be presented as

$$0 \leq \mu\left(s[t]|\theta^{\mu}\right) \leq 1, \ \forall s[t]. \tag{35}$$

However, the additional noise in the exploration in (34) may violate the action value ranges in (25). Specifically, we generate random action in the value range of [0,1] in $5e^6$ steps and perform the noise using the Ornstein-Uhlenbeck process to the action. The probability density functions (PDF) of the action and action with noise are illustrated in Fig. 2. Although the action varies from 0 to 1, the action plus noise can go out of the range and hit the new value range of (-1,2), which violate the constraints in (25).

Therefore, we employ another way to explore the training samples in this study. Instead of using the additional noise, we embed the parameter space noise to the DDPG algorithm for exploration as proposed in [24].

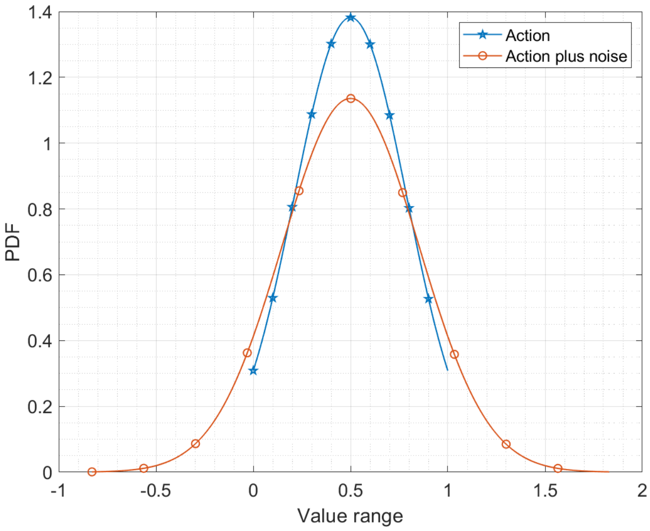


FIGURE 2: PDF of action with noise.

2) Parameter noise in the DDPG algorithm

This method explores the training samples by adding noise to the network parameters instead of the action space. The agent employs a perturbed actor-network, $\tilde{\mu}\left(s|\tilde{\theta^{\mu}}\right)$ with the parameter $\tilde{\theta}^{\mu}$, to decide action interacting with the environment and trains the non-perturbed actor-network, $\mu(s|\theta^{\mu})$. Then, the decided action is given as

$$a[t] = \tilde{\mu}\left(s[t]|\tilde{\theta}^{\mu}\right). \tag{36}$$

The parameter of the perturbed actor-network is achieved by applying additive Gaussian noise to the non-perturbed actor-network, which can be given as

$$\tilde{\theta}^{\mu} = \theta^{\mu} + \mathcal{N}(0,1), \tag{37}$$

where $\mathcal{N}(0,1)$ is the additive Gaussian noise with mean 0 and variance 1.

Algorithm 1: HAMEC−RSMA algorithm

```
Set up the environment and model parameters.
Initialize the neural network parameters $\theta^{\mu}, \tilde{\theta^{\mu}}, \theta^{Q}, \theta^{Q'}, \theta^{\mu'}$.
Set the number of episodes $E$ and number of steps in episode.
for e = 1,...,$E$ do
    Perturb the actor-network parameter as (40).
    Observe initial state $s[t]$ of the episode as (26).
    while in episode do
        Interacting:
            Decide action $a[t] = \tilde{\mu}\left(s[t]|\tilde{\theta}^{\mu}\right)$.
            Perform $a[t]$ and get $r[t]$, $s[t+1]$.
            Store the tuple $< s[t], a[t], r[t], s[t+1] >$ into buffer.
            Update next state $s[t] \leftarrow s[t+1]$.
        Training:
            Randomly sample experiences from replay buffer.
            Update parameter $\theta^{Q}$ by minimizing loss in (30).
            Update parameter $\theta^{\mu}$ by policy gradient in (32).
            Target networks soft update as (33)
    end while
    Adapt noise scale value as (38).
end for
return the trained networks $\theta^{\mu^*}$.
```

In addition, the parameter space noise requires a scale value ρ to adjust the variance in the action space. The noise scale value is adapted over time and can be calculated as

$$\rho_{k+1} = \begin{cases} \varsigma\rho_k & \text{, if } d(\mu,\tilde{\mu}) \leq \vartheta, \\ \frac{1}{\varsigma}\rho_k & \text{, otherwise,} \end{cases} \tag{38}$$

where ς and ϑ are a scaling factor and a threshold value, respectively, $d(\mu,\tilde{\mu})$ denotes the distance between the non-perturbed and perturbed policy, which can be calculated as

$$d(\mu,\tilde{\mu}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_s\left[(\mu(s)_i - \tilde{\mu}(s)_i)^2\right]}, \tag{39}$$

where N is the dimension of the action space, and $\mathbb{E}_s[.]$ is estimated from a batch of states from the replay buffer. Then, the perturbed actor-network parameter at episode k is calculated as

$$\tilde{\theta}^{\mu} = \theta^{\mu} + \rho_k\mathcal{N}(0,1), \tag{40}$$

We illustrate the whole framework in the Fig. 3, which has five neural networks: two critic-networks and three actor-networks. The perturbed actor-network is used to decide the action $a[t]$ to interact with the observed state from the environment $s[t]$ during exploration. Then, The samples are stored into the replay buffer and used for training the agent using the DDPG algorithm. The procedure is described in the algorithm 1.

At the beginning of each episode, the actor-network is perturbed with the noise scale value using (40). Based on observation $s[t]$ at each step, the perturbed actor-network decides an action $a[t]$ interacting with the environment and gets back the reward $r[t]$ and the next state $s[t+1]$. A tuple including $s[t]$, $a[t]$, $r[t]$, and $s[t+1]$ is then stored to the replay buffer for training the agent. In each training step, the agent randomly samples a mini-batch of experiences from the replay buffer and calculates to update the parameter of the critic-network and the non-perturbed actor-network by minimizing the loss as (30) and policy gradient as (32), respectively. Accordingly, the target networks are updated using the soft update as (33). At the end of each episode, the agent calculates the distance between the non-perturbed and perturbed policy by (39) to update the noise scale value using (38). After training, the trained non-perturbed actor-network is obtained to test the model.

V. PERFORMANCE EVALUATION

A. SIMULATION SETTINGS

To evaluate the performance of the proposed HAMEC−RSMA framework, we simulate an environment including a HAP hovers at an altitude of 20 (Km) that serves 4 AUs flying in a range of 500 (m) at an altitude of 200 (m). Each AU is assumed to fly randomly with a velocity of 15 (m/s) in a certain area without violating the others. The networks in this model have two hidden layers, which include 512 and 1024 nodes in the critic-networks, and 128 and 256 nodes in the actor-networks. The model parameters, and

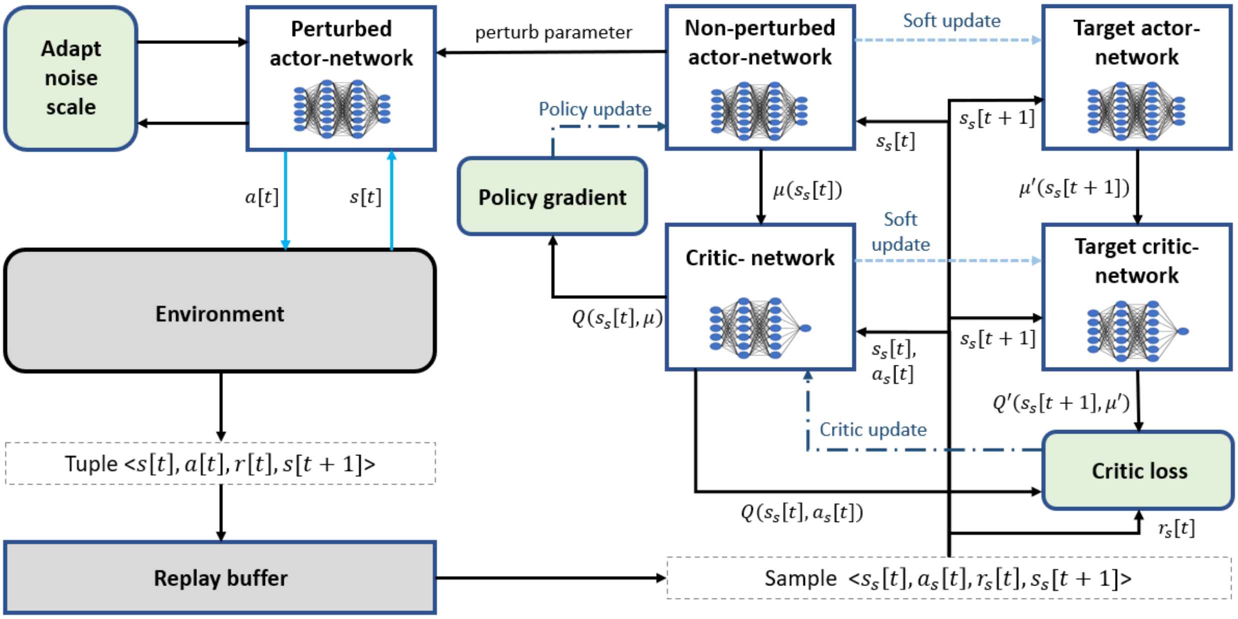


FIGURE 3: HAMEC-RSMA framework.

TABLE 1: Simulation Parameters

Parameter	Value
Size of the task, $w_u[t]$	[1-1.5] Mbits
Required computation resource, $c_u[t]$	[1000-1200] cycles/bit
Number of AU clusters	8
Number of AUs per cluster, U	4
Transmission bandwidth, $\mathcal{B}$	200 MHz
Channel power gain, β_0	1.42×10^{-4}
Transmission power, $P_u^{\max}$	20 dBm
Path loss exponent, α	2
Local computing resource, f_u^l	1.5 GHz
Computation resource of HAP, f_{hu}^c	9 GHz
Energy coefficient, κ_u	1e-28
Noise power, σ^2	-170 dBm/Hz
Negative penalty value, ∇	-20
Scaling factor, ς	1.01
Threshold value, ϑ	0.2
Initial noise scale value, ρ_0	0.1
Soft update constant, τ	$1e^{-3}$
Discount factor, γ	0.99
Replay buffer size	$1e^6$
Number of step in each episode	300

the environment parameters are given in the table 1. To demonstrate the efficiency of the proposed framework, we compare the performance of the proposed framework with some benchmark schemes, which are defined as follows.

- *HAMEC-NOMA*: In this scheme, the communication between the AUs and the HAP using the non-orthogonal multiple access (NOMA) technique is applied for comparison to demonstrate the efficiency of the RSMA technique. We also use the DDPG algorithm with parameter noise to solve the problem.
- *Full local executing* (FLE): We define this scheme for comparison to demonstrate the efficiency of the HAMEC model to the network. In this scheme, all the tasks are executed locally at the AUs, implying that there is no HAMEC in the aerial network.
- *DDPG with additional noise* (DDPG-AN): In this scheme, we apply the original DDPG algorithm proposed in [23] to the model, where the exploration is ensured by the additional noise as (35). To deal with the training samples violation issue as analyzed in subsection IV-B1, we directly trim the action to the constraint range. Then, the decided action can be presented as

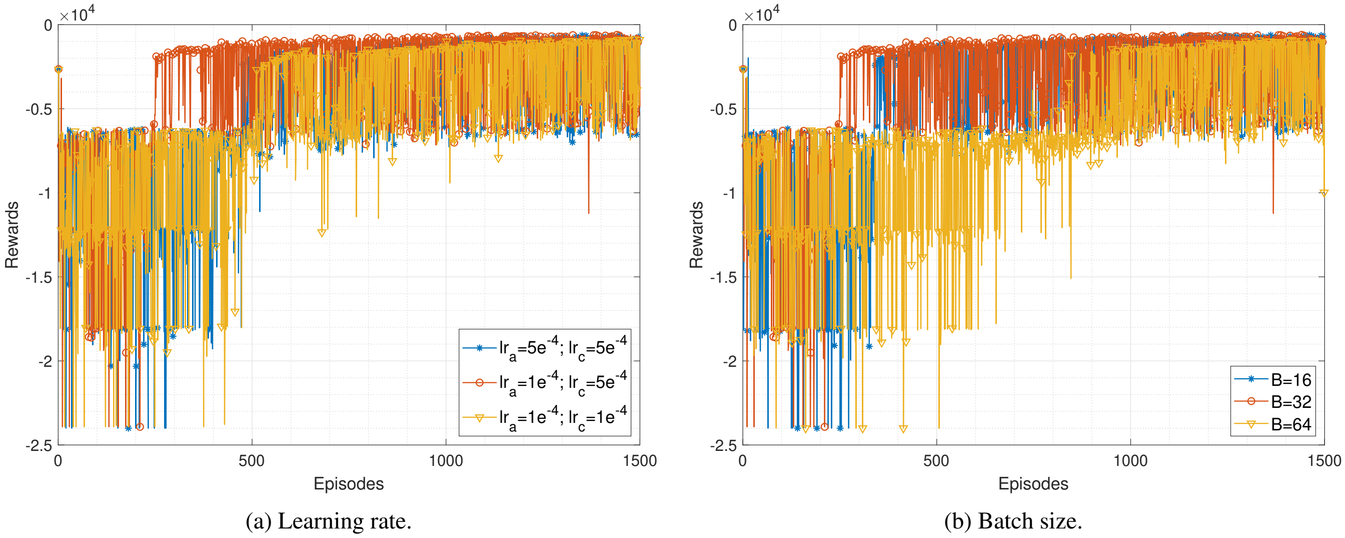
$$a[t] = \begin{cases} 1 & \text{, if } \mu\left(s[t]|\theta^\mu\right) + \mathcal{N}[t] > 1, \\ 0 & \text{, if } \mu\left(s[t]|\theta^\mu\right) + \mathcal{N}[t] < 0, \\ \mu\left(s[t]|\theta^\mu\right) + \mathcal{N}[t] & \text{, otherwise.} \end{cases} \tag{41}$$

- *Random action* (RA): In this scheme, the actions are chosen randomly within the constraint value range to interact with the environment.

B. CONVERGENCE ANALYSIS

Firstly, we evaluate the proposed algorithm convergence by varying some training hyper-parameters. The environment in this training is a scenario with 4 AUs moving randomly, the task size and the required computation resource are chosen randomly in the range of [1-1.5] (Mbits) and [1000-1200] (cycles/bit), respectively. The training reward of the model when changing the learning rate and the mini-batch size are illustrated in Fig. 4.

The learning rates of the actor-network (lr_a) and the critic-network (lr_c) are chosen in three cases (lr_a, lr_c) = $\{(5e^{-4}, 5e^{-4}), (1e^{-4}, 5e^{-4}), (1e^{-4}, 1e^{-4})\}$. As the results in Fig. 4a, the rewards converge to a definite range after several hundred episodes and then slightly increase in all three cases.



(a) Learning rate.

(b) Batch size.

FIGURE 4: Convergence concerning hyper-parameter.

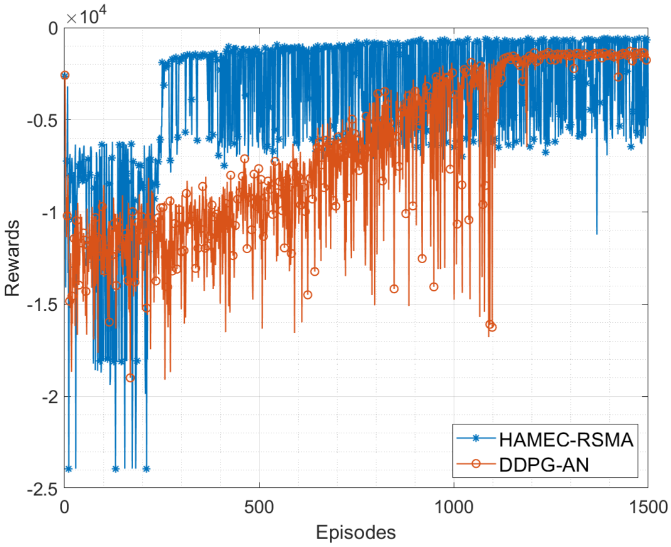


FIGURE 5: Comparison training exploration of parameter space noise and action space noise.

However, the case $(\mathrm{lr_a}, \mathrm{lr_c}) = (1e^{-4}, 5e^{-4})$ gives better performance than the others when it reaches the definite range after about 300 episodes and achieves the highest reward value in three cases. Therefore, we choose this case of learning rate for the simulation.

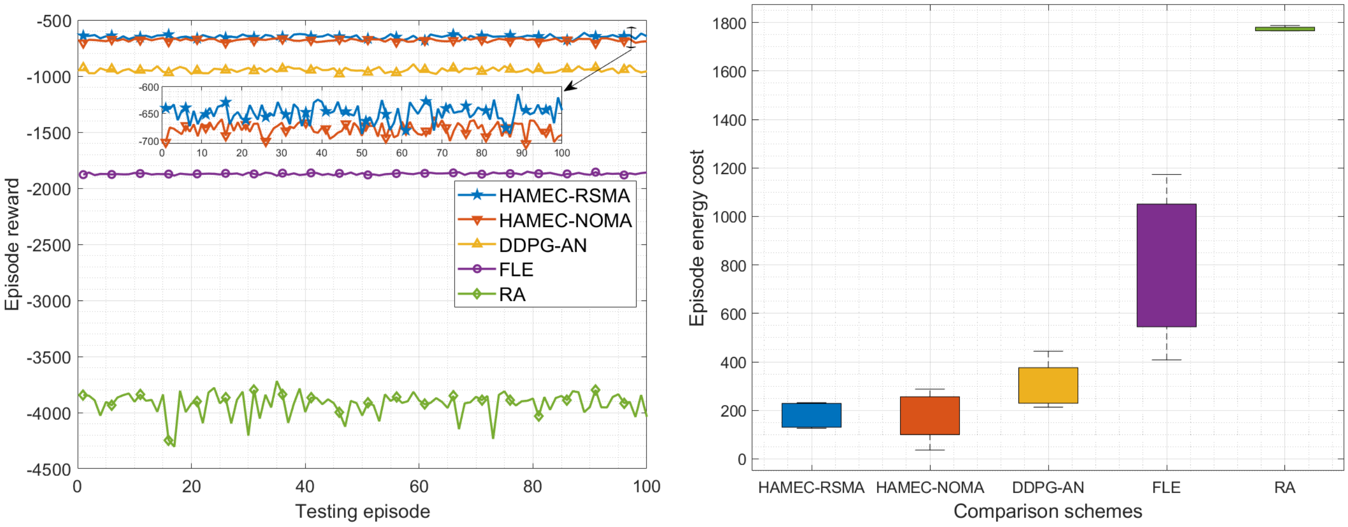
To evaluate the efficiency of the mini-batch size to the training, we train the model with threes values as $B = \{16, 32, 64\}$ and get the result in Fig. 4b. In the case $B = 64$, the model starts to converge after about 950 episodes, which is very slow compared to the others. Because of the dynamic environment, the large mini-batch size causes noise in the training data and leads to slow convergence. In the remaining cases, the model converges earlier, and the case $B = 32$ gives the best performance. Hence, we select $B = 32$ to train this model.

Then, we compare the efficiency of the parameter space noise and action space noise (DDPG-AN) to the exploration in training. The training rewards are illustrated in Fig. 5. According to the results, the reward for employing action space noise grows over time and stabilizes around 1150 training episodes. The reward when utilizing parameter space noise, on the other hand, is not consistent during training, implying that it is still exploring new experiences. Furthermore, the reward of parameter space noise is higher than the reward of action space noise. Because in DDPG-AN, we explicitly trim the action to the constraint range, which limits the exploration and may lead the model to become stuck at a local point, reducing the model performance. However, parameter space noise can improve this issue, and the results reveal that parameter space noise performs better than action space noise in this scenario.

C. PERFORMANCE EVALUATION

The performance of the proposed HAMEC−RSMA framework is compared to the other schemes in this sub-section. We test the model in 100 episodes, and the results are illustrated in Fig. 6. The episode rewards of the schemes are given in Fig. 6a, with HAMEC−RSMA consistently having the highest reward. The results indicate that the RSMA technique can improve the performance of the NOMA technique, where the reward of HAMEC−RSMA is about 5.1% greater than HAMEC-NOMA. The result also demonstrates the efficiency of the parameter space noise compared to the action space noise, where the proposed framework gives the result approximately 32.11% better than DDPG-AN. In addition, the HAMEC model significantly improve the network performance, where it is about 65.57% and 83.56% higher than FLE and RA schemes, respectively.

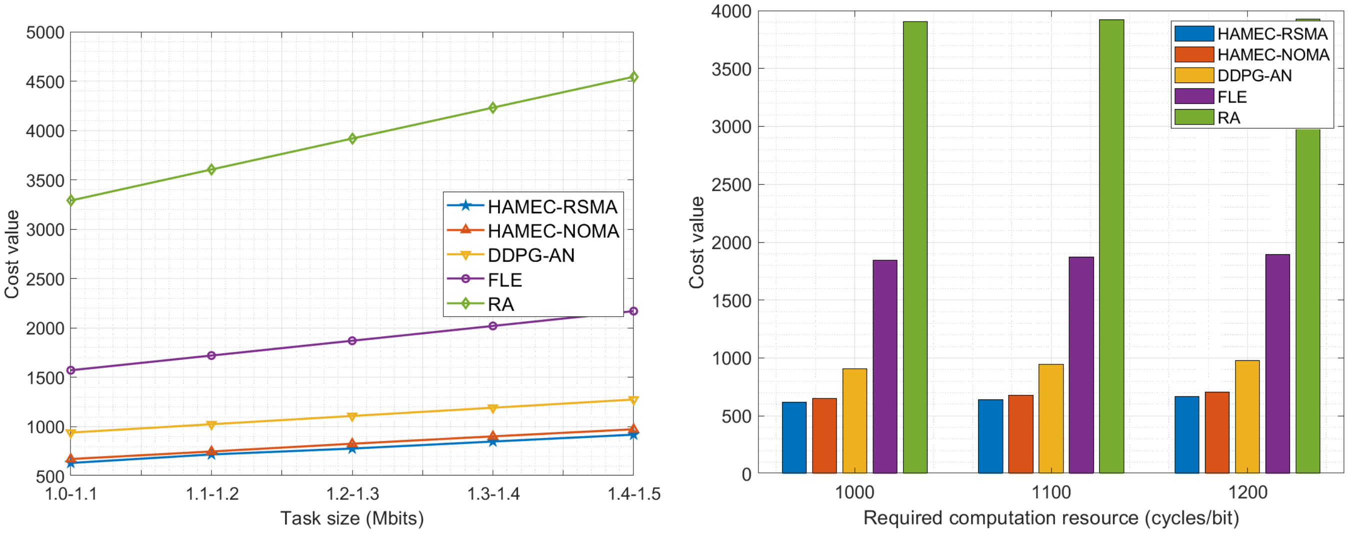
Besides, we evaluate the energy cost fairness between the



(a) Episode reward value.

(b) Energy cost fairness.

FIGURE 6: Performance in testing 100 episodes.



(a) Episode cost value according to task size.

(b) Episode cost value according to task required resource.

FIGURE 7: Cost value when changing the task.

AUs by measuring the energy costs in 100 testing episodes, which are illustrated in Fig. 6b. The average energy cost of each AU in the HAMEC−RSMA and HAMEC-NOMA is low compared with the others. Furthermore, HAMEC−RSMA gives better fairness than HAMEC-NOMA. Because in the NOMA, an AU with poor channel gain has to use more transmit power to increase the data rate, resulting in an imbalance in transmit power. In the RSMA, each AU can have a high rate path and a low rate path, depending on the splitting ratio and decoding order, which makes data transmission fair. As a result, the total power used to transmit data will be fair among the AUs.

Next, we examine the system performance when increasing the task size, where the task size increases from 1.0 to 1.5 (Mbits), with the cost values shown in Fig. 7a. Firstly, the proposed framework performs best in all cases where the cost value is the lowest. In addition, the cost value rises as the task size increase, where the cost increase approximately 9.81% when the task size increases 100 (Kbits) in the HAMEC−RSMA scheme. Because with a given capacity, increasing the task size leads to an addition in the time for processing or the need for more power to process the task, which grows the cost value of competing for the task.

Also, Fig. 7b illustrates the impact of the required computation resource of the task on the system performance. Our proposed framework also yields the lowest cost compared with the other schemes in all the evaluation cases. Similar to the task size, increasing the task required resource makes the cost value higher due to an increase in the processing time or the power. In particular, the cost increase approximately 4% when the task required resource increases 100 (cycles/bit) in the `HAMEC-RSMA` scheme.

VI. CONCLUSION

In this research, we have considered the HAMEC-enhanced aerial system with uplink RSMA communication. Here, we formulated an optimization problem involving the offloading decision, splitting ratio, transmit power, and decoding order to minimize task processing costs in terms of the total latency and energy consumption. To overcome the problem, we deployed a reinforcement learning model named `HAMEC-RSMA`, which trains the agent using the DDPG algorithm. Since the action space noise in the DDPG algorithm may violate variable constraints of the problems, we applied the parameter space noise for exploration to improve the training performance. Numerical results demonstrated the efficiency of the proposed `HAMEC-RSMA` framework, where it outperforms the existing benchmark schemes. As future work, the integration of RSMA and MEC should be considered in an industrial IoT scenario with dense user equipment. In addition, a combination of `HAMEC-RSMA` with other emerging technologies potential in 6G networks such as intelligent reflecting surface, massive multiple-input and multiple-output (MIMO) deserves to be investigated thoroughly.

REFERENCES

[1] G. Gui, M. Liu, F. Tang, N. Kato, and F. Adachi, “6G: Opening new horizons for integration of comfort, security, and intelligence,” IEEE Wireless Communications, vol. 27, no. 5, pp. 126–132, 2020.

[2] 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.

[3] S. Dang, O. Amin, B. Shihada, and M.-S. Alouini, “What should 6G be?,” Nature Electronics, vol. 3, no. 1, p. 20–29, 2020.

[4] P. Boccadoro, D. Striccoli, and L. A. Grieco, “An extensive survey on the Internet of drones,” Ad Hoc Networks, vol. 122, p. 102600, 2021.

[5] S. Jung, W. J. Yun, M. Shin, J. Kim, and J.-H. Kim, “Orchestrated scheduling and multi-agent deep reinforcement learning for cloud-assisted multi-UAV charging systems,” IEEE Transactions on Vehicular Technology, vol. 70, pp. 5362–5377, June 2021.

[6] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, “Mobile edge computing: A survey,” IEEE Internet of Things Journal, vol. 5, no. 1, pp. 450–465, 2018.

[7] B. Shang, Y. Yi, and L. Liu, “Computing over space-air-ground integrated networks: Challenges and opportunities,” IEEE Network, vol. 35, no. 4, pp. 302–309, 2021.

[8] Z. Lin, M. Lin, B. Champagne, W.-P. Zhu, and N. Al-Dhahir, “Secure and energy efficient transmission for RSMA-based cognitive satellite-terrestrial networks,” IEEE Wireless Communications Letters, vol. 10, no. 2, pp. 251–255, 2021.

[9] M. S. Hossain, C. I. Nwakanma, J. M. Lee, and D.-S. Kim, “Edge computational task offloading scheme using reinforcement learning for IIoT scenario,” ICT Express, vol. 6, no. 4, pp. 291–299, 2020.

[10] Z. Kuang, Z. Ma, Z. Li, and X. Deng, “Cooperative computation offloading and resource allocation for delay minimization in mobile edge computing,” Journal of Systems Architecture, vol. 118, p. 102167, 2021.

[11] N.-N. Dao, D.-N. Vu, W. Na, J. Kim, and S. Cho, “SGCO: Stabilized green crosshaul orchestration for dense IoT offloading services,” IEEE Journal on Selected Areas in Communications, vol. 36, pp. 2538–2548, November 2018.

[12] T. P. Truong, A.-T. Tran, T. M. T. Nguyen, T.-V. Nguyen, A. Masood, and S. Cho, “MEC-enhanced aerial serving networks via hap: A deep reinforcement learning approach,” in 2022 International Conference on Information Networking (ICOIN), pp. 319–323, 2022.

[13] L. Zhang and N. Ansari, “Optimizing the operation cost for UAV-aided mobile edge computing,” IEEE Transactions on Vehicular Technology, vol. 70, no. 6, pp. 6085–6093, 2021.

[14] Z. Yu, Y. Gong, S. Gong, and Y. Guo, “Joint task offloading and resource allocation in UAV-enabled mobile edge computing,” IEEE Internet of Things Journal, vol. 7, no. 4, pp. 3147–3159, 2020.

[15] M. Choi, J. Kim, and J. Moon, “Dynamic power allocation and user scheduling for power-efficient and delay-constrained multiple access networks,” IEEE Transactions on Wireless Communications, vol. 18, pp. 4846–4858, October 2019.

[16] M. Choi, D. Yoon, and J. Kim, “Blind signal classification for non-orthogonal multiple access in vehicular networks,” IEEE Transactions on Vehicular Technology, vol. 68, pp. 9722–9734, October 2019.

[17] A. Rahmati, Y. Yapici, N. Rupasinghe, I. Guvenc, H. Dai, and A. Bhuyan, “Energy efficiency of RSMA and NOMA in cellular-connected mmWave UAV networks,” in 2019 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1–6, 2019.

[18] Z. Yang, M. Chen, W. Saad, and M. Shikh-Bahaei, “Optimization of rate allocation and power control for rate splitting multiple access (RSMA),” IEEE Transactions on Communications, vol. 69, no. 9, pp. 5988–6002, 2021.

[19] W. Jaafar, S. Naser, S. Muhaidat, P. C. Sofotasios, and H. Yanikomeroglu, “On the downlink performance of RSMA-based UAV communications,” IEEE Transactions on Vehicular Technology, vol. 69, no. 12, pp. 16258–16263, 2020.

[20] S. A. Tegos, P. D. Diamantoulakis, and G. K. Karagiannidis, “On the performance of uplink rate-splitting multiple access,” IEEE Communications Letters, pp. 1–1, 2022.

[21] Z. Yang, M. Chen, W. Saad, W. Xu, and M. Shikh-Bahaei, “Sum-rate maximization of uplink rate splitting multiple access (RSMA) communication,” IEEE Transactions on Mobile Computing, pp. 1–1, 2020.

[22] 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. 13196–13208, 2021.

[23] T. P. Lillicrap, J. J. Hunt, A. e. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” arXiv e-prints, p. arXiv:1509.02971, Sept. 2015.

[24] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, “Parameter space noise for exploration,” arXiv e-prints, p. arXiv:1706.01905, 2018.

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, South Korea, in 2022. His research interests include machine learning, mobile edge computing, and wireless communication.

NHU-NGOC DAO (Senior Member, IEEE) 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 the B.S. degree in electronics and telecommunications from the Posts and Telecommunications Institute of Technology, Hanoi, Viet Nam, in 2009. Prior to joining the 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 currently serves as an Editor of the *Scientific Reports*. His research interests include network softwarization, mobile cloudization, intelligent systems, and the Intelligence of Things.

SUNGRAE CHO received B.S. and M.S. degrees in Electronics Engineering from Korea University, Seoul, South Korea, in 1992 and 1994, respectively, and Ph.D. degree in Electrical and Computer Engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2002. He is a Professor with the School of Computer Science and Engineering, Chung-Ang University (CAU), Seoul, South 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, South Korea, in 2003. From 1994 to 1996, he was a Research Staff Member with Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea. From 2012 to 2013, he held a Visiting Professorship with the National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA. His current research interests include wireless networking, ubiquitous computing, and ICT convergence. He has been a Subject Editor of IET Electronics Letter since 2018 and was an Area Editor of Ad Hoc Networks Journal (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 the IEEE MASS, and as a Program Committee Member, such as IEEE ICC, GLOBECOM, VTC, MobiApps, SENSORNETS, and WINSYS.

• • •