Abstract

Sengly Muy
Department of Intelligent Energy Industry
Major in AI/IoT/Big Data-Based New Energy Industry
The Graduated School of Chung-Ang University

Recently, the space-air-ground integrated network (SAGIN) has gained more attention from researchers due to the success of the deployment by Star Link company, which provided reliable internet for the mobile users on ground by using beam-forming of satellites. SAGIN is a network technology that simply integrates satellites, aircraft, and ground-based communication systems to ensure connectivity for wireless communication system. However, SAGIN faces many challenges due to the dynamic of the network environment with the limitations in resources such as bandwidth, power, and processing capabilities. These challenges can impact the overall performance of the wireless communication, including reliability, data rate, latency, and energy efficiency. Therefore, many researchers are intensively working on developing solutions to improve the performance and reliability of SAGIN with fully utilize the resource effectively. Consequently, this study aims to design the effective and reliable machine learning (ML) algorithms for SAGIN in order to improve the quality of services in SAGIN, such as coverage, fairness, data rate, and energy consumption. First, we considered a network scenario consisting of low-earth orbit (LEO) satellites, high-altitude platforms (HAPs),

and unmanned aerial vehicles (UAVs), that equipped with array antennas enable beam forming to communicate with moving ground users (GUs). After that, we formulate a problem model with the three QoS problems, such as coverage, fairness, data rate, and energy consumption. This problem involved determining the trajectory, bandwidth, power, and beam direction of LEO satellites, HAPs, and UAVs, which makes this problem more complicated and difficult to solve. In order to solve this problem, we have developed three centralized deep reinforcement learning (DRL) algorithms: SA-DQL, SA-DAC, and SA-DDPG. Furthermore, we proposed a novel decentralized DRL approach known as multi-agent DAC with experienced DNN (MA-DAC-DNN), which aimed to solve the problem in a distributed way. Finally, we evaluated our proposed MA-DAC-DNN by comparing with centralized DRL algorithms. Through the simulation, the results show that our proposed algorithm outperforms other algorithms in terms of optimality and training convergence. These results indicate that decentralized DRL outperforms centralized DRL in optimizing multiple QoS for SAGIN. Our proposed algorithm has been proven to be scalable and efficient, making it ideal for deployment in extensive networks with the huge number of devices.

Keywords: SAGIN, multi-object optimization, centralized DRL, decentralized DRL.

Table of Contents

List of Tables

List of Figures

1. Introduction

1.1. Overview

Space-Air-Ground Integrated Network (SAGIN) is a technology that integrates satellites, aircraft, and ground-based communication systems to provide seamless connectivity across various environments [1]. SAGIN, which Star-Link successfully deployed, has shown promising results in enhancing network reliability and communication capabilities, particularly in remote areas. Additionally, it enables real-time data transmission, remote monitoring, and cross-platform communication, improving efficiency and coordination in operations. Advancements in artificial intelligence (AI) and machine learning (ML) are being incorporated to further optimize SAGIN performance, providing more reliable and efficient results in various network environments. The integration of ML with SAGIN will enhance the performances that make SAGIN more reliable in industries such as manufacturing, defense, healthcare, and emergency response. SAGIN is expected to revolutionize industries by offering reliable connectivity, high data rates, global coverage, low latency, and enhanced security. However, the high mobility of low earth orbit (LEO) satellites, high-altitude platforms (HAPs), unmanned aerial vehicles (UAVs), and ground users (GUs) presents significant challenges.

SAGIN, a multi-domain communication system, faces difficulties such as a dynamic environment [2] and limited resources [3]. Here, the dynamic environment includes frequent spacecraft movement, varying signal strengths, and potential interference from other systems, requiring expert monitoring and adjustment to ensure smooth communication and reliable network performance [4]. Additionally, the integration of various platforms adds complexity to effective network management. Advanced algorithms and communication systems can help organizations respond to dynamic network environments and maintain operational effectiveness. Moreover, the limited resources, such as bandwidth, processing power, and storage capacity, pose challenges to optimize system performance and efficiency [5]. Implementing optimized data transfer protocols, memory storage strategies, and backhauling techniques can help reduce bandwidth constraints and high latency connections [6]. Additionally, utilizing lightweight algorithms, distributed processing, and edge computing can further reduce processing burdens in space and air network elements.

SAGIN's quality of service (QoS) is critical due to the various industries and sectors requiring high levels of knowledge. The rapidly changing technological landscape poses a challenge to keeping up with the latest advancements and ensuring efficient services. Key QoS metrics in SAGIN include reliability [7], availability [8], mobility [9], throughput [10], spectrum efficiency [11], fairness [12]-[13], energy efficiency [14], coverage [15], latency [16], jitter [17], and security [18]. Reliability ensures uninterrupted connectivity for applications impacting lives, safety, or critical infrastructure. Availability minimizes outages for emergency response, remote control of critical systems, and real-time communication. Mobility ensures seamless handover between network segments for mobile devices. Throughput ensures a sufficient data transfer rate for high-resolution video streaming, data backhaul, and large file transfers. Spectrum efficiency maximizes data transmission per unit of available radio spectrum. Fairness ensures equitable resource allocation and service quality across different users and applications. Security protects against cyberattacks, data breaches, and unauthorized access, requiring robust security measures and resilient infrastructure.

One solution to address these challenges is implementing an autonomous system to optimize network QoS and resource allocation without human intervention. This technology adapts to changing network conditions in real-time, enhancing efficiency and security in mobile communications [19]. Intelligent connectivity for SAGIN uses ML algorithms to enable autonomous networking, allowing real-time decisions suitable for SAGIN. This optimizes network performance and ensures seamless communication across different domains. Autonomous networking enhances performance and resilience by reducing human intervention and enabling effective operation in dynamic environments. Examples of autonomous networking include self-configuring networks, self-healing networks, self-optimizing networks, and distributed decision-making and collaboration. These features enable seamless communication, efficient resource allocation, and better management of network adjustment, ensuring smooth communication and reliability.

ML is a key component of autonomous networking, allowing systems to analyze data, identify patterns, and make decisions without explicit programming [20]. ML can be categorized into supervised learning, unsupervised learning, and reinforcement learning (RL), each with a unique approach to analyzing data and improving network performance. RL is particularly effective for analyzing data and enhancing network performance due to its process of trial, error, and adjustment. In RL, an agent learns to make decisions through interaction with an environment, receiving rewards or

penalties based on the outcomes of its actions. Deep reinforcement learning (DRL), a subset of RL, uses deep neural networks (DNN) to process data and make decisions based on rewards and penalties. DRL has demonstrated success in fields like gaming, robotics, and natural language processing, but designing DRL algorithms is challenging due to the complexity and dynamics of the environment, requiring careful consideration for stability and convergence.

In ML, DRL can be divided into centralized DRL, where a single agent makes adjustments for all devices, and decentralized DRL, where each agent or device makes its own adjustments. Centralized DRL is easy to organize and provides effective results, but it faces scalability issues and increased complexity. Conversely, decentralized DRL allows more flexibility and adaptability in dynamic environments. Consequently, decentralized DRL is more suitable for SAGIN as it allows each agent to react rapidly to changes in the dynamic network environment and adapt to the large number of devices in SAGIN.

1.2. Related Works

In this study, we have reviewed several research studies that focused on optimizing single QoS or multiple QoS optimization in SAGIN by using different type of techniques. These techniques include traditional optimization technique, centralized DRL techniques, and decentralized DRL techniques. To improve QoS in SAGIN, there are many studies that applied traditional optimization algorithms to address both single- and multiple QoS optimization problems. For instance, in [21], the authors suggested a non-preemptive priority queuing system and a transmission control scheme with cross-layer optimization. This created a strong two-stage stochastic optimization problem that was used to cut down on delays in rural areas. In [22], the authors optimized ground data processing center data collection and transmission using time expanding graphs, considering resource constraints and flow restrictions, and collaborating with HAPs and LEO satellites for remote areas. In [23], convex approximation and semi-definite programming were used to find the best way to divide up the power for the NOMA downlink, the uplink transmit power, and the satellite beamforming. In [15], the downlink coverage for civil aircraft augmented-SAGIN (CAA-SAGIN) was looked into, and analytical expressions, approximations, and bounds were checked to make sure they were correct. The authors of [24] proposed integrated relative energy efficiency (IREE) as a tool to measure traffic profiles and network capacities from an energy efficiency perspective, comparing its green trade-off with

conventional design methods. Also, in [25], a study on data offloading in SAGINs introduced an approximation algorithm that finds half of the best solution for power allocation, balancing energy use, and mean make span. There was research on a relay-aided device-to-device (D2D) MIMO scheme in [26] that looked at how to make D2D communication more energy and spectral efficient (SE).

Furthermore, many researchers have used centralized DRL algorithms to address and solve the problem of SAGIN. In [27], the authors suggested using speech emotion recognition (SER) and the AlexNet network model to enhance the vehicle's speech acoustic model and improve traffic flow for self-driving cars in a 5G-enabled SAGIN-based intelligent transportation system (ITS). In [14], a plan was put forward to use RL and the Lyapunov optimization method to choose services for the civil aircraft-enabled SAGIN (CAE-SAGIN) that would use less energy by making better use of resources and requests. In addition, [28] introduced a deep deterministic policy gradient (DDPG)-based network selection algorithm to help networks make better decisions on their own. The goal was to improve resource utilization and load balance in an integrated network with many options. Last but not least, [29] solved the issue of very reliable and fast slicing in multi-access edge computing (MEC) systems for the next generation of IoT and mobile apps in SAGIN by creating a new deep learning model based on the message passing-graph neural network (MPNN).

Moreover, there are many studies that have proposed DRL approaches to enhance SAGIN performance. In [30], the authors proposed distributed DRL storage resource management to enhance reliability and communication by providing storage resources for multiple agents to train the model. [31] also suggested using distributed DRL to cut down on transmission delays and task offloading pressure. This would lead to better long-term revenue-to-cost ratios for nodes, links, and average age compared to older methods. The authors of [32] suggested a learning-based orbital edge offloading (LOEF) method based on the actor-critic framework of multi-agent DRL. This method would help each UAV figure out the best way to offload computing tasks in SAGIN for the Internet of Things (IoRT).

1.3. Our Contributions

Based on our knowledge, there is no research addressing the multiple QoS trade-offs regarding coverage, data rate fairness, and energy efficiency using decentralized DRL for SAGIN. Therefore, we aim to fill this research gap by proposing a novel decentralized DRL approach that can effectively and distributive address multiple QoS problems in SAGIN. The following describes the contributions of our research to this study:

1. **Network design**: First, we consider a network scenario where a constellation of LEO satellites transmits the information to the moving GUs, deploying multiple HAPs and UAVs to relay the information. Additionally, we assume that LEO satellites, HAPs, and UAVs are adapted with array antennas that generate multiple beam-forming signals to improve communication from space to ground.
2. **Multiple QoS optimization problems**: Then, we construct the optimization problem model with the goal of maximizing the multiple QoS problems, such as coverage, data rate fairness, and energy consumption, by jointly optimizing the trajectory, bandwidth, power, and beam direction of LEO satellites, HAPs, and UAVs.
3. **Centralized DRL**: After that, we provide an overview of the conventional centralized DRL algorithms, such as deep Q-learning (DQL), deep actor critic (DAC) learning, and deep deterministic policy gradient (DDPG) learning, in order to solve the problem of multiple

QoS optimization in SAGIN. And we analyze the advantages and limitations of each algorithm in the context of convergence, optimality, and scalability in SAGIN.

4. **Propose decentralized DRL**: Moreover, we proposed a multi-agent DRL approach called Multi-Agent DAC, which enables multiple agents to collaborate and share experiences, improving convergence and scalability in SAGIN. Based on our experience, we observed that DRL agents often choose actions requiring lengthy computations to find the optimal solution. Therefore, we implemented an enhanced version called Multi-Agent DAC with Experienced DNN (MA-DAC-DNN), incorporating a policy design aimed at reducing training process repetition.
5. **Performance evaluation**: Finally, we evaluated our proposed decentralized DRL algorithm called MA-DAC-DNN and compared it to conventional centralized DRL algorithms, including SA-DQL, SA-DAC, and SA-DDPG. The simulation results show that our proposed algorithm outperforms these others in terms of optimality and training convergence. This simulation result suggest that our approach could be a promising solution for addressing multiple QoS optimizations in complex systems.

2. System Model and Problem Formulation

2.1. System Model

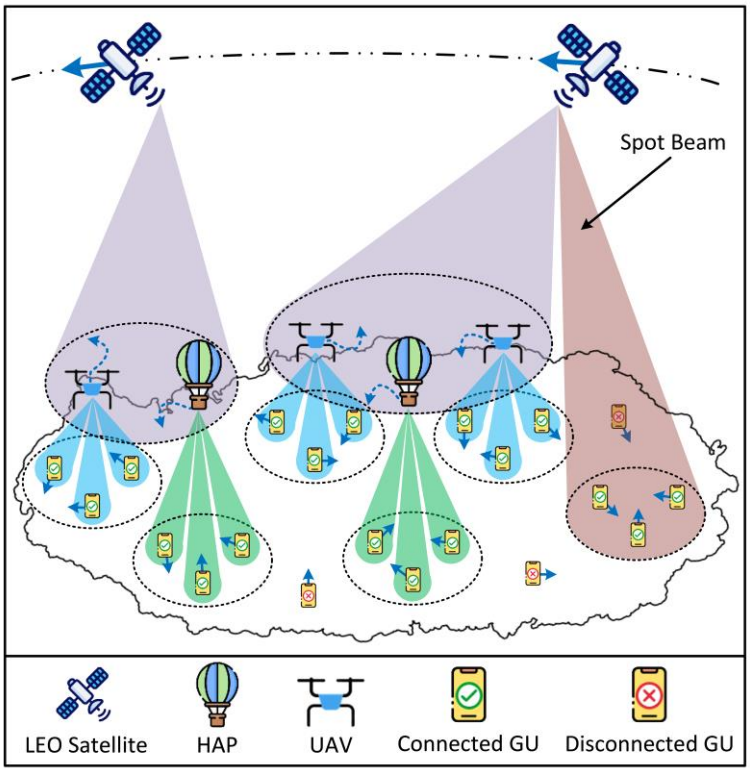


Figure 1: System model.

In this study, we consider a network scenario following the SAGIN, which consists of a constellation of LEO satellites, H HAPs, U UAVs, and I GUs. Additionally, we assume that LEOs, HAPs, and UAVs are equipped with array antennas capable of generating multiple beamforming patterns to support the GUs. Due to the immense number of GUs, a single constellation of LEOs is unable to support direct connections between LEOs and GUs; therefore, HAPs and UAVs are employed as relays to extend coverage and enhance network capacity, as shown in **Figure 1**. Here, a constellation of LEOs

orbits uniformly at a constant speed V^{LEO}, maintaining an equal arc distance D^{LEO}, resulting in a period of $T^{\mathrm{LEO}} = D^{\mathrm{LEO}} / V^{\mathrm{LEO}}$. Therefore, the time period for each LEO can be denote as $t \in \mathcal{T} = \left[0, T^{\mathrm{LEO}}\right]$.

Let $s \in \mathcal{S} = \{1, 2, ..., S\}$, $h \in \mathcal{H} = \{1, 2, \ldots, H\}$, $u \in \mathcal{U} = \{1, 2, \ldots, U\}$, and $i \in \mathcal{I} = \{1, 2, \ldots, I\}$ are denoted as the indices of LEO satellites, HAPs, UAVs, and GUs, respectively. Furthermore, the locations of LEO satellite s, HAP h, UAV u, and GU i can be denoted as $L_s^t = \left(x_s^t, y_s^t, Z_s\right)$, $L_h^t = \left(x_h^t, y_h^t, Z_h\right)$, $L_u^t = \left(x_u^t, y_u^t, Z_u\right)$, and $L_i^t = \left(x_i^t, y_i^t, 0\right)$, respectively. Here, Z_s, Z_h, and Z_u are the altitudes of LEO satellites, HAPs, and UAVs, respectively. Moreover, we assume that the LEO satellites generate a spot-beam on the ground to transmit information to GUs. Let us denote $L_c^t = (x_c^t, y_c^t, 0)$ as the location of the spot-beam on the ground with the center c at time t.

In this study, we assume that all GUs are randomly distributed following the pixel of road inside the closed polygon Ω. For example, **Figure 2** shows the generated location of GUs following the pixel of the roads uniformly inside a close polygon of Jeju Island. Here, the closed polygon with K points can be defined as

$$\Omega = \left\{\overline{\rho_1\rho_2}, \overline{\rho_2\rho_3}, \ldots, \overline{\rho_{K-1}\rho_K}, \overline{\rho_K\rho_1}\right\}, \forall k \in \{1, 2, \ldots, K\}, \quad (1)$$

where $\rho_k = \left(x_k, y_k\right)$ is the point k on the xy-axis, and $\overline{\rho_k\rho_{k+1}}$ is a segmentation from point k and point $k+1$. Moreover, all GUs are

considered to move with a randomly uniform direction angle $\text{Uniform}(0, 2\pi)$ and distance D^{GU}. Notice that the distance between two points X and Y can be calculated as $d_{X,Y} = |L_X - L_Y|$, where $|\cdot|$ is the Euclidean norm.

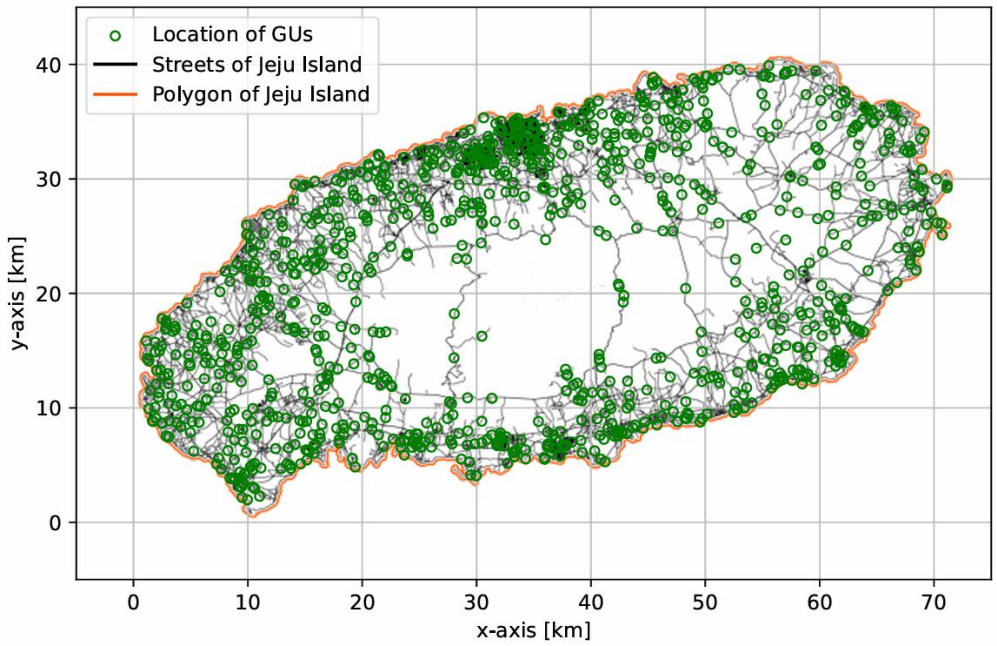


Figure 2: Generated GUs on the roads of Jeju Island.

2.1.1. Space-to-Ground Channel Model

In this study, we consider LEO satellites equipped with array antennas that generate a spot-beam center c which aim to communicate with GUs. Moreover, we consider that antenna gain, path loss, and fading are used to model the channel from LEO satellites to GUs. Therefore, the channel model from LEO satellite s to GU i using spot-beam center c at time t can be expressed as

$$\left|g_{s,i}^{t}\right|^{2}=\left|E_{i}\left(\theta_{s,c}^{t},\phi_{s,c}^{t}\right)\right|^{2}\times PL_{s,i}^{t}\times FD_{s,i}^{t},\tag{2}$$

where $\left|E_{i}\left(\theta_{s,c}^{t},\phi_{s,c}^{t}\right)\right|^{2}$, $PL_{s,i}^{t}$, and $FD_{s,i}^{t}$ are the effective transmitter's antenna gain, the path-loss, and the fading channel from LEO satellite s to GU i using spot-beam center c at time t, respectively. Here, $\theta_{s,c}^{t}$ and $\phi_{s,c}^{t}$ are the elevation and azimuth angle from the LEO satellite s to the spot-beam center c at time t, respectively.

Let $\vec{v}_{s,c}^{t}=\left(L_{c}^{t}-L_{s}^{t}\right)_{x,y,z}=\left(r_{s,c}^{t},\theta_{s,c}^{t},\phi_{s,c}^{t}\right)_{r,\theta,\phi}$ and $\vec{v}_{s,i}^{t}=\left(L_{i}^{t}-L_{s}^{t}\right)_{x,y,z}=\left(r_{s,i}^{t},\theta_{s,i}^{t},\phi_{s,i}^{t}\right)_{r,\theta,\phi}$ be the vector from LEO satellite s to spot-beam center c and GU i at time t in Cartesian and Spherical coordinates, respectively. Therefore, the transmitter's antenna gain from LEO satellite s to GU i at time t can be approximated as, [33],

$$\left|E_{i}\left(\theta_{s,c}^{t},\phi_{s,c}^{t}\right)\right|^{2}\approx G_{\max}-12\left(\frac{\measuredangle\left(\vec{v}_{s,c}^{t},\vec{v}_{s,i}^{t}\right)}{\theta_{3dB}}\right)^{2},\tag{3}$$

where $G_{\max}$ represents the maximum transmitter's gain of the beam that observed at the spot-beam center. And, θ_{3dB} is the 3dB angle of the spot-beam that represents the half-power beam-width. Here, $\measuredangle\left(\vec{v}_{s,c}^{t},\vec{v}_{s,i}^{t}\right)$ is the angle between LEO satellite s to the spot-beam center c and GU i at time t, which can be calculated as

$$\measuredangle\left(\vec{v}_{s,c}^{t},\vec{v}_{s,i}^{t}\right)=\cos^{-1}\left(\frac{\vec{v}_{s,c}^{t}\cdot\vec{v}_{s,i}^{t}}{\left\|\vec{v}_{s,c}^{t}\right\|\left\|\vec{v}_{s,i}^{t}\right\|}\right). \tag{4}$$

Moreover, $PL_{s,i}^{t}$ is the path-loss from LEO satellite s to GU i at time t can be calculated as, [34],

$$PL_{s,i}^{t}=\frac{\beta_0}{\left(d_{s,i}^{t}\right)^{\alpha}}, \tag{5}$$

where α is the path-loss exponent, β_0 is the channel power at the reference distance 1m, and $d_{s,i}^{t}$ is the distance from LEO satellite s to GU i at time *t.* And, $FD_{s,i}^{t}$ is the channel fading from LEO satellite s to GU i at time *t*, characterized by small-scale fading following the Rician distribution, [35].

2.1.2. Aerial-to-Ground Channel Model

In this study, we consider HAPs and UAVs equipped with array antennas that generate multiple independently steered beams, as illustrated in **Figure 3**. Multiple main lobe beams can directly target GU i by adjusting the phase shifters of the multiple sub-array antennas. Consequently, the effective channel gain from HAP h and UAV u to GU i at time t can be expressed as

$$\left|g_{h,i}^{t}\right|^{2}=\left|E_i\left(\theta_h^{t},\phi_h^{t}\right)\right|^{2}\times PL_{h,i}^{t}\times FD_{h,i}^{t}, \tag{6}$$

and

$$\left|g_{u,i}^{t}\right|^{2}=\left|E_i\left(\theta_u^{t},\phi_u^{t}\right)\right|^{2}\times PL_{u,i}^{t}\times FD_{u,i}^{t}, \tag{7}$$

where $\left|E_i\left(\theta_h^t,\phi_h^t\right)\right|^2$ and $\left|E_i\left(\theta_u^t,\phi_u^t\right)\right|^2$ are the effective beam-forming from HAP h and UAV u to GU i at time t with the elevation angle θ_h^t and θ_u^t, and azimuth angle ϕ_h^t and ϕ_u^t, respectively. Here, the beam-forming from HAP h and UAV u to GU i at time t, can be expressed as, [36],

$$\begin{aligned} &E_i\left(\theta_h^t,\phi_h^t\right)= \\ &\left(1,\ldots,\exp\left(j\frac{2\pi}{\lambda}D_s\sin\left(\theta_h^t\right)\left(m\cos\left(\phi_h^t\right)+n\sin\left(\phi_h^t\right)\right)\right),\ldots\right. \\ &\qquad\left.\exp\left(j\frac{2\pi}{\lambda}D_s\sin\left(\theta_h^t\right)\left(M\cos\left(\phi_h^t\right)+N\sin\left(\phi_h^t\right)\right)\right)\right)^T, \end{aligned} \tag{8}$$

and

$$\begin{aligned} &E_i\left(\theta_u^t,\phi_u^t\right)= \\ &\left(1,\ldots,\exp\left(j\frac{2\pi}{\lambda}D_s\sin\left(\theta_u^t\right)\left(m\cos\left(\phi_u^t\right)+n\sin\left(\phi_u^t\right)\right)\right),\ldots\right. \\ &\qquad\left.\exp\left(j\frac{2\pi}{\lambda}D_s\sin\left(\theta_u^t\right)\left(M\cos\left(\phi_u^t\right)+N\sin\left(\phi_u^t\right)\right)\right)\right)^T, \end{aligned} \tag{9}$$

where j, λ, and D_s are the complex number, the wavelength of the radio frequency, and the distance spacing between antenna elements, respectively. Here, $m\in\{1,2,\ldots,M\}$ and $n\in\{1,2,\ldots,N\}$ denote the coordinates of antenna elements along the x-axis and y-axis, respectively. Moreover, $PL_{h,i}^t=\beta_0/\left(d_{h,i}^t\right)^\alpha$, $PL_{u,i}^t=\beta_0/\left(d_{u,i}^t\right)^\alpha$, $FD_{h,i}^t$ and $FD_{u,i}^t$ are the path-loss and Rician fading from HAP h and UAV u to GU i at time t, respectively.

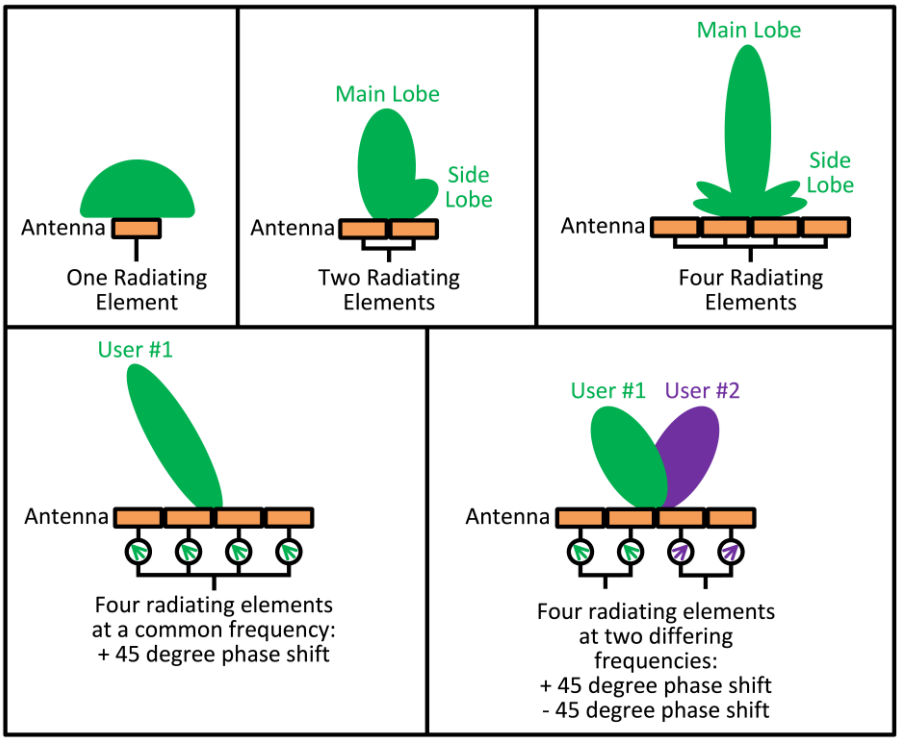


Figure 3: Example of beam-forming of array antennas.

2.1.3. Space-to-Aerial Channel Model

In this study we assume that the channel model from LEO satellite s to HAP h and UAV u are following the path-loss and fading, which can be expressed as

$$\left|g_{s,h}^{t}\right|^2 = PL_{s,h}^{t} \times FD_{s,h}^{t}, \tag{10}$$

and

$$\left|g_{s,u}^{t}\right|^2 = PL_{s,u}^{t} \times FD_{s,u}^{t}. \tag{11}$$

Similarly, the path-loss from LEO satellite s to HAP h and UAV u can be calculated as $PL_{s,h}^{t} = \beta_0 / \left(d_{s,h}^{t}\right)^{\alpha}$ and $PL_{s,u}^{t} = \beta_0 / \left(d_{s,u}^{t}\right)^{\alpha}$, respectively. And,

the channel fading from the LEO satellite s to HAP h and UAV u at time t $\left(FD_{s,h}^{t}\text{ and }FD_{s,u}^{t}\right)$ are modeled as Rician fading.

2.1.4. Data Rate Model

Let p_s^t, p_h^t and p_u^t be the transmit power of LEO satellite s, HAP h, and UAV u at time t, respectively. Thus, we can calculate the SINR, signal-to-interference-plus-noise Ratio, from the LEO satellite s, HAP h, and UAV u to the GU i at time t as following

$$\Gamma_{s,i}^{t}=\frac{p_s^t\times\left|g_{s,i}^{t}\right|^2\times G^{RX}}{I_{s'}^{t}+\sigma_A^2}, \tag{12}$$

$$\Gamma_{h,i}^{t}=\frac{p_h^t\times\left|g_{h,i}^{t}\right|^2\times G^{RX}}{I_{h'}^{t}+\sigma_A^2}, \tag{13}$$

and

$$\Gamma_{u,i}^{t}=\frac{p_u^t\times\left|g_{u,i}^{t}\right|^2\times G^{RX}}{I_{u'}^{t}+\sigma_A^2}, \tag{14}$$

where $I_{s'}^{t}$, $I_{h'}^{t}$, and $I_{u'}^{t}$ are the interference from other transmitter devices of LEO satellite s', HAP h', and UAV u' at time t, respectively. Moreover, G^{RX} and σ_A^2 are the receiver's gain and the power spectral density of Additional White Gaussian Noise (AWGN), respectively. By using Shannon–Hartley theorem [37], the data rate from LEO satellite s, HAP h, and UAV u to GU i can be calculated as

$$R_{s,i}^{t} = \beta_s B \log_2\left(1+\Gamma_{s,i}^{t}\right), \tag{15}$$

$$R_{h,i}^{t} = \beta_h B \log_2\left(1+\Gamma_{h,i}^{t}\right), \tag{16}$$

and

$$R_{u,i}^{t} = \beta_u B \log_2\left(1+\Gamma_{u,i}^{t}\right), \tag{17}$$

where β_s, β_h, and β_u are the bandwidth allocation of LEO satellite s, HAP h, and UAV u, respectively.

Similarly, the SINR from LEO satellite s to HAP h and UAV u at time t can be represented as

$$\Gamma_{s,h}^{t} = \frac{p_s^t \times \left|g_{s,h}^{t}\right|^2 \times G^{RX}}{I_{s'}^{t} + \sigma_A^2}, \tag{18}$$

and

$$\Gamma_{s,u}^{t} = \frac{p_s^t \times \left|g_{s,u}^{t}\right|^2 \times G^{RX}}{I_{s'}^{t} + \sigma_A^2}, \tag{19}$$

where $I_{s'}^{t}$ is the interference from other transmitter devices of LEO satellite s' at time t. Therefore, the data rate from LEO satellite s to HAP h and UAV can be calculated as

$$R_{s,h}^{t} = \beta_s B \log_2\left(1+\Gamma_{s,h}^{t}\right), \tag{20}$$

and

$$R_{s,u}^{t} = \beta_s B \log_2\left(1+\Gamma_{s,u}^{t}\right), \tag{21}$$

where β_s is the bandwidth allocation of LEO satellite s. It is should be notice that the total data rate at time t from LEO satellite s to HAP h or UAV u must be higher than the total data rate from HAP h or UAV u to GU i, which can be expressed as

$$\sum_{i\in\mathcal{I}} R_{h,i}^{t} \le \sum_{s\in\mathcal{S}} R_{s,h}^{t},\ \forall h \in \mathcal{H}, \tag{22}$$

and

$$\sum_{i\in\mathcal{I}} R_{u,i}^{t} \le \sum_{s\in\mathcal{S}} R_{s,u}^{t},\ \forall u \in \mathcal{U}. \tag{23}$$

Moreover, we assume that the end-to-end (E2E) data rate of GUs from LEO satellites is determined based on the maximum data rate from LEO satellites, HAPs, or UAV, which can be expressed as

$$R_i^t = \max_{s\in\mathcal{S},h\in\mathcal{H},u\in\mathcal{U}} \left\{ R_{s,i}^{t}, R_{h,i}^{t}, R_{u,i}^{t} \right\}. \tag{24}$$

2.1.5. Energy Consumption Model

In the network scenario, LEO satellites, HAPs, and UAVs require power to move (flying and hovering) and transmit data. Since LEO satellites orbit around the earth, LEO satellites require power for transmitting data only. Moreover, HAPs are assumed to be floating, and therefore, they need power for flying and transmitting data only. Otherwise, UAVs require power for flying, hovering, and transmitting data. Therefore, the total energy consumption for transmitting data by using LEO satellites, HAPs, and UAVs can be calculated as

$$E_{TX}^{t} = \sum_{s\in\mathcal{S}} \int_0^t p_s^{\tau} d\tau + \sum_{h\in\mathcal{H}} \int_0^t p_h^{\tau} d\tau + \sum_{u\in\mathcal{U}} \int_0^t p_u^{\tau} d\tau \,. \tag{25}$$

According to the studying [38]-[39], the power consumption of HAPs and UAVs for flying and hovering can be assumed to be fixed. Therefore, the total energy consumption for flying and hovering of HAPs and UAVs is calculated as follows:

$$E_{MOVE}^{t} = \sum_{h\in\mathcal{H}} \int_0^t \delta_h^{\tau} P_{FLY}^{HAP} d\tau + \sum_{u\in\mathcal{U}} \int_0^t \left(\delta_u^{\tau} P_{FLY}^{UAV} + \left(1-\delta_u^{\tau}\right) P_{HOV}^{UAV} \right) d\tau, \tag{26}$$

where P_{FLY}^{HAP}, P_{FLY}^{UAV}, and P_{HOV}^{UAV} are the constant power consumption for flying of HAP, flying of UAVs, and hovering of UAVs, respectively. Here, $\delta_h^t \in \{0,1\}$ and $\delta_u^t \in \{0,1\}$ are the indicator function for indicating the flying HAP h and UAV u at time t, respectively.

2.2. Multiple Objectives Optimization

The purpose of this study is to investigate the multi-objectives of QoS optimization for energy efficiency of wireless coverage and fairness. Let us denote $R_{\min}$ as the minimum requirement data rate threshold for GUs. Therefore, the indicator that establish the connection for GU i at time t can be defined as

$$\alpha_i^t = \begin{cases} 1, & R_i^t \geq R_{\min} \\ 0, & \text{otherwise.} \end{cases} \tag{27}$$

In this study, we aim to optimize three QoS in SAGIN, as described as follow:

Objective #1: Base on the link indicator α_i^t, the number of connected GUs can be calculated as

$$N^t = \sum_{i \in \mathcal{I}} \alpha_i^t. \tag{28}$$

Then, the ratio of connected GUs can be calculated as

$$\bar{N}^t = \frac{N^t}{I}. \tag{29}$$

Objective #2: By using Jain's fairness formula, the data rate fairness of GU can be calculated as

$$\bar{J}^t = \frac{\left(\sum_{i \in \mathcal{I}} \alpha_i^t R_i^t\right)^2}{N^t \sum_{i \in \mathcal{I}} \left(\alpha_i^t R_i^t\right)^2}. \tag{30}$$

Objective #3: Moreover, we can calculate the total energy consumption for the whole network environment as follows:

$$E_{TOTAL}^t = E_{TX}^t + E_{MOVE}^t. \tag{31}$$

Then, the normalized total energy consumption can be formulated as

$$\bar{E}^t = \frac{E_{TOTAL}^t}{E_{\max}}. \tag{32}$$

2.3. Problem Formulation

Let us denote $\mathbf{L}^t = \left(L_h^t, L_u^t\right)$, $\mathbf{P}^t = \left(p_s^t, p_h^t, p_u^t\right)$, $\mathbf{B}^t = \left(\beta_s^t, \beta_h^t, \beta_u^t\right)$, and $\mathbf{E}^t = \left(E_i\left(\theta_{s,c}^t, \phi_{s,c}^t\right), E_i\left(\theta_h^t, \phi_h^t\right), E_i\left(\theta_u^t, \phi_u^t\right)\right)$ as the set of the control variables, such as the location, the transmit power, the bandwidth allocation, and the effective beam of LEO satellites, HAPs, and UAVs, respectively. In order to optimize the three QoS together, we use multi-object optimization problem by utilizing the scalarization technique with the weights ω_1, ω_2, and ω_3 for the coverage ratio, Jain's fairness of data rate, and the normalized energy consumption, respectively. Therefore, we can formulate the problem as follow:

$$\max_{\mathbf{L}^t, \mathbf{P}^t, \mathbf{B}^t, \mathbf{E}^t} \omega_1 \bar{N}^t + \omega_2 \bar{J}^t - \omega_3 \bar{E}^t \tag{33}$$

subjected to

$$\text{C1} : \left(x_h^t, y_h^t\right), \left(x_u^t, y_u^t\right), \left(x_i^t, y_i^t\right) \in \Omega,$$

$$\text{C2} : p_s^t \le P_{\max}^{LEO}, p_h^t \le P_{\max}^{HAP}, p_u^t \le P_{\max}^{UAV},$$

$$\text{C3} : \sum_{i \in \mathcal{I}} R_{h,i}^t \le \sum_{s \in \mathcal{S}} R_{s,h}^t, \sum_{i \in \mathcal{I}} R_{u,i}^t \le \sum_{s \in \mathcal{S}} R_{s,u}^t,$$

$$\text{C4} : \alpha_i^t = 1 \text{for } R_i^t \ge R_{\min}; \text{otherwise, } \alpha_i^t = 0,$$

$$\text{C5} : \beta_s^t, \beta_h^t, \beta_u^t \in [0,1],$$

$$\text{C6} : \theta_{s,c}^t, \theta_h^t, \theta_u^t \in [0, \Theta_{\max}],$$

$$\text{C7} : \phi_{s,c}^t, \phi_h^t, \phi_u^t \in [0, 2\pi),$$

$$\text{C8} : \omega_1 + \omega_2 + \omega_3 = 1,$$

$$\text{C9} : \omega_1, \omega_2, \omega_3 \ge 0,$$

$$\text{C10}: p_s^t, p_h^t, p_u^t \ge 0,$$

where $P_{\max}^{LEO}$, $P_{\max}^{HAP}$, and $P_{\max}^{UAV}$ are the maximum transmit power of LEO satellites, HAPs, and UAVs, respectively. And, $\Theta_{\max}$ is the maximum elevation angle of beam-forming.

3. Machine Learning

A branch of AI known as machine learning (ML) entails creating statistical models and algorithms that computers can use to learn from data and make inferences or predictions without human intervention, [20]. ML enables computers to gradually enhance their performance by learning from past experiences and analyzing data. ML algorithms can be categorized into supervised, unsupervised, and reinforcement learning based on the type of data available for training, [40]-[41]-[42]. Many applications rely on the algorithms of ML, including automatic systems, image recognition, optimization, natural language processing, etc.

Reinforcement Learning (RL) can be described as a Markov Decision Process (MDP) as the tuple of $\langle s^t, a^t, r^t, \Pr, \pi^t \rangle$, where s^t, a^t, r^t, $\Pr$, and π^t are the set of states, actions, rewards, transition probabilities, and policy, respectively, [43]. In RL or MDP, the agent RL learns how to take action based on the current state by using the policy to interact with the environment, which aims to achieve the optimal accumulated reward. Here, the state represents the current environment configuration, allowing the agent to make the decisions. This decision or action is made using the policy, which is the set of possible adjustments or movements that an agent can take within a given state. After taking action, the environment provides feedback that shows the outcomes (positive or negative experienced) of the action, including the next state and

the reward. With these experiences (states, actions, and rewards), the agent uses stochastics to model the dynamic of the environment by characterizing the probability of changing states after performing a specific action as the transition probability. This process continues until the system achieves the optimal policy.

In RL, the value functions (state value function, action value function, and advantage value function) have been used to model the transition probability corresponding to the dynamic environment, [44]. The value function is developed based on the expected value of discounted returns based on the given state and policy. The Monte Carlo methods, temporal difference methods, and function approximation methods have been used in order to evaluate and improve the model, [45]-[46]-[47]. While the environment becomes more complicated and the dimension of the data increases, RL is unable to solve the problem. As a result, DRL has been developed, which is the combination of the RL algorithms and deep neural networks (DNN). This combination has shown promising results in various fields such as robotics, gaming, and natural language processing. As shown in **Figure 4**, DRL can be categorized into centralized and decentralized DRL; centralized DRL involving a single decision-making entity receiving input from multiple sources; And, decentralized DRL involving multiple decision-making agents acting independently based on their local information.

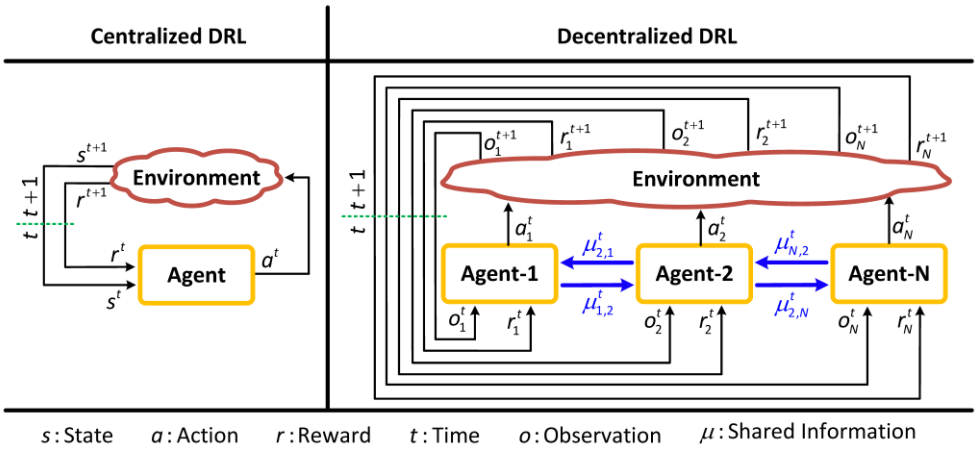


Figure 4: Centralized DRL vs. decentralized DRL.

3.1. Centralized DRL

Centralized DRL is a methodology in the field of artificial intelligence (AI) and ML that leverages a central decision-making agent to coordinate and control multiple decentralized agents. In this approach, a central agent observes the environment and makes decisions that interact with the environment directly. By centralizing the learning process, DRL can achieve improved network performance and efficient resource allocation, [48]. However, implementing centralized DRL also poses challenges such as increased computational complexity and potential privacy concerns due to the need for data sharing among devices.

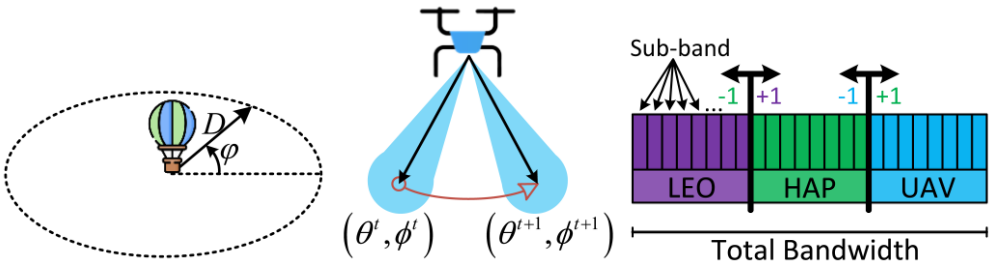
In this study, we consider the central agent located at the LEO satellite, which aims to manage the entire system of SAGIN. Furthermore, the key components that are important for the centralized DRL framework, which

consist of state, action, reward, replay memory, and error, are designed as follows:

State design: The state in DRL represents the current context or situation in which the agent operates. In general, state is designed based on the observable information from the system. Thus, we design the state space as

$$s^t \in \mathcal{S} = \left\{ \hat{g}_{s,h}^t, \hat{g}_{s,u}^t, \hat{g}_{s,i}^t, \hat{g}_{h,i}^t, \hat{g}_{u,i}^t, \alpha_i^t, \mathbf{L}^{t-1}, \mathbf{P}^{t-1}, \mathbf{E}^{t-1}, \mathbf{B}^{t-1} \right\}, \tag{34}$$

where $\hat{g}_{s,h}^t = \left| g_{s,h}^t \right|^2$, $\hat{g}_{s,u}^t = \left| g_{s,u}^t \right|^2$, $\hat{g}_{s,i}^t = \left| g_{s,i}^t \right|^2$, $\hat{g}_{h,i}^t = \left| g_{h,i}^t \right|^2$, and $\hat{g}_{u,i}^t = \left| g_{u,i}^t \right|^2$ are the effective channel gain from LEO satellite s to HAP h, LEO satellite s to UAV u, LEO satellite s to GU i, HAP h to GU i, and UAV u to GU i, respectively. Here, $\mathbf{L}^{t-1}$, $\mathbf{P}^{t-1}$, $\mathbf{E}^{t-1}$, and $\mathbf{B}^{t-1}$ represent the set of the locations of HAPs and UAV; the transmit power of LEO satellites and HAPs, and UAV; the beam-forming of LEO satellite, HAPs, and UAVs; and the bandwidth allocation for LEO satellite, HAPs, and UAV at the previous time step $(t-1)$, respectively.



(a) Location adjustment. **(b)** Beam adjustment. **(c)** Bandwidth adjustment.

Figure 5: The action of DRL.

Action design: The action in DRL represents as the potential choices available for the agent within a specific state. In general, the action space is designed to adjust all controllable variables. In our problem, the location, transmit power, beam-forming pattern, and bandwidth allocation are required to adjust in order to maximize the objective function. It is important to note that the action can be continuous or discrete, depending on the architecture of the algorithm of DRL that utilized. In this study, we define the action space as

$$a^t \in \mathcal{A} = \left\{ \mathbf{a}\left(\mathbf{L}^t\right), \mathbf{a}\left(\mathbf{P}^t\right), \mathbf{a}\left(\mathbf{E}^t\right), \mathbf{a}\left(\mathbf{B}^t\right) \right\}. \tag{35}$$

Here, the action $\mathbf{a}\left(\mathbf{L}^t\right)$ represents the adjustment of the location for HAPs and UAVs, which include flying and hovering. The adjustment of the location for HAPs and UAVs are characterized by moving direction with angle $\varphi \in [0, 2\pi)$ and the moving distance $D \in \left\{0, D_{\max}\right\}$, where $D_{\max}$ is the maximum distance proportional to the maximum flying speed. **Figure 5a** depicts the example of the adjustment of the location for a HAP. And, the action $\mathbf{a}\left(\mathbf{P}^t\right)$ represents the adjustment of the transmit power for LEO satellites, HAPs, and UAVs. The transmit power are quantified into $(L+1)$ portions, where each portion of the transmit power for LEO, HAP, and UAV are $\frac{1}{L} P_{\max}^{LEO}$, $\frac{1}{L} P_{\max}^{HAP}$, and $\frac{1}{L} P_{\max}^{UAV}$, respectively. Thus, the action $\mathbf{a}\left(\mathbf{P}^t\right)$ is the adjustment by increasing or decreasing the transmit power by one portion. And, the action $\mathbf{a}\left(\mathbf{E}^t\right)$

represents the adjustment of the beamforming angles for LEO satellite and aerial, characterized by an azimuth angle $\phi \in [0, 2\pi)$ and an elevation angle $\theta \in [0, \Theta_{\max}]$, as illustrated in **Figure 5b**. And, the action $\mathbf{a}(\mathbf{B}^t)$ represents the adjustment of bandwidth allocation for LEO satellite and aerial. Initially, the total bandwidth B is divided into $(M+1)$ sub-bands, allowing for adjustments by either increasing or decreasing one sub-band, as illustrated in **Figure 5c**.

Reward design: In DRL, the reward is serving as feedback based on the agent's actions within a state, guiding future decision-making. In this study, the reward is representing the improvement of the objective function. Thus, we design the reward function as the objective function as follow

$$r^t \in \mathcal{R} = \begin{cases} \left(\omega_1 \bar{N}^t + \omega_2 \bar{J}^t - \omega_3 \bar{E}^t\right) & \text{, satisfied C1-C10,} \\ 0 & \text{, otherwise.} \end{cases} \tag{36}$$

Replay memory: The replay memory is an essential data structure that stores past agent experiences that facilitating learning and decision improvement through experienced replay. Generally, the replay memory is storing the tuple of the current state s^t, action a^t, next state s^{t+1}, and reward r^t.

Temporal difference (TD) error: TD error in RL is a measure of the difference between the predicted value of a state or action and the actual observed value. It is used to update the value function in RL algorithms by

adjusting predictions based on the difference between expected and actual rewards received. The TD error equation can be expressed as

$$\left[r^t + \gamma V\left(s^{t+1}\right)\right] - V\left(s^t\right). \tag{37}$$

Here, $r^t + \gamma V\left(s^{t+1}\right)$ is the target value; and, $V\left(s^t\right)$ is the current state value. Additionally, γ is the discount factor.

3.1.1. Single Agent Deep Q-Learning

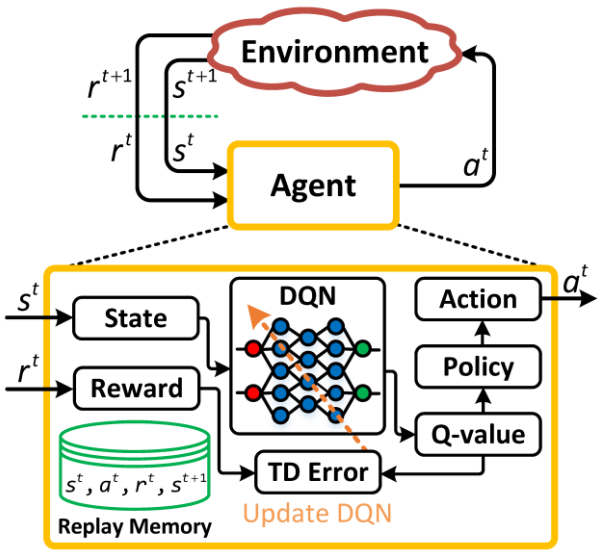


Figure 6: Centralized DQL architecture.

The Single-Agent DQL (SA-DQL) is a type of ML approach that involves training a central agent to make decisions for all controllable devices within the system [49], as shown in **Figure 6**. This centralized approach allows the central agent to aggregate data from all devices, enabling more accurate information and comprehensive learning. The central agent then utilizes this collective knowledge to make decisions, leading to improved network

performance and resource allocation. The main components in SA-DQL can be described as following.

Action value function: The action value function or the Q-value function is a function estimation of the expected cumulative reward that achievable by taking a particular action within a given state, crucial for guiding the agent's decisions. An update to the Q-value algorithm relies on the Bellman equation, a basic value iteration update that takes into account both the current value and new information through a weighted average:

$$Q^{\text{new}}\left(s^t, a^t\right) = \left(1-\eta\right) Q^{\text{old}}\left(s^t, a^t\right) + \eta\left(r + \gamma \max_{a \in A} Q\left(s^t, a\right)\right), \tag{38}$$

where η and γ are the learning rate and is the discount factor, respectively.

Deep Q-network (DQN): In Q-learning, the action is selected based on the Q-table that stores the Q-values mapping with the specific actions $\left\{Q\left(s^t, a_1\right), Q\left(s^t, a_2\right), \ldots\right\}$. When the state and action space increase, the amount of memory needed to store them also increases exponentially. Therefore, a designed deep neural network called DQN is used to approximate Q-value functions, where the input of DQN is the state and the output is the Q-value that corresponds with the specific action. Therefore, the Q-value function can be approximated as DQN with weights $\boldsymbol{\theta}$ as follows:

$$Q\left(s^t, a^t\right) \approx Q\left(s^t, a^t; \boldsymbol{\theta}\right). \tag{39}$$

Here, the DQN is trained to minimize the loss function as follow:

$$L(\boldsymbol{\theta}) = \mathbb{E}\left[\left(y^{\text{targ}} - Q\left(s^t, a^t; \boldsymbol{\theta}\right)\right)^2\right], \tag{40}$$

where $y^{\text{targ}} = r^t + \gamma \max_{a \in \mathcal{A}} Q\left(s^t, a; \boldsymbol{\theta}\right)$ represents the target of the Q-value. Notice that because of the mapping between Q-values and actions, the actions must be discrete value.

Policy: The ϵ-greedy policy is a simple and popular method that uses the stochastic policy approach to balance exploration and exploitation by choosing the next action with a probability of ϵ as follows:

$$a^{t+1} = \begin{cases} \underset{a \in \mathcal{A}}{\text{argmax}}\, Q\left(s^t, a; \boldsymbol{\theta}\right) & , \text{ with Prob. } (1-\epsilon) \\ \text{Random action} & , \text{ with Prob.}(\epsilon). \end{cases} \tag{41}$$

The following **Algorithm 1** represents the pseudocode for SA-DQL.

Algorithm 1: Single agent deep Q-learning.

INITIALIZATION:
  Weight of DQN, $\boldsymbol{\theta}$
  Replay memory, $\mathcal{D}$
WHILE not converge **DO**
  OBSERVE state s^t
  SELECT and **EXECUTE** action a^t using ϵ - policy
  OBSERVE:
    Next state s^{t+1}
    Reward r^t
    STORE $\left(s^t, a^t, r^t, s^{t+1}\right)$ in $\mathcal{D}$
  IF s^{t+1} is terminal **THEN**
    RESET environment
  IF $\mathcal{D}$ is full **THEN**
    SAMPLE mini-batch, $\mathcal{B} = \left\{\left(s^t, a^t, r^t, s^{t+1}\right)\right\}$ from $\mathcal{D}$
    COMPUTE target value

$$y^{\text{targ}} = r^t + \gamma \max_{a \in \mathcal{A}} Q\left(s^t, a; \boldsymbol{\theta}\right)$$

    COMPUTE gradient of loss function

$$\nabla_{\boldsymbol{\theta}} L\left(\boldsymbol{\theta}\right) = \nabla_{\boldsymbol{\theta}} \mathbb{E}\left[\left(y^{\text{targ}} - Q\left(s^t, a^t; \boldsymbol{\theta}\right)\right)^2\right]$$

    UPDATE DQN with learning rate η

$$\boldsymbol{\theta}^{\text{new}} \leftarrow \boldsymbol{\theta}^{\text{old}} - \eta \nabla_{\boldsymbol{\theta}} L\left(\boldsymbol{\theta}\right)$$

3.1.2. Single Agent Deep Actor Critic

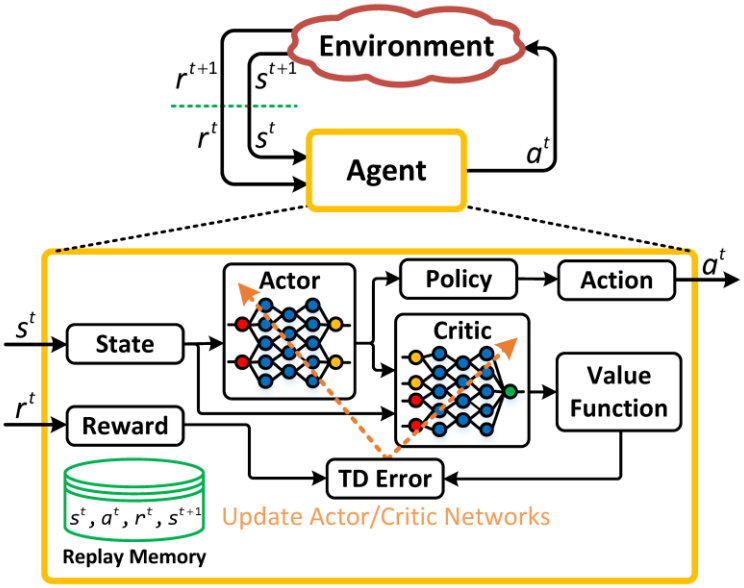


Figure 7: Centralized DAC architecture.

The single-agent DAC (SA-DAC) algorithm overcomes the limitations of centralized DQL by introducing a policy-based and value-based approach, [50]. It utilizes two DNNs: an actor network and a critic network, to effectively handle both discrete and continuous high-dimensional action spaces in DQL. This is illustrated in **Figure 7**. The actor network is responsible for selecting actions based on the current state, while the critic network evaluates the chosen actions. By combining these two networks, DAC provides a more comprehensive solution for RL tasks with complex action spaces. The main components of SA-DAC can be described as follows.

Actor network: In DAC learning, the actor network is responsible for selecting actions based on the current state of the environment. The actor network is employed to explore a policy π, that maps the state space $\mathcal{S}$ to the action space $\mathcal{A}$. Here, the policy π is a function of the current state s^t and is parameterized by $\boldsymbol{\theta}_{actor}$. Furthermore, the chosen action based on policy π can be expressed as $\pi\left(a \mid s^t; \boldsymbol{\theta}_{actor}\right)$. Notice that the actor network uses the policy function to update the weight parameters $\boldsymbol{\theta}_{actor}$ and improve the selection of actions based on the feedback received from the critic network. Here the policy function can be expressed as

$$P\left(\boldsymbol{\theta}_{actor}\right) = \mathbb{E}\left[\log \pi\left(a \mid s^t; \boldsymbol{\theta}_{actor}\right) \sum_{t=1}^{T} \gamma^t r^t \right], \tag{42}$$

where $\gamma \in (0,1)$ is denoted as the discount factor.

Critic network: The critic network plays a crucial role in DAC as it provides feedback on the quality of selected actions. The critic is employed to estimate the value function $V\left(s_t, \boldsymbol{\theta}_{critic}\right)$ with the weight parameters $\boldsymbol{\theta}_{critic}$. When the actor network selects an action a_t, the agent executes it in the environment and sends the current observation s_t to the critic, along with feedback from the environment. These feedbacks include the reward r_t and the next time state, s_{t+1}. Thus, the target value can be calculated as

$$y^{\text{targ}} = r^t + \gamma V\left(s^{t+1}; \boldsymbol{\theta}_{critic}\right). \tag{43}$$

In order to evaluate and update the value function, the TD error or the loss function have been used. Here, the loss function can be expressed as

$$L\left(\boldsymbol{\theta}_c\right) = \mathbb{E}\left[\left(y^{\text{targ}} - V\left(s^t; \boldsymbol{\theta}_{critic}\right)\right)^2\right]. \tag{44}$$

The following **Algorithm 2** represents the pseudocode of SA-DAC algorithm.

Algorithm 2: Single agent deep actor critic.

INITIALIZATION:
  Weight of actor network, $\boldsymbol{\theta}_{actor}$
  Weight of critic network, $\boldsymbol{\theta}_{critic}$
  Replay memory, $\mathcal{D}$
WHILE not converge **DO**
  OBSERVE state s^t
  SELECT and **EXECUTE** action a^t
  OBSERVE:
    Next state s^{t+1}
    Reward r^t
    STORE $\left(s^t, a^t, r^t, s^{t+1}\right)$ in $\mathcal{D}$
  IF s^{t+1} is terminal **THEN**
    RESET environment
  IF $\mathcal{D}$ is full **THEN**
    SAMPLE mini-batch $\mathcal{B} = \left\{\left(s^t, a^t, r^t, s^{t+1}\right)\right\}$ from $\mathcal{D}$
    COMPUTE target value

$$y^{\text{targ}} = r^t + \gamma V\left(s^{t+1}; \boldsymbol{\theta}_{critic}\right)$$

    COMPUTE gradient of loss function

$$\nabla_{\boldsymbol{\theta}_{critic}} L\left(\boldsymbol{\theta}_{critic}\right) = \nabla_{\boldsymbol{\theta}_{critic}} \mathbb{E}\left[\left(y^{\text{targ}} - V\left(s^t; \boldsymbol{\theta}_{critic}\right)\right)^2\right]$$

    COMPUTE gradient of policy function

$$\nabla_{\boldsymbol{\theta}_{actor}} P\left(\boldsymbol{\theta}_{actor}\right) = \nabla_{\boldsymbol{\theta}_{actor}} \mathbb{E}\left[\log \pi\left(a \mid s^t; \boldsymbol{\theta}_{actor}\right) \sum_{t=1}^{T} \gamma^t r^t\right]$$

    UPDATE actor and critic network with learning rate η

$$\boldsymbol{\theta}_{critic}^{\text{new}} \leftarrow \boldsymbol{\theta}_{critic}^{\text{old}} - \eta \nabla_{\boldsymbol{\theta}_{critic}} L\left(\boldsymbol{\theta}_{critic}\right)$$

$$\boldsymbol{\theta}_{actor}^{\text{new}} \leftarrow \boldsymbol{\theta}_{actor}^{\text{old}} - \eta \nabla_{\boldsymbol{\theta}_{actor}} P\left(\boldsymbol{\theta}_{actor}\right)$$

3.1.3. Single Agent Deep Deterministic Policy Gradient

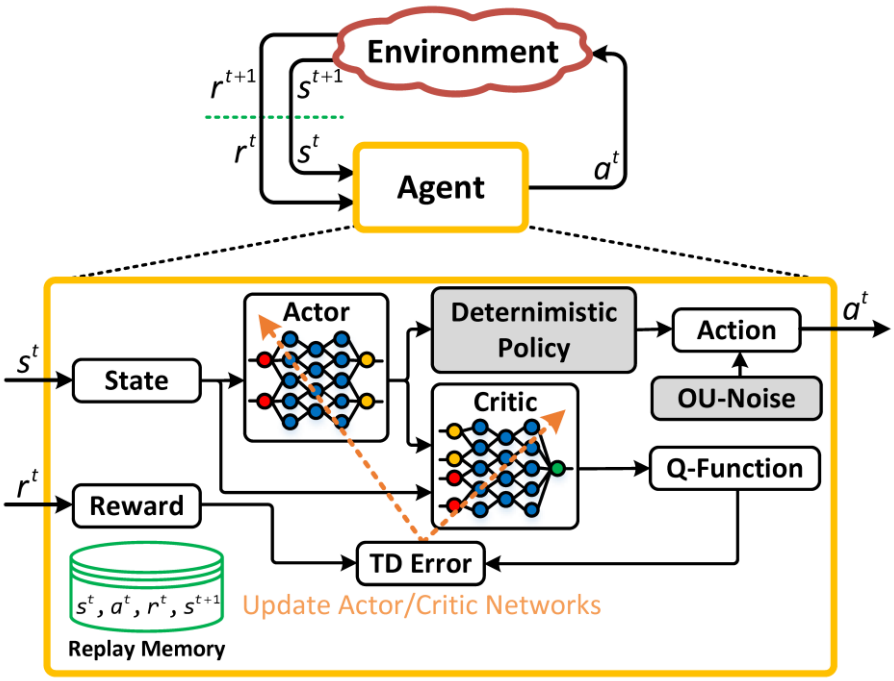


Figure 8: Centralized DDPG architecture.

The single-agent deep deterministic policy gradient (SA-DDPG) is a RL algorithm that combines the deterministic policy gradient (DPG) and DQL [51], as shown in **Figure 8**. Here, DPG uses a deterministic policy function to select actions in continuous domains, where it maps a state to an action without any randomness. DPG learns the policy function by following the gradient of the action-value function, or Q-value, which estimates the expected return of taking an action in a state. The main components in SA-DDPG can be described as follows.

Actor and critic network: Similarly, DDPG utilizes actor and critic networks which are processed similarly to the DAC algorithm. Here, the target value function is calculated as

$$y^{\text{targ}} = r^t + \gamma \max_{a \in \mathcal{A}} Q\left(s^t, a; \boldsymbol{\theta}_{\text{critic}}\right). \tag{45}$$

Furthermore, to update the weight parameters of the critic network, the loss function is used, which can be expressed as

$$L\left(\boldsymbol{\theta}_{\text{c}}\right) = \mathbb{E}\left[\left(y^{\text{targ}} - Q\left(s^t, a^t; \boldsymbol{\theta}_{\text{critic}}\right)\right)^2\right]. \tag{46}$$

Moreover, to update the weight parameters of the actor network, the policy function is utilized, which can be expressed as

$$P\left(\boldsymbol{\theta}_{\text{actor}}\right) = Q\left(s^t, a^t; \boldsymbol{\theta}_{\text{actor}}\right) \pi\left(a \mid s^t; \boldsymbol{\theta}_{\text{actor}}\right). \tag{47}$$

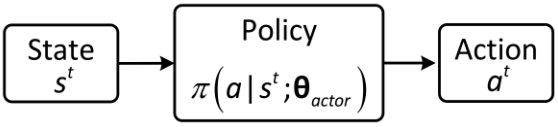
Ornstein–Uhlenbeck (OU) noise: OU noise is a random process that effectively guides the trained agent towards an optimal value. In RL algorithms, especially in continuous control tasks, OU noise is frequently employed to include exploration and smoothness in the action of the agent. This method helps the agent prevent getting stuck in local minima during training. The Ornstein-Uhlenbeck process x_t is defined by the following stochastic differential equation

$$\frac{dx^t}{dt} = -\theta x^t + \sigma \eta(t), \tag{48}$$

where $\theta > 0$ and $\sigma > 0$ are constant parameters representing the mean and the noise scale, respectively, and $\eta(t)$ is denoted as the white noise.

Deterministic policy: In DDPG, a deterministic policy refers to a policy that directly maps states to actions without any randomness or stochastic, as show in **Figure 9**. Unlike stochastic policies, which output a probability distribution over actions, deterministic policies output a single action for each state. The following **Algorithm 3** is the pseudocode for single-agent deep deterministic policy gradient learning.

(a) Stochastic Policy.



(b) Deterministic Policy.

Figure 9: Stochastic vs. deterministic policy.

Algorithm 3: Single agent deep deterministic policy gradient.

1 : **INITIALIZATION**:
2 : Weight of actor network, $\boldsymbol{\theta}_{actor}$
3 : Weight of critic network, $\boldsymbol{\theta}_{critic}$
4 : Replay memory, $\mathcal{D}$
5 : **WHILE** not convergence **DO**
6 : **OBSERVE** state s^t
7 : **SELECT** and **EXECUTE** action a^t by adding OU-noise
8 : **OBSERVE**:
9 : Next state s^{t+1}
10 : Reward r^t
11 : **Store** $\left(s^t, a^t, r^t, s^{t+1}\right)$ in $\mathcal{D}$
12 : **IF** s^{t+1} is terminal **THEN**
13 : **RESET** environment
14 : **IF** $\mathcal{D}$ is full **THEN**
15 : **SAMPLE** mini-batch $\mathcal{B} = \left\{\left(s^t, a^t, r^t, s^{t+1}\right)\right\}$ from $\mathcal{D}$
16 : **COMPUTE** target value

$$y^{\text{targ}} = r^t + \gamma \max_{a \in \mathcal{A}} Q\left(s^t, a; \boldsymbol{\theta}_{critic}\right)$$

17 : **COMPUTE** gradient of loss function

$$\nabla_{\boldsymbol{\theta}_{critic}} L\left(\boldsymbol{\theta}_{critic}\right) = \nabla_{\boldsymbol{\theta}_{critic}} \mathbb{E}\left[\left(y^{\text{targ}} - Q\left(s^t, a^t; \boldsymbol{\theta}_{critic}\right)\right)^2\right]$$

18 : **COMPUTE** gradient of policy function

$$\nabla_{\boldsymbol{\theta}_{actor}} P\left(\boldsymbol{\theta}_{actor}\right) = \nabla_{\boldsymbol{\theta}_{actor}} Q\left(s^t, a^t; \boldsymbol{\theta}_{actor}\right) \pi\left(a \mid s^t; \boldsymbol{\theta}_{actor}\right)$$

19 : **UPDATE** actor and critic network with learning rate η

$$\boldsymbol{\theta}_{critic}^{\text{new}} \leftarrow \boldsymbol{\theta}_{critic}^{\text{old}} - \eta \nabla_{\boldsymbol{\theta}_{critic}} L\left(\boldsymbol{\theta}_{critic}\right)$$

$$\boldsymbol{\theta}_{actor}^{\text{new}} \leftarrow \boldsymbol{\theta}_{actor}^{\text{old}} - \eta \nabla_{\boldsymbol{\theta}_{actor}} P\left(\boldsymbol{\theta}_{actor}\right)$$

3.2. Decentralized DRL

Decentralized DRL is a framework in which multiple agents, each with their own policies and value functions, interact in a shared environment to jointly achieve the same objective, [52]-[53]. Decentralized DRL, in contrast to centralized DRL, distributes the decision-making of individual agents to make independent decisions based on their local observations. Even though each agent in a decentralized DRL operates independently, they often communicate and coordinate with each other, either through direct messages or by sharing information about the environment. Decentralized DRL enables agents to develop both individual strategies and collaborative behaviors, resulting in the formation of cohesive actions within a complex and ever-changing environment. This is particularly beneficial in scenarios involving multi-agent systems that rely on decentralized decision-making.

On the other hand, the design of multi-agent DRL is challenging due to the unexpected behavior (lack of synchronization) and coordination of each agent. Moreover, not all agents can receive the shared information from other agents due to the long communication distances. One of the most challenging aspects of multi-agent DRL is the difficulty in achieving convergence due to the non-stationary nature of the system. Additionally, it requires very high time complexity to achieve convergence with high optimality compared to centralized DRL. Therefore, this study aims to propose a design for multi-agent

DRL that can achieve better performance than centralized DRL in terms of optimality and convergence.

3.2.1. Proposed Decentralized DRL

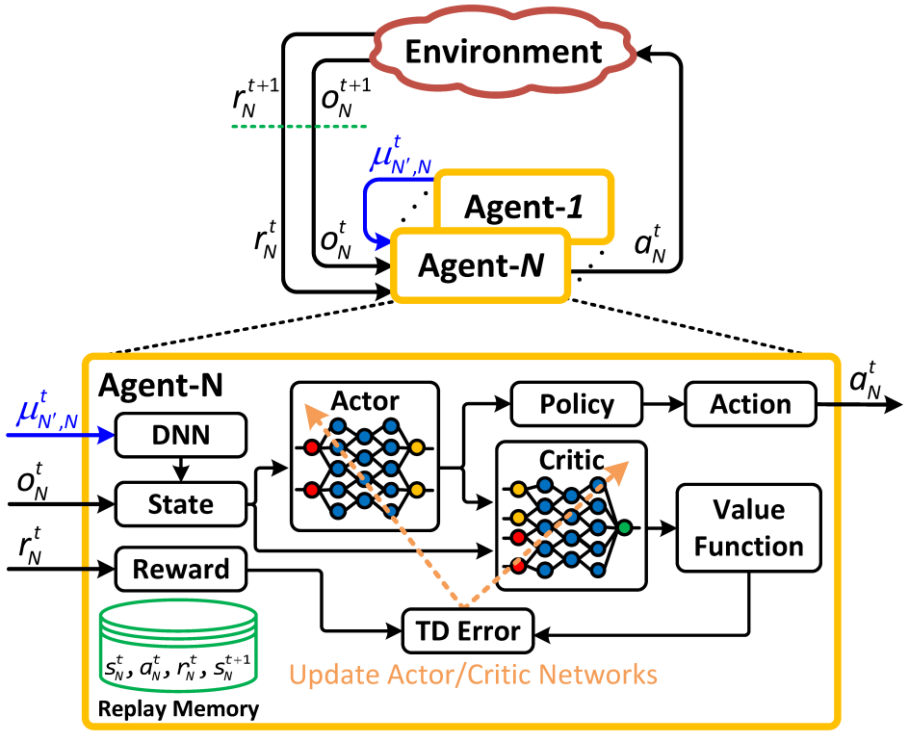


Figure 10: Proposed multi-agent DAC architecture.

In this section, we conducted research on the architecture of the multi-agent DAC (MA-DAC) following the study in [54]. As illustrated in **Figure 10**, multiple agents aim to optimize network performance through a decentralized approach. Each agent has its own actor and critic network, enabling it to learn from the environment and other agents. In this architecture, we assume that all agents communicate and share information with each other in real-time, aiming to enhance their decision-making abilities and synchronize

with each other. This framework enables agents to adapt and respond to dynamic environments more effectively, making it suitable for real-world applications with complex environments.

3.2.1.1. Proposed State Design

In this study, we assume that all agents are located on LEO satellites, HAPs, and UAVs. Denote agent s, agent h, and agent u as the agent that located on LEO satellite s, HAP h, and UAV u, respectively. It is important to note that the agent s is different from the agents h and agent u. For example, agent s adjusts the transmission power, spot-beam center, and bandwidth allocation. In contrast, agent h and agent u are required to adjust not only the transmitting parameters but also their movement.

In decentralized architecture, all agents are required to observe the information and take action from the environment and other agents. Therefore, the agent s requires to observe the information from environment and observe the information from other agents. The real-time observable information from the environment includes the channel from LEO satellites to HAPs and UAVs $\left(\hat{g}_{s,h}^{t} \text{ and } \hat{g}_{s,u}^{t}\right)$, the transmit power $\left(p_s^t\right)$, the spot-beam center $\left(E_i\left(\theta_{s,c}^{t}, \phi_{s,c}^{t}\right)\right)$, and the bandwidth allocation $\left(\beta_s^t\right)$. Thus, the observable information from the environment of the agent s at time t can be denoted as

$$o_s^t = \left\{ \hat{g}_{s,h}^t, \hat{g}_{s,u}^t, p_s^t, E_i \left(\theta_{s,c}^t, \phi_{s,c}^t \right), \beta_s^t \right\}. \tag{49}$$

Notice that all LEO satellites are shared their information with each other. Moreover, the agent h and the agent u share their previous information with the agent s. Here, the information from the previous time slot of HAPs and UAVs, such as the location $\left(L_h^{t-1} \text{ and } L_u^{t-1} \right)$, the channel gain $\left(\hat{g}_{h,i}^{t-1} \text{ and } \hat{g}_{u,i}^{t-1} \right)$, the connected GU indicator $\left(\alpha_i^{t-1} \right)$, the transmit power $\left(p_h^{t-1} \text{ and } p_u^{t-1} \right)$, the beamforming angle $\left(E_i \left(\theta_h^{t-1}, \phi_h^{t-1} \right) \text{ and } E_i \left(\theta_u^{t-1}, \phi_u^{t-1} \right) \right)$, and the bandwidth allocation $\left(\beta_h^{t-1} \text{ and } \beta_u^{t-1} \right)$, are shared with the agent s. Thus, the observable information from another agent can be denoted as

$$\mu_s^{t-1} = \left\{ L_h^{t-1}, L_u^{t-1}, \hat{g}_{h,i}^{t-1}, \hat{g}_{u,i}^{t-1}, \alpha_i^{t-1}, p_h^{t-1}, p_u^{t-1}, E_i \left(\theta_h^{t-1}, \phi_h^{t-1} \right), E_i \left(\theta_u^{t-1}, \phi_u^{t-1} \right), \beta_h^{t-1}, \beta_u^{t-1} \right\}. \tag{50}$$

Therefore, the state of the agent s on the LEO satellite s at time t can be designed as

$$s_s^t \in \mathbb{S}_s = \left\{ o_s^t, \mathrm{DNN}\left(\mu_s^{t-1} \right) \right\}, \tag{51}$$

where $\mathrm{DNN}\left(\mu_s^{t-1} \right)$ represents the deep neural network that estimate the current information based on the previous information.

The agent h and agent u can observe information from both the environment and other agents. The real-time observable information from the environment includes the channel from LEO satellites to HAPs/UAVs and HAPs/UAVs to GUs $\left(\hat{g}_{s,h}^t, \hat{g}_{s,u}^t, \hat{g}_{h,i}^t \text{ and } \hat{g}_{u,i}^t \right)$, the transmit power of

HAPs/UAVs $\left(p_h^t \text{ and } p_u^t\right)$, the beam-forming $\left(E_i\left(\theta_h^t, \phi_h^t\right) \text{ and } E_i\left(\theta_u^t, \phi_u^t\right)\right)$, and the bandwidth allocation $\left(\beta_h^t \text{ and } \beta_u^t\right)$. Thus, the observable information from the environment of the agent h and agent u can be denoted as

$$o_h^t = \left\{\hat{g}_{s,h}^t, \hat{g}_{h,i}^t, p_h^t, E_i\left(\theta_h^t, \phi_h^t\right), \beta_h^t\right\}, \tag{52}$$

and

$$o_u^t = \left\{\hat{g}_{s,u}^t, \hat{g}_{u,i}^t, p_u^t, E_i\left(\theta_u^t, \phi_u^t\right), \beta_u^t\right\}. \tag{53}$$

Moreover, the agent s located on LEO satellite s shares their previous information with the agent h and agent u located on HAP h and UAV u. Here, the information from the previous time slot, such as the channel from LEO satellite s to HAPs/UAVs and HAPs/UAVs to GUs $\left(\hat{g}_{s,h}^{t-1}, \hat{g}_{s,u}^{t-1}, \hat{g}_{h,i}^{t-1} \text{ and } \hat{g}_{u,i}^{t-1}\right)$ the connected GU indicator $\left(\alpha_i^{t-1}\right)$, the transmit power $\left(p_s^{t-1}, p_h^{t-1} \text{ and } p_u^{t-1}\right)$, the beamforming angle $\left(E_i\left(\theta_{s,c}^{t-1}, \phi_{s,c}^{t-1}\right),\right.$ $\left.E_i\left(\theta_h^{t-1}, \phi_h^{t-1}\right), \text{ and } E_i\left(\theta_u^{t-1}, \phi_u^{t-1}\right)\right)$, and the bandwidth allocation $\left(\beta_s^{t-1}, \beta_h^{t-1}\right.$ $\left., \text{ and } \beta_u^{t-1}\right)$, are shared with the agent h and agent u. Thus, the observable information from another agent of the agent h and agent u at previous time can be designed as

$$\mu_h^{t-1} = \left\{\hat{g}_{s,u}^{t-1}, \hat{g}_{u,i}^{t-1}, \alpha_i^{t-1}, p_s^{t-1}, E_i\left(\theta_{s,c}^{t-1}, \phi_{s,c}^{t-1}\right), \beta_s^{t-1}\right\}. \tag{54}$$

and

$$\mu_u^{t-1} = \left\{\hat{g}_{s,h}^{t-1}, \hat{g}_{h,i}^{t-1}, \alpha_i^{t-1}, p_s^{t-1}, E_i\left(\theta_{s,c}^{t-1}, \phi_{s,c}^{t-1}\right), \beta_s^{t-1}\right\}. \tag{55}$$

Therefore, the state of the agent h and agent u at time t can be formulated as

$$s_h^t \in \mathbb{S}_h = \left\{ o_h^t, \mathrm{DNN}\left(\mu_h^{t-1}\right) \right\}, \tag{56}$$

and

$$s_u^t \in \mathbb{S}_u = \left\{ o_u^t, \mathrm{DNN}\left(\mu_u^{t-1}\right) \right\}, \tag{57}$$

where $\mathrm{DNN}\left(\mu_h^{t-1}\right)$ and $\mathrm{DNN}\left(\mu_u^{t-1}\right)$ are the deep neural network that estimate the current information based on the previous information.

3.2.1.2. Proposed Action Design

Similar to the centralized ML, the action of agent s at time t is designed as

$$a_s^t \in \mathbb{A}_s = \left\{ \mathbf{a}\left(p_s^t\right), \mathbf{a}\left(E_i\left(\theta_{s,c}^t, \phi_{s,c}^t\right)\right), \mathbf{a}\left(\beta_s^t\right) \right\}, \tag{58}$$

where $\mathbf{a}\left(p_s^t\right)$, $\mathbf{a}\left(E_i\left(\theta_{s,c}^t, \phi_{s,c}^t\right)\right)$, and $\mathbf{a}\left(\beta_s^t\right)$ are the adjustment of the transmit power, spot-beam angle, and bandwidth allocation, respectively. Moreover, the action of agent h and agent u at time t are designed as

$$a_h^t \in \mathbb{A}_h = \left\{ \mathbf{a}\left(L_h^t\right), \mathbf{a}\left(p_h^t\right), \mathbf{a}\left(E_i\left(\theta_h^t, \phi_h^t\right)\right), \mathbf{a}\left(\beta_h^t\right) \right\}, \tag{59}$$

and

$$a_u^t \in \mathbb{A}_u = \left\{ \mathbf{a}\left(L_u^t\right), \mathbf{a}\left(p_u^t\right), \mathbf{a}\left(E_i\left(\theta_u^t, \phi_u^t\right)\right), \mathbf{a}\left(\beta_u^t\right) \right\}, \tag{60}$$

where $\mathbf{a}\left(L_h^t\right)$, $\mathbf{a}\left(L_u^t\right)$, $\mathbf{a}\left(p_h^t\right)$, $\mathbf{a}\left(p_u^t\right)$, $\mathbf{a}\left(E_i\left(\theta_h^t,\phi_h^t\right)\right)$, $\mathbf{a}\left(E_i\left(\theta_u^t,\phi_u^t\right)\right)$, $\mathbf{a}\left(\beta_h^t\right)$, and $\mathbf{a}\left(\beta_h^t\right)$ are the adjustable variables for location of HAPs/UAVs, transmit power of HAPs/UAVs, beam-forming of HAPs/UAVs, and bandwidth allocation of HAPs/UAVs, respectively.

3.2.1.3. Proposed Reward Design

In this study, we design the reward function as the combination of the objective function with the multi-variate normal distribution as follow

$$r^t \in \mathcal{R}^t = \begin{cases} \left(\omega_1 \bar{N}^t + \omega_2 \bar{J}^t - \omega_3 \bar{E}^t\right) \bar{f}_{\vec{\mu}_i}\left(\vec{x}\right) & \text{, satisfied C1-C10,} \\ 0 & \text{, otherwise,} \end{cases} \tag{61}$$

where $\bar{f}_{\vec{\mu}_i}\left(\vec{x}\right) = \dfrac{\sum_{i\in\mathcal{I}} f_{\vec{\mu}_i}\left(\vec{x}\right)}{\max\left(\sum_{i\in\mathcal{I}} f_{\vec{\mu}_i}\left(\vec{x}\right)\right)}$ represents the normalized Probability Density Function (PDF) of the multi-variate normal distribution. Here, the PDF of the multi-variate normal distribution can be defined as

$$f_{\vec{\mu}_i}\left(\vec{x}\right) = \frac{\exp\left(-\frac{1}{2}\left(\vec{x}-\vec{\mu}_i\right)^T \Sigma^{-1}\left(\vec{x}-\vec{\mu}_i\right)\right)}{2\pi\sqrt{|\Sigma|}}, \tag{62}$$

where $\vec{\mu}_i = \left(x_i, y_i\right)$ is the mean vector represents the location of GUs, and Σ is the symmetric covariance matrix which is positive definite, where the standard deviation σ_x^2 and σ_y^2 on x-axis and y-axis, respectively. Notice that

$|\Sigma| = \det \Sigma$ is the determinant of Σ also known as the generalized variance. Hint, the high of the density of GUs correspond to the high reward function.

3.2.1.4. Proposed Experienced DNN Policy

A stochastic policy is an effective technique to enhance learning adaptability (exploration and exploitation) and performance in DRL agents by using probabilities to select actions for a given state. In DRL, several designs of stochastic policies include:

- **Epsilon-greedy** [55]: Utilizes a small probability (epsilon) to select a random action for exploration and a specific action for exploitation.
- **SoftMax** [56]: Assigns probabilities to each potential action based on their estimated values to improve the exploration rate.
- **Boltzmann exploration** [57]: Utilizes the SoftMax function to assign probabilities to actions, balancing between exploration and exploitation.
- **Gaussian exploration** [58]: Adds noise to the action selection process to explore different actions with higher expected values.
- **Novelty-based exploration** [59]: Increases the probability of selecting unexplored states to improve the efficiency of discovering novel solutions.

- **DNN** [60]: Uses DNNs as function approximators for the probability of action in a stochastic policy. The DNN policy offers a significant advantage in identifying complex environmental patterns.

Based on our experienced, we have noticed that the agent usually chooses actions that lead to high computation in order to find the optimal solution. To address this issue, we propose a simple approach called the experienced DNN policy. The experienced DNN policy utilizes a DNN architecture, as illustrated in **Figure 11**. At the beginning of training, we initialized the weight parameters of the experienced DNN as zero, which represents that the agent does not have any experienced. After further training, the agent will identify these experiences for exploration more efficiently, following the probability of all previous actions. Here, the design of DNN can be described as follows: The input layer of experienced DNN represents the state of an agent; the hidden layer is composed of fully connected neurons with a sigmoid activation function; and the output layer determines the stochastic characteristic of the action by indicating its probability of selected action. When the action is detected, the experienced of DNN will decrease the probability of the specific action from the beginning of the training and adjustment parameters. The purpose of implementing the experienced DNN policy is to enhance the decision-making process and reduce training time. With the coordination of replay memory, this approach enables the policy to learn from previous states and actions, thereby leading to more efficient

exploration. The Experienced DNN policy also aims to increase overall system performance by lowering computational load through the utilization of stored experiences.

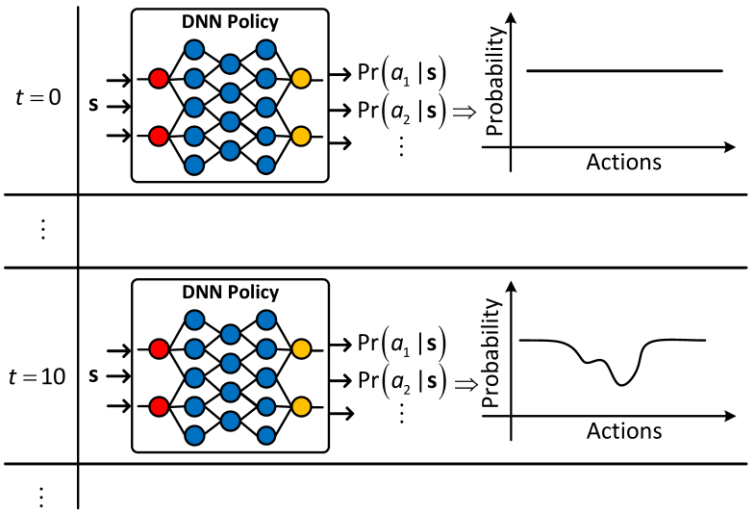


Figure 11: Proposed experienced DNN policy.

4. Simulation and Performance Evaluation

In this section, we simulated a region representing Jeju Island in South Korea that was captured using Google Maps and Google Street. Notice that because we used Google Maps as a reference, the region may not completely coincide with the actual map. Moreover, in this simulation, we employed the number of HAPs and UAVs equally and set the starting point from the center of the Jeju island. Furthermore, we assume that orbital LEO satellites are moving on the x-axis and located in South Korea, which is 200 km away from Jeju Island. It is important to note that the high velocity of LEO satellites causes the spot beam on the ground to change rapidly. **Table 1** presents the complete environment that has been prepared for this simulation.

In this simulation, we run on a computer with the following specifications: AMD Ryzen 5 3600 6-Core Processor and Nvidia GeForce RTX 2080 Ti. We use OpenAI's open-source code called Stable-Baselines 3, which is simple to implement and develop. **Table 2** shows the configuration of the SA-DQL's algorithm. Here, the exploration coefficient epsilon decays from 1 to 0.5 over ten percent of the total number of episodes, which is set to 3000 episodes. The SA-DQL algorithm also utilizes a learning rate of 0.001 and a discount factor of 0.99 for optimal performance. **Table 3** shows the configuration of the SA-DAC's algorithm. Here, we used the entropy regulation technique to balance exploration and exploitation, ensuring a more stable learning process. The entropy coefficient is gradually reduced over time

to encourage the agent to explore different actions and learn optimal policies efficiently. **Table 4** shows the configuration of the SA-DDPG's algorithm. The OU noise parameters are crucial for maintaining exploration in the SA-DDPG algorithm, allowing the agent to discover new strategies and avoid getting stuck in local optima. By setting the mean to zero and the noise scale to π, we ensure a balance between exploration and exploitation for more effective learning. Moreover, we configure our proposed MA-DAC-DNN following **Table 5**. Here, the number of neurons in experienced DNN is set to 1024, with two hidden layers to ensure all possible actions. Additionally, the estimated DNN for the information from the other agent is set to 1024 with two hidden layers as well. In this setup, we use the sigmoid function as the activation function for all the ML algorithms due to the smooth approximation compared to the ReLU activation function. Moreover, we use adaptive moment estimation (Adam) as the optimizer to reduce the loss function over and over again during training in this simulation. Adam is a mix of RMSprop and stochastic gradient descent with momentum.

Table 1: Simulation environment setup.

Parameters	Value
Number of HAPs	1-7 devices
Number of UAVs	1-7 devices
Number of GUs	100-1000 devices
LEO satellite's altitude	550 Km
HAP's altitude	10 Km
UAV's altitude	1 Km
LEO satellite's speed	10 Km/s
HAP's speed	30 Km/h
UAV's speed	30 Km/h
LEO satellite's arc distance	70 Km
LEO satellite's 3dB angle	1 Degree
HAP's 3dB angle	30 Degree
UAV's 3dB angle	75 Degree
LEO satellite's maximum transmit power	30 dBm
HAP's maximum transmit power	30 dBm
UAV's maximum transmit power	30 dBm
UAV's flying power	300 W
UAV's hovering power	170 W
LEO satellite's antenna gain	32 dB
Number of array antennas	10×10
Speed of light	2.99×10^{8} m/s
Radio Frequency, Ka-band	3×10^{9} Hz
Bandwidth	100 kHz
GU's moving speed	1 Km/h
Noise power ASWG	-174 dBm
Minimum data rate threshold	1.5 Mbps
Polygons	Jeju Island

Table 2: Single agent DQL's configuration.

Parameters	**Value**
Policy	ϵ-policy
Activation function	Sigmoid
Buffer size	10000
Number of neurons	[128, 128]
Learning rate	0.0001
Learning start	128
Batch size	128
Soft update coefficient	1
Discount factor	0.99
Train frequency	1 time/step
Gradient steps	1
Target update interval	1
Exploration fraction	0.1
Exploration initial	1
Exploration final	0.5
Maximum gradient normalized	10
Device	GPU
Optimizer	Adam

Table 3: Single agent DAC's configuration.

Parameters	**Value**
Policy	ϵ -policy
Activation function	Sigmoid
Number of neurons in actor network	[128, 128]
Number of neurons in critic network	[128, 128]
Learning rate	0.0001
Learning start	128
Batch size	128
Buffer size	10000
Soft update coefficient	0.005
Discount factor,	0.99
Train frequency	1 time / step
Target network update interval	1 time / episode
Gradient steps	1
Optimize memory usage	False
Entropy regularization coefficient	auto
Target entropy	auto
Use generalized state dependent exploration	False
Use generalized state dependent exploration at warmup	False
Sample a new noise	False
Optimizer	Adam
Device	GPU

Table 4: Single agent DDPG's configuration.

Parameters	**Value**
Policy	ϵ -policy
Activation function	Sigmoid
Number of neurons in actor network	[128, 128]
Number of neurons in critic network	[128, 128]
Learning rate	0.0001
Learning start	128
Batch size	128
Buffer size	10000
Soft update coefficient	0.005
Discount factor	0.99
Train frequency	1 time / step
Target network update interval	1 time / episode
Gradient steps	1
Action noise	Ornstein Uhlenbeck
Mean of noise	0
Scale of noise	π
Optimize memory usage	False
Optimizer	Adam
Device	GPU

Table 5: Multi-agent DAC with DNN's configuration.

Parameters	**Value**
Policy	ϵ -policy
Activation function	Sigmoid
Number of neurons in actor network	[128, 128]
Number of neurons in critic network	[128, 128]
Number of neurons in experienced DNN	[1024, 1024]
Learning rate	0.0001
Learning start	128
Batch size	128
Buffer size	1000000
Soft update coefficient	0.005
Discount factor	0.99
Train frequency	1 time / step
Target network update interval	1 time / episode
Optimize memory usage	False
Optimizer	Adam
Device	GPU

Figure 12 shows the comparison graphs for the training SA-DQL, SA-DAC, SA-DDPG, and MA-DAC-DNN as the accumulated reward versus the number of episodes. Notice that in this simulation, we use 1000 time-steps for each episode, which takes about an hour to complete 3000 episodes. Furthermore, in this simulation, we set the number of HAPs and UAVs to two and the number of GUs to 100. The SA-DQL graph shows that training convergence is the slowest. This is due to the Epsilon greedy policy, and we can see that the fluctuation is decreasing slowly. Moreover, we can observe that the training convergence of SA-DAC and SA-DDPG suggests faster convergence, with SA-DDPG outperforming SA-DQL. It should be noted that the result of SA-DDPG fluctuates when compared to SA-DAG because the DDPG algorithm uses the OU noise to improve the exploration. Our proposed algorithm converges faster compared to other algorithms while consuming less than 250 episodes, and it has a low fluctuation compared to the SA-DQL and SA-DDPG. Overall, the proposed algorithm shows promising results in terms of training convergence speed and stability, making it a strong algorithm for further research and real-world environment.

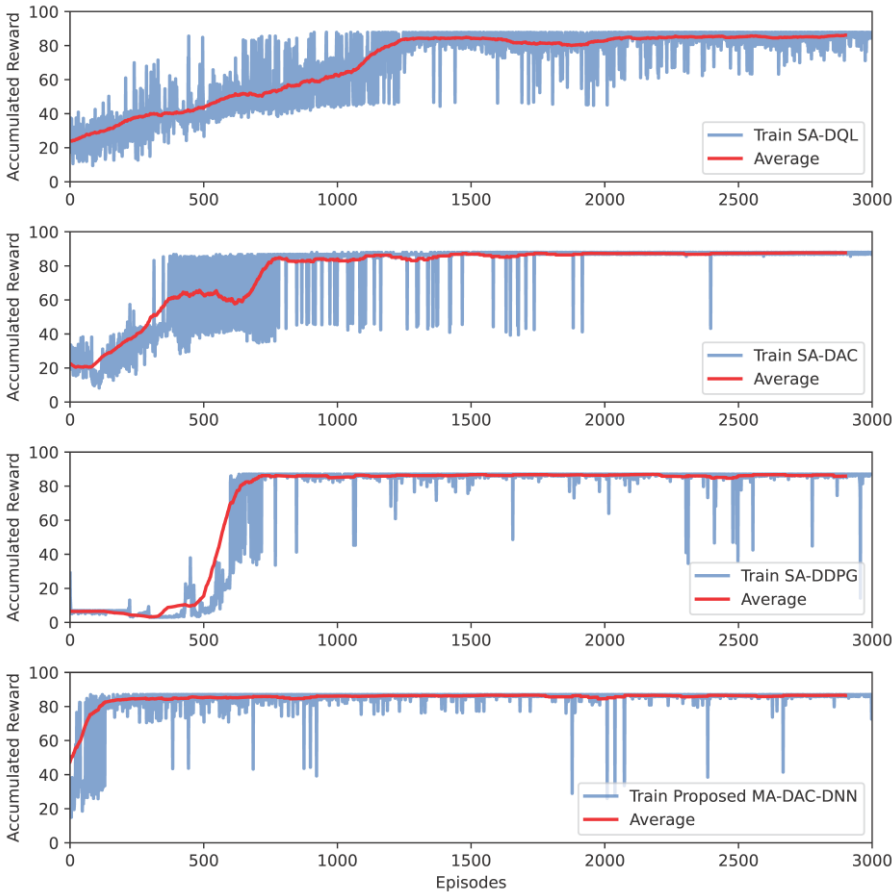
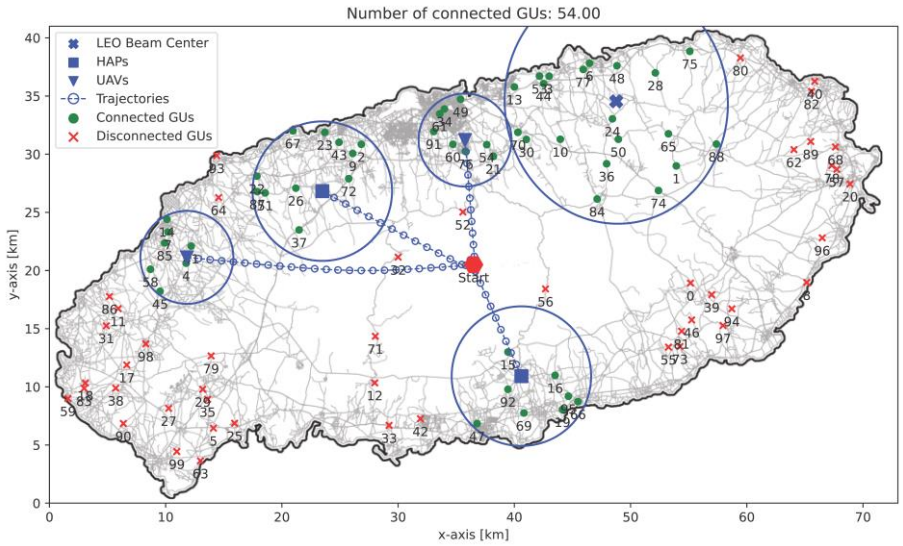
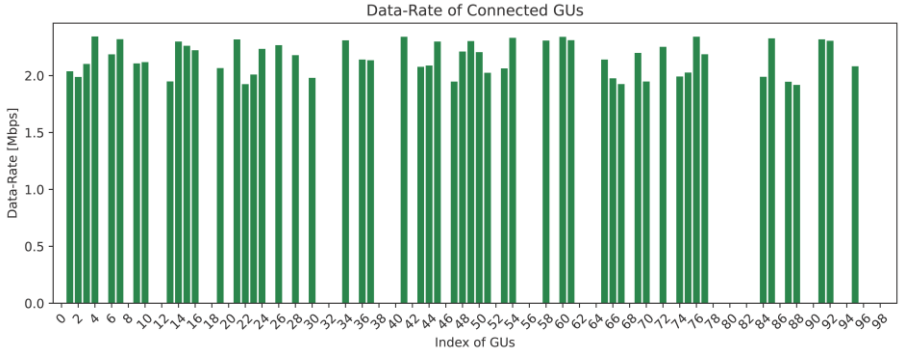


Figure 12: Training convergence.

Figure 13 and **Figure 14** show the optimal trajectory and data rate of SAGIN for our proposed MA-DAC-DNN by using two and five HAPs and UAVs. As shown in both figures, the trajectories of HAPs and UAVs are reliable due to the connected GUs and data rate, indicating that our proposed algorithm is effective at optimizing the data rate for GUs with the deference number of deployed HAPs and UAVs. This reliability is very important for ensuring connectivity and efficient data transmission in wireless communication systems. Moreover, we noticed that the optimal hovering locations of HAPs and UAVs are not at the center of the connected GUs due to the energy consumption QoS. For example, if HAPs travel a long distance from their starting point, they will consume a lot of energy. Furthermore, the bar graphs show the outcome of Jain's fairness of the data rate of connected GUs, indicating that the data rates of all connected GUs are fairly optimized. Furthermore, it is clear that as we deploy more HAPs and UAVs, the number of GUs connected to networks increases. It should also be noted that, in order to save energy, the spot-beam center of LEO satellites is located far away from the starting point, reducing system energy consumption.

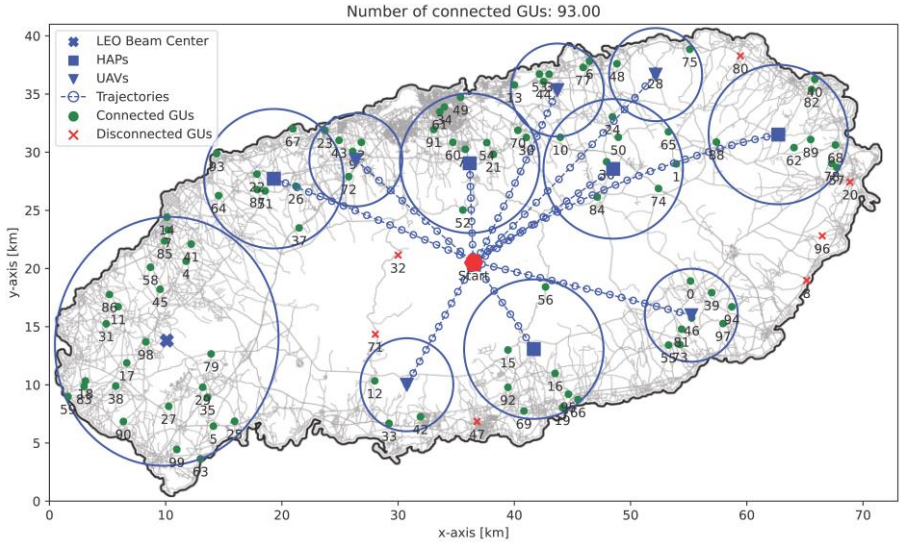


(a) Trajectories and Coverage Result.

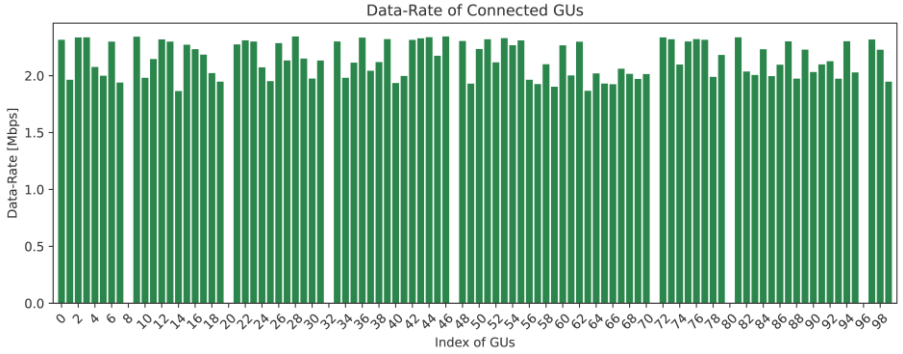


(b) Data-Rate Result.

Figure 13: Simulation result with two HAPs and two UAVs.



(a) Trajectories and Coverage Result.



(b) Data-Rate Result.

Figure 14: Simulation result with five HAPs and five UAVs.

Figure 15 illustrates the simulation of our proposed MA-DAC-DNN compared to SA-DQL, SA-DAC, and SA-DDPG in terms of the objective function versus the increase in the number of HAPs and UAVs from 1 to 7 devices. The graph shows that the objective is increased while the number of HAPs and GUs increases due to the efficient resource allocation and decision-making capabilities of ML models. Here, our proposed algorithm shows higher objective function values compared to other ML algorithms, indicating that our proposed algorithm has better performance in terms of optimizing the three QoS optimizations in SAGIN. This result indicates that our proposed algorithm is more effective in maximizing network coverage and minimizing interference compared to other ML algorithms.

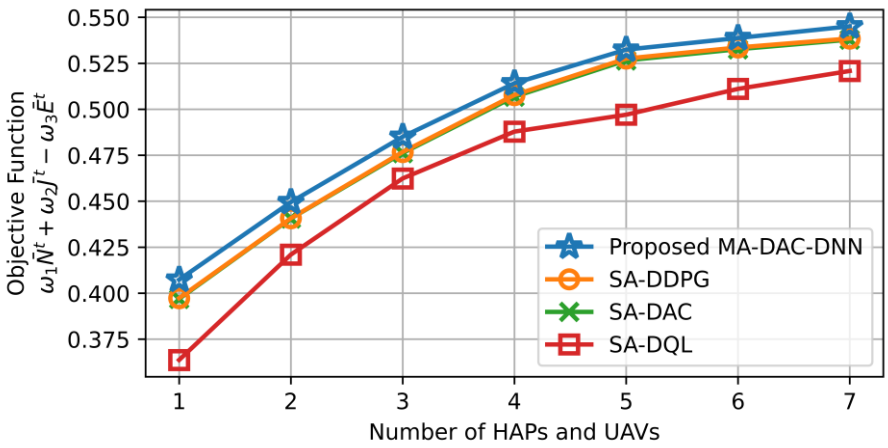


Figure 15: The objective function vs. number of aerials.

Figure 16 illustrates the simulation of our proposed MA-DAC-DNN compared to SA-DQL, SA-DAC, and SA-DDPG in terms of the coverage ratio versus the increase in the number of HAPs and UAVs from 1 to 7 devices. It is clear that the coverage is increasing due to the increased number of HAPs and UAVs that serve the communication from LEO satellites to GUs. Here, the graph shows that our proposed algorithm outperforms other ML algorithms and is increasing toward 1, which represents all GUs being served. This result indicates that our proposed algorithm is more efficient in utilizing resources, with a high potential for improving coverage and resource allocation in SAGIN.

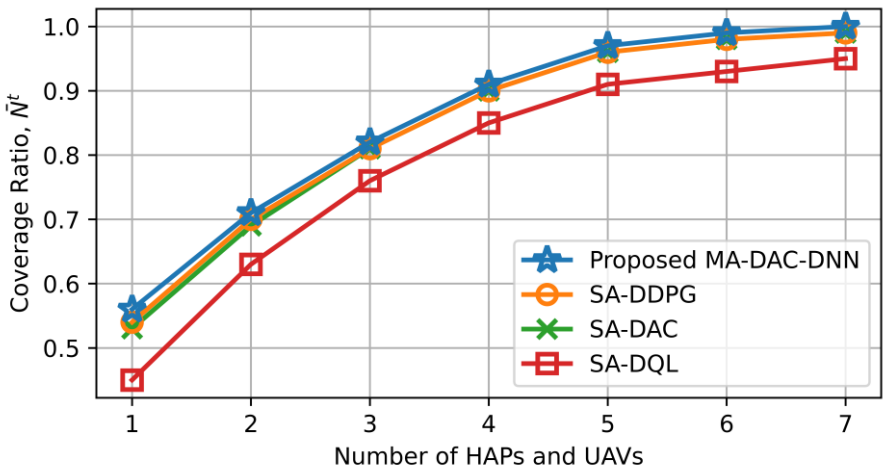


Figure 16: The coverage ratio vs. number of aerials.

Figure 17 illustrates the simulation of our proposed MA-DAC-DNN compared to SA-DQL, SA-DAC, and SA-DDPG in terms of Jain fairness data-rate versus the increase in the number of HAPs and UAVs from 1 to 7 devices. This result shows that our proposed algorithm outperforms other algorithms and decreases as the number of HAPs and UAVs increases. This decrease corresponds to an increase in the number of connected GUs, leading to a decrease in data rate fairness. Specifically, this result indicates that our proposed algorithm achieves better fairness in data rate allocation as the network scales up with more HAPs and UAVs, maintaining a balance in data rate distribution among connected GUs.

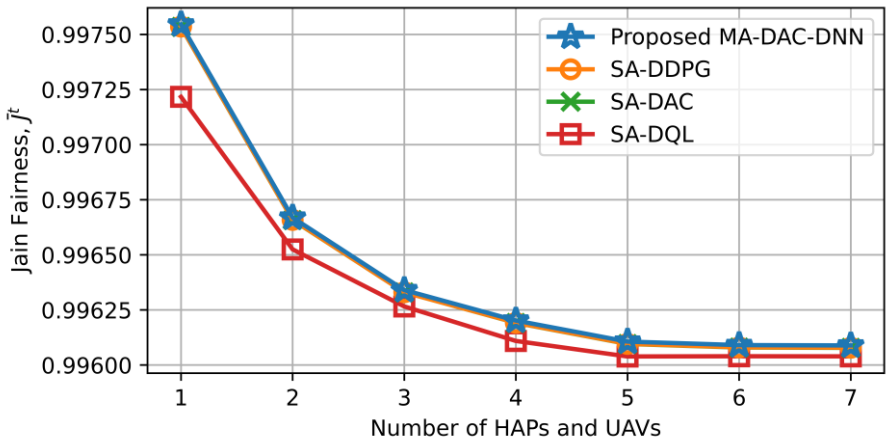


Figure 17: The data-rate fairness vs. number of aerials.

Figure 18 illustrates the simulation of our proposed MA-DAC-DNN compared to SA-DQL, SA-DAC, and SA-DDPG in terms of normalized energy consumption versus the increase in the number of HAPs and UAVs from 1 to 7 devices. This result shows that our proposed algorithm uses lower energy compared to other algorithms. This indicates that our proposed algorithm is more efficient in managing energy consumption as the number of devices increases, making it a promising solution for future HAP and UAV networks.

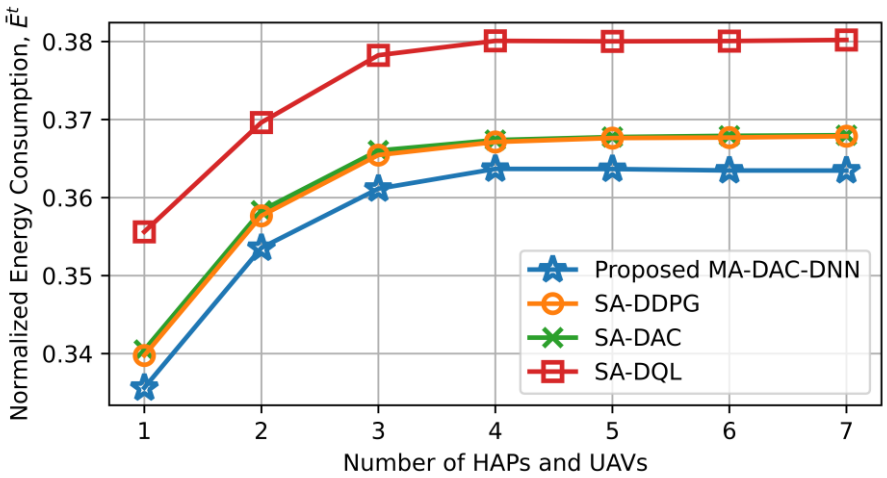


Figure 18: The normalized energy consumption vs. number of aerials.

Figure 19 illustrates the simulation of our proposed MA-DAC-DNN compared to SA-DQL, SA-DAC, and SA-DDPG in terms of the objective function versus the increase in the number of GUs from 100 to 1000 devices. The objective function is decreasing due to the randomly generated GUs following the edge of Jeju Island. Here, MA-DAC-DNN shows higher objective function values compared to other ML algorithms, indicating that our proposed algorithm has better performance in terms of multi-objective optimization in SAGIN. This result indicates that our proposed algorithm is more effective in improving network performance and minimizing interference compared to existing algorithms.

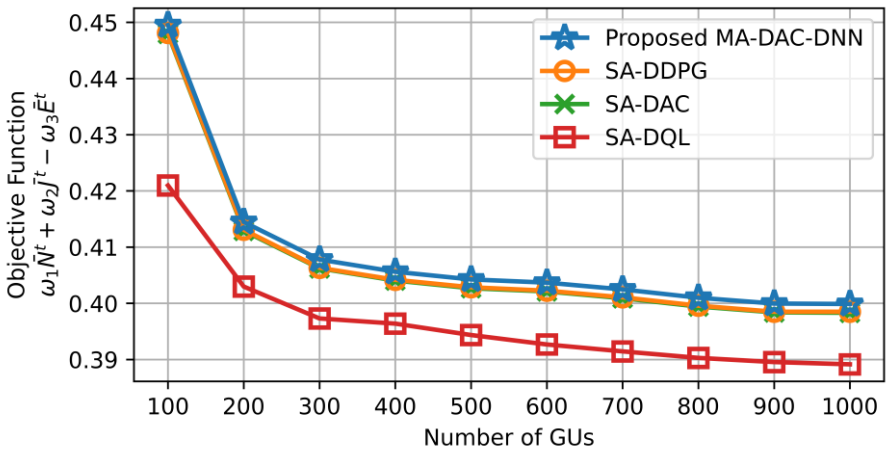


Figure 19: The objective function vs. number of GUs.

Figure 20 illustrates the simulation of our proposed MA-DAC-DNN compared to SA-DQL, SA-DAC, and SA-DDPG in terms of the coverage ratio versus the increasing number of GUs from 100 to 1000 devices. Similar to the previous result, the graph of the coverage ratio is increasing due to the randomly generated GUs at the edge of Jeju Island. Again, this result shows that our proposed algorithm outperforms other ML algorithms, which represent all GUs being connected. This result indicates that our proposed algorithm is more efficient in utilizing resources, with a high potential for improving coverage and resource allocation in SAGIN.

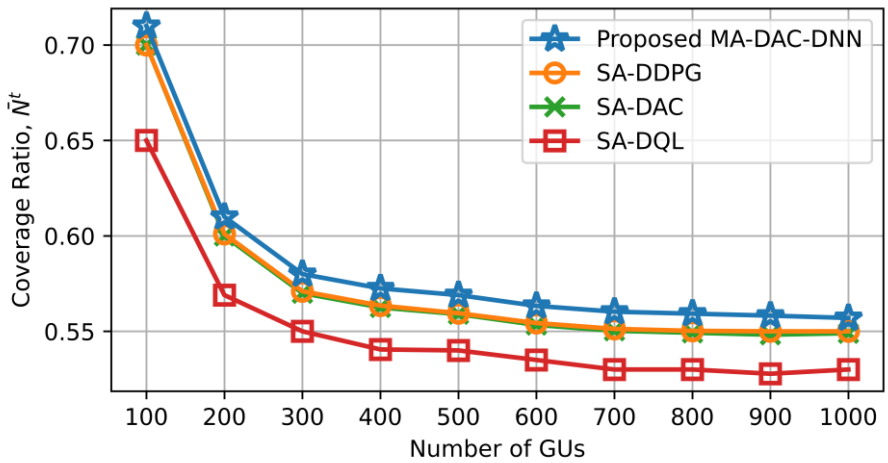


Figure 20: The coverage ratio vs. number of GUs.

Figure 21 illustrates the simulation of our proposed MA-DAC-DNN compared to SA-DQL, SA-DAC, and SA-DDPG in terms of Jain fairness data rate versus the increasing number of GUs from 100 to 1000 devices. This result shows that our proposed algorithm outperforms other algorithms and decreases as the number of GUs increases. This result indicates that our proposed algorithm achieves better fairness in data rate allocation as the network scales up with a high density of GUs, maintaining the fairness of the data rate for all connected GUs.

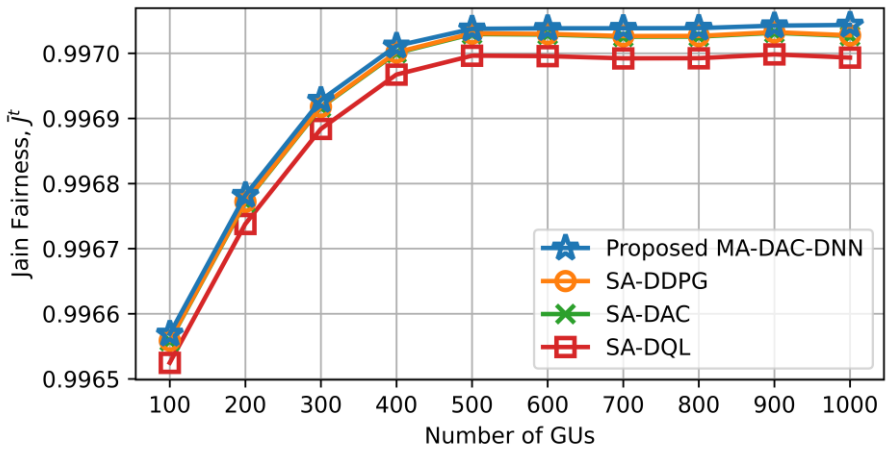


Figure 21: The data-rate fairness vs. number of GUs.

Figure 22 illustrates the simulation of our proposed MA-DAC-DNN compared to SA-DQL, SA-DAC, and SA-DDPG in terms of normalized energy consumption versus an increasing number of GUs from 100 to 1000 devices. This result shows that our proposed algorithm uses less energy compared to other algorithms. This indicates that our proposed algorithm is more efficient at managing energy consumption as the number of devices increases, making it a promising solution for a wide range of network scenarios.

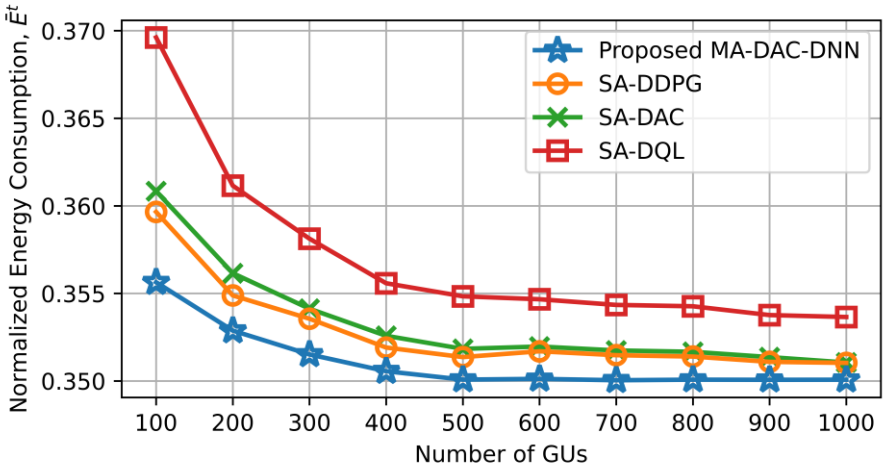


Figure 22: The normalized energy consumption vs. number of GUs.

5. Conclusion

In this study, we investigated and compared multiple QoS optimizations for SAGIN using different ML algorithms. We observed that no existing research addresses the multiple QoS trade-offs, such as coverage, data rate fairness, and energy efficiency, using decentralized ML for autonomous networking in SAGIN. To fill this research gap, we introduced a novel decentralized ML approach that optimizes these challenges by addressing the trade-offs among coverage, fairness, data rate, and energy consumption. First, we constructed a network scenario comprising LEO satellites, HAPs, and UAVs equipped with array antennas for beamforming communication with mobile GUs. Our problem statement aimed to optimize the trade-offs among coverage, fairness, data rate, and energy consumption by jointly optimizing the trajectory, bandwidth, power allocation, and beam direction of LEO satellites, HAPs, and UAVs. To solve the problem of multiple QoS optimizations, we designed three traditional centralized DRL such as single-agent deep Q-learning (SA-DQL), single-agent deep actor-critic (SA-DAC), and single-agent deep deterministic policy gradient (SA-DDPG). Furthermore, we proposed a novel multi-agent DRL approach called multi-agent DAC with experienced DNN (MA-DAC-DNN), which is intended to reduce training process redundancy and improve convergence during training. Finally, through the simulation, we have evaluated our proposed MA-DAC-DNN compared to the centralized DRL algorithms. The simulation results show that our proposed

algorithm outperforms conventional centralized DRL algorithm by achieving a higher optimality in addressing multiple QoS problems and demonstrating superior training convergence. These results indicate MA-DAC-DNN as a promising solution for scalable and efficient deployment in extensive networks with a huge number of devices.

Reference

[1] B. Li, Z. Fei, and Y. Zhang, "UAV Communications for 5G and Beyond: Recent Advances and Future Trends," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2241–2263, 2019, doi: 10.1109/JIOT.2018.2887086.

[2] Z. Niu, X. S. Shen, Q. Zhang, and Y. Tang, "Space-air-ground integrated vehicular network for connected and automated vehicles: Challenges and solutions," *Intell. Converg. Netw.*, vol. 1, no. 2, pp. 142–169, 2020, doi: 10.23919/ICN.2020.0009.

[3] H. Cao, S. Garg, G. Kaddoum, M. Alrashoud, and L. Yang, "Efficient resource allocation of slicing services in softwarized space-aerial-ground integrated networks for seamless and open access services," *IEEE Trans. Veh. Technol.*, 2023.

[4] H. Wang, X. Xia, T. Song, and Y. Xing, "Survey on Space-air-ground Integrated Networks in 6G," in *2021 IEEE/CIC International Conference on Communications in China (ICCC Workshops)*, IEEE, 2021, pp. 315–320.

[5] R. Zhang *et al.*, "Generative AI for Space-Air-Ground Integrated Networks (SAGIN)," *ArXiv Prepr. ArXiv231106523*, 2023.

[6] C. Zhou *et al.*, "Deep reinforcement learning for delay-oriented IoT task scheduling in SAGIN," *IEEE Trans. Wirel. Commun.*, vol. 20, no. 2, pp. 911–925, 2020.

[7] J. Liu, Y. Shi, Z. M. Fadlullah, and N. Kato, "Space-air-ground integrated network: A survey," *IEEE Commun. Surv. Tutor.*, vol. 20, no. 4, pp. 2714–2741, 2018.

[8] J. Ye, S. Dang, B. Shihada, and M.-S. Alouini, "Space-air-ground integrated networks: Outage performance analysis," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 12, pp. 7897–7912, 2020.

[9] F. Tang, H. Hofner, N. Kato, K. Kaneko, Y. Yamashita, and M. Hangai, "A deep reinforcement learning-based dynamic traffic offloading in space-air-ground integrated networks (SAGIN)," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 276–289, 2021.

[10] H. Wu and J. Yan, "QoS provisioning in space information networks: Applications, challenges, architectures, and solutions," *IEEE Netw.*, vol. 35, no. 4, pp. 58–65, 2021.

[11] N. Yang, D. Guo, Y. Jiao, G. Ding, and T. Qu, "Lightweight Blockchain-Based Secure Spectrum Sharing in Space–Air–Ground-Integrated IoT Network," *IEEE Internet Things J.*, vol. 10, no. 23, pp. 20511–20527, 2023, doi: 10.1109/JIOT.2023.3288121.

[12] T. Carter, S. E. Wolfe, Y. Nam, and S. G. Lawson, "Front porch roll calls: an innovative approach to community-oriented policing in Saginaw, MI," *Polic. Int. J.*, vol. 46, no. 5/6, pp. 766–779, 2023.

[13] S. Huaizhou, R. V. Prasad, E. Onur, and I. Niemegeers, "Fairness in wireless networks: Issues, measures and challenges," *IEEE Commun. Surv. Tutor.*, vol. 16, no. 1, pp. 5–24, 2013.

[14] Q. Chen, W. Meng, S. Han, C. Li, and H.-H. Chen, "Reinforcement learning-based energy-efficient data access for airborne users in civil aircrafts-enabled SAGIN," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 2, pp. 934–949, 2021.

[15] Q. Chen, W. Meng, S. Han, C. Li, and T. Q. Quek, "Coverage Analysis of SAGIN With Sectorized Beam Pattern Under Shadowed-Rician Fading Channels," *IEEE Trans. Commun.*, 2023.

[16] P. Zhang, Y. Zhang, N. Kumar, and C.-H. Hsu, "Deep reinforcement learning algorithm for latency-oriented iiot resource orchestration," *IEEE Internet Things J.*, vol. 10, no. 8, pp. 7153–7163, 2022.

[17] F. Tang, C. Wen, X. Chen, and N. Kato, "Federated learning for intelligent transmission with space-air-ground integrated network (SAGIN) toward 6G," *IEEE Netw.*, 2022.

[18] B. Li, R. Liang, W. Zhou, H. Yin, H. Gao, and K. Cai, "LBS meets blockchain: an efficient method with security preserving trust in SAGIN," *IEEE Internet Things J.*, vol. 9, no. 8, pp. 5932–5942, 2021.

[19] Z.-H. Zhou, *Machine learning*. Springer Nature, 2021.

[20] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science*, vol. 349, no. 6245, pp. 255–260, 2015.
[21] Q. Chen, W. Meng, S. Han, C. Li, and H.-H. Chen, “Robust task scheduling for delay-aware IoT applications in civil aircraft-augmented SAGIN,” *IEEE Trans. Commun.*, vol. 70, no. 8, pp. 5368–5385, 2022.
[22] Z. Jia, M. Sheng, J. Li, and Z. Han, “Toward data collection and transmission in 6G space–air–ground integrated networks: Cooperative HAP and LEO satellite schemes,” *IEEE Internet Things J.*, vol. 9, no. 13, pp. 10516–10528, 2021.
[23] Q. Gao, M. Jia, Q. Guo, X. Gu, and L. Hanzo, “Jointly Optimized Beamforming and Power Allocation for Full-Duplex Cell-Free NOMA in Space-Ground Integrated Networks,” *IEEE Trans. Commun.*, 2023.
[24] T. Yu, S. Zhang, X. Chen, and X. Wang, “A Novel Energy Efficiency Metric for Next-Generation Green Wireless Communication Network Design,” *IEEE Internet Things J.*, vol. 10, no. 2, pp. 1746–1760, 2022.
[25] L. He, J. Li, Y. Wang, J. Zheng, and L. He, “Balancing Total Energy Consumption and Mean Makespan in Data Offloading for Space-Air-Ground Integrated Networks,” *IEEE Trans. Mob. Comput.*, 2022.
[26] N. AbuAli, M. Ndong, and M. Hayajneh, “Relay-Aided D2D MIMO Scheme (RAS) for Achieving Energy Efficiency in Satellite-Air-Ground Integrated Networks (SAGIN) Schéma D2D MIMO assisté par relais (RAS) pour atteindre l’efficacité énergétique dans les réseaux intégrés satellite-air-sol (SAGIN),” *IEEE Can. J. Electr. Comput. Eng.*, 2023.
[27] L. Tan *et al.*, “Speech emotion recognition enhanced traffic efficiency solution for autonomous vehicles in a 5G-enabled space–air–ground integrated intelligent transportation system,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2830–2842, 2021.
[28] K. Fan, B. Feng, X. Zhang, and Q. Zhang, “Network selection based on evolutionary game and deep reinforcement learning in space-air-ground integrated network,” *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 3, pp. 1802–1812, 2022.
[29] A. Asheralieva, D. Niyato, and X. Wei, “Ultra-Reliable Low-Latency Slicing in Space-Air-Ground Multi-Access Edge Computing Networks for Next-Generation Internet of Things and Mobile Applications,” *IEEE Internet Things J.*, 2023.
[30] C. Wang, L. Liu, C. Jiang, S. Wang, P. Zhang, and S. Shen, “Incorporating distributed DRL into storage resource optimization of space-air-ground integrated wireless communication network,” *IEEE J. Sel. Top. Signal Process.*, vol. 16, no. 3, pp. 434–446, 2021.
[31] P. Zhang, Y. Li, N. Kumar, N. Chen, C.-H. Hsu, and A. Barnawi, “Distributed deep reinforcement learning assisted resource allocation algorithm for space-air-ground integrated networks,” *IEEE Trans. Netw. Serv. Manag.*, 2022.
[32] S. Zhang, A. Liu, C. Han, X. Liang, X. Xu, and G. Wang, “Multi-agent Reinforcement Learning-Based Orbital Edge Offloading in SAGIN Supporting Internet of Remote Things,” *IEEE Internet Things J.*, 2023.
[33] G. Maral, M. Bousquet, and Z. Sun, *Satellite communications systems: systems, techniques and technology*. John Wiley & Sons, 2020.
[34] S. Kurt and B. Tavli, “Path-Loss Modeling for Wireless Sensor Networks: A review of models and comparative evaluations,” *IEEE Antennas Propag. Mag.*, vol. 59, no. 1, pp. 18–37, 2017.
[35] C. Xiao, Y. R. Zheng, and N. C. Beaulieu, “Novel sum-of-sinusoids simulation models for Rayleigh and Rician fading channels,” *IEEE Trans. Wirel. Commun.*, vol. 5, no. 12, pp. 3667–3679, 2006.
[36] L. Zhu, J. Zhang, Z. Xiao, X. Cao, D. O. Wu, and X.-G. Xia, “3-D beamforming for flexible coverage in millimeter-wave UAV communications,” *IEEE Wirel. Commun. Lett.*, vol. 8, no. 3, pp. 837–840, 2019.

[37] In *Classical and Quantum Information Theory*, Cambridge University Press, 2009, pp. 264–282. doi: 10.1017/cbo9780511803758.016.
[38] J. Sun, "Time Series Data Mining System Based on Anomaly Detection," in *The Sixth International Conference on Information Management and Technology*, 2021, pp. 1–4.
[39] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Trans. Wirel. Commun.*, vol. 18, no. 4, pp. 2329–2345, 2019.
[40] N. Burkart and M. F. Huber, "A survey on the explainability of supervised machine learning," *J. Artif. Intell. Res.*, vol. 70, pp. 245–317, 2021.
[41] M. Khanum, T. Mahboob, W. Imtiaz, H. A. Ghafoor, and R. Sehar, "A survey on unsupervised machine learning algorithms for automation, classification and maintenance," *Int. J. Comput. Appl.*, vol. 119, no. 13, 2015.
[42] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.
[43] M. Van Otterlo and M. Wiering, "Reinforcement learning and markov decision processes," in *Reinforcement learning: State-of-the-art*, Springer, 2012, pp. 3–42.
[44] H. P. Van Hasselt, M. Hessel, and J. Aslanides, "When to use parametric models in reinforcement learning?," *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.
[45] J. Hammersley, *Monte carlo methods*. Springer Science & Business Media, 2013.
[46] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, pp. 9–44, 1988.
[47] B. Matzliach, I. Ben-Gal, and E. Kagan, "Detection of static and mobile targets by an autonomous agent with deep Q-learning abilities," *Entropy*, vol. 24, no. 8, p. 1168, 2022.
[48] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, "A survey on distributed machine learning," *Acm Comput. Surv. Csur*, vol. 53, no. 2, pp. 1–33, 2020.
[49] T. Hester *et al.*, "Deep q-learning from demonstrations," in *Proceedings of the AAAI conference on artificial intelligence*, 2018.
[50] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, PMLR, 2018, pp. 1861–1870.
[51] C. Qiu, Y. Hu, Y. Chen, and B. Zeng, "Deep deterministic policy gradient (DDPG)-based energy harvesting wireless communications," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8577–8588, 2019.
[52] S. Hwang, H. Lee, J. Park, and I. Lee, "Decentralized Computation Offloading with Cooperative UAVs: Multi-Agent Deep Reinforcement Learning Perspective," *IEEE Wirel. Commun.*, vol. 29, no. 4, pp. 24–31, 2022, doi: 10.1109/MWC.003.2100690.
[53] K. Li, W. Ni, Y. Emami, and F. Dressler, "Data-Driven Flight Control of Internet-of-Drones for Sensor Data Aggregation Using Multi-Agent Deep Reinforcement Learning," *IEEE Wirel. Commun.*, vol. 29, no. 4, pp. 18–23, 2022, doi: 10.1109/MWC.002.2100681.
[54] H. Soni, R. Vyas, and K. K. Hiran, "Self-Autonomous Car Simulation Using Deep Q-Learning Algorithm," in *2022 International Conference on Trends in Quantum Computing and Emerging Business Technologies (TQCEBT)*, 2022, pp. 1–4. doi: 10.1109/TQCEBT54229.2022.10041614.
[55] M. Tokic and G. Palm, "Value-difference based exploration: adaptive control between epsilon-greedy and softmax," in *Annual conference on artificial intelligence*, Springer, 2011, pp. 335–346.
[56] J. Mei, C. Xiao, C. Szepesvari, and D. Schuurmans, "On the global convergence rates of softmax policy gradient methods," in *International Conference on Machine Learning*, PMLR, 2020, pp. 6820–6829.
[57] N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu, "Boltzmann exploration done right," *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[58] J. Fan and W. Li, “Safety-guided deep reinforcement learning via online gaussian process estimation,” *ArXiv Prepr. ArXiv190302526*, 2019.

[59] J. Fu, J. Co-Reyes, and S. Levine, “Ex2: Exploration with exemplar models for deep reinforcement learning,” *Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[60] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, “Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning,” in *2018 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2018, pp. 7559–7566.

국문초록

최근 우주-공중 지상 통합 네트워크(SAGIN)는 위성의 빔 포밍을 사용하여 지상의 모바일 사용자에게 신뢰할 수 있는 인터넷을 제공한 스타 링크 회사의 배포 성공으로 인해 연구자들에게 더 많은 관심을 받고 있습니다. SAGIN 은 단순히 위성, 항공기 및 지상 기반 통신 시스템을 통합하여 무선 통신 시스템의 연결을 보장하는 네트워크 기술입니다. 그러나 SAGIN 은 대역폭, 전력 및 처리 기능과 같은 리소스의 제한으로 인해 네트워크 환경의 동적 특성으로 인해 많은 문제에 직면해 있습니다. 이러한 문제는 신뢰성, 데이터 속도, 지연 시간 및 에너지 효율성을 포함한 무선 통신의 전반적인 성능에 영향을 미칠 수 있습니다. 따라서 많은 연구자들이 리소스를 효과적으로 활용하여 SAGIN 의 성능과 신뢰성을 개선하기 위한 솔루션을 개발하기 위해 집중적으로 노력하고 있습니다. 결과적으로, 이 연구는 SAGIN 의 서비스 품질을 향상시키기 위해 SAGIN 을 위한 효과적이고 신뢰할 수 있는 기계 학습(ML) 알고리즘을 설계하는 것을 목표로 합니다. 먼저, 어레이 안테나를 장착하여 움직이는 지상 사용자(GU)와 통신할 수 있는 빔 포밍을 가능하게 하는 저지구 궤도(LEO) 위성, 고고도 플랫폼(HAP) 및 무인 항공기(UAV)로 구성된 네트워크 시나리오를 고려했습니다. 그 후 커버리지, 공정성, 데이터 속도 및 에너지 소비와 같은 세 가지 QoS 문제에 대한 문제 모델을 공식화합니다. 이 문제는 LEO 위성, HAP 및 UAV 의 궤적, 대역폭, 전력 및 빔 방향을 결정하는 것과 관련이 있으므로 이 문제를 더 복잡하고 해결하기가 어렵습니다. 이 문제를 해결하기 위해 SA-DQL, SA-DAC 및 SA-DDPG 의 세 가지 중앙 집중식 심층 강화

학습(DRL) 알고리즘을 개발했습니다. 또한, 분산 방식으로 문제를 해결하는 것을 목표로 하는 경험이 풍부한 DNN(MA-DAC-DNN)을 갖춘 다중 에이전트 DRL 접근 방식으로 알려진 새로운 분산형 DRL 접근 방식을 제안했습니다. 마지막으로, 저희는 중앙 집중식 DRL 알고리즘과 비교하여 제안한 MA-DAC-DNN 을 평가했습니다. 시뮬레이션을 통해 결과는 저희가 제안한 알고리즘이 최적성 및 훈련 수렴 측면에서 다른 알고리즘보다 성능이 뛰어나다는 것을 보여줍니다. 이러한 결과는 분산형 DRL 이 SAGIN 을 위한 여러 QoS 를 최적화하는 데 중앙 집중식 DRL 보다 성능이 뛰어나다는 것을 나타냅니다. 저희가 제안한 알고리즘은 확장 가능하고 효율적인 것으로 입증되어 엄청난 양의 장치가 있는 광범위한 네트워크에 배포하기에 이상적입니다.

핵심어: SAGIN, 다중 객체 최적화, 중앙 집중식 DRL, 분산형 DRL.