drones

MDPI

Article

Robust UAV-Oriented Wireless Communications via Multi-Agent Deep Reinforcement Learning to Optimize User Coverage

Mahfizur Rahman Khan †, Gowtham Raj Veeraswamy Premkumar † and Bryan Van Scoy *

Department of Electrical and Computer Engineering, Miami University, Oxford, OH 45056, USA; khanm12@miamioh.edu (M.R.K.); veerasg@miamioh.edu (G.R.V.P.)
* Correspondence: bvanscoy@miamioh.edu
† These authors contributed equally to this work.

Abstract: In this study, we deploy drones as dynamic base stations to address the issue of optimizing user coverage in areas without fixed base station infrastructure. To optimize drone placement, we employ Deep Q-Learning, beginning with a centralized approach due to its simplicity and ease of training. In this centralized approach, all drones are trained simultaneously. We also employ a decentralized technique in which each drone acts autonomously while sharing a common neural network, allowing for individualized learning. In addition, we explore the impacts of jamming on UAVs and provide a reliable approach for mitigating this interference. To boost robustness, we employ stochastic user distributions, which train our policy to successfully respond to a wide range of user situations.

Keywords: machine learning (ML); artificial intelligence (AI); unmanned aerial vehicle (UAV); reinforcement learning (RL); deep reinforcement learning (DRL); deep Q-learning (DQL); multi-agent deep reinforcement learning (MADRL); multi-agent deep Q-learning (MADQL); Internet of Things (IoT); Line of Sight (LoS)

Academic Editors: Tomotaka Kimura and Chinthaka Premachandra

Received: 6 March 2025
Revised: 15 April 2025
Accepted: 16 April 2025
Published: 22 April 2025

Citation: Khan, M.R.; Premkumar, G.R.V.; Van Scoy, B. Robust UAV-Oriented Wireless Communications via Multi-Agent Deep Reinforcement Learning to Optimize User Coverage. *Drones* **2025**, *9*, 321. https://doi.org/10.3390/drones9050321

1. Introduction

Unmanned Aerial Vehicles (UAVs) are being explored for their potential applications in various fields. Their uses include telecommunications, military and security activities, and entertainment [1]. UAVs are becoming more practical, dependable, and economical, which makes UAV-based solutions competitive in new markets. The telecommunications industry is expected to have the highest drone value in 2021, with an estimated 6.3 billion USD according to [2]. Along with providing network programmability, orchestration, and edge cloud capabilities, UAVs can also be used to establish line-of-sight (LoS) connections, provide scalable proximity services, and hold these functions on demand. As a result, the integration of unmanned aerial platforms into the mobile network ecosystem is gaining traction in both industry and academia.

UAVs are among the flying platforms whose applications are expanding quickly. Specifically, due to their intrinsic qualities, like mobility, adaptability, and altitude adaptation, UAVs have a number of important potential uses in wireless systems. UAVs have the potential to improve wireless network coverage, capacity, dependability, and energy-efficiency by serving as airborne base stations. UAVs can also function inside a cellular network as flying mobile terminals. These cellular-connected UAVs can be used for a variety of tasks, such as item delivery and real-time video broadcasting [3]. To overcome

some of the limitations of current technologies, we see flying base stations carried by UAVs as an essential addition to a heterogeneous 5G ecosystem. In areas of the world without reliable cellular infrastructure, UAVs are starting to show promise as an economically viable method of delivering wireless access. Moreover, UAV base station deployment makes the most sense when considering events that require wireless services for a short amount of time. This is particularly true for temporary events, when it is evident that creating permanent small cell networks for short-term demands is financially unfeasible, such as sporting events and festivals [4]. For longer-term, more sustainable coverage in such rural areas, High-Altitude Platform (HAP) UAVs can be of assistance. During transitory events like football tournaments or presidential inaugurations, as well as in hotspots, mobile UAVs can offer on-demand connectivity, a high-data-rate wireless service, and traffic offloading opportunities [5,6]. With reference to this, AT&T and Verizon previously declared their intention to employ drones in flight to temporarily enhance internet coverage during the Super Bowl and college football national championships [7]. In addition to ultra-dense tiny cell networks, flying base stations can undoubtedly be a valuable resource. UAVs have the potential to save lives as well because of their quick and adaptable deployment. This makes them a great option in disaster and search and rescue situations, as well as other situations when stationary infrastructures, such as communication networks, are damaged or unavailable. This may occur when an earthquake or other natural disaster damages cell towers, and drones serve as flying base stations (BSs) to allow first responders to reestablish contact [8]. It is obvious that all UAVs need to have well-designed communication capabilities or be able to provide communication services in order to realize their full potential and contribute to the civilization of the future. Additionally, flying UAVs have the ability to travel continually in order to cover a particular region completely in the shortest amount of time. Thus, in public safety scenarios, using UAV-mounted base stations can be a suitable alternative for quick and pervasive communication.

Connecting UAVs to cellular networks provides a number of advantages. With cellular command and control links, autonomous UAVs can be remotely piloted or monitored from thousands of kilometers away because of the widespread availability of mobile networks. It can also facilitate information-sharing with air traffic control or allow UAV-to-UAV communication. Cellular networks can also offer increased dependability and throughput, as well as improved privacy and security, in comparison to conventional direct ground-to-UAV communication. A significant financial benefit of using existing cellular networks is that it may be more affordable to do so than to build new, different infrastructure configurations [9].

There are several distinct design and research problems associated with UAV-aided communication as compared to fixed terrestrial networks [10]. In the first place, compared to the conventional two-dimensional (2D) deployment of terrestrial base stations, the deployment of aerial base stations in three-dimensional (3D) space offers an extra degree of freedom (DoF). The diverse context of propagation is another significant obstacle [11]. In addition, restricted flight time and inherent flight dynamics are two more complications associated with aerial BSs and relays that affect the communication performance or quality of service (QoS). In order to address these issues and direct the study toward UAV-assisted communications, scheduling, research allocation, and multiple access protocols must be modified. UAV BS placement optimization or comprehensive end-to-end trajectory optimization can both account for mobility. Because of the extra DoF, even placing UAVs is a more difficult task than placing stationary base stations. Figure 1 shows an example of UAV based wireless communication.

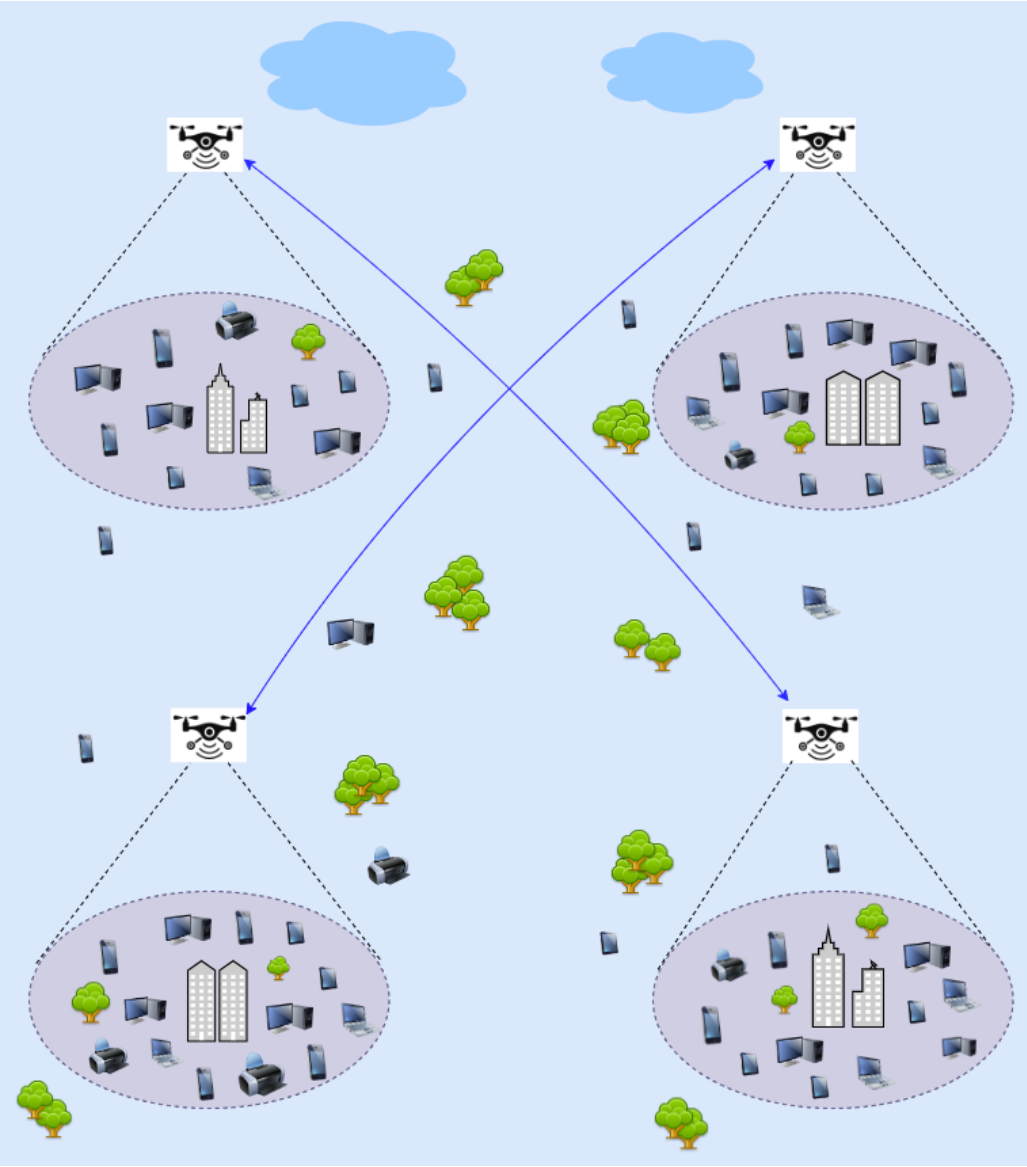

Figure 1. In order to effectively provide maximum user connectivity, UAVs maintain minimal information exchange among themselves while providing wireless coverage.

The foundation of reinforcement learning is particularly well-suited to the problems associated with the deployment of autonomous UAVs in communication networks, since its main idea involves an autonomous agent making choices (like trajectory planning) in order to maximize a goal (like QoS for an aerial BS) in an uncharted area. Many of the applications of UAV placement and trajectory planning, including interference-aware multi-UAV path planning [12], data collection in the context of mobile crowdsensing [13], and maximizing communications coverage in UAV-aided networks [14], could benefit from the application of deep reinforcement learning (DRL). Another application of UAV networks is resource allocation, in addition to path planning. Multi-agent reinforcement learning (MARL) is employed in [15] to choose the communicative user of each UAV automatically, whether or not the UAVs communicate information. The integration of intelligent reflecting surfaces with UAVs in the context of 6G presents a number of issues that RL can address [16].

1.1. Research Gaps

Most existing research focuses solely on either centralized or decentralized approaches for user coverage, without a comprehensive comparison of both. Our work addresses

this gap by evaluating and contrasting both methods (Sections 4 and 7). Additionally, while most reinforcement learning studies train and test their models on the same user distribution, we deliberately use a stochastic environment during training to ensure that our model performs well under varied user distribution scenarios (Section 5). Finally, we also consider jamming attacks during the operation of dynamic base stations—a topic with limited prior research (Section 6). We further review the relevant literature in Section 2.

1.2. Contributions

The main contributions of this paper are as follows:

1. We use both centralized and decentralized multi-agent deep Q-learning to determine the positions for a set of UAVs that maximize user connectivity in an ad-hoc communication network problem.
2. We develop a simple algorithm to train a policy for a stochastic environment, such as when users are distributed according to a known probability distribution. Moreover, we empirically explore how well a policy designed for one distribution of environments performs when applied to environments sampled from a different distribution.
3. We develop a model in which a UAV is jammed and loses connection with other UAVs after reaching its optimal position, and investigate how this affects user coverage and network connectivity.

2. Literature Review

The field of wireless communication has advanced significantly with the introduction of UAVs as enablers of expanded communication services, especially in situations requiring quick deployment and great flexibility. UAVs can function as BSs or relays, providing on-demand communication services that are essential for Internet of Things (IoT) applications, post-disaster recovery networks, and reducing abrupt traffic congestion in cellular networks [17]. UAV-enabled small cells have been optimized through research into the strategic deployment of UAVs with regard to altitude and inter-UAV distances, offering a way to reduce UAV deployment while guaranteeing coverage for all ground users [18]. The methods used to deploy UAVs are critical, particularly when considering UAV-enabled small cells, as the effective and efficient operation of the network is greatly impacted by the placement of UAVs. Past research has demonstrated the ability to reduce the number of UAVs needed by using a sequential UAV placement technique with set heights, improving network coverage and cutting down on wasteful spending [19]. However, this tactic might also add processing latency, which could negatively affect network throughput [20]. We now discuss various techniques to solve this problem, which are summarized in Table 1.

Table 1. Comparison of techniques.

Technique	Methodology	Strengths	Limitations
Optimization	Solve NP-hard problems using iterative approximation	Provides near-optimal solutions; well-studied	High computational overhead; limited scalability
Single-Agent Deep Q-Learning	Train single RL agent to optimize UAV positions	Simple formulation; lower computational cost	Does not effectively capture inter-UAV coordination
Distributed Q-Learning (MARL)	Each UAV learns its own policy with limited coordination	Scalability; flexibility in dynamic environments	Challenges in achieving global coordination
Proposed Methods	Both centralized and decentralized multi-agent deep Q-learning	Comprehensive comparison; trained in stochastic environments; includes jamming resilience	Centralized approach scales poorly; decentralized coordination complexity

2.1. Conventional Method

Enhancing network efficiency and lowering energy usage would provide major contributions to the developing field of UAV-assisted multi-access edge computing (MEC). In order to address the need for energy-efficient resource allocation, ref. [21] presented a system that utilized MEC-enabled UAVs and ground-based base stations. The system used the Block Successive Upperbound Minimization (BSUM) method, which represents a novel approach to resource allocation. A different study [22] demonstrated how non-orthogonal multiple access (NOMA) may be integrated with UAV flying base stations. It achieved this by using a path-following algorithm to solve a non-convex optimization problem, thereby demonstrating the advantages of NOMA in improving wireless communication. Wu et al. created an effective iterative technique to optimize user scheduling, UAV trajectory, and power control in order to maximize throughput in multi-UAV enabled systems [23]. This method showed improvements, optimizing user performance while navigating the intricacies of optimization. In order to dramatically lower latency and increase spectral efficiency in 3D wireless networks, the author [24] presented a novel framework for 3D cellular network planning and cell association. This framework uses kernel density estimation and optimal transport theory, providing a ground-breaking approach to 3D cellular architecture. In contrast to most previous research that focuses on homogeneous UAV fleets, ref. [25] proposes a deployment approach for a heterogeneous UAV network. UAVs with different service capabilities, transmission powers, and battery lifetimes are used in this network. In this heterogeneous UAV network, user coverage is optimized using a traditional approximation algorithm.

2.2. Optimization Approach

The problem of controlling and deploying UAVs in communication contexts can also be formulated as a non-convex optimization problem [26]. These issues are inherently complicated and frequently categorized as NP-hard [27], which highlights how challenging it is to solve them using traditional optimization techniques. This difficulty helps to explain why RL is becoming more and more popular as a substitute method for handling certain optimization problems. With regard to UAV-based communication systems, reinforcement learning's flexibility and learning-based approach present viable ways to move beyond the conceptual and computational obstacles that come with more conventional optimization methods.

2.3. Machine Learning Approach

Intelligent machine learning-based UAV control is necessary to improve the efficiency of UAV-enabled communication networks. With consideration of the UAVs' structural designs, neural network architectures have been examined for UAV trajectories. For relay placement intended to maximize flow rate, ref. [28] presented a Riemannian multi-armed bandit (RMAB) reinforcement learning model. In order to support as many ground devices as feasible, a single UAV was explored by Fahim et al. [29]. The main emphasis of this study was the trade-off between the UAVs' increased coverage area and continued connectivity. Nevertheless, the study in [30] did not include the employment of UAVs as stationary base stations. A distributed Q-learning method was provided by Klaine et al. [31] to identify the optimal UAV deployment sites that maximize ground user coverage in the presence of time-varying user distributions. Liu et al. [32] employed double Q-learning to design routes that would optimize customer satisfaction for users with time-constrained service requirements. By jointly optimizing the multiuser communication scheduling, user association, UAV trajectory, and power control, a combined trajectory and communication design for multi-UAV enabled wireless networks was developed in [23]. The issue of

flying trajectory planning for many UAVs in order to provide ground personnel with emergency communication services was examined by Yang et al. [33]. In order to monitor a catastrophe area, Xu et al. [34] investigated the problem of deploying a network of connected unmanned aerial vehicles (UAVs) that consists of K homogeneous UAVs in the air while maximizing the total data rates of all users. The downlink of a UAV-assisted cellular network, whereby several collaborating UAVs service various ground user equipment (UE) under the supervision of a central ground controller via wireless fronthaul links, is highlighted in [35]. Deep Q-learning is used to find the UAVs positions. In order to optimize resource allocation in UAV networks, the author in [36] presented a MARL technique that allows UAVs to autonomously choose communication techniques that maximize long-term rewards with the least amount of inter-UAV communication. In order to enhance user connectivity in UAV networks, a distinct study [37] proposed a fully decentralized deep Q-learning method. The study demonstrated increased performance through differential levels of UAV information sharing and customized reward functions.

2.4. Jamming

Due to the broadcasting nature of wireless communications, UAV-assisted wireless communication networks are particularly vulnerable to spectrum jamming assaults, which pose a serious threat to network operation. Malicious users exploit this vulnerability by launching three forms of jammer attacks: constant, intermittent, and reactive. Constant jamming occurs when jamming signals are continuously sent, intermittent jamming involves sending signals periodically, and reactive jamming occurs when jamming signals target the region of the spectrum inhabited by legitimate users while monitoring their transmission [38]. In this study [39], a hidden Markov model (HMM)-based jamming detection technique is suggested, with the goal of detecting reactive short-period jamming for UAV-assisted wireless communications without requiring prior knowledge of thesignal or channel characteristics.

2.5. Motivation

Recent advances in drone technology have created new prospects for UAV deployment in wireless communication networks. UAVs, which range from drones to small airplanes and airships, represent an innovative approach to delivering dependable and cost-effective wireless communication options in a variety of real-world circumstances. UAVs can revolutionize traditional terrestrial networks by operating as aerial BSs and providing on-demand wireless communications to specific areas. This trend of using UAVs for wireless communication indicates the importance of rethinking research issues, prioritizing networking and the handling of resources over control and positioning difficulties [40,41]. Machine learning (ML) has emerged as a significant method for improving UAV-enabled communication networks, providing autonomous and intelligent solutions. While most of the existing research focuses on UAV deployment and trajectory designs, the optimization of resource allocation strategies such as transmit power and subchannels has mainly occurred in time-independent settings. Furthermore, the rapid movement of UAVs hinders the collection of accurate dynamic environmental information, making the design of dependable UAV-enabled wireless communications difficult. Furthermore, as network size increases, the centralized methodologies presented in prior research face computing issues [3,42–44]. MARL provides a distributed approach to intelligent resource management in UAV-enabled communication networks, particularly when UAVs only have local information. MARL allows agents to consider particular application-specific requirements while modeling local interactions, addressing distributed modeling and computation problems. While MARL applications in cognitive radio networks and wireless regional area networks have

demonstrated promise, its potential in multi-UAV networks remains untapped, notably in resource allocation [36,45,46]. Thus, applying MARL to UAV-enabled communication networks offers a viable approach for intelligent resource management.

3. Background

The growing field reflects the increasing interest and extensive research carried out in recent years by integrating artificial intelligence (AI) and machine learning (ML) into UAV-assisted communications [47,48]. The three categories for the machine learning algorithms are specifically supervised, unsupervised, and reinforcement learning (RL). While algorithms in an unsupervised environment extract knowledge from patterns in the unlabeled data, supervised scenarios have an data set with labeled input–output pairs accessible to direct the learning process. In contrast to ML, which relies on labeled data, RL learns directly from the environment [49].

3.1. Reinforcement Learning

RL comes under the third category of machine learning, where the RL agent interacts with an unknown environment and takes the actions and receives the corresponding reward based on the action that was taken and the current state of the system [50]. The states describe the agent's circumstances in the environment. The agent can interact with the world or environment through a set of actions [50]. The action space can be either discrete or continuous, indicating the nature of actions from which the agent can choose. For example, a robot has a continuous action space, which means that the robot's actions are not limited to fixed steps, whereas playing a game of chess has a discrete action space [51]. At each time step t, the state s_t of the system and immediate reward r_t are observed and this information is used to choose the action a_t that will maximize the cumulative future reward obtained by the agent. The RL agent makes decisions regarding the future actions to take. The main job of the agent is to obtain as large a reward as possible based on the action that it takes.

From Figure 2, the agent interacts with the environment and makes decisions about future actions. In many RL problems, the reward structure might be sparse, and the feedback regarding whether the agent is making good moves or not is obtained only after a significant amount of interaction with the environment. This delayed reward information makes the reinforcement learning problem more challenging. In RL, the agent has a control strategy, which is called a policy. The policy is the probability that an action is taken in a given state. The policy can either be deterministic or probabilistic. The agent follows the policy to maximize the expected sum of future rewards, where the sum represents the rewards received at different times.

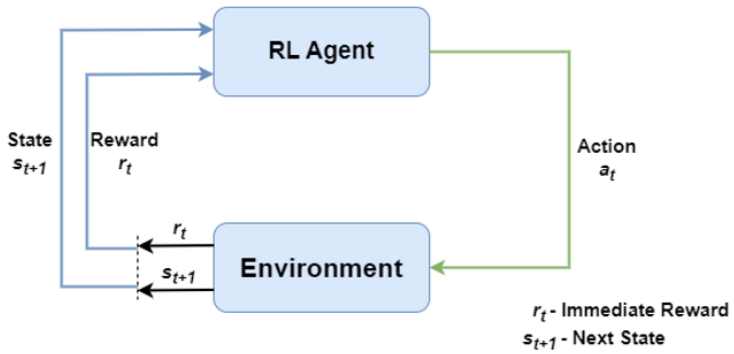

Figure 2. Reinforcement learning flow.

3.2. Deep Q-Learning

Q-learning is effective in small state and action spaces, but it may not find the optimal policy in the vast state and action spaces found in complex systems. To address this problem, the Deep Q-Learning (DQL) technique is presented [52]. The reinforcement learning algorithm DQL [53] breaks the dependence of the memory complexity of the learning process from the size of the state–action space by approximating the Q-table and taking advantage of the universal approximation property of deep neural networks (DNNs). DNNs are utilized in deep reinforcement learning to control high-dimensional state spaces. The agent feeds the deep neural network the current state's feature vector at each time step. This network is known as a deep Q-network (DQN) because it can predict the Q-value for every action and state. The DQN must be trained in order to generate meaningful predictions, which requires the definition of a loss function. There is no label indicating the proper Q-value for any given state–action pair, so this is not a supervised learning problem. It is therefore not easy to define an appropriate loss function. Two identical DNNs (one serving as the main network and the other as the target network) are initialized with the same settings at the beginning of the Deep Q-Learning process. Afterward, the agent chooses the action with the highest estimated Q-value and stores the experience in a replay buffer, together with the reward r_t and the state transition from s_t to s_{t+1}. This allows the DQN to be trained to estimate the Q-values more precisely. The deep Q-network framework [52] is shown in Figure 3.

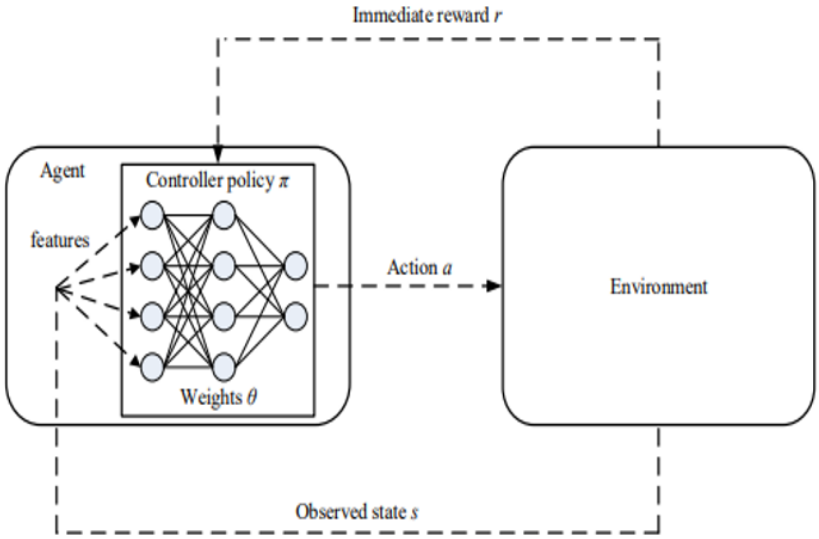

Figure 3. Deep Q-Learning framework.

The value against which the loss function will be assessed, or the target update, is defined as follows [54]:

$$y_t = \begin{cases} r_t & \text{if } s_{t+1} \text{ is terminal,} \\ r_t + \gamma \max_a Q_\theta(s_{t+1}, a) & \text{otherwise.} \end{cases} \quad (1)$$

The target y_t is compared to the output of the Q-network by means of a mean squared error (MSE) loss [55]

$$L(\theta) = \frac{1}{K} \sum_{t=1}^{K} (y_t - Q_\theta(s_t, a_t))^2. \quad (2)$$

Using batch gradient descent, the loss function $L(\theta)$ is optimized in relation to the model parameters θ, where K is the batch size denoting the quantity of transitions sampled from

the experience replay buffer. The discounted maximum Q-value of the subsequent state s_{t+1} and all feasible actions a are added together to provide the goal value for the t-th transition, y_t. The current state–action pair's projected Q-value is indicated as $Q_\theta(s_t, a_t)$, whereas r_t stands for the immediate reward obtained following action a_t in state s_t. The significance of future rewards is determined using the discount factor γ. The expression $\max_a Q_\theta(s_{t+1}, a)$ represents the maximum expected Q-value for all feasible actions a in the next state s_{t+1}, as determined by the current value function estimate Q_θ.

3.3. Multi-Agent Reinforcement Learning

Multiple agents working together to impact a shared environment is known as multi-agent reinforcement learning (MARL). More specifically, each agent acts at each state, and these actions collectively decide the next state of the environment and each agent's reward. Furthermore, each agent may only perceive its own reward, which may differ from another agent's. When coordination, collaboration, or rivalry among agents is necessary, such as in autonomous vehicles, games, robotics, and social issues, MARL can be used in a variety of contexts [56]. In recent years, a wide range of strategies have been put out to take advantage of the benefits and overcome the obstacles presented by the fast-growing field of MARL. For example, agents can communicate with one another to exchange information [57], competent agents can act as teachers for learners [58], or learners can observe and mimic skilled agents [59]. In a multi-agent system, the surviving agents can assume some of the responsibilities of the failing agent or agents. This suggests that MARL is resilient by nature. Additionally, most multi-agent systems are highly scalable due to their design, which makes it simple to add new agents to the system. Stability, scalability, and communication are a few issues that MARL must deal with, which are contingent upon the structure of the learning process. When agents in an environment work together to improve the average return for the group as a whole, there is an increase in interest. This collaborative approach emphasizes how important it is to work together. The multi-agent reinforcement learning framework [60] is shown in Figure 4.

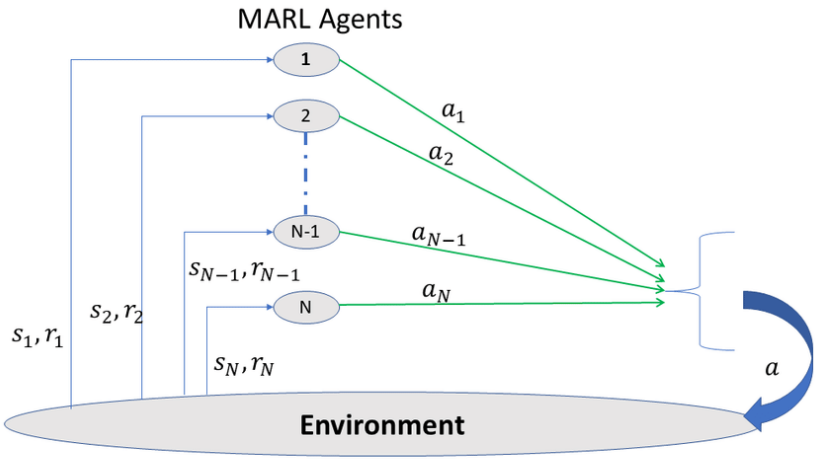

Figure 4. Illustration of a MARL framework featuring N agents. Each agent contributes to the environment by selecting actions a_i based on its individual state S_i and reward R_i. Image from [60].

3.3.1. Centralized

Developing a central controller that decides what each agent should do and receives all of the agent's rewards is an alluring option. The problem can be reduced to an MDP, which is easily handled by single-agent reinforcement learning methods as the controller has access to all of the agents' data. Nevertheless, the installation of a central controller could

be expensive in many real-world situations. In addition, the communication overhead at the single controller scales with the number of agents, since each agent requires a connection with the central controller in order to send information. The multi-agent system's resilience to malicious attacks and scalability may be negatively impacted by this [61]. Using observations from every agent, a central agent or network would learn a shared policy that optimizes how well the team does on each task in a centralized learning approach. Throughout training, rather than each agent improving its own policy, the central agent would provide feedback to each agent depending on the team's success. Every agent would benefit from this input, learning how to act in a way that advances the team's goals.

3.3.2. Decentralized

Another method in MARL is decentralized learning, in which each agent independently learns and modifies its own policy or value function depending on its local observations and actions. The benefits of this strategy include scalability, privacy, and robustness, as it can manage dynamic and heterogeneous agents without relying on a global state or central authority. Nevertheless, because decentralized learning necessitates greater agent cooperation and communication and may face non-stationarity and partial observability, it also has several drawbacks, including complexity, inconsistency, and inefficiency [61]. Without a network or central agent to manage the learning process, each agent's policy would be modified in a decentralized learning approach based on its own experience. Each agent would converse with other agents in the vicinity to exchange information about the surroundings and their planned course of action while undergoing training. Each agent would be able to learn to anticipate the other agents' movements as a result.

4. Problem Setup

We consider a target region A with M number of users located on the ground, as shown in Figure 5. All users on the ground are stationary. The target area measures L by L squares. The K number of users congregate near the four hotspots. The remaining $M - K$ users are uniformly distributed across the entire target area A. To ensure LoS for communication with ground users, N UAVs fly horizontally inside the target region at a constant altitude H. Two-dimensional Cartesian coordinates are used to identify the UAV and user positions inside the designated target region. Each UAV is identical and has a single directional antenna that concentrates the transmission energy inside an aperture angle θ. A UAV's ground coverage is considered to be a disk with radius $r = H \tan(\frac{\theta}{2})$. The UAV does not cover users who are outside of its coverage disk. One UAV may cover up to the number of users assigned to it at once. Bandwidth is not provided uniformly among UAVs; it is allotted based on user demand and a UAV's available bandwidth. UAVs operate independently, observing individual states and communicating little information, primarily sharing rewards. The time of the UAV movement is separated into discrete time stages, allowing the UAVs to make sequential decisions. Table 2 represent the default values of the parameters.

Table 2. Description of parameters and default values.

Parameter	Default Value	Description
N	5	Number of UAVs
M	100	Number of users
K	90	Number of hotspot users
H	350 m	Height of UAVs
θ	60°	Aperture angle of UAVs
L	1000 m	Length and width of target area

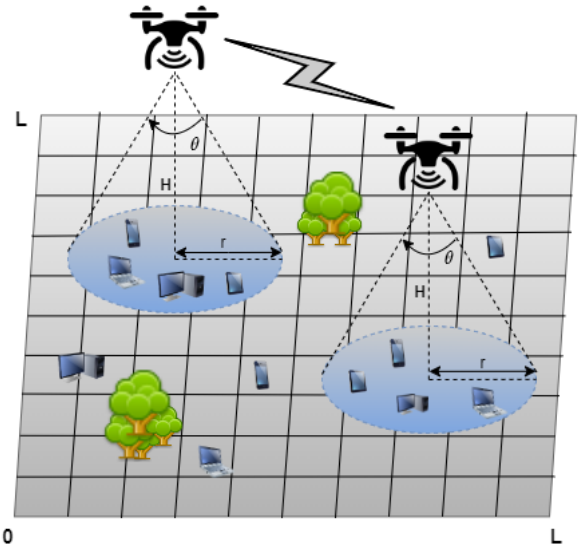

Figure 5. UAVs serving ground users by sharing local information. The domain is discretized into a grid of length L, each UAV is at a fixed height H, and the coverage radius is r.

4.1. User Distribution Framework

The user distribution approach simulates how users are distributed throughout distinct places within a defined coverage area. In the system, the target area is L by L square meters and there are M users. We consider all the ground users to be stationary. To make the model simpler and facilitate easier calculation and analysis, the target region is discretized, and UAVs execute discrete actions. Each grid space size is 100. The system designates four primary hotspots. It then loops through each hotspot, providing random coordinates for users within a predetermined radius from the hotspot. These coordinates are initially created in polar form and then transformed to Cartesian coordinates for convenience of representation. The K users are in those hotspots. The remaining $M - K$ users are distributed uniformly within the entire target area, including the hotspot positions. The goal of this technique is to construct a realistic simulation of the user distribution among various regions within the defined coverage area, providing geographical dynamics and insights into user behaviors. The user distribution framework is depicted in Figure 6.

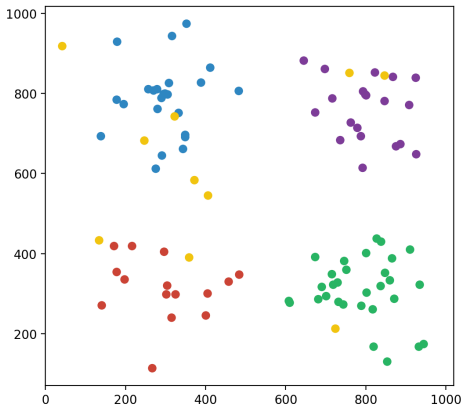

Figure 6. User distribution framework; hotspot users are in represented by red, green, blue, and purple, and randomly distributed users are in yellow.

4.2. Initialization of UAVs

In this system, M users are covered by N UAVs. The standard height for all UAVS is fixed at H meters. All of the UAVs are initially positioned at the grid's origin $(0, 0)$.

4.3. User Coverage Method

To cover as many users as possible, we use the two-sweep method described in Algorithm 1. In wireless communication systems, the user association problem is frequently solved using the two-sweep method [37]. In order to assign users to access points in the most effective way, two independent stages of operation are carried out. During the first sweep, which occurs in a 2D environment with the UAVs and users, each individual user included in the set U of all users makes a connection request to the closest UAV in the set I of all UAVs. One user is able to send a single request at a time, preventing many UAVs from serving the same user. Once the UAVs have received the connection requests C_r, they can identify which users are inside their coverage radius (i.e., within a Euclidean distance d_{ij} that is less than or equal to the coverage radius r). The distance between user j and UAV i is the Euclidean distance

$$d_{ij} = \sqrt{(x_{ui} - x_j)^2 + (y_{ui} - y_j)^2}.$$

Here, UAV i coordinates are represented by x_{ui} and y_{ui}, whereas user j coordinates are indicated by x_j and y_j. The users inside each UAV's service area are then sorted according to their distance from the UAV. As the UAV fills to capacity C, users are admitted in order of increasing distance. If a user is not connected to any UAVs in the initial sweep, a second sweep is performed to find the closest UAV that can cover the user. The user cannot access the network if there is no UAV nearby to cover them.

Algorithm 1 User Association algorithm of decentralized DQN

Initialization:
Input: Connection request C_r, User association q for $u \in U$,
User count for a UAV C for $i \in I$, Coverage radius for UAV r, Distance between UAV and user d, Closest UAV Closest-UAV, Maximum capacity of a UAV M_c
for each user $u \in U$ **do**
    Closest-UAV $\leftarrow$ argsort(d) for all $i \in I$
    if $d \leq r$ **then**
        $C_r \leftarrow 1$
    end if
end for
for each UAV $i \in I$ **do**
    Closest-user $\leftarrow$ argsort(d) for all $u \in U$ and $C_r = 1$
    for each user $u \in$ Closest-user **do**
        if $q < C$ **then**
            Set $q \leftarrow 1$
            Set $C \leftarrow 1$
        end if
    end for
end for
for each user $u \in U$ **do**
    if any user is not associated with any UAV **then**
        Closest-UAV $\leftarrow$ argsort(d) for all $i \in I$
        for each $i \in$ Closest-UAV **do**
            if $C \leq M_c$ and $d \leq r$ **then**
                Set $q \leftarrow 1$
                Set $C \leftarrow 1$
            end if
        end for
    end if
end for

4.4. Design

For a UAV mesh network, we developed a decentralized multi-agent deep Q-learning framework in which each UAV is autonomous and accountable for its subsequent actions in order to maximize the total user connectivity. The state space, action space, and reward function of UAV design will be covered in this section.

4.5. State Space

Since the state of the environment provides the basis for the agent behavior and is also used to represent and map the environment, its design is very significant. The placements of UAVs directly determine how many users are covered in each epoch, and this has a major effect on the goal of maximization. For simplicity, we discretize the whole area into equal small grid cells, where each cell indicates a possible UAV agent location. Each UAV agent's state is defined at each time step using its unique position coordinates within this grid. While the UAVs fly horizontally, we simply take into account the 2D coordinates (x, y); the epoch duration is 100 steps in the grid. The movements of UAVs are limited to within the target region. During training, the DQN uses agents local states as an input, as well as the individual and overall coverage rate of agents.

4.6. Action Space

Actions include both an agent's output and the environmental input. Each drone has five different possible actions. These motions include moving one step in any direction (left, right, forward, backward), or remaining still. We do not consider the velocity of the UAV movement in the dynamics, which allows us to have better control over the UAV movement inside the state space. This reduces the complexity of directing their paths and interactions in the environment. In addition, there is a border condition in the target area. Any UAV will return to its initial state if it takes any action to leave the grid from a border state. When calculating the reward, a flag is utilized to track the UAVs that attempt to leave the grid and penalize them. Each UAV's available actions can be expressed as follows:

$$a_i \in \{\text{left}, \text{right}, \text{forward}, \text{backward}, \text{remaining stand-still}\},$$

where a_i denotes the action of UAV i.

4.7. Reward Function

Reward is the feedback given to the agent after they have acted in response to specific environmental conditions. The reward R is determined by the number of users covered by a UAV because the objective is to maximize the number of covered users. The reward function has multiple additional rewards and negative reward conditions. Any attempt by a UAV to cross the border will result in a penalty. The UAV will receive a further reward if the overall coverage rate exceeds a threshold value; if the overall coverage rate falls below the threshold value, the UAV will be penalized. The UAVs attempt to maximize their own reward by sharing reward information among themselves. This is called the average reward. The average reward R is given by the following:

$$R = \begin{cases} \frac{1}{N}\sum_{i=1}^{N}(R_i - p_i) & \text{if flag}_i \neq 0, \\ \frac{1}{N}\sum_{i=1}^{N}(R_i + q_i) & \text{if overall\_coverage\_rate} > x, \\ \frac{1}{N}\sum_{i=1}^{N}(R_i - r_i) & \text{if overall\_coverage\_rate} < x, \\ \frac{1}{N}\sum_{i=1}^{N} R_i & \text{otherwise,} \end{cases}$$

where R_i is the reward for the i-th agent, p is the penalty value for the ith agent who is tried to go out of bounds, r is the penalty value for the i-th agent when the overall coverage rate is below the threshold level, q is the additional reward value for the i-th agent when overall coverage rate is below the threshold level, x is the threshold value of the overall coverage rate, and N is the total number of agents.

4.8. Training

To train agents to make decisions in complicated environments, we propose utilizing the DQN architecture. Using the Rectified Linear Unit (ReLU) activation functions to introduce non-linearity, the neural network model consists of two fully linked hidden layers with 400 neurons each. Given the current states and the individual and total coverage rate of the environment's agents, this architecture enables the network to approximate Q-values for various actions. We train all the agents in a decentralized manner using a single neural network model.

The reinforcement learning agent facilitates the process of learning by coordinating its interactions with its environment and gradually improving its ability to make decisions. The main goal of the agent is to optimize its actions in order to maximize the cumulative rewards. This is accomplished by carefully balancing exploration and exploitation. In order to learn more about the environment and the rewards associated with different activities, the agent is frequently encouraged to experiment with a range of actions and behaviors during the exploration phase. This exploration is often guided by a policy known as an epsilon greedy policy, described in Algorithm 2, which finds a balance between the exploration process and the exploitation of activities that were shown to be effective. Once the agent has completed the exploration phase, they enter the planning phase. In this phase, the agent updates its policy or plan to maximize its expected reward using the knowledge it gathered during the exploration phase.

Algorithm 2 Epsilon-greedy policy

```
Input: Random variable r ∈ (0, 1); epsilon value ϵ ∈ (0, 1); Q-function; state
if r ≤ ϵ then
    action ← random available action
else
    action ← maximum Q-value in Q[state];
end if
```

To improve sample efficiency and stabilize learning, experiences are stored in a replay buffer during training. These experiences comprise state–action pairings, accompanying rewards, future states, and terminal flags. Using an MSE loss function that was optimized using the Adam optimizer, the agent's primary goal is to reduce the difference between the target and forecast Q-values. Multiple epochs of environmental interactions are combined into each iterative episode that makes up the training process. To stabilize training and reduce overestimation bias, the target network is frequently updated to reflect the parameters of the main network. Optimizing the learning dynamics and convergence speed involves fine-tuning hyperparameters including target network update rate, epsilon, batch size, discount factor, and learning rate. In order to analyze the agent's performance and track its learning progress, episode rewards and other pertinent metrics are tracked during the training process. Through iteratively updating the neural network parameters with experiences taken from the replay buffer, the agent progressively discovers the most optimal policies to maximize the cumulative rewards in the given environment.

Agent–environment interaction can be divided into sub-sequences whenever the RL algorithms provide a concept of time steps. Until a terminal state or a stopping criterion has

been attained, these subsequences, which are referred to as episodes, consist of recurring interactions between the agent and the environment. The current episode ends when all UAVs have moved and are in their ideal places for that particular episode. A maximum of 750 episodes, each with up to 100 epochs, make up the training. The tested agent goes through 100 epochs of testing. The training is carried out on a Windows 11 server with an Intel Core i7-7700 CPU (Intel Corporation, Santa Clara, CA, USA) running at 3.60 GHz and 16 GB of RAM using Python 3.11. The MADQL algorithm was built using the PyTorch library. Table 3 summarizes the key parameters, which are set according to the UAV simulation requirements for optimal performance. In addition, the Figure 7 illustrates the entire methodology used to position the UAVs within the environment to maximize user coverage.

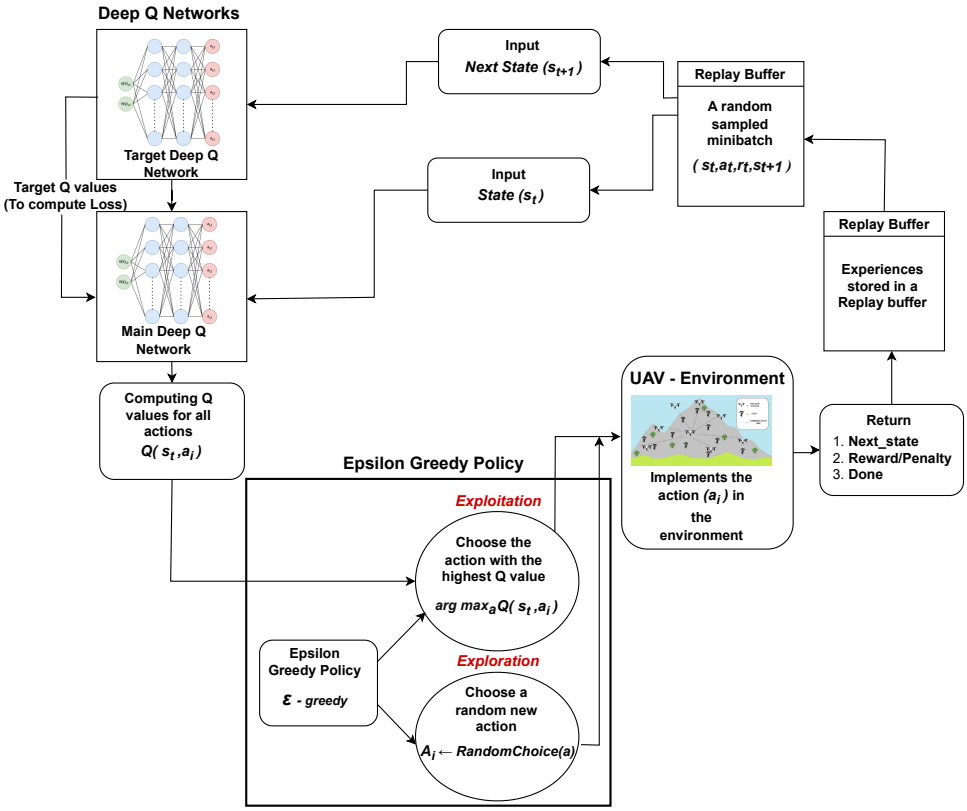

Figure 7. Flowchart of the proposed multi-agent reinforcement learning workflow.

Table 3. Simulation parameters.

Parameter/Software	Value/Detail
Simulation Domain	1000 × 1000 units, discretized into a 10 × 10 grid
Number of Users	100 (distributed via hotspots and random scattering)
Number of UAVs	5
UAV Height	350 units
Coverage Angle	60° (coverage radius = 350 × tan(60°/2))
UAV Capacities	[10, 15, 20, 25, 30]
Learning Algorithm	Centralized and Decentralized Deep Q-Learning
Neural Network Architecture	Two hidden layers with 400 neurons each, ReLU activation
RL Hyperparameters	Discount factor = 0.95; learning rate = 3.5×10^{-4}
	Epochs = 100; Replay buffer size = 125,000;
	Batch size = 512; $\epsilon = 0.10$
Software	Python 3.11.12, Gym 0.25.2, NumPy 2.0.2, Matplotlib 3.10.0, PyTorch 2.6.0
Hardware	Simulated on CPU (with potential GPU configuration)

4.9. Example Simulation Result

This study explores the findings, in which each UAV maximizes its individual reward while also sharing rewards with others to maximize the global reward and cover the maximum number of distributed users, as the reward is based on the number of users covered.

In Figure 8, the convergence curve is shown. The optimal state of the UAVs is displayed in Figure 9, showing their successful collaboration in maximizing the reward function and enabling the UAVs to cover up to 93 users.

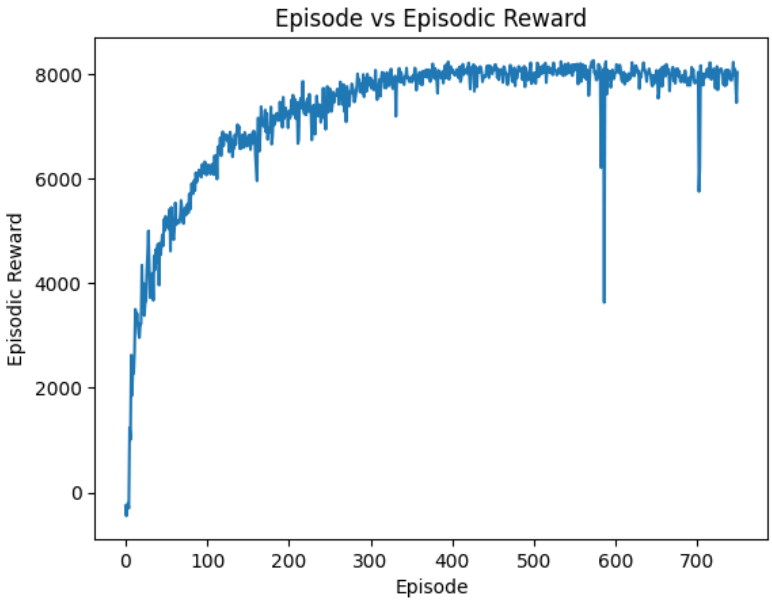

Figure 8. Convergence plot of connected users per episode.

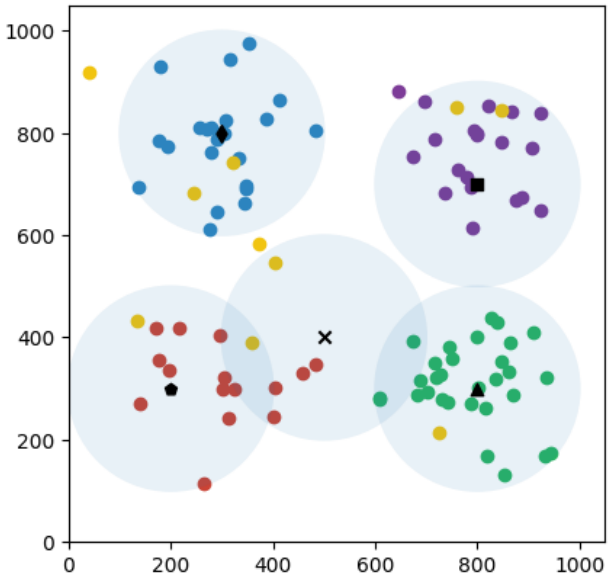

Figure 9. Simulation result with the reward information exchanged among the UAVs. Each dot represents a user, with colors indicating the hotspots used to generate the user distribution (see Section 4.1), with yellow indicating users randomly sampled in the environment. Each of the five UAVs are indicated by a black shape, with their coverage area indicated by light blue.

To improve positional accuracy, we reduced the grid resolution from 100 m to 50 m in our most recent UAV simulation update. The updated technique involves the UAV taking 100 m steps for the first 60 steps before switching to 50 m steps for the final 80 steps.

Figure 10 depicts the convergence plot for this approach. This change enables the UAV to make more precise movements as it approaches the optimal position, increasing user coverage from 93 to 95. Figure 11 shows the optimal position of the UAVs. Despite this development, the improved grid resolution and smaller step sizes cause a substantial increase in simulation time. The previous setup took roughly 200 min, whereas the current configuration takes around 300 min. As a result, achieving more exact optimal positioning requires longer simulation times, emphasizing the importance of balancing precision and computational efficiency.

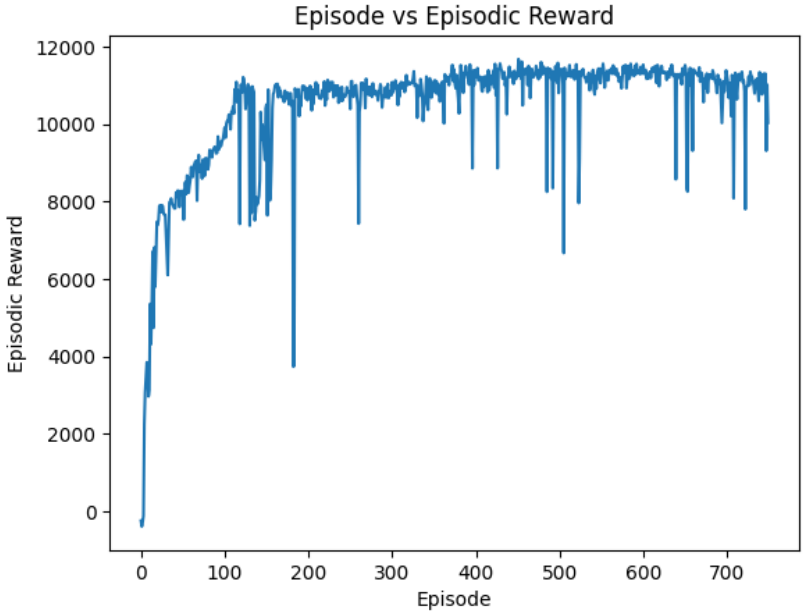

Figure 10. Convergence plot of connected users per episode when the grid space is 50 m.

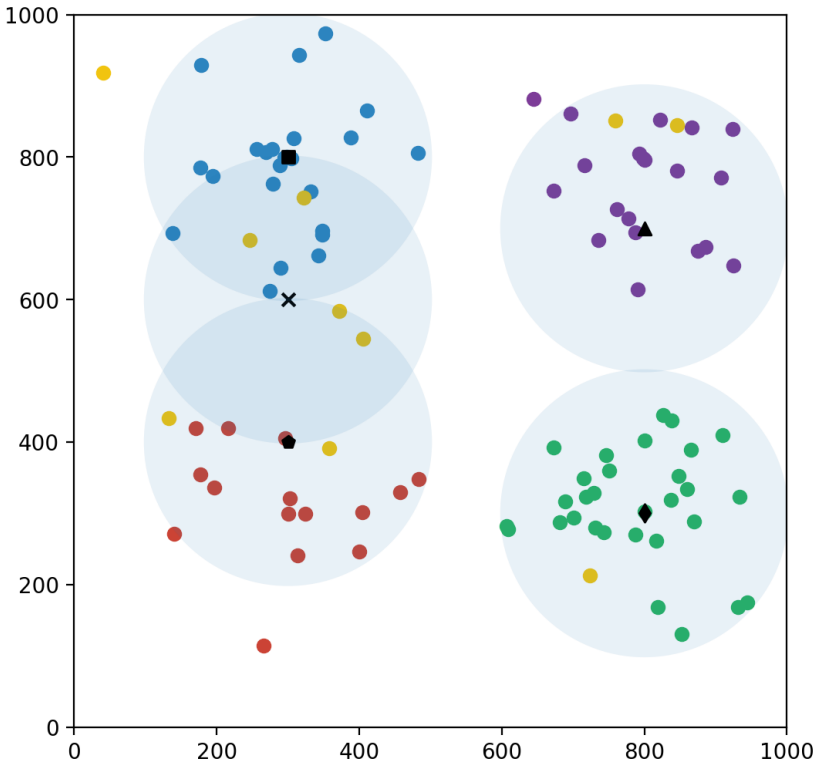

Figure 11. Simulation result with reward information exchange among the UAVs when the grid space is 50 m. For the meaning of the colors and shapes, please see the caption of Figure 9.

In summary, achieving more precise optimal positioning necessitates longer simulation times, highlighting the need to balance precision with computational efficiency.

5. Training Methods and Policy Adaptability in Different Environmental Settings

In this section, we focus on the stochastic environment and investigate various user distributions inside a specific target area, with the goal of improving our model's adaptability and robustness [62]. To accurately imitate real-world settings, we purposely add diversity in user distribution across different episodes when training our model. This deliberate variation is critical, providing our model with the capacity to handle the many user distribution circumstances seen in practical applications. By exposing our model to a variety of user distributions during training, we hope to foster resilience and improve its capacity to be generalized across different types of user distributions. We hope to gain a thorough understanding of how our model responds to changing levels of uncertainty and adapts to a variety of environmental conditions, ultimately improving its performance in real-world applications characterized by stochastic dynamics.

We expand our investigation by applying the training data from one setting to another and analyzing the results using histograms. Our goal is to evaluate the performance of the model learned in one environment when applied to a different, but comparable, environment. By displaying the results with histograms, we can see how well the model generalizes across settings and responds to changes in environmental conditions. This research allows for us to assess the model's resilience and transferability, providing information on its ability to manage stochastic environments.

5.1. Motivation

Our approach is motivated by the computational expenses associated with training reinforcement learning models. Given the computational resources necessary for training, we intend to accelerate the process by training the model in a given environment and then evaluating it in similar scenarios. This method allows us to use the knowledge learned during training to similar situations, lowering the computational overhead required when building numerous models from scratch for each environment. By focusing on training the model in one environment and testing it in others, we hope to increase efficiency while maintaining the model's effectiveness and flexibility across multiple related scenarios. This strategy not only minimizes computational resources, but it also makes it easier to use reinforcement learning techniques in real-world circumstances.

5.2. Stochastic User Distribution

In Section 4, user positions remained consistent across all episodes of training. In this section, we consider two additional distribution scenarios. To distinguish the various user distributions, we refer to the user distribution from Section 4, in which the user positions remain constant as a type I distribution. Figure 12 provides a sample of a type II user distribution in which user positions are randomly sampled in each episode while hotspot coordinates and user counts remain unchanged. This design enables an investigation of how user distribution patterns change over numerous episodes while the overall structure of the hotspots remains unchanged. In contrast, Figure 13 displays a sample of a type III user distribution scenario in which both the placements and the number of user counts are randomly sampled in each episode but the hotspot coordinates remain constant. This dynamic design allows for research on how changes in user counts and positions affect the overall distribution landscape while keeping hotspot locations consistent across episodes.

These different distribution techniques provide useful insights into the dynamic nature of user distribution and its consequences for system efficiency.

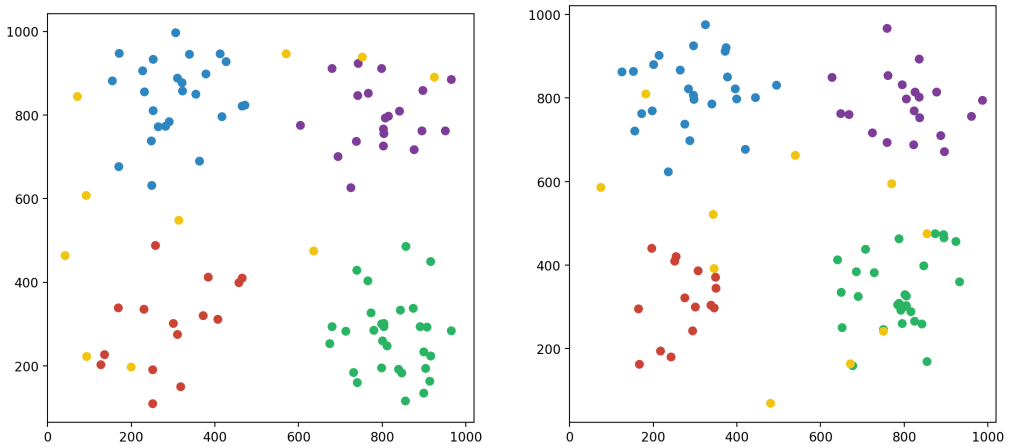

Figure 12. Type II user distribution samples in which user positions are randomly sampled in each episode while keeping hotspot coordinates and user counts fixed. For the meaning of the colors and shapes, please see the caption of Figure 9.

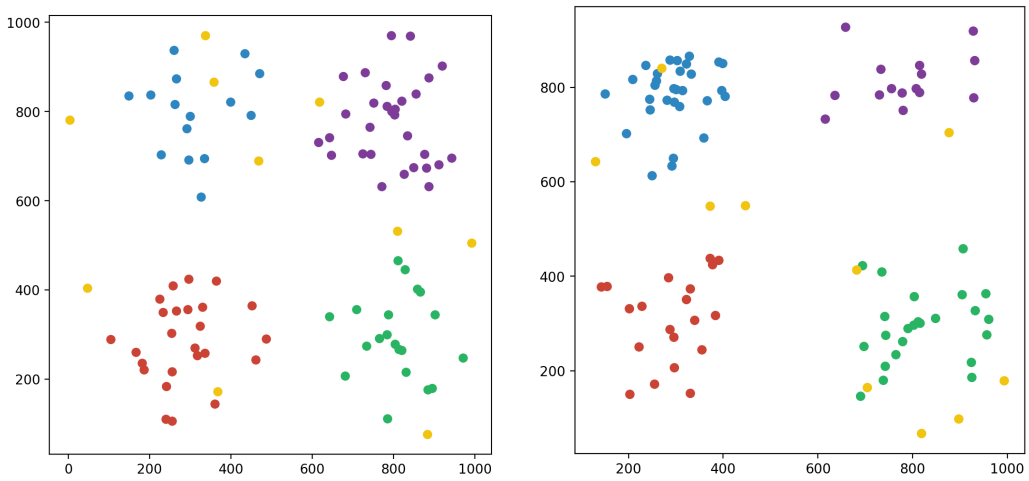

Figure 13. Type III user distribution samples in which user positions and the number of user counts are randomly sampled in each episode while the hotspot coordinates are kept constant. For the meaning of the colors and shapes, please see the caption of Figure 9.

5.3. Training

To optimize the UAV's decision-making within the network, we employed the same training procedure as in Section 4. The strategy used for user distribution during the training phase, however, differs significantly. In contrast to the fixed user distribution employed in the earlier training cycles, we introduced dynamic adjustments to the user distribution following each episode. This adaptable approach is in line with real-world situations, in which user populations may fluctuate over time as a result of events, shifting mobility patterns, or modifications to the surrounding environment. As they were trained under a variety of user distribution scenarios, the UAV agents are exposed to a broad range of working environments, which improves their ability to adapt to changing user environments. After conducting a thorough experiment and analyzing the simulation results, we evaluate how well the training framework performs in enabling UAV agents to navigate and serve users in a variety of distribution patterns, which helps to optimize network performance in dynamic and unpredictable environments.

5.4. Simulation Results

For three learning algorithms in a UAV-assisted network, Figure 14 shows a comparative study of the episodic rewards received across 750 episodes. Because the user distribution remained constant throughout every training event, Type I exhibits a rapid learning efficiency that is typified by a swift ascent to a stable high-reward plateau. As seen by the longer progress to equivalent reward levels with larger fluctuations, Type II, which has a user sample that varies with each training event, shows a more measured learning curve and moderate stability. With the highest degree of variety, Type III exhibits comparable reward attainment but a more intricate learning environment. Type III introduces variability in both user samples and the number of users per hotspot during training. In spite of these variations, all three kinds show convergence, highlighting how well the learning algorithms adapt to different training scenarios.

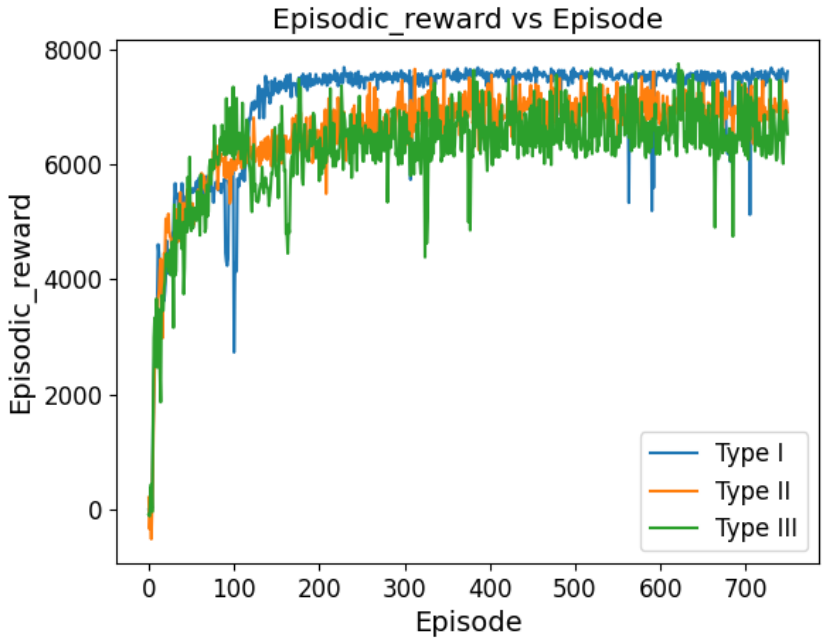

Figure 14. Comparison of convergence plot.

The variable spread of and peaks in the distributions in the histograms presented below show how the number of users connected per episode varied among the simulation settings. The red dashed line, which represents the mean number of users connected, moves around the charts to show the various average connectivity in each case.

We conducted an empirical study to examine the effectiveness of a policy designed for one distribution of environments on environments taken from a different distribution. This investigation was essential for evaluating the training policy's robustness and adaptability in a variety of environmental circumstances.

The test results obtained for a policy that was learned on a Type I user distribution and then applied to the same training distribution are shown in Figure 15 (top left). Applying the taught policy to a similar user distribution situation resulted in the policy connecting 86 users in each testing episode. When the same policy was applied to a Type II user distribution, as shown in Figure 15 (top center), a broader frequency distribution with a lower mean of about 82 users connected is revealed. This suggests that the user sampling variability may have somewhat lowered performance. Lastly, due to changes in both user sampling and the number of users per hotspot, Figure 15 (top right) shows an even greater decline in user connectivity when the policy was applied to the Type III user distribution, with the mean connecting to around 76 users.

Test results are shown in Figure 15 (middle left) for a policy learned on a Type II user distribution that, when applied to a Type I distribution, achieved connectivity for 83 users. The performance of the identical Type II-trained strategy with its native Type II distribution is shown in Figure 15 (middle center), where the mean connectivity decreased to about 82 users and the distribution is more dispersed. When evaluated under Type III conditions, the Type II-trained strategy in Figure 15 (middle right) shows an even broader spread, with the mean number of people linked falling to about 80. These outcomes demonstrate how flexible the strategy is, preserving a high level of user connectivity in a range of distributions.

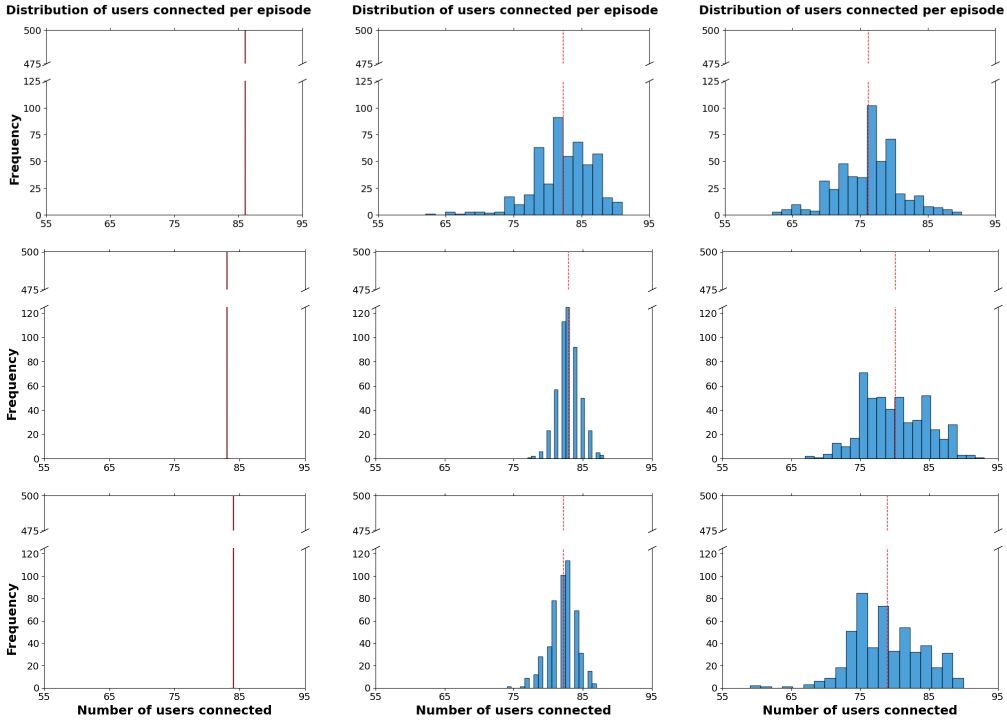

Figure 15. Histograms showing the distribution of the number of users connected after applying the trained policy for 500 episodes. The horizontal axis represents the number of users connected, while the vertical axis shows how frequently each number of connected users occurred. The average number of connected users is shown by a red dashed line on each graph. Each column of histograms indicates a different type of user distribution (Type I, II, and III), and each row applies a different policy (trained on Type I, II, and III user distributions). In the first column (corresponding to Type 1 user distributions), the black and red lines are identical since there is only a single user distribution so all simulations achieve the same number of connected users.

After training with a Type III user distribution, the policy connected an average of 84 users per episode in a Type I scenario. The test results are shown in Figure 15 (bottom left). Under Type II conditions, the identical technique shows a greater dispersion in connectivity with a mean of about 81 users in Figure 15 (bottom middle). After additional testing under Type III settings, the number of connected users showed the widest range, with the mean falling to roughly 79, as illustrated in Figure 15 (bottom right).

The testing results show that the user connectivity policy was robust when applied to different user distributions; in all circumstances, the policy covered about 80 percent of users. The policy was able to maintain a high degree of consistency every time. This performance consistency, which accounts for a significant majority of the users, irrespective of the distribution method, highlights the algorithm's resilience and potential utility in a range of operational environments.

6. Robustness Against Jamming Attacks

Unmanned aerial vehicles (UAVs) have shown great promise in addressing a variety of communication network difficulties [62]. Jamming attacks remain one of the main problems in wireless networks, despite the significant technological advancements in this field. The widespread occurrence of wireless networks based on UAVs has made jamming assaults a significant obstacle to the effective implementation of these technologies [63]. The term "jamming attack" describes the illegal creation of interference to an ongoing communication in order to cause disruptions or deceive users of wireless networks. In order to interfere with the wireless networks' regular operation, the jammer sends out jamming signals. Wireless networks are therefore still susceptible to a variety of jamming attack methods. Because of the UAV's great degree of adaptability, it is possible to mitigate the jamming attack or perhaps completely prevent its detrimental effects. Nonetheless, the jammer may target the UAV itself in an attempt to impede the regular operation of UAV-based communication networks [64].

The authors of [65] presented a UAV-aided anti-jamming system for cellular networks. Reinforcement learning algorithms are used by the UAV to select relay policies for users in cellular networks. To counteract the jamming attack, the UAV routes traffic from the jammed base station to a backup base station. In [66], the authors examined an anti-jamming communication within a UAV swarm when jamming was present. The UAV maximizes its data reception by taking use of the degree of freedom in frequency, velocity, antenna, and regional domain while a jammer targets the network. In [67,68] suggested a combined optimization for the UAV trajectory and transmission power in anti-jamming communication networks. The optimization problems are solved using a Q-learning-based anti-jamming approach and a stackelberg framework.

6.1. Problem Setup

In our current system model, any UAV can be jammed by malicious entities while providing services to users, resulting in a drop in overall coverage because the jammed UAV can no longer serve users. When a UAV is jammed, it loses its capacity to communicate with other UAVs, which means it cannot send or receive information or signals from the remaining UAVs. The jammed UAV can only hover indefinitely, with no further activity. In this approach, jamming occurs after the UAV has taken 30 steps within the grid from the origin, allowing it to achieve its optimal location before being jammed.

Our key objective is to enable the UAV to quickly react, reposition itself, and take the optimal position to cover the maximum number of users.

6.2. Methodology

To optimize user coverage in the event of UAV jamming, we incorporated a straightforward yet efficient heuristic into our proposed strategy, as described in Algorithm 3. Any UAV that becomes jammed instantly experiences a zero percent individual coverage rate. As soon as one of the UAVs becomes jammed, the other UAVs will identify its location. Then, the individual coverage rate of each remaining UAV will be determined. The UAV with the lowest coverage rate will travel in the direction of the jammed UAV if any of these serving UAVs have individual coverage rates that are lower than the jammed UAV's initial coverage rate. The UAVs take one step at a time toward the jammed UAV. The UAV with the lowest coverage rate applies vector calculations to precisely navigate to the jammed UAV's location. The UAV that goes to the jammed UAV's location resumes normal operation, seeking to restore coverage to the previously served area. If the jammed UAV's coverage rate prior to jamming was less than or equal to the remaining UAVs' coverage rates, no UAV

will approach the jammed UAV's position. This is because the overall goal is to maintain or increase user coverage across the network.

This technique was included in the main algorithm to assist in the dynamic relocation of UAVs when one is jammed. Its integration prevents the main algorithm from getting stuck in the local minima, allowing for UAVs to continuously adjust their positions to optimize user coverage, which is the primary goal.

Algorithm 3 UAV Handling in Jammed UAV Scenario

Input: Number of UAVs N, Index of jammed UAV j, User count per UAV $C[i]$ for $i \in [0, N)$, Current and target positions for UAVs.
if $j \neq$ None **then**
    Determine Valid UAV Indices:
    $V \leftarrow \{i \mid i \in [0, N), i \neq j, \text{UAV}_i \text{ not in transit}\}$
    Compare User Counts:
    $u_j \leftarrow C[j]$ ▷ Number of users for the jammed UAV
    $u_{min} \leftarrow \min(C[V])$ ▷ Minimum number of users among valid UAVs
    if $u_j > u_{min}$ **then**
        if assigned_uav = None **then**
            Select UAV with Minimum Users:
            uav_min $\leftarrow \arg\min(C[V])$
            assigned_uav $\leftarrow$ uav_min
        else
            uav_min $\leftarrow$ assigned_uav
        end if
        Obtain Current and Target Positions:
        current_position $\leftarrow$ Position[uav_min]
        target_position $\leftarrow$ Position[j]
        if current_position $\neq$ target_position **then**
            Calculate Movement Direction:
            direction_vector $\leftarrow$ target_position $-$ current_position
            step_x $\leftarrow$ sign(direction_vector[horizontal])
            step_y $\leftarrow$ sign(direction_vector[vertical])
            Update UAV Position:
            next_position $\leftarrow$ current_position $+$ [step_x, step_y]
            Position[uav_min] $\leftarrow$ next_position
        else State **Reset Assigned UAV:**
            assigned_uav $\leftarrow$ None
        end if
    end if
end if

6.3. Simulation Results

The simulation outcomes illustrate the positions of UAVs before and after a jamming event, providing a visual representation of both UAVs and users in Figure 16. Here, we present one specific case, while similar figures and their numerical data will be provided for other cases. In the left figure, the UAVs are initially positioned optimally, covering a total of 86 users. However, after UAV 1, marked with a star, is jammed, its individual coverage rate drops to zero, leading to a significant decrease in overall user coverage, as shown in Figure 16, dropping to 45 users. In response, the remaining UAVs adjust their positions within a few time steps, following the relocation strategy outlined by our algorithm. The figure on the right depicts the new positions after the relocation, where the UAV marked with a triangle moves to the location of the jammed star-marked UAV to cover the affected users. This adaptive response enables the network to recover, achieving a new coverage level of 68 users, as indicated in Figure 17.

Figure 18 depicts the dynamic behavior of five UAVs in reaction to jamming events, with each UAV jammed independently after a time step of 30. The plot depicts how the number of users connected to each UAV varies before and after the jamming, showing the

overall network coverage rate and the subsequent recovery process. Initially, all UAVs performed optimally, and the total network coverage rate reached a maximum of 86 users. However, when a jamming event occurred that targeted one UAV at a time (as represented by the red dashed line at time step 30), the coverage rate quickly declined, indicating that the jammed UAV was unable to provide service. When the first UAV (blue line) was jammed, its coverage rate decreased from around 86 users to 45, resulting in a considerable decrease in overall coverage. However, due to the algorithm's relocation method, the remaining UAVs modified their placements, resulting in a steady recovery of overall coverage that stabilized at around 68 users. For the second UAV (orange line), the remaining UAVs shifted their coverage, settling at around 68 users. The third UAV (green line) maintained a relatively consistent coverage, starting at roughly 76 users, with minimal influence from jamming due to its ideal placement. When UAV four was jammed, its coverage decreased from roughly 86 to around 57 users. After the jamming occurred, the relocation mechanism allowed the remaining UAVs to shift their positions, gradually restoring the overall coverage to around 68 users. Similarly, when UAV five was jammed, the coverage rate dropped dramatically, to roughly 66 users, before returning to 71 users following relocation. These findings show that the algorithm is effective at dynamically reallocating UAVs to enhance user coverage, even in the face of severe disruptions such as jamming attempts.

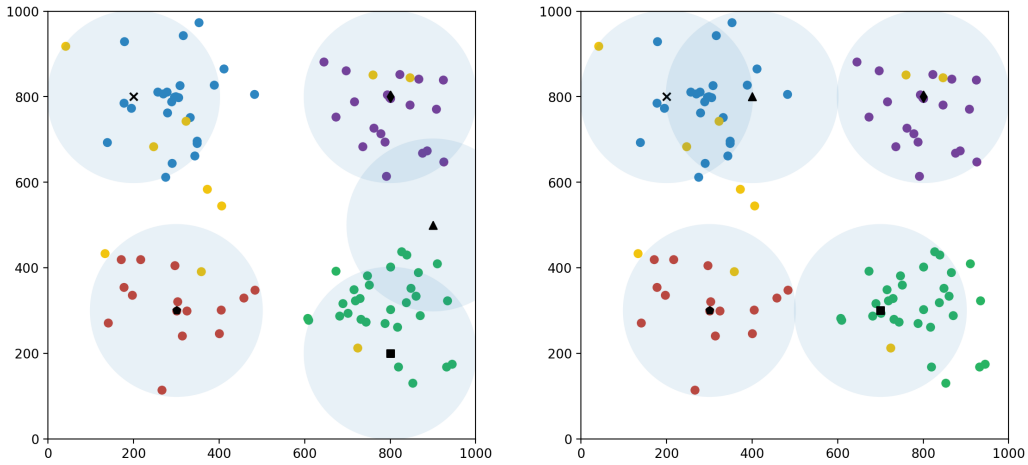

Figure 16. UAV relocation to maintain maximum user coverage before and after a jamming incident. For the meaning of the colors and shapes, please see the caption of Figure 9.

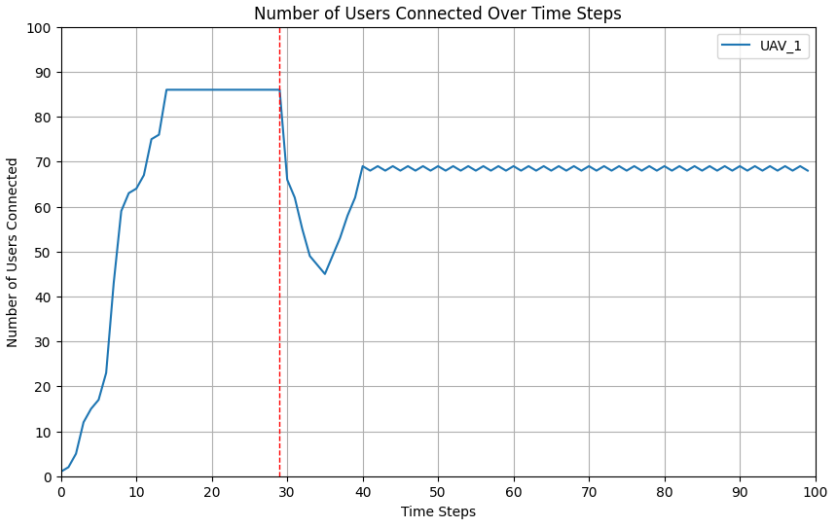

Figure 17. Number of connected users over time (blue). The vertical red dashed line indicates the time of the jamming attack.

After each jamming incident, the network eventually settled on a new equilibrium coverage level that iwa somewhat lower than the initial peak but much greater than the immediate post-jamming condition. This result demonstrates the relocation strategy's effectiveness in maintaining a strong network performance and maximizing user connectivity under adverse conditions.

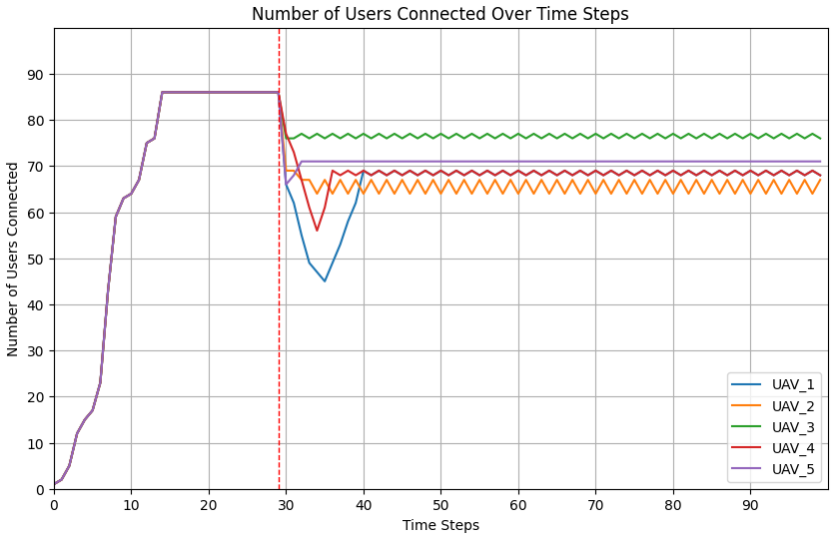

Figure 18. The number of connected users for each of the five UAVs over time. The vertical red dashed line indicates the time of the jamming attack. Before the jamming attack, all UAVs steadily increased their number of connected users. The number of users decreased noticeably after the jamming event. Over time, the UAVs adapted to counteract the impacts of the jamming, stabilizing their user connections.

7. Centralized Problem Approach

In contrast to the distributed approach shown in the previous section, we now consider the use of a centralized algorithm to place the UAVs in order to maximize the number of connected users. Here, a centralized controller computes the actions of all UAVs. This approach is illustrated in Figure 19.

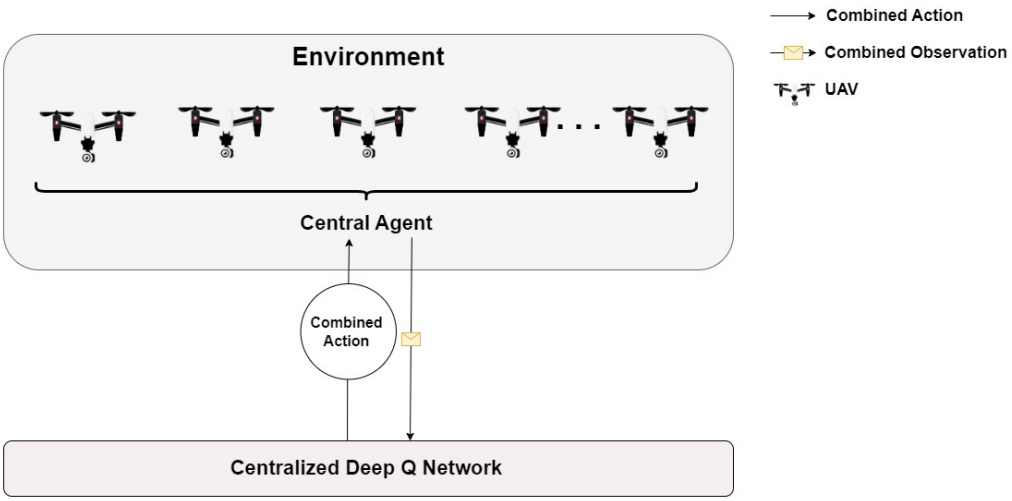

Figure 19. Centralized RL approach where all the UAVs are controlled by a single agent.

7.1. Centralized Algorithm Workflow

The centralized approach consists of a central neural network that can control the actions of all the UAVs [26,69]. The global state of all the UAVs, $s_{global}(t)$, provides the

inputs to the main network. The main deep Q network approximates the optimal Q values for all the possible joint actions of the UAVs and the Q values for the current state are predicted. The target network, in parallel, provides the target Q values that are used in the computation of the loss function and the main network weights are adjusted based on this. The best set of joint actions that holds the maximum Q value is chosen, and this action is considered for all the UAVs in the environment. The epsilon greedy policy acts as the deciding factor and the trade-off between exploration and exploitation is determined using the ϵ parameter.

After choosing the best joint action, it is sent to the environment and the actions are implemented. The environment, in return, provides the combined reward/penalty for the collective action taken, the next transitioned state information, and information regarding whether the terminal state was reached. This collective information is stored in the replay buffer from which the sample of memories that contains these information is taken and used to train the neural network.This iterative process is carried out in each step for several episodes, more than the multi-agent algorithm, as a large amount of data need to be handled. The calculation of the optimal value takes time since the network has to find the optimal joint actions in a combined fashion for all the UAVs. This strategy is used in optimal UAV position placement to obtain the maximum user coverage using a single controller. The total runtime of this algorithm is longer than that of the multi-agent algorithm, as a large amount of data are handled by the neural network. The centralized algorithm flow is shown in Algorithm 4.

Algorithm 4 Centralized DQL Approach to UAV position optimization

Initialization:
Initialize the centralized main DQN $Q(\mathbf{s}_{\text{global}}, \mathbf{a}_{\text{joint}}; \theta)$;
Initialize the centralized target DQN $\hat{Q}(\mathbf{s}_{\text{global}}, \mathbf{a}_{\text{joint}}; \hat{\theta})$ with weights $\hat{\theta} = \theta$;
Initialize the centralized experience replay buffer $\mathcal{B}$;
Learning Process:
for each episode $e = 1 \dots E$ **do**
    Initialize the global state for all UAVs to starting state $\mathbf{s}_{\text{global},0}$;
    for each timestep $t = 1 \dots T$ **do**
        Execute centralized ϵ-greedy policy to select joint action $\mathbf{a}_{\text{joint},t}$;
        Take joint action $\mathbf{a}_{\text{joint},t}$, observe joint reward r_t, and next global state $\mathbf{s}_{\text{global},t+1}$;
        Store transition $(\mathbf{s}_{\text{global},t}, \mathbf{a}_{\text{joint},t}, r_t, \mathbf{s}_{\text{global},t+1})$ in $\mathcal{B}$;
        if enough data in $\mathcal{B}$ **then**
            Sample minibatch and perform a gradient descent step on Q;
            Periodically update $\hat{Q}$ with weights from Q;
        end if
    end for
end for

7.2. Centralized Deep Neural Network

The centralized deep neural network is responsible for the calculation of the Q values for the joint action. This is shown in Algorithm 5. The neural network in the multi-agent approach is computationally easy as the input dimension and output dimension of the neural network are not large. However, in the centralized approach, the input dimension of the neural network depends on the total number of UAVs employed in the environment. The output dimension of both the network grows larger in accordance with the number of actions and number of added UAVs. This neural network structure is used in both the main network and the target network of the centralized algorithm. The centralized neural network structure is shown in Figure 20.

Algorithm 5 Training centralized DQN for UAV

1: Initialize replay buffer $\mathcal{B}$, centralized action-value function Q with weights θ, and target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
2: **for** each training iteration **do**
3: Sample minibatch of transitions $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')$ from $\mathcal{B}$
4: Set $y = \begin{cases} r & \text{for terminal } \mathbf{s}' \\ r + \gamma \max_{\mathbf{a}'} \hat{Q}(\mathbf{s}', \mathbf{a}'; \theta^-) & \text{otherwise} \end{cases}$
5: Perform a gradient descent step on $(y - Q(\mathbf{s}, \mathbf{a}; \theta))^2$ with respect to θ
6: Every C steps reset $\hat{Q}$ to Q: $\theta^- \leftarrow \theta$
7: **end for**

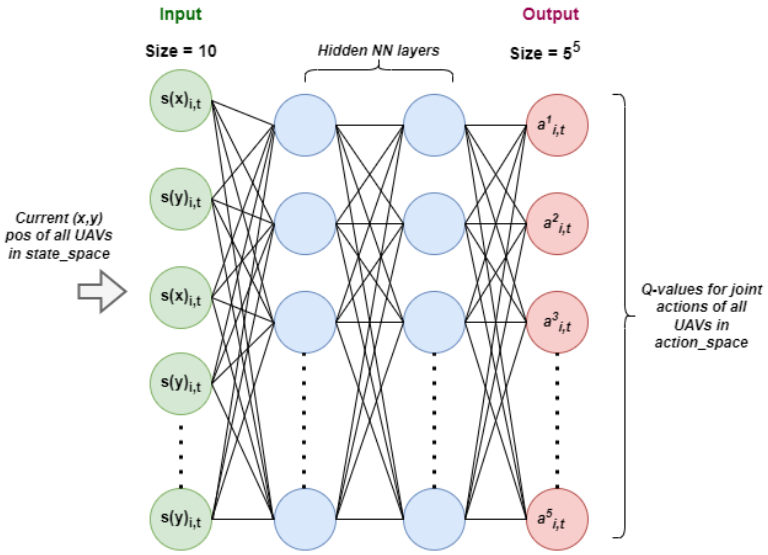

Figure 20. Centralized deep Q network structure for the main network and target network. Here, $\mathbf{s}(x)_{i,t}$ and $\mathbf{s}(y)_{i,t}$ denote the states of all the UAVs, and $\mathbf{a}_{i,t}$ denote the Q values for the combined joint action.

7.3. Main Network

The main network is responsible for decision-making using the Q values, which determine the best joint action to take. The dimension of the input layer corresponds to the state space. For example, if there are five UAVs in the environment and each UAV has two coordinates, then the dimension of the state space would be (5×2), which is 10. This would provide the input dimension of the main network and output the Q values for the joint action. If there are k different actions, then the dimensions of the output layer would be k^M. That means if there were five UAVs and five actions for each UAV, then the output dimension of the neural network would be 5^5. This is a very large value compared to the output of the multi-agent network.

7.4. Target Network

The target network has the same architecture as the main network. The target network is used in the calculation of the target Q values for combined joint action for the next state. This sets a target value for the main network, which is used in the calculation of loss function. The error is the measure of the difference between the expected cumulative reward for the joint action and the current cumulative reward after taking the current joint action. The target Q value is calculated as

$$Q_{\text{target}}(\mathbf{s}_{\text{global}}, \mathbf{a}_{\text{joint}}) = r(\mathbf{s}_{\text{global}}, \mathbf{a}_{\text{joint}}) + \gamma \max_{\mathbf{a}'_{\text{joint}}} Q_{\text{target network}}(\mathbf{s}'_{\text{global}}, \mathbf{a}'_{\text{joint}}) \quad (3)$$

where $\mathbf{s}_{\text{global}}$ is the current global state, $\mathbf{a}_{\text{joint}}$ is the joint action taken by all UAVs, $r(\mathbf{s}_{\text{global}}, \mathbf{a}_{\text{joint}})$ is the immediate reward received after taking action $\mathbf{a}_{\text{joint}}$ in state $\mathbf{s}_{\text{global}}$, $\mathbf{s}'_{\text{global}}$ is the next global state resulting from action $\mathbf{a}_{\text{joint}}$, γ is the discount factor, and $Q_{\text{target network}}(\mathbf{s}'_{\text{global}}, \mathbf{a}'_{\text{joint}})$ is the Q value predicted by the target network for the next state and all possible joint actions. The training of the centralized neural network is shown in Algorithm 5.

7.5. Epsilon Greedy Policy

Similar to multi-agent deep Q learning, the centralized approach uses an epsilon greedy policy to determine the joint action to be taken. The difference here is that the central controller uses a single epsilon greedy policy to determine whether the exploration or exploitation strategy is to be used at each time step. If the epsilon value is greater than the generated random number, then the agent chooses exploration; if the value of ϵ is lower, the agent is prompted to choose the action with the highest Q value, as shown in Algorithm 2.

7.6. Reward Function

In centralized algorithm the reward is based on the total number of users covered by all the UAVs. Instead of assigning a reward and penalty to each individual UAV, we assign a cumulative reward combining all the UAVs, and the penalties are assigned based on the violation conditions. The centralized reward function is shown in Algorithm 6.

Algorithm 6 Reward calculation for the centralized approach

```
Input: N_assoc,i, number of UAVs M, penalty_flag
reward ← sum(N_assoc,i)
for k from 0 to M − 1 do
    if penalty_flag[k] = 1 then
        reward ← reward − 2                ▷ Penalty for boundary exceedance
    end if
end for
Output: reward
```

7.7. Training of the Decentralized Model

The five decentralized models in the multi-agent deep Q network are first trained separately, with a separate neural network controlling each model. The decentralized model architecture has an input layer of size 2, two hidden layers with 400 neurons each and ReLU activation functions, and an output layer of size 5.

After training the decentralized models, neural network weights are saved individually and the averages of the saved decentralized model weights for each compatible layer are computed. In this case, the compatible layers are the hidden layers of the neural network. As their dimensions are different from the centralized models, the input and output layers of the neural network are incompatible.

7.8. Initialization of the Centralized Model

In the centralized model, the weights of each individual decentralized neural network are initialized into a central neural network, which then trains the neural network. This is shown in Figure 21. The centralized neural network architecture has an input layer of size 10, two hidden layers with 400 neurons each and ReLU activation functions, and an output layer of size 3125.

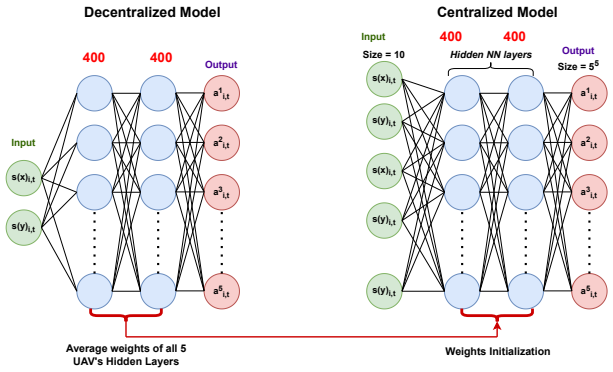

Figure 21. Neural Network initialization for centralized deep Q network structure.

In order to initialize the centralized model, we updated the state dictionary containing the weights of the centralized neural network with the average weights of the decentralized neural network. The centralized neural network's hidden layer structure contains similarities to the decentralized neural network's weight structure. Therefore, these layers were compatible. We calculated and set the weights of the various decentralized neural networks as the average for the centralized model. After training the decentralized models, each weight of the neural network was saved. The averages of the saved decentralized model weights for the compatible layers were then computed. The layers of the neural network that were compatible in this case were its hidden layers. As the dimensions of the input and output layers of neural networks are different from those of the centralized model, they cannot be initialized.

7.9. Centralized DQL Results

Table 4 describes the simulation parameters used in the centralized approach. Figure 22 (left) presents the trend of episode vs. the connected user in episodes obtained from the centralized learning algorithm. This graph highlights the smoothed short-term fluctuations and highlights the long-term trends. Initially, the learning graph increases rapidly, representing the learning process, and after approximately 600 episodes, the algorithm stabilizes and shows convergence towards the optimal policy. This depicts the trial-and-error nature of the learning process where, despite these fluctuations, the algorithm shows an upward trend and convergence.

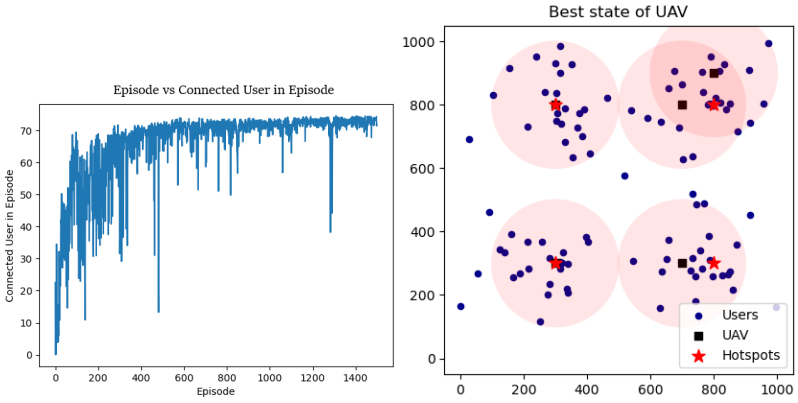

Figure 22. Simulation results of the centralized algorithm; (**Left**) number of connected users in each episode by all the agents; (**Right**) best state of all UAVs.

Table 4. Simulation parameters of the centralized approach.

Parameter	Value
Number of epoch N_{epoch}	100
Dimension of state space	10
Dimension of action space	$5^5 = 3125$
Coverage radius	$h \cdot \tan\left(\frac{\Omega}{2}\right) = 202.07$ m
DQN structure	$N_{layers} = 4, N_{nodes} = 400$
Altitude of UAV h	350 m
Usable bandwidth	$0.9 \times BW_{UAV} = 3.6$ GHz
Learning rate α	0.10×10^{-4}
Resource block per UAV	20
Number of hotspots	4
Bandwidth of resource block	180 KHz
UAV bandwidth	4 GHz
Grid space	100 m
Number of user U	100
Replay buffer size	125,000
Batch size	512

Figure 22 (right) shows the best state of all the UAVs in the centralized approach. Here, the red circles denote the coverage radius of each UAV and the users that fall under the coverage radius of an UAV are considered for connection with that particular UAV depending upon the constraints of the UAV.

7.10. Neural Network Initialization

The performance of the centralized algorithm with the hidden neural network layers initialized using the weights of decentralized network is similar to that of the centralized algorithm. A major difference, however, is the time required for the model to learn the policy. As the centralized neural network is large, it requires extensive training and tuning of the hyper-parameters. The layers of the neural network and the learning rate are directly related to the improved outcomes obtained by the model. Therefore, both the centralized and the decentralized neural network layers must be compatible in order to initialize the weights. In the current case, we have hidden neural network layers of the same size with a similar number of nodes. If we change any one of the neural network layers during the tuning process, it would be not compatible for the initialization, and this approach would not be effective. The previous centralized algorithm had a runtime of 157 min, while the initialization approach took 63 min to run the centralized algorithm with the initialized weights. If we consider the total runtime, including the decentralized network training, the combined runtime is 125 min. The results show that the initialization of the hidden layers in the centralized neural network was helpful and did not degrade the performance; instead, the learning process stabilized faster even if the learning rate was increased.

Figure 23 shows the output of the centralized and decentralized neural network in a small-scale problem. This particular output shows that the centralized UAV algorithm performs better when the problem is not large. As the problem becomes larger, the data involved increase and this causes the algorithm to learn slower and take a longer time to achieve a better performance. Here, the output of the centralized algorithm shows a smooth learning process, whereas the decentralized algorithm shows a rough learning curve. One of the main reasons for this is because the learning rate employed in the decentralized network is updating faster. Compared to the five-UAV scenario, three UAVs provide a better outcome, with a higher learning rate. Due to this, even if there were some initial spikes in the learning, after a few episodes, the learning stabilizes. The final best outcome of

the centralized algorithm covers 54 users of a total user count of 60, and the outcome of the decentralized algorithm was 48 covered users. This shows that the centralized algorithm performs better than the decentralized algorithm for smaller-scale scenarios.

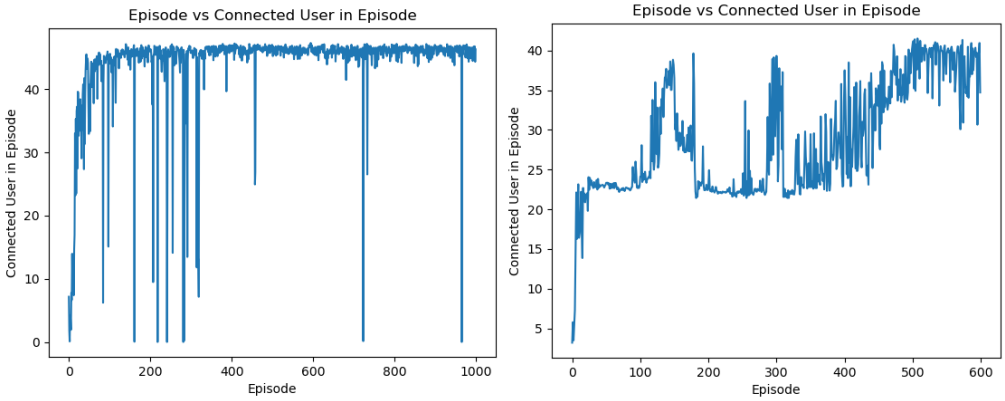

Figure 23. Simulation results of centralized (**left**) and decentralized (**right**) neural network at a small scale using three UAVs.

7.11. Comparison with Decentralized Approach

The centralized approach, while effective for smaller networks, faces increasing computational complexity as the number of UAVs and users grows due to the exponential expansion of the joint state and action spaces. In contrast, the decentralized approach distributes the computational load across individual UAVs, which allows it to scale more gracefully in larger networks. Although decentralized systems may encounter challenges with inter-agent coordination as the network size increases, our simulation results demonstrate that they maintain a robust performance and adaptability even with higher numbers of UAVs and users. Thus, our work provides a clear trade-off: centralized methods can offer optimal coordination in limited scenarios, whereas decentralized methods are more suitable for large-scale, dynamic environments.

8. Conclusions

To summarize, this paper advanced the cause of improving user connectivity in ad hoc communication networks by strategically deploying UAVs under the control of a multi-agent and centralized deep Q-learning framework. The robustness of the suggested approach is demonstrated through the creation of a simple algorithm for policy-training in stochastic settings and the thorough testing of that algorithm over a range of environmental distributions. Furthermore, the empirical analysis of policy adaptation under various distributions confirms the taught policy's adaptability and robustness. This paper also focused on instances in which jamming attacks result in a considerable reduction in total user coverage rate. Under such adverse conditions, we created an algorithm that adjusts the placements of UAVs that are not impacted by jamming while in service. The main goal of this algorithm is to optimize user coverage by reallocating UAVs to keep as many ground users connected as is feasible during instances of jamming. This method significantly mitigates the negative consequences of jamming assaults by guaranteeing that the network can adapt and respond to disturbances, hence increasing the overall system's resilience and reliability. This work establishes a baseline for the flexibility of UAV network rules in the dynamic conditions of real-world situations, laying the groundwork for further research in this field. The practical implementation of multi-agent reinforcement learning algorithms for UAV base stations presents a variety of additional challenges. UAV flight time, for instance, is limited by battery capacity, making efficient energy management and optimized trajectory planning essential in real-world deployments. Moreover, limited

memory and computational abilities present practical challenges in UAV base station deployment, which are interesting challenges for future work.

Author Contributions: Conceptualization, B.V.S.; funding acquisition, B.V.S.; investigation, M.R.K. and G.R.V.P.; project administration, B.V.S.; software, M.R.K. and G.R.V.P.; supervision, B.V.S.; validation, M.R.K. and G.R.V.P.; visualization, M.R.K. and G.R.V.P.; writing—original draft, M.R.K. and G.R.V.P.; writing—review and editing, B.V.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available in the following public GitHub repository: https://github.com/vanscoy/data-uav-rl/ (accessed on 6 March 2025).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

UAV	Unmanned Aerial Vehicle
RL	Reinforcement Learning
LoS	Line of Sight
HAP	High-Altitude Platform
BS	Base Station
DoF	Degree of Freedom
QoS	Quality of Service
MARL	Multi-Agent Reinforcement Learning
DRL	Deep Reinforcement Learning
IoT	Internet of Things
MEC	Multi-Access Edge Computing
BSUM	Block Successive Upperbound Minimization
NOMA	Non-Orthogonal Multiple Access
RMAB	Riemannian Multi-Armed Bandit
UE	User Equipment
HMM	Hidden Markov Model
ML	Machine Learning
AI	Artificial Intelligence
DQL	Deep Q-Learning
DNN	Deep Neural Network
MSE	Mean Squared Error
ReLU	Rectified Linear Unit

References

1. Zeng, Y.; Zhang, R.; Lim, T.J. Wireless communications with unmanned aerial vehicles: Opportunities and challenges. *IEEE Commun. Mag.* **2016**, *54*, 36–42. [CrossRef]
2. Vinogradov, E.; Sallouha, H.; De Bast, S.; Azari, M.; Pollin, S. Tutorial on UAV: A blue sky view on wireless communication. *arXiv* **2019**, arXiv:1901.02306.
3. Mozaffari, M.; Saad, W.; Bennis, M.; Nam, Y.H.; Debbah, M. A tutorial on UAVs for wireless networks: Applications, challenges, and open problems. *IEEE Commun. Surv. Tutorials* **2019**, *21*, 2334–2360. [CrossRef]
4. Bucaille, I.; Héthuin, S.; Munari, A.; Hermenier, R.; Rasheed, T.; Allsopp, S. Rapidly deployable network for tactical applications: Aerial base station with opportunistic links for unattended and temporary events absolute example. In Proceedings of the MILCOM 2013-2013 IEEE military communications conference, San Diego, CA, USA, 18–20 November 2013; pp. 1116–1120.
5. Bor-Yaliniz, I.; Yanikomeroglu, H. The new frontier in RAN heterogeneity: Multi-tier drone-cells. *IEEE Commun. Mag.* **2016**, *54*, 48–55. [CrossRef]
6. Lyu, J.; Zeng, Y.; Zhang, R. UAV-aided offloading for cellular hotspot. *IEEE Trans. Wirel. Commun.* **2018**, *17*, 3988–4001. [CrossRef]

7. Fuller, D. *AT&T Detail Network Testing of Drones in Football Stadiums*; Android Headlines: Riverside, CA, USA, 2016.
8. Namuduri, K. Flying cell towers to the rescue. *IEEE Spectr.* **2017**, *54*, 38–43. [CrossRef]
9. Zeng, Y.; Lyu, J.; Zhang, R. Cellular-connected UAV: Potential, challenges, and promising technologies. *IEEE Wirel. Commun.* **2018**, *26*, 120–127. [CrossRef]
10. Saad, W.; Bennis, M.; Mozaffari, M.; Lin, X. *Wireless Communications and Networking for Unmanned Aerial Vehicles*; Cambridge University Press: Cambridge, UK, 2020.
11. Khawaja, W.; Guvenc, I.; Matolak, D.W.; Fiebig, U.C.; Schneckenburger, N. A survey of air-to-ground propagation channel modeling for unmanned aerial vehicles. *IEEE Commun. Surv. Tutorials* **2019**, *21*, 2361–2391. [CrossRef]
12. Challita, U.; Saad, W.; Bettstetter, C. Interference management for cellular-connected UAVs: A deep reinforcement learning approach. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 2125–2140. [CrossRef]
13. Liu, C.H.; Dai, Z.; Zhao, Y.; Crowcroft, J.; Wu, D.; Leung, K.K. Distributed and energy-efficient mobile crowdsensing with charging stations by deep reinforcement learning. *IEEE Trans. Mob. Comput.* **2019**, *20*, 130–146. [CrossRef]
14. Hu, Y.; Chen, M.; Saad, W.; Poor, H.V.; Cui, S. Distributed multi-agent meta learning for trajectory design in wireless drone networks. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 3177–3192. [CrossRef]
15. Cui, J.; Ding, Z.; Deng, Y.; Nallanathan, A. Model-free based automated trajectory optimization for UAVs toward data transmission. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.
16. Geraci, G.; Garcia-Rodriguez, A.; Azari, M.M.; Lozano, A.; Mezzavilla, M.; Chatzinotas, S.; Chen, Y.; Rangan, S.; Di Renzo, M. What will the future of UAV cellular communications be? A flight from 5G to 6G. *IEEE Commun. Surv. Tutorials* **2022**, *24*, 1304–1335. [CrossRef]
17. Liu, Y.; Liu, K.; Han, J.; Zhu, L.; Xiao, Z.; Xia, X.G. Resource allocation and 3-D placement for UAV-enabled energy-efficient IoT communications. *IEEE Internet Things J.* **2020**, *8*, 1322–1333. [CrossRef]
18. Lyu, J.; Zeng, Y.; Zhang, R.; Lim, T.J. Placement optimization of UAV-mounted mobile base stations. *IEEE Commun. Lett.* **2016**, *21*, 604–607. [CrossRef]
19. Xiao, Z.; Zhu, L.; Liu, Y.; Yi, P.; Zhang, R.; Xia, X.G.; Schober, R. A survey on millimeter-wave beamforming enabled UAV communications and networking. *IEEE Commun. Surv. Tutorials* **2021**, *24*, 557–610. [CrossRef]
20. Moradi, M.; Sundaresan, K.; Chai, E.; Rangarajan, S.; Mao, Z.M. SkyCore: Moving core to the edge for untethered and reliable UAV-based LTE networks. In Proceedings of the 24th Annual International Conference on Mobile Computing and Networking, New Delhi, India, 29 October–2 November 2018; pp. 35–49.
21. Ei, N.N.; Alsenwi, M.; Tun, Y.K.; Han, Z.; Hong, C.S. Energy-efficient resource allocation in multi-UAV-assisted two-stage edge computing for beyond 5G networks. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 16421–16432. [CrossRef]
22. Nasir, A.A.; Tuan, H.D.; Duong, T.Q.; Poor, H.V. UAV-enabled communication using NOMA. *IEEE Trans. Commun.* **2019**, *67*, 5126–5138. [CrossRef]
23. Wu, Q.; Zeng, Y.; Zhang, R. Joint trajectory and communication design for multi-UAV enabled wireless networks. *IEEE Trans. Wirel. Commun.* **2018**, *17*, 2109–2121. [CrossRef]
24. Mozaffari, M.; Kasgari, A.T.Z.; Saad, W.; Bennis, M.; Debbah, M. Beyond 5G with UAVs: Foundations of a 3D wireless cellular network. *IEEE Trans. Wirel. Commun.* **2018**, *18*, 357–372. [CrossRef]
25. Li, S.; Xiang, C.; Xu, W.; Peng, J.; Xu, Z.; Li, J.; Liang, W.; Jia, X. Coverage Maximization of Heterogeneous UAV Networks. In Proceedings of the 2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS), Hong Kong, China, 18–21 July 2023; pp. 120–130.
26. Veeraswamy Premkumar, G.R.; Van Scoy, B. Optimal Positioning of Unmanned Aerial Vehical (UAV) Base Stations using Mixed-Integer Linear Programming. *Drones* **2025**, *9*, 44. [CrossRef]
27. Shakeri, R.; Al-Garadi, M.A.; Badawy, A.; Mohamed, A.; Khattab, T.; Al-Ali, A.K.; Harras, K.A.; Guizani, M. Design challenges of multi-UAV systems in cyber-physical applications: A comprehensive survey and future directions. *IEEE Commun. Surv. Tutorials* **2019**, *21*, 3340–3385. [CrossRef]
28. Nasim, I.; Ibrahim, A.S. Relay Placement for Maximum Flow Rate via Learning and Optimization Over Riemannian Manifolds. *IEEE Trans. Mach. Learn. Commun. Netw.* **2023**, *1*, 197–209. [CrossRef]
29. Fahim, A.; Gadallah, Y. An optimized LTE-based technique for drone base station dynamic 3D placement and resource allocation in delay-sensitive M2M networks. *IEEE Trans. Mob. Comput.* **2021**, *22*, 732–743. [CrossRef]
30. Orfanus, D.; De Freitas, E.P.; Eliassen, F. Self-organization as a supporting paradigm for military UAV relay networks. *IEEE Commun. Lett.* **2016**, *20*, 804–807. [CrossRef]
31. Klaine, P.V.; Nadas, J.P.; Souza, R.D.; Imran, M.A. Distributed drone base station positioning for emergency cellular networks using reinforcement learning. *Cogn. Comput.* **2018**, *10*, 790–804. [CrossRef]
32. Liu, X.; Chen, M.; Yin, C. Optimized trajectory design in UAV based cellular networks for 3D users: A double Q-learning approach. *J. Commun. Inf. Netw.* **2019**, *4*, 24–32. [CrossRef]

33. Yang, P.; Cao, X.; Xi, X.; Du, W.; Xiao, Z.; Wu, D. Three-dimensional continuous movement control of drone cells for energy-efficient communication coverage. *IEEE Trans. Veh. Technol.* **2019**, *68*, 6535–6546. [CrossRef]
34. Xu, W.; Sun, Y.; Zou, R.; Liang, W.; Xia, Q.; Shan, F.; Wang, T.; Jia, X.; Li, Z. Throughput maximization of UAV networks. *IEEE/ACM Trans. Netw.* **2021**, *30*, 881–895. [CrossRef]
35. Luong, P.; Gagnon, F.; Tran, L.N.; Labeau, F. Deep reinforcement learning-based resource allocation in cooperative UAV-assisted wireless networks. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 7610–7625. [CrossRef]
36. Cui, J.; Liu, Y.; Nallanathan, A. Multi-agent reinforcement learning-based resource allocation for UAV networks. *IEEE Trans. Wirel. Commun.* **2019**, *19*, 729–743. [CrossRef]
37. Tripathi, S.; Zhang, R.; Wang, M. Distributed User Connectivity Maximization in UAV-Based Communication Networks. In Proceedings of the GLOBECOM 2023—2023 IEEE Global Communications Conference, Kuala Lumpur, Malaysia, 4–8 December 2023; pp. 3753–3758.
38. Pelechrinis, K.; Iliofotou, M.; Krishnamurthy, S.V. Denial of service attacks in wireless networks: The case of jammers. *IEEE Commun. Surv. Tutorials* **2010**, *13*, 245–257. [CrossRef]
39. Zhang, C.; Zhang, L.; Mao, T.; Xiao, Z.; Han, Z.; Xia, X.G. Detection of Stealthy Jamming for UAV-Assisted Wireless Communications: An HMM-Based Method. *IEEE Trans. Cogn. Commun. Netw.* **2023**, *9*, 779–793. [CrossRef]
40. Kamnani, K.; Suratkar, C. A review paper on Google Loon technique. *Int. J. Res. Sci. Eng.* **2015**, *1*, 167–171.
41. Wu, Q.; Xu, J.; Zhang, R. Capacity characterization of UAV-enabled two-user broadcast channel. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 1955–1971. [CrossRef]
42. Wu, Q.; Zhang, R. Common throughput maximization in UAV-enabled OFDMA systems with delay consideration. *IEEE Trans. Commun.* **2018**, *66*, 6614–6627. [CrossRef]
43. Challita, U.; Saad, W.; Bettstetter, C. Cellular-connected UAVs over 5G: Deep reinforcement learning for interference management. *arXiv* **2018**, arXiv:1801.05500.
44. Chen, M.; Saad, W.; Yin, C. Liquid state machine learning for resource allocation in a network of cache-enabled LTE-U UAVs. In Proceedings of the GLOBECOM 2017-2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017; pp. 1–6.
45. Li, H. Multi-agent Q-learning of channel selection in multi-user cognitive radio systems: A two by two case. In Proceedings of the 2009 IEEE International Conference on Systems, Man and Cybernetics, San Antonio, TX, USA, 11–14 October 2009; pp. 1893–1898.
46. Galindo-Serrano, A.; Giupponi, L. Distributed Q-learning for aggregated interference control in cognitive radio networks. *IEEE Trans. Veh. Technol.* **2010**, *59*, 1823–1834. [CrossRef]
47. Russell, S.J.; Norvig, P. *Artificial Intelligence: A Modern Approach*; Pearson: London, UK, 2016.
48. Xiao, K.; Zhao, J.; He, Y.; Yu, S. Trajectory prediction of UAV in smart city using recurrent neural networks. In Proceedings of the ICC 2019—2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6.
49. Mitchell, T.M. Artificial neural networks. *Mach. Learn.* **1997**, *45*, 127.
50. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; The MIT Press: Cambridge, MA, USA, 2018.
51. Lai, M. Giraffe: Using Deep Reinforcement Learning to Play Chess. *arXiv* **2015**, arXiv:1509.01549.
52. Luong, N.C.; Hoang, D.T.; Gong, S.; Niyato, D.; Wang, P.; Liang, Y.C.; Kim, D.I. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Commun. Surv. Tutorials* **2019**, *21*, 3133–3174. [CrossRef]
53. Fitzek, F.H.; Granelli, F.; Seeling, P. *Computing in Communication Networks: From Theory to Practice*; Academic Press: London, UK, 2020.
54. Hu, J.; Zhang, H.; Song, L.; Han, Z.; Poor, H.V. Reinforcement learning for a cellular internet of UAVs: Protocol design, trajectory control, and resource management. *IEEE Wirel. Commun.* **2020**, *27*, 116–123. [CrossRef]
55. Zhou, S.; Cheng, Y.; Lei, X.; Peng, Q.; Wang, J.; Li, S. Resource allocation in UAV-assisted networks: A clustering-aided reinforcement learning approach. *IEEE Trans. Veh. Technol.* **2022**, *71*, 12088–12103. [CrossRef]
56. Busoniu, L.; Babuska, R.; De Schutter, B. A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. Syst. Man, Cybern. Part C Appl. Rev.* **2008**, *38*, 156–172. [CrossRef]
57. Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In Proceedings of the Tenth International Conference on Machine Learning, Amherst, MA, USA, 27–29 July 1993; pp. 330–337.
58. Clouse, J.A. Learning from an automated training agent. In *Adaptation and Learning in Multiagent Systems*; Springer: Berlin/Heidelberg, Germany, 1996; p. 195.
59. Price, B.; Boutilier, C. Accelerating reinforcement learning through implicit imitation. *J. Artif. Intell. Res.* **2003**, *19*, 569–629. [CrossRef]
60. Muscinelli, E.; Shinde, S.S.; Tarchi, D. Overview of distributed machine learning techniques for 6G networks. *Algorithms* **2022**, *15*, 210. [CrossRef]
61. Zhang, K.; Yang, Z.; Liu, H.; Zhang, T.; Basar, T. Fully decentralized multi-agent reinforcement learning with networked agents. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 5872–5881.

62. Khan, M.R. Distributed UAV-Based Wireless Communications Using Multi-Agent Deep Reinforcement Learning. Master's Thesis, Miami University, Oxford, OH, USA, 2024.
63. Almasoud, A. Jamming-aware optimization for UAV trajectory design and internet of things devices clustering. *Complex Intell. Syst.* **2023**, *9*, 4571–4590. [CrossRef]
64. Pirayesh, H.; Zeng, H. Jamming attacks and anti-jamming strategies in wireless networks: A comprehensive survey. *IEEE Commun. Surv. Tutorials* **2022**, *24*, 767–809. [CrossRef]
65. Lu, X.; Xiao, L.; Dai, C.; Dai, H. UAV-aided cellular communications with deep reinforcement learning against jamming. *IEEE Wirel. Commun.* **2020**, *27*, 48–53. [CrossRef]
66. Peng, J.; Zhang, Z.; Wu, Q.; Zhang, B. Anti-jamming communications in UAV swarms: A reinforcement learning approach. *IEEE Access* **2019**, *7*, 180532–180543. [CrossRef]
67. Xu, Y.; Ren, G.; Chen, J.; Zhang, X.; Jia, L.; Feng, Z.; Xu, Y. Joint power and trajectory optimization in UAV anti-jamming communication networks. In Proceedings of the ICC 2019—2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–5.
68. Lv, S.; Xiao, L.; Hu, Q.; Wang, X.; Hu, C.; Sun, L. Anti-jamming power control game in unmanned aerial vehicle networks. In Proceedings of the GLOBECOM 2017-2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017; pp. 1–6.
69. Premkumar, G.R.V. Centralized Deep Reinforcement Learning and Optimization in UAV Communication Networks Towards Enhanced User Coverage. Master's Thesis, Miami University, Oxford, OH, USA, 2024.