Generative AI-augmented Graph Reinforcement Learning for Adaptive UAV Swarm Optimization

Bishmita Hazarika, *Member, IEEE*, Piyush Singh, *Graduate Student Member, IEEE*, Keshav Singh, *Member, IEEE*, Simon L. Cotton, *Senior Member, IEEE*, Hyundong Shin, *Fellow, IEEE,* Octavia A. Dobre, *Fellow, IEEE*, and Trung Q. Duong, *Fellow, IEEE*

***Abstract*—Unmanned aerial vehicles (UAVs) are essential for providing communication and computation services in disaster recovery scenarios where traditional infrastructure is compromised. However, challenges related to energy efficiency, real-time adaptability, coverage, load balancing, and safe navigation persist, particularly in dynamic disaster environments. In this study, we propose a comprehensive framework that integrates Generative AI (GenAI) with graph neural networks (GNN) to dynamically generate hover points for waypoint-based UAV navigation and realistic task generation based on environmental conditions. The GNN-based collision avoidance mechanism further ensures safe navigation by allowing UAVs to avoid obstacles and no-fly zones while coordinating with neighboring UAVs in real-time. To optimize UAV swarm operations, we introduce a multi-agent graph reinforcement learning (MAGRL) framework, enabling UAVs to maximize overall system utility by refining hover point selection, task allocation, and load balancing in response to environmental changes. A graph attention mechanism enhances UAV coordination, improving communication efficiency and decision-making. Extensive simulations show that the proposed GenAI-GNN and MAGRL framework significantly outperforms existing methods in task completion, energy efficiency, and overall system utility in disaster recovery scenarios.**

This work was supported by the Canada Excellence Research Chair (CERC) Program CERC-2022-00109. The work of K. Singh was supported by the National Science and Technology Council of Taiwan under Grant NSTC 112-2221-E-110-038-MY3 and NSTC 113-2218-E-110-009. The work of S. L. Cotton was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) through the EPSRC Hub on All Spectrum Connectivity (EP/X040569/1 and EP/Y037197/1). The work of O. A. Dobre was supported by Canada Research Chairs Program CRC-2022-00187.

B. Hazarika and O. A. Dobre are with the Faculty of Engineering and Applied Science, Memorial University, St. John's, Canada. (email: {bhazarika, odobre}@mun.ca).

P. Singh is with the Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung 804, Taiwan. (Email: piyush230497@gmail.com).

K. Singh is with the Institute of Communications Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan, and also with the Department of Electronic Engineering, Kyung Hee University, Yongin-si, Gyeonggi-do 17104, South Korea (Email: keshav.singh@mail.nsysu.edu.tw).

S. L. Cotton is with the Centre for Wireless Innovation (CWI), Queen's University of Belfast, Queen's Road, Belfast, BT3 9DT, UK (e-mail: simon.cotton@qub.ac.uk).

H. Shin is with the Department of Electronics and Information Convergence Engineering, Kyung Hee University, 1732 Deogyeong-daero, Giheung-gu, Yongin-si, Gyeonggi-do 17104, Republic of Korea (e-mail: hshin@khu.ac.kr).

T. Q. Duong is with the Faculty of Engineering and Applied Science, Memorial University, St. John's, NL A1C 5S7, Canada, and is also with the School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast, U.K. (E-mail: tduong@mun.ca).

***Index Terms*—Unmanned Aerial Vehicles (UAV), Disaster Recovery, Generative AI (GenAI), Multi-Agent Reinforcement Learning (MARL), Graph Neural Networks (GNN)**

I. Introduction

NATURAL disasters like earthquakes, floods, and wildfires cause severe disruptions to communication infrastructure, making real-time coordination and response difficult. In such critical scenarios, swift deployment of communication and computation systems is crucial for emergency response, hazard monitoring, and public safety. Integrating UAVs with IoT networks has emerged as a promising solution to provide wide-area coverage, mobile edge computing (MEC), and sensing capabilities in areas lacking ground infrastructure. Despite their mobility and scalability, challenges in energy management, real-time adaptability, and task offloading limit their effectiveness in disaster recovery. Studies like [1] and [2] focus on optimizing UAV trajectories, while others such as [3] and [4] propose energy-efficient deployment strategies. Yet, these approaches often fail to account for the unpredictable and rapidly evolving nature of disaster zones, where real-time adaptability is crucial. Task offloading solutions from works like [5] and [6] typically assume static conditions, and [7] focuses on energy-aware strategies without fully considering the complexities of real-time UAV coordination. Moreover, traditional trajectory planning in disaster scenarios, as explored in [8] and [9], is often risky due to obstacles, debris, and no-fly zones (NFZs), making continuous real-time adjustments computationally expensive and energy-intensive [10]. This underscores the need for an adaptive framework that integrates energy efficiency, task prioritization, and real-time UAV coordination to effectively address the demands of disaster recovery operations.

To tackle the complexity and unpredictability of disaster scenarios, deep reinforcement learning (DRL) is increasingly used for UAV deployment, trajectory optimization, and task management, as in [11] and [12]. However, DRL's effectiveness is limited by its need for large datasets, long convergence times [13], and challenges in balancing exploration and exploitation [14], which are critical in dynamic disaster environments [15]. Multi-agent DRL (MADRL) enhances coordination among UAVs [16], [17], but struggles with slow convergence and computational burdens as agent numbers increase. Approaches like LSTM

networks [18], mean-field reinforcement learning [19], and centralized training with decentralized execution [20] aim to boost scalability and adaptability but still confront high computational demands and limited flexibility.

To overcome these limitations, graph reinforcement learning (GRL) has emerged as a more structured approach for handling multi-agent coordination in complex environments [21], [22]. Studies like [23], and [24] show their potential to enhance multi-agent collaboration by enabling UAVs to share state information and adjust tasks in real time. GRL frameworks excel in optimizing UAV operations by allowing dynamic reconfiguration based on real-time information, making them well-suited for the evolving demands of disaster recovery. Unlike DRL, GRL leverages the communication network's structure, providing better coordination in scenarios where task demands and UAV mobility fluctuate [25]. However, the iterative nature of policy optimization in GRL means that convergence times can be slow, especially when the graph structure changes frequently due to task demands or UAV movement. This slow convergence poses a significant challenge in disaster scenarios, where real-time adaptability is essential.

Another approach gaining traction in UAV swarm coordination is federated learning (FL), which allows decentralized learning across multiple agents without the need for centralized data collection [26], [27]. FL has been employed in various UAV applications to optimize tasks such as resource allocation and swarm coordination, as seen in works by [28]–[30]. While promising, FL faces significant challenges in disaster recovery. Its reliance on periodic aggregation of local models becomes impractical due to the intermittent connectivity and rapidly shifting conditions of UAV swarms in such scenarios [31]. This slows model synchronization, making it unsuitable for real-time, time-sensitive decision-making. Moreover, FL's dependence on reliable and consistent data poses difficulties in chaotic disaster environments, where data availability is scarce. Although some studies, such as [32] and [33], rely on simulations to generate training data for FL models, simulated environments often fail to capture the complexity and anomalies present in real-world disaster situations. As a result, the performance of FL models trained on simulated data often doesn't generalize well to real disaster conditions, underscoring the need for real, diverse datasets that capture the unpredictability of disasters.

Generative AI (GenAI) can address some of these limitations by creating diverse and complex simulated data that more closely mimics real-world disaster conditions. Recent studies, such as those by [34] and [35], have demonstrated the potential of GenAI in enhancing wireless network operations by generating realistic data that improves model training and decision-making. GenAI can dynamically generate simulated data that captures the unique characteristics of different disaster zones, filling the gap left by traditional simulation approaches that fail to account for the chaotic and evolving nature of disaster environments [36], [37]. However, although GenAI shows promise for data generation in disaster recovery, it is still underutilized, especially in UAV swarm operations. It primarily generates data without tackling the required coordination and adaptability in dynamic disaster zones. This highlights the need for solutions that merge data generation with strategies for efficient, adaptive UAV coordination in disaster scenarios.

The key gap in existing research lies in addressing the complexities of disaster scenarios, including dynamic task demands, real-time adaptability, and unpredictable environments. Current methods like DRL and MADRL struggle with slow convergence and high computational demands. GRL cannot adapt in real-time, and FL relies on stable connectivity and reliable data, which are often absent in disasters. Traditional trajectory planning raises risks and energy use due to obstacles and NFZs. While GenAI offers a promising solution by generating safe hover points as an alternative to free UAV trajectory planning in high-risk areas, it alone cannot address the challenges of real-time coordination and adaptability. This leaves a gap in developing fully integrated solutions for efficient UAV swarm deployment in disaster management scenarios.

Thus, in this study, we bridge these gaps by integrating GenAI with a graph-based model for dynamic hover point and realistic task generation and combining it with a multi-agent graph reinforcement learning framework, enhanced by a graph attention mechanism to improve UAV communication efficiency, optimize energy, and ensure real-time adaptability and efficient task offloading in disaster scenarios. The key contributions of this study are outlined below:

- We establish a comprehensive framework for UAV swarm deployment in disaster recovery scenarios to assist ground terminals, formulating an optimization problem that maximizes an overall utility function. This utility function integrates key factors such as task offloading, energy efficiency, coverage, and load balancing, while also accounting for critical constraints like task deadlines and the dynamic, unpredictable nature of disaster zones.
- We introduce a GenAI model integrated with graph neural networks (GNN) to dynamically generate hover points based on the physical environment, offering a robust alternative to traditional UAV trajectory planning. This approach optimizes energy consumption, avoids NFZs, and incorporates a GNN-based collision avoidance strategy, ensuring safer and more efficient navigation. Additionally, GenAI is utilized for realistic task generation, enabling the system to handle diverse and complex tasks reflective of real-world disaster scenarios, further enhancing operational effectiveness.
- We develop a multi-agent graph reinforcement learning (MAGRL) framework, tightly integrated with the GenAI-GNN model, to optimize UAV swarm operations in disaster scenarios. This framework adapts dynamically by refining hover point selection, task allocation, and load balancing in response to real-time environmental changes, ensuring resilient and efficient UAV coordination under unpredictable conditions.
- We incorporate a graph attention mechanism within the MAGRL framework to further enhance coordination

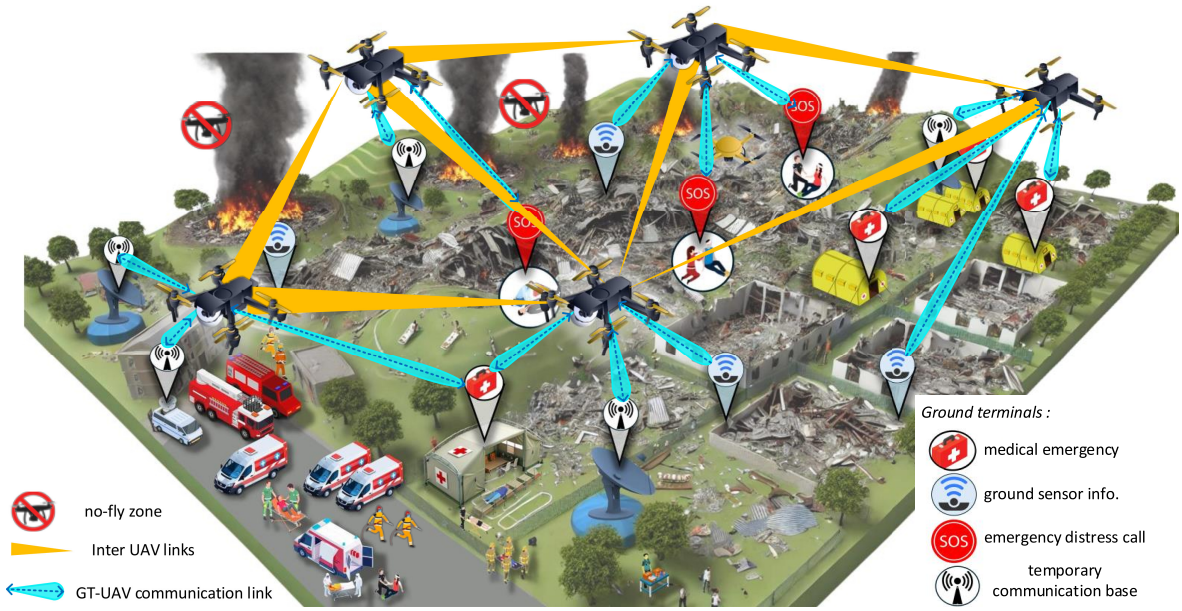


Fig. 1: Illustration of a UAV swarm providing support to various ground terminals in a disaster-stricken area.

between UAVs. This mechanism enables UAVs to adjust movements, task scheduling, and load balancing intelligently, optimizing communication and decision-making while maximizing overall system utility in rapidly evolving disaster-affected regions.

- Finally, we conduct extensive simulations to evaluate the effectiveness of the proposed GenAI-GNN and MAGRL framework with a graph attention mechanism, demonstrating significant improvements in task completion rates, energy efficiency, total utility, and overall UAV swarm performance.

Structure of the paper: The flow of the paper is organized as follows: Section II discusses the network and task model, while Section III describes the UAV operation model. Section IV covers the utility model, and Section V defines the problem statement. In Section VI, we present the solution approach, followed by Section VII, which introduces the generative AI-graph model for UAV deployment. Section VIII details the multi-agent graph reinforcement learning for UAV swarm operations. Section IX provides the performance evaluation, and Section X shares the result analysis. Finally, Section XI presents the conclusion of the paper.

II. Network and Task Model

We consider a disaster-affected urban area where traditional communication infrastructure has been severely damaged or rendered non-functional. The affected area is modeled as a two-dimensional space of $D_x \times D_y$ square meters, populated with N_{GT} GTs dispersed across the region. These GTs represent critical devices such as emergency response units, environmental sensors, medical equipment, and public communication nodes, all of which require reliable communication and computation offloading services. To support these GTs, a swarm of N_{UAV} UAVs is deployed. Each UAV is equipped with communication modules and MEC capabilities, allowing them to function as both aerial base stations and mobile computing nodes. The UAVs operate at a fixed altitude h_u, facilitating optimal communication coverage and task handling. Tasks at the GTs are generated dynamically based on real-time disaster needs and are often resource-intensive. These tasks may involve video processing for situational awareness, sensor data analysis for hazard monitoring, medical data processing, or communication support for emergency response teams. Due to the limited processing capacity of GTs, many of these tasks are offloaded to the UAV swarm for timely completion. Fig. 1 illustrates a disaster scenario where UAVs establish inter-UAV communication links and connect with various ground terminals, including medical emergency stations, ground sensors, emergency distress call responders, and temporary communication bases. The scene highlights the UAVs' role in coordinating rescue efforts, managing task offloading, and ensuring seamless communication, all while navigating through NFZ marked due to fire hazards.

Each UAV possesses limited resources utilized during communication, computation, movement, and coordination within the swarm. The resources of UAV i include its central processing unit (CPU) processing power, denoted as $f_i(t)$, which represents the available CPU cycles per second used for handling computation tasks offloaded from GTs. Additionally, each UAV has a finite energy budget $E_{\max}(t)$, which diminishes over time due to energy consumption from communication, computation, movement, and hovering. Moreover, UAVs have a task queue with limited buffer capacity, allowing them to handle only a finite number of offloaded tasks from GTs

at any time. Each UAV's position at time t is given by $\mathbf{p}_u(t) = (x_u(t), y_u(t), h_u)$, while the position of GT k is fixed at $\mathbf{p}_k = (x_k, y_k)$. The UAVs form a dynamic, adaptable swarm that adjusts its configuration in real time to respond to the evolving needs of the disaster scenario.

A. UAV Mobility Model

The UAVs are capable of moving to different positions within the disaster area to optimize coverage and resource allocation. The deployment strategy aims to ensure that the UAVs collectively maximize wireless coverage while minimizing energy consumption and communication latency. Each UAV must dynamically adjust its position to balance the load among neighboring UAVs, cover as many ground terminals as possible, and avoid NFZs. The movement of UAV i from position $\mathbf{p}_i(t)$ to $\mathbf{p}_i(t+\Delta t)$ is governed as the following:

$$\mathbf{p}_i(t + \Delta t) = \mathbf{p}_i(t) + \mathbf{v}_i(t) \cdot \Delta t \,, \tag{1}$$

where $\mathbf{v}_i(t)$ is the velocity vector of UAV i at time t. The velocity is constrained by a maximum speed v_{max}:

$$\|\mathbf{v}_i(t)\| \leq v_{max} \,. \tag{2}$$

Additionally, UAVs must respect NFZs present in the disaster area. When moving between positions, a UAV will avoid entering these NFZs by adjusting its trajectory. The UAVs can only hover at predefined hover points, and within each time slot, a UAV will hover at a fixed point, shifting to another point only if necessary.

B. Task Model

In the disaster recovery scenario, GTs generate computational tasks that are either processed locally if sufficient resources are available, or fully offloaded to UAVs when the GT is overloaded or when it is more efficient to utilize UAV resources. Let T_k represent the task generated by GT k at time t. The task T_k requires the processing of D_k bits of data, and each bit requires μ_k CPU cycles to process. Each task has a deadline τ_k by which it must be completed.

1) Local Processing at GT: GTs can process their tasks locally using their computational resources [7]. The time required to process task T_k locally at GT k is given by:

$$T_{lo,k}(t) = \frac{D_k \cdot \mu_k}{f_k(t)} \,, \tag{3}$$

where $f_k(t)$ represents the available computational resources (in CPU cycles per second) at GT k.

2) Offloaded Processing at UAV: Alternatively, GTs can offload their tasks to UAVs for processing. If task T_k is offloaded to UAV i, the total time for offloading and processing consists of the time to transmit the task data to the UAV and the time to process the task at the UAV [38]. The total time for UAV i to complete the task is given by:

$$T_{cmp,k,i}(t) = T_{tr,i}(t) + \frac{D_k \cdot \mu_k}{f_i(t)} \,, \tag{4}$$

where $T_{tr,i}(t)$ is the task transmission time, computed as:

$$T_{tr,i}(t) = \frac{D_k}{r_{i,k}(t)} \,. \tag{5}$$

Here, $r_{i,k}(t)$ is the data transmission rate between UAV i and GT k, which is computed as described in (14) in Section III-B.

3) Computation Load: Each UAV i has a total computational load $L_i(t)$ at time t, which is the sum of the processing demands of all the tasks offloaded to it from the GTs within its coverage area. The total load $L_i(t)$ on UAV i at time t is expressed as:

$$L_i(t) = \sum_{k \in \mathcal{K}_i(t)} D_k \cdot \mu_k \,, \tag{6}$$

where $\mathcal{K}_i(t)$ is the set of GTs offloading tasks to UAV i.

4) Task Completion: For both local and offloaded processing, the task must be completed before its deadline:

$$T_{lo,k}(t) \leq \tau_k \quad \text{or} \quad T_{cmp,k,i}(t) \leq \tau_k \tag{7}$$

.

III. UAV Operation Model

This section covers the operational aspects of UAV swarms in disaster recovery, highlighting sensing, communication, and energy management. It includes UAV detection and interaction with ground targets, data transmission via air-to-ground (ATG) links, and energy expenditure for movement, communication, computation, and coordination.

A. Sensing Model

Each UAV i has a probabilistic sensing range R_s, influenced by environmental conditions, obstacles, and terrain. The coverage of UAV i at time t for a specific GT k is defined as the probability that UAV i can successfully provide communication and sensing services to GT k. This probability is affected by both the line-of-sight (LOS) conditions and the presence of obstacles in the environment (i.e., non-line-of-sight (NLOS) conditions). The basic probability that UAV i successfully detects GT k at time t, assuming no obstacles, is:

$$P_d(k, i, t) = P_{\text{LOS}}(k, i) \cdot \exp\left(-\frac{\|\mathbf{p}_k - \mathbf{p}_i(t)\|}{R_s}\right) \,, \tag{8}$$

where $P_{\text{LOS}}(k, i)$ represents the probability that there is LOS between UAV i and GT k. This LOS probability is influenced by environmental factors and the elevation angle $\theta_{k,i}$ between UAV i and GT k, which is given by:

$$P_{\text{LOS}}(k, i) = \frac{1}{1 + a \exp\left(-b(\theta_{k,i} - a)\right)} \,, \tag{9}$$

Here, $\theta_{k,i}$ represents the elevation angle between GT k and UAV i [39], calculated as:

$$\theta_{k,i} = \frac{180}{\pi} \arctan\left(\frac{h_u - h_k}{d_{k,i}}\right) \,, \tag{10}$$

where h_k is the altitude of GT k (considered as ground level, so $h_k = 0$ in this study), and $d_{k,i} = \|\mathbf{p}_k - \mathbf{p}_i(t)\|$ is the horizontal

distance between UAV i and GT k. The parameters a and b are environment-dependent values that vary based on urban, suburban, or rural settings, as described in previous empirical research [40]. To account for obstacles in the environment, we introduce an obstacle-aware detection probability $P_d^{\text{obs}}(k,i,t)$, which modifies the basic detection probability by considering the effects of obstacles between UAV i and GT k:

$$P_d^{\text{obs}}(k,i,t) = P_d(k,i,t) \cdot (1 - O_{k,i}(t)), \tag{11}$$

where $O_{k,i}(t)$ is a factor between 0 and 1 that represents the severity of the obstacle's impact on communication. Higher values of $O_{k,i}(t)$ indicate greater obstruction, such as buildings or trees, that reduces the effective coverage [41].

B. Communication Model

Communication between UAVs and GTs follows an ATG channel model, accounting for both LoS and NLoS conditions. The channel gain between UAV i and GT k is expressed as:

$$G_{i,k}(t) = \frac{P_{\text{LOS}}(k,i)}{L_{\text{LOS}}(k,i)} + \frac{(1 - P_{\text{LOS}}(k,i))}{L_{\text{NLOS}}(k,i)}, \tag{12}$$

where $L_{\text{LOS}}(k,i)$ and $L_{\text{NLOS}}(k,i)$ represent the path losses under LoS and NLoS conditions, respectively. $P_{\text{LOS}}(k,i)$ denotes the probability that a LoS connection exists between UAV i and GT k, which depends on environmental factors and the elevation angle [41]. The signal-to-interference-plus-noise ratio (SINR) for GT k, served by UAV i, is computed as:

$$\gamma_{i,k}(t) = \frac{G_{i,k}(t) \cdot P_t}{I_{i,k}(t) + \sigma^2}, \tag{13}$$

where $G_{i,k}(t)$ is the channel gain between UAV i and GT k, P_t is the transmission power of UAV i, $I_{i,k}(t)$ is the interference from other UAVs serving neighboring GTs, and σ^2 is the noise power. The data transmission rate $r_{i,k}(t)$ depends on the available bandwidth B and the SINR, expressed as:

$$r_{i,k}(t) = B \log_2 (1 + \gamma_{i,k}(t)), \tag{14}$$

where B is the bandwidth allocated to the communication between UAV i and GT k. The time taken for UAV i to transmit a given amount of data $D_{i,k}$ to GT k is computed as:

$$T_{tr,i}(t) = \frac{D_{i,k}}{r_{i,k}(t)}, \tag{15}$$

where $D_{i,k}$ represents the data size to be transmitted from UAV i to GT k.

C. Energy Consumption Model

The total energy consumed by UAV i at time t includes energy for movement, communication, computation, hovering, and swarm coordination.

1) Movement and Hovering Energy: The energy consumed by UAV i at time t for movement between hover points and hovering is expressed as:

$$E_{\text{mh},i}(t) = \mathcal{P}_{\text{hov}} \cdot T_{\text{hov},i}(t) + \mathcal{P}_{\text{mov}} \cdot \|\mathbf{p}_i(t + \Delta t) - \mathbf{p}_i(t)\|. \tag{16}$$

Here, $\mathcal{P}_{\text{hov}}$ represents the power consumed by UAV i while hovering, and $T_{\text{hov},i}(t)$ is the time duration for which the UAV hovers. The term $\mathcal{P}_{\text{mov}} \cdot \|\mathbf{p}_i(t+\Delta t) - \mathbf{p}_i(t)\|$ accounts for the energy required to move between hover points, where $\mathcal{P}_{\text{mov}}$ is the power consumed per unit distance, and $\|\mathbf{p}_i(t + \Delta t) - \mathbf{p}_i(t)\|$ is the distance between the UAV's position at times t and $t + \Delta t$.

On the other hand, in the context of the generative adversarial network (GAN) model for generating optimal hover points, the energy consumption for UAV i when shifting from its current position to a newly generated hover point $\mathbf{p}_i^{\text{hover}}$ is:

$$E_{\text{move},i} = \mathcal{P}_{\text{mov}} \cdot \|\mathbf{p}_i^{\text{hover}} - \mathbf{p}_i(t)\|, \tag{17}$$

where $\|\mathbf{p}_i^{\text{hover}} - \mathbf{p}_i(t)\|$ represents the distance between the UAV's current position $\mathbf{p}_i(t)$ and the newly generated hover point $\mathbf{p}_i^{\text{hover}}$. While both equations deal with movement energy, (16) captures a general model of energy consumption during movement and hovering, whereas (17) focuses specifically on the energy required for shifting the UAV to a new hover point as generated by the GAN model.

2) Computation Energy: The energy consumed by UAV i for processing the offloaded task is modeled as:

$$E_{\text{comp},i}(t) = \kappa \left(f_i(t)\right)^2 \cdot T_{\text{process},i}(t), \tag{18}$$

where κ is a scaling factor depending on the efficiency of the UAV's processor, and $T_{\text{process},i}(t)$ is the task processing time.

3) Swarm Communication and Coordination Energy: The energy consumed by UAV i for intra-swarm communication and coordination is modeled as:

$$E_{\text{swm},i}(t) = \mathcal{P}_{\text{swm}} \cdot T_{c,i}(t) \cdot N_{nb,i}(t), \tag{19}$$

where $\mathcal{P}_{\text{swm}}$ is the power required for communication, $T_{c,i}(t)$ is the time spent coordinating with neighboring UAVs, and $N_{nb,i}(t)$ is the number of UAVs within communication range.

4) Communication Energy: The energy consumed for communication, $E_{\text{com},i}(t)$, is dynamically computed based on the transmission power and time. It is modeled as:

$$E_{\text{com},i}(t) = P_t \cdot T_{\text{tr},i}(t), \tag{20}$$

where $T_{\text{tr},i}(t)$ is the transmission time during which UAV i communicates with the GTs at time t, computed as:

$$T_{\text{tr},i}(t) = \frac{D_{i,k}}{B \log_2 (1 + \gamma_{i,k}(t))}. \tag{21}$$

IV. Utility Model

The goal of our system is to optimize the deployment and performance of UAVs in a disaster management framework by maximizing coverage, minimizing energy consumption, and ensuring effective load balancing through computation

offloading. The system's utility is defined through three key components: coverage utility, energy utility, and load balancing/computation offloading utility.

A. Coverage and Energy Utility

The coverage utility aims to maximize the coverage of GTs by UAVs, while the energy utility seeks to minimize the total energy consumed by UAVs during their operation. The coverage utility for UAV i at time t is defined based on the detection probability of GTs within its coverage area. The total coverage provided by UAV i is influenced by environmental factors, such as obstacles, and the presence of other UAVs, which may lead to redundant coverage. The utility for UAV i is expressed as:

$$U_{cov,i}(t) = \sum_{k \in \mathcal{K}_i(t)} \left(\left(1 - \frac{1}{N_{UAV,k}(t)} \right) \cdot P_d^{\text{obs}}(k,i,t) \right), \tag{22}$$

where $N_{UAV,k}(t)$ is the number of UAVs providing coverage to GT k at time t. The term $\left(1 - \frac{1}{N_{UAV,k}(t)}\right)$ serves as a discount factor to avoid over-counting coverage when multiple UAVs cover the same GT.

Next, we define the energy utility for UAV i at time t, which seeks to minimize the energy consumed for movement, communication, hovering, and computation. The total energy consumed by UAV i at time t is:

$$E_i(t) = E_{mh,i}(t) + E_{com,i}(t) + E_{comp,i}(t) + E_{swm,i}(t)\,. \tag{23}$$

Thus, the energy utility, aiming to minimize total energy consumption, is expressed as:

$$U_{eng,i}(t) = -E_i(t)\,. \tag{24}$$

B. Load Balancing and Computation Offloading Utility

The load balancing/computation offloading utility ensures that computational tasks generated by GTs are efficiently processed, either locally at the GT or offloaded to UAVs. The goal is to minimize the time required to complete these tasks while balancing the computational load across UAVs and meeting task deadlines. Each GT generates tasks that must be processed locally or offloaded to a UAV. The load balancing utility for UAV i is defined as:

$$U_{of,i}(t) = \sum_{k \in \mathcal{K}_i(t)} \left(\frac{1}{L_i(t)} \cdot \mathbb{1}\left(\min\left(T_{lo,k}(t), T_{cmp,k,i}(t)\right) \leq \tau_k\right) \right) \tag{25}$$

where τ_k is the task deadline for GT k.

C. Total Utility

The total utility for each UAV i at time t combines the coverage utility, energy utility, and load balancing/computation offloading utility, and is expressed as:

$$U_{total,i}(t) = U_{cov,i}(t) + U_{eng,i}(t) + U_{of,i}(t)\,. \tag{26}$$

Thus, this study aims to maximize $U_{total,i}(t)$ for each UAV in the swarm, ensuring efficient operation in the disaster area.

V. Problem Statement

In this paper, we focus on optimizing the deployment and operation of a UAV swarm in a disaster recovery environment. The objectives are to maximize GT coverage, minimize UAV energy consumption, and ensure efficient load balancing for computation tasks offloaded from the GTs. The UAV swarm dynamically adjusts its position to maximize coverage, accounting for environmental obstacles, interference, and redundant coverage. Simultaneously, energy consumption from communication, movement, hovering, and computation must be minimized to prolong UAV operation. Additionally, the UAVs must efficiently distribute computational loads, ensuring that tasks are either processed locally or offloaded to UAVs for timely completion before their respective deadlines.

Mathematically, we aim to maximize the total utility function $U_{total,i}(t)$ for each UAV i at time t, which integrates the coverage utility $U_{cov,i}(t)$, energy utility $U_{eng,i}(t)$, and load balancing/computation offloading utility $U_{of,i}(t)$. This optimization is subject to constraints on UAV mobility, energy, communication, and task deadlines. This formulation can be expressed as:

$$\begin{aligned}
(\mathcal{P}): \quad & \max_{\{\mathbf{p}_i(t), f_i(t)\}} \sum_{i=1}^{N_{UAV}} U_{total,i}(t) \\
\text{s.t.} \quad & (C.1) \quad \|\mathbf{p}_i(t) - \mathbf{p}_i(t+\Delta t)\| \leq v_{max} \cdot \Delta t \quad \forall i, \\
& (C.2) \quad \mathbf{p}_i(t) \notin \text{NFZ}, \quad \forall t, \forall i, \\
& (C.3) \quad E_i(t) \leq E_{\max,i} \quad \forall i, \\
& (C.4) \quad T_{lo,k}(t) \leq \tau_k \,||\, T_{cmp,k,i}(t) \leq \tau_k, \forall k, i, \\
& (C.5) \quad f_i(t) \leq f_{\max,i} \quad \forall i, \\
& (C.6) \quad \sum_{i=1}^{N} \rho_i \cdot f'_m \leq f_m \quad \forall m, \\
& (C.7) \quad T_{cmp,k,i}(t) + T_{tr,i}(t) \leq \tau_k \quad \forall k, i.
\end{aligned} \tag{27}$$

where $(C.1)$ ensures that UAVs must adhere to mobility limits, specifically ensuring they do not exceed the maximum allowable speed v_{max} when moving between hover points. $(C.2)$ mandates that UAVs avoid NFZs during movement. $(C.3)$ governs the total energy consumption of each UAV i—including movement, communication, computation, and swarm coordination—ensuring it does not exceed the energy budget $E_{\max,i}$. $(C.4)$ ensures that computational tasks meet their deadlines, whether processed locally at the GTs or offloaded to the UAVs. $(C.5)$ limits each UAV's CPU resources, $f_i(t)$, ensuring they do not exceed the maximum processing capability $f_{\max,i}$. In $(C.6)$, the computational load $L_i(t)$ on UAV i must not exceed its available processing capacity, where ρ_i is a binary variable: $\rho_i = 1$ indicates the task is offloaded to the UAV, and $\rho_i = 0$ indicates local processing at the GT. Finally, $(C.7)$ ensures that, if a task is offloaded to a UAV, the combined data transmission and computation time is less than or equal to the task deadline τ_k.

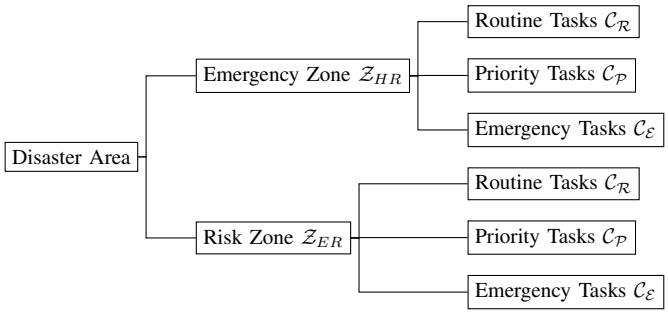
VI. Solution Approach

The proposed solution framework addresses key challenges in UAV deployment for disaster recovery and consists of the components described below:

A. UAV Resource Scheduling

In disaster-affected areas, some zones are more critical than others, with high-risk zones requiring immediate attention due to severe damage or emergencies. To optimize resource allocation, the disaster area is divided into sub-zone risk categories based on real-time monitoring. This approach allows the UAV swarm to prioritize tasks in more urgent areas. In this paper, we assume the risk levels of areas are constant and do not change throughout the simulation.

The UAVs, acting as mobile computing nodes, must efficiently allocate their CPU resources to maximize system performance while adhering to energy constraints and task deadlines. To optimize UAV resource allocation, the disaster area is divided into two risk categories: *Emergency Response* ($\mathcal{Z}_{ER}$), which includes life-threatening areas requiring immediate attention, and *High Risk* ($\mathcal{Z}_{HR}$), which covers critical areas needing prompt but less urgent action. Tasks generated within these zones are further classified into three types: *Emergency Tasks* ($\mathcal{C}_{\mathcal{E}}$), which involve immediate life-saving operations; *Priority Tasks* ($\mathcal{C}_{\mathcal{P}}$), important tasks such as situational awareness or environmental monitoring, and *Routine Tasks* ($\mathcal{C}_{\mathcal{R}}$), which consist of lower-priority operations like system maintenance. This layered approach ensures UAV resources are allocated efficiently based on task urgency and zone severity. Below is a tree diagram illustrating the hierarchical structure of the disaster zone and task prioritization.



To optimize CPU resource allocation, we design a scheduling algorithm to ensure UAVs efficiently process tasks within their coverage area. Let $T_i = \{T_1, \ldots, T_k\}$ represent the tasks under UAV i's coverage. The UAV prioritizes tasks based on sub-zone risk, urgency, and available resources such as energy, CPU capacity, and task deadlines. For each task T_k (size D_k, deadline τ_k), the UAV processes it if resources allow; otherwise, it looks for a smaller task T_j (size D_j, deadline τ_j) that fits its capacity. If no tasks can be processed, the UAV flags the region for neighboring UAVs to assist in the next timestamp, ensuring efficient, energy-conscious task processing across the swarm. The detailed scheduling algorithm is provided in **Algorithm 1**.

Algorithm 1 Task Scheduling and UAV Resource Allocation

Input: $T_i = \{T_1, \ldots, T_k\}$, $E_{\max}$, f_i, $Z = \{\mathcal{Z}_{ER}, \mathcal{Z}_{HR}\}$, $\mathcal{C} = \{\mathcal{C}_{\mathcal{E}}, \mathcal{C}_{\mathcal{P}}, \mathcal{C}_{\mathcal{R}}\}$, τ_k, task sizes D_k.
Sort $\mathcal{T}_k$ by sub-zone risk and task type in decreasing order of priority
for each UAV i in swarm **do**
  for each task $T_k \in T_i$ **do**
    if $E_{\max} \geq$ energy required for $T_k(D_k)$ and f_i can process T_k before deadline τ_k **then**
      Process task T_k
      $E_{\max} \leftarrow E_{\max} -$ energy used for $T_k(D_k)$
    else
      Search smaller task T_j such that $D_j < D_k$ and $E_{\max} \geq$ energy for T_j, and f_i can meet τ_j
      if T_j found **then**
        Process task T_j
        $E_{\max} \leftarrow E_{\max} -$ energy used for $T_j(D_j)$
      else
        Flag the region for additional UAV support using GNN.
      end if
    end if
  end for
end for
At the next time slot, neighboring UAVs adjust their positions to assist flagged regions

Algorithm 2 UAV Movement Between Hover Points

Input: Current hover point H_{current}, target hover point H_{target}, NFZ map, positions of UAVs, energy levels E_i
Compute direct path from H_{current} to H_{target} using A^*.
if direct path intersects NFZ **then**
  Compute alternate path using A^* to bypass NFZ, considering Euclidean distance as cost.
end if
Monitor positions of neighboring UAVs.
if potential collision detected with UAV j **then**
  if H_{target} of UAV i falls in $\mathcal{Z}_{ER}$ and UAV j does not **then**
    UAV j reroutes to avoid collision.
  else if H_{target} of both UAV i and UAV j fall in same risk zone **then**
    if UAV i has more energy than UAV j **then**
      UAV j reroutes.
    else
      UAV i reroutes.
    end if
  end if
end if
Move UAV along the computed safe path to H_{target}

B. UAV Movement with NFZ and Collision Avoidance

To ensure safe movement in disaster zones, UAVs navigate between predefined hover points (generated using

GenAI, described in Section VII), employing waypoint-based navigation to avoid free trajectories that could be disrupted by obstacles. Hover points provide a structured path, minimizing energy use and allowing dynamic position adjustments based on real-time conditions.

For path planning, UAVs use the A^* algorithm, which considers Euclidean distance as the cost function. If an NFZ blocks the direct path, the UAV reroutes using A^*, even if it results in a longer route [42]. Additionally, UAVs employ a collision-avoidance mechanism based on graph communication using the GNN. UAVs share their position and intended path with neighboring UAVs, allowing the swarm to predict potential collisions and adjust paths accordingly. If a potential collision is detected, UAVs follow a priority rule: the UAV targeting an emergency zone has higher priority, and if both UAVs have the same target zone, the UAV with more energy takes priority, while the other reroutes. This hover point deployment simplifies navigation and energy management, making it ideal for disaster areas where free-flight paths are impractical. UAVs stay at a hover point for a time slot, then shift based on real-time data, calculating the safest path to avoid NFZs and collisions using pre-defined inputs. The UAV movement algorithm is described in **Algorithm 2**.

C. Adaptive UAV Swarm Optimization for Disaster Recovery

To tackle the inherent challenges of UAV deployment in disaster recovery, such as optimal hover point selection, dynamic task generation, and resource management within energy and NFZ constraints, we develop a GenAI-based graph model. Unlike conventional approaches, the use of GenAI ensures that hover points are generated according to real-world environmental factors, including terrain and NFZs, rather than being randomly simulated. Additionally, it generates tasks at GTs that realistically represent the demands in a disaster scenario. The GNN within the model facilitates real-time sharing of key information between UAVs, such as energy levels, task loads, and intended paths, allowing for efficient coordination, collision avoidance, and hover point adjustment. The full description of the GenAI-graph model and its integration with GNN is detailed in Section VII.

Building on this, we develop a multi-agent graph reinforcement learning framework with attention mechanisms to optimize UAV decision-making. This framework continuously updates UAV policies through actor-critic learning, ensuring better task completion, efficient energy usage, and adaptability to changing environmental conditions. By focusing on the most relevant neighbors and tasks, the model effectively integrates hover point and task generation from the generative AI approach, explained in Section VIII.

VII. Generative AI-Graph Model for UAV Deployment

In this section, we propose a GAN integrated with a GNN to address the problem of UAV deployment [43], hover point generation, and task generation in a disaster recovery scenario. The GAN component is responsible for generating optimal hover points for UAVs, avoiding NFZs, minimizing energy consumption, and ensuring task generation at the GTs is realistic and efficient. The GNN component facilitates structured interactions between UAVs and GTs, enabling real-time adaptive hover point shifting as UAVs reposition themselves dynamically over time.

A. GAN Architecture Overview

The proposed GAN architecture consists of two main components: a *Generator* and a *Discriminator*. The generator generates potential hover point locations for UAVs and assists in task generation at the GTs, while the discriminator evaluates the quality and feasibility of these generated solutions [44].

1) Generator: The generator G takes as input the current state of UAVs and GTs, encoded as a latent vector z_i, as well as the graph structure $\mathcal{G}$, which represents communication links between UAVs and GTs. The generator outputs potential hover points $\mathbf{p}_i^{\text{hover}}$ for each UAV i and task generation parameters for each GT k.

The latent vector z_i encodes relevant UAV information, such as energy levels, current positions, and computational load. The graph $\mathcal{G}$, comprising UAV nodes and GT nodes, captures multi-agent interactions related to UAV movements and GT task demands. A GNN is employed within the generator to propagate information across the graph structure, enabling coordinated UAV-GT interactions for deployment decisions. The hover points generated for UAVs are continuously adapted based on real-time conditions, such as task loads, energy constraints, and risk level changes within the disaster zone. The hover point for UAV i is defined as:

$$\mathbf{p}_i^{\text{hover}} = G(z_i, \mathcal{G}), \tag{28}$$

where z_i is the latent vector encoding UAV i's state, and $\mathcal{G}$ is the graph structure representing UAV-GT interactions. For task generation at GT k, the generator produces task parameters based on the current state of GT k:

$$T_k = G(z_k, \mathcal{G}), \tag{29}$$

where z_k represents the latent state of GT k, and T_k indicates the generated task load for GT k based on current conditions. The generator is trained to minimize the discrepancies between the generated hover points and valid solutions, ensuring that hover points avoid NFZs, optimize energy usage, and meet task deadlines.

2) Discriminator: The discriminator D evaluates the feasibility of the generated hover points and task generation. It checks that hover points avoid NFZs, energy consumption is within acceptable limits, and tasks generated at GTs are realistic to replicate real-world disaster scenarios. Hover point feasibility is given by:

$$D_{\text{NFZ}}(\mathbf{p}_i^{\text{hover}}) = \begin{cases} 1, & \text{if } \mathbf{p}_i^{\text{hover}} \notin \text{NFZ}, \\ 0, & \text{if } \mathbf{p}_i^{\text{hover}} \in \text{NFZ}, \end{cases} \tag{30}$$

To ensure UAVs do not collide while navigating toward hover points, a collision avoidance term is introduced:

$$D_{\text{c}}(\mathbf{p}_i^{\text{hover}}) = \begin{cases} 1, & \text{if UAV } i \text{ doesn't collide with other UAV,} \\ 0, & \text{if UAV } i \text{ path collides with other UAVs.} \end{cases} \tag{31}$$

The discriminator also checks whether the energy consumption remains within the UAV's energy budget:

$$D_{\text{en}}(\mathbf{p}_i^{\text{hover}}) = \begin{cases} 1, & \text{if } E_{\text{move},i} \leq E_{\text{max},i}, \\ 0, & \text{if } E_{\text{move},i} > E_{\text{max},i}. \end{cases} \tag{32}$$

For task generation, the discriminator evaluates whether the generated tasks at GTs are feasible based on available resources and deadline constraints:

$$D_{\text{ts}}(T_k) = \begin{cases} 1, & \text{if } T_k \text{ meets constraints,} \\ 0, & \text{otherwise.} \end{cases} \tag{33}$$

3) Training Process: The generator G is trained to minimize the differences between the generated solutions and valid solutions, while the discriminator D is trained to distinguish between valid and invalid solutions. Since D_{NFZ}, D_{en}, D_{ts}, and D_{c} are binary (0 or 1), the expected values $\mathbb{E}$ represent the average outcome over multiple training samples. The generator loss function is given by:

$$\mathcal{L}_G = \mathbb{E}\left[1 - D_{\text{NFZ}}\right] + \mathbb{E}\left[1 - D_{\text{en}}\right] + \mathbb{E}\left[1 - D_{\text{ts}}\right] + \mathbb{E}\left[1 - D_{\text{c}}\right], \tag{34}$$

where each term reflects whether the generated solution violates the corresponding constraint (NFZ, energy, task feasibility, or collision avoidance). The message-passing mechanism of the GNN enables the UAVs to share task loads, energy status and intended positions with their neighbors. This ensures that the generated hover points are not only optimal for task coverage but also avoid collisions and respect the energy constraints of each UAV.

Consequently, the discriminator loss function is given as:

$$\mathcal{L}_D = \mathbb{E}\left[\log\left(D\left(\mathbf{p}_i^{\text{hover}}, T_k\right)\right)\right] + \mathbb{E}\left[\log\left(1 - D\left(G(z_i, \mathcal{G})\right)\right)\right], \tag{35}$$

where the discriminator learns to classify the hover point and task allocation as valid or invalid while ensuring collision-free paths for UAVs.

B. Graph-Based Representation

The graph $\mathcal{G}$ represents the interactions between UAVs and GTs. Each node in the graph represents a UAV or a GT, while the edges represent communication links. Each UAV node i contains features such as $E_{\text{max}}(t)$, $f_i(t)$, and $\mathbf{p}_i(t)$. GT nodes k contain features such as total task load D_k and deadline τ_k. A GNN propagates information through message passing, updating the UAV and GT embeddings iteratively as:

$$\mathbf{h}_i^{(t+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} W \mathbf{h}_j^{(t)} + b\right), \tag{36}$$

Algorithm 3 GenAI-GNN for UAV Deployment and Task Generation

Input: UAV states $\mathbf{s}_i(t)$, GT states $\mathbf{s}_k(t)$, $\mathcal{G}$, z_i, risk levels.
Initialize generator G and discriminator D.
Output: $\mathbf{p}_i^{\text{hover}}$, T_k, updated UAV-GT graph structure $\mathcal{G}$.
Generator Process:
for each UAV i **do**
    Encode UAV state: $z_i = \text{Encode}(\mathbf{s}_i(t))$.
    Generate hover point $\mathbf{p}_i^{\text{hover}} = G(z_i, \mathcal{G})$ as per (37).
    Adapt hover points based on real-time task loads, energy constraints, and risk levels.
end for
for each GT k **do**
    Encode GT state: $z_k = \text{Encode}(\mathbf{s}_k(t))$.
    Generate task parameters $T_k = G(z_k, \mathcal{G})$ as in (33).
end for
for each UAV i **do**
    Propagate information via message passing on graph $\mathcal{G}$ to update UAV embeddings:
    Update UAV embeddings based on task loads, energy, and positions as per (36).
end for
Discriminator Process:
for each hover point $\mathbf{p}_i^{\text{hover}}$ **do**
    Check NFZ constraints using (30).
end for
for each UAV i and j (neighboring UAVs) **do**
    Evaluate collision risk using (31).
end for
for each UAV i **do**
    Ensure energy constraints as per (32).
end for
Training Process:
for each training step **do**
    Update the generator by minimizing loss in (34).
    Update the discriminator by minimizing loss in (35).
end for
Repeat until convergence.

where $\mathcal{N}(i)$ is the neighborhood of UAV i, W is a learnable weight matrix, and σ is a non-linear activation function. Additionally, UAVs share their intended paths $\mathbf{p}_i^{\text{target}}(t)$ and energy levels with neighboring UAVs to facilitate collision avoidance. The GNN updates the UAV embeddings based on both current positions and intended paths, ensuring that UAV i will dynamically reroute if a collision with UAV j is predicted. This can be represented as:

$$\mathbf{h}_i^{\text{path}}(t+1) = \sigma\left(\sum_{j \in \mathcal{N}(i)} W \mathbf{p}_j^{\text{target}}(t) + W^{\text{eng}} E_{\text{max}_j}(t) + b\right), \tag{37}$$

where $E_{\text{max}_j}(t)$ is the current energy level of UAV j. The detailed GenAI-based GNN model is given in **Algorithm 3**.

VIII. Multi-Agent Graph Reinforcement Learning for UAV Swarm Operations

This section presents the MAGRL framework for optimizing UAV swarm operations in disaster recovery. By integrating the GAN-GNN model for initial hover point and task generation, MAGRL enables UAVs to make coordinated decisions that maximize task completion, balance computational loads, minimize energy consumption, and improve coverage. UAVs and GTs form a natural graph through their communication links, which MAGRL uses via GNNs to share information and adapt to real-time environmental changes. The GAN-GNN model generates initial hover points and task allocations, considering environmental constraints like NFZs and task demands. As the UAV operations progress, the GNN enables dynamic refinement of hover points and task reallocation to respond to changes in its environmental factors.

We propose further enhancing this decision-making process through a multi-agent graph attention actor-critic (MAGAC) framework, combining graph attention mechanisms for improved coordination with actor-critic learning for continuous policy updates. This approach allows UAVs to focus on relevant neighbors, adjust strategies based on environmental feedback, and optimize overall system performance.

A. Preliminaries

1) State: The state $s_i(t)$ for UAV i includes local information: $\mathbf{p}_i(t)$, $E_{\max}(t)$, $f_i(t)$, $L_i(t)$, and $\mathbf{p}_i^{\text{hover}}$ from the GAN. Neighboring UAV states—$\mathbf{p}_j(t)$, $E_j(t)$, $f_j(t)$, $L_j(t)$ for all $j \in \mathcal{N}(i)$—and GT task loads T_k and deadlines D_k for $k \in \mathcal{K}_i(t)$ are included through GNN-based message passing on graph $\mathcal{G}(t)$. Each task T_k is associated with a deadline τ_k, which UAVs must meet to avoid penalties. This deadline information is shared across the UAV network through the GNN, enabling neighboring UAVs to assist when necessary.

2) Action: The action space $A_i(t)$ consists of movement, computation, and communication actions. UAV i's movement action $a_i^{\text{move}}(t)$ shifts it to a new position $\mathbf{p}_i(t+1)$, including the GAN-generated hover point. Computation action $a_i^{\text{comp}}(t)$ allocates $f_i(t)$ for local processing or offloading. Communication action $a_i^{\text{comm}}(t)$ adjusts transmission power P_t to maintain UAV-GT links.

3) Reward: The reward function $r_i(t)$ for UAV i aims to maximize overall system utility while penalizing undesired behavior, expressed as:

$$\begin{aligned} r_i(t) = U_{\text{total},i}(t) &+ \lambda_{\text{plty}} \cdot \mathbb{1}_{\text{NFZ}} \\ &+ \lambda_{\text{dln}} \cdot (\mathbb{1}_{\text{miss}}(\tau_k) - \mathbb{1}_{\text{comp}}(\tau_k)) \\ &+ \eta H(\pi_i(t)), \end{aligned} \tag{38}$$

where $U_{\text{total},i}(t)$ represents the total utility, $\mathbb{1}_{\text{NFZ}}$ applies a penalty for UAV i violating NFZs, λ_{dln} controls the penalties for missed deadlines $\mathbb{1}_{\text{miss}}(\tau_k)$ and rewards for task completion $\mathbb{1}_{\text{comp}}(\tau_k)$, and $H(\pi_i(t))$ is the policy entropy encouraging exploration, with η as the exploration weight.

The MAGAC algorithm is described in detail across two parts: the first part outlines the overall framework and operational procedures for optimizing UAV swarm operations, described in **Algorithm 4**, while the second part provides a comprehensive overview of the reinforcement learning processes, including policy and value function updates, described in **Algorithm 5**. The following subsections provide a detailed explanation of these components.

B. Policy and Value Function Approximation

In the MAGRL framework, each UAV maintains a policy $\pi_i(a_i|s_i)$ and a value function $V_i(s_i)$, both parameterized using neural networks. The policy $\pi_i(a_i|s_i)$ maps states to action probabilities, guiding UAV i's decisions at each time step. It is updated using a policy gradient method, where the advantage function weights the gradient of the expected log probability of the selected action. The policy gradient is given by:

$$\nabla_{\theta_i} \pi_i(a_i|s_i) = \mathbb{E}\left[\nabla_{\theta_i} \log \pi_i(a_i|s_i) \cdot \delta_i(t)\right], \tag{39}$$

where $\delta_i(t)$ is the temporal difference (TD) error, which serves as the basis for updating the policy, computed as:

$$\delta_i(t) = r_i(t) + \beta V_i(s_i(t+1)) - V_i(s_i(t)). \tag{40}$$

where β is the discount factor. For stable training, gradient clipping is applied to ensure the gradients do not exceed a specified threshold. The gradient is clipped as:

$$\nabla_{\text{clip}} = \text{clip}\left(\nabla_{\theta_i}, -c_{\text{clip}}, c_{\text{clip}}\right), \tag{41}$$

where c_{clip} represents the gradient clipping threshold. Additionally, learning rate decay is applied after each update to dynamically reduce the learning rate as

$$\eta_{\text{new}} = \gamma_{\text{decay}} \cdot \eta_{\text{old}}, \tag{42}$$

where γ_{decay} is the decay factor controlling the rate of reduction over time. The value function $V_i(s_i)$ estimates the expected cumulative reward from state s_i and is updated by minimizing the squared TD error as:

$$L_{\text{value},i} = \left(V_i(s_i(t)) - (r_i(t) + \beta V_i(s_i(t+1)))\right)^2. \tag{43}$$

C. Graph Attention Mechanism

To efficiently aggregate information from neighboring UAVs and GTs, we employ a graph attention network (GAT) within the MAGRL framework. The GAT allows each UAV to focus on the most relevant neighboring UAVs and GTs during the decision-making process, enhancing the quality of information propagation within the swarm [45]. The state embedding for UAV i from (36) is updated using the attention mechanism as:

$$\mathbf{h}_i = \phi\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} W \mathbf{h}_j\right), \tag{44}$$

where ϕ is a non-linear activation function, W is a learnable weight matrix, and α_{ij} is the attention coefficient that determines the importance of UAV j's information to UAV i. The embedding $\mathbf{h}_j$ refers to the current state embedding of neighboring UAV j or GT j. The attention coefficient α_{ij} is

Algorithm 4 MAGAC *Part 1*: Initialization & State Update

Input: UAV states $\mathbf{s}_i(t)$, GT states $\mathbf{s}_k(t)$, graph structure $\mathcal{G}(t)$, hover points $\mathbf{p}_i^{\text{hover}}$, energy budgets $E_{\max,i}$, CPU resources f_i, task deadlines τ_k, initial policies $\pi_i(a_i|s_i)$, value functions $V_i(s_i)$.
Output: Updated UAV states and embeddings $\mathbf{h}_i(t)$.
Initialization:
Initialize generator G and discriminator D of the GenAI-GNN system.
Initialize policy $\pi_i(a_i|s_i)$ and value function $V_i(s_i)$ for each UAV i.
Initialize GAT for information propagation in the GNN.
Initialize hover points $\mathbf{p}_i^{\text{hover}}$ from GenAI model.
Initialize the UAV-GT graph structure $\mathcal{G}$ based on communication links.
for each training episode **do**
    Initialize $convergence\_flag \leftarrow False$.
    Initialize $t \leftarrow 0$.
    while not $convergence\_flag$ **do**
        Increment t.
        Step 1: Policy and State Update
        for each UAV i **do**
            Encode UAV state $\mathbf{s}_i(t)$ with position $\mathbf{p}_i(t)$, energy $E_{\max,i}$, CPU f_i, and task load L_i.
            Gather neighboring UAV states $\mathbf{s}_j(t)$ for $j \in \mathcal{N}(i)$ via GNN message passing on $\mathcal{G}(t)$.
            Update UAV state embeddings $\mathbf{h}_i(t)$ using attention mechanism (44).
            Propagate task deadlines τ_k and task loads T_k from neighboring UAVs/GTs as in (46).
            Encode GT states $\mathbf{s}_k(t)$ and update graph $\mathcal{G}$.
        end for
    end while
end for

Algorithm 5 MAGAC *Part 2*: Action Selection, Reward Calculation, and Training

Input: Updated UAV states and embeddings $\mathbf{h}_i(t)$.
Output: Optimal $\pi_i(a_i|s_i)$ over $\mathbf{p}_i(t+1)$.
while not $convergence\_flag$ **do**
    Step 2: Action Selection
    for each UAV i **do**
        Select movement action $a_i^{\text{move}}(t)$ using policy $\pi_i(a_i|s_i)$ based on $\mathbf{p}_i^{\text{hover}}$.
        Select computation action $a_i^{\text{comp}}(t)$ based on task load L_i and CPU f_i.
        Select communication action $a_i^{\text{comm}}(t)$ to adjust transmission power P_t.
    end for
    Step 3: Reward Calculation
    for each UAV i **do**
        Calculate the $r_i(t)$ using $U_{\text{total},i}(t)$ and penalties for NFZ violations and task deadlines as in (38).
        Store rewards for policy update and learning.
    end for
    Step 4: Policy and Value Update
    for each UAV i **do**
        Compute TD error $\delta_i(t)$ as in (40).
        Update policy gradient $\nabla_{\theta_i}\pi_i(a_i|s_i)$ as in (39).
        Minimize $L_{\text{value},i}$ by updating $V_i(s_i)$ as in (43).
        Compute Compute total loss $\mathcal{L}_i$ using (51).
        Update model parameters with ψ as the learning rate: $\theta_i \leftarrow \theta_i - \psi\nabla\mathcal{L}_i$,
    end for
    Step 5: Training Process
    Update the generator G using loss $\mathcal{L}_G$ in (34).
    Update the discriminator D using loss $\mathcal{L}_D$ in (35).
    Refine hover points $\mathbf{p}_i(t+1)$ based on generator outputs and current states.
    Step 6: Convergence Check
    if average reward over last N episodes $\geq \epsilon$ **then**
        Set $convergence\_flag \leftarrow True$.
    end if
    if maximum iteration $t \geq T_{max}$ **then**
        Set $convergence\_flag \leftarrow True$.
    end if
end while

computed to assign varying importance to different neighbors and is given by:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(a^\top[W\mathbf{h}_i \parallel W\mathbf{h}_j]))}{\sum_{k\in\mathcal{N}(i)}\exp(\text{LeakyReLU}(a^\top[W\mathbf{h}_i \parallel W\mathbf{h}_k]))}, \quad (45)$$

where a is a learnable attention vector, $\parallel$ denotes the concatenation operation, and $\mathbf{h}_i$, $\mathbf{h}_j$, and $\mathbf{h}_k$ are the state embeddings of UAV i, neighboring UAV j, and another neighbor k, respectively. In this equation, the attention coefficient α_{ij} plays a critical role in determining the importance of the information from neighboring node j to node i. The objective of this coefficient is to dynamically weigh the influence of each neighbor's features based on their relevance to the current node's task or state. This mechanism allows the model to focus more on important neighbors, enhancing the learning efficiency and adaptability of the network in diverse scenarios. The numerator computes the importance of UAV j to UAV i using the LeakyReLU activation, and the denominator normalizes these importance scores across all neighbors in $\mathcal{N}(i)$, ensuring that the attention coefficients α_{ij} sum to 1. Additionally, task deadlines τ_k and task loads T_k from neighboring UAVs and GTs are incorporated into the embedding update to assist in task reallocation:

$$\mathbf{h}_i^{\text{task}}(t+1) = \sigma\left(\sum_{j\in\mathcal{N}(i)} W^{\text{task}}\mathbf{T}_j(t) + W^{\text{dln}}\tau_j(t) + b\right), \quad (46)$$

where $\mathbf{T}_j(t)$ represents the task load from neighboring GTs or UAVs, and $\tau_j(t)$ represents the task deadline. The GNN aids in collision avoidance by relaying intended paths and alerts,

which are used to modify UAV i's embedding as shown in the following equation:

$$\mathbf{h}_i^{\text{path}}(t+1) = \sigma\left(\sum_{j\in\mathcal{N}(i)} W\mathbf{p}_j^{\text{target}}(t) + \hat{W}\mathbb{1}_{\text{collide}}(i,j) + b\right), \tag{47}$$

where $\hat{W}$ represents a learnable weight matrix applied to $\mathbb{1}_{\text{collide}}(i,j)$ which is an indicator function that checks whether UAV i and UAV j are on a collision course.

D. Multi-Agent Advantage Actor-Critic (MAGAC)

We adopt the multi-agent graph attention actor-critic framework to optimize the policies and value functions for each UAV. MAGAC utilizes both actor and critic networks to improve the UAV's decision-making process. The advantage function $A_i(s_i, a_i)$ measures the relative value of an action and is defined as:

$$A_i(s_i, a_i) = Q_i(s_i, a_i) - V_i(s_i)\,, \tag{48}$$

where $Q_i(s_i, a_i)$ represents the action-value function, approximated using the TD error. The policy loss is given by:

$$L_{\text{policy},i} = -\mathbb{E}\left[\log \pi_i(a_i|s_i) A_i(s_i, a_i)\right]\,, \tag{49}$$

where $\mathbb{E}$ indicates that the $L_{\text{policy},i}$ is computed as the expected value of the log probability of taking action a_i given state s_i weighted by the advantage function $A_i(s_i, a_i)$. The value network is updated by minimizing the squared TD error as:

$$L_{\text{value},i} = \mathbb{E}\left[\left(V_i(s_i(t)) - (r_i(t) + \beta V_i(s_i(t+1)))\right)^2\right]\,. \tag{50}$$

The total loss for UAV i is a combination of the policy loss, value loss, and entropy regularization to encourage exploration:

$$\mathcal{L}_i = L_{\text{policy},i} + c_v L_{\text{value},i} - c_e H(\pi_i)\,, \tag{51}$$

where the entropy $H(\pi_i)$ is used to encourage exploration by ensuring that the policy does not become too deterministic early on, computed as:

$$H(\pi_i) = -\sum_{a_i} \pi_i(a_i|s_i) \log \pi_i(a_i|s_i)\,, \tag{52}$$

the notations c_v and c_e are hyperparameters that control the importance of the value loss and entropy regularization, respectively, in the total loss function.

E. Integration with GAN-GNN for Hover Point Shifting

The integration of GAN-GNN with MAGRL occurs at multiple stages of the UAV swarm operation. Initially, the GAN generates optimal hover points $\mathbf{p}_i^{\text{hover}}$ for UAVs, ensuring an effective starting position by minimizing energy consumption and avoiding NFZs. These hover points serve as inputs to the GNN, which propagates task demands and neighboring UAV information through message passing. The GNN-based embedding for UAV i, denoted by $\mathbf{h}_i(t)$, is updated at each time step based on neighboring UAVs and task information as shown in (44)

The updated $\mathbf{h}_i(t)$ are then fed into the policy $\pi_i(a_i|s_i)$ and value networks $V_i(s_i)$, which refine the UAV's hover point decisions and task allocation. Specifically, the movement action $a_i^{\text{move}}(t)$ is selected based on the updated policy:

$$a_i^{\text{move}}(t) = \arg\max_{a_i} \pi_i(a_i|s_i)\,. \tag{53}$$

This adjusts the UAV's position to a new hover point $\mathbf{p}_i(t+1)$, either moving toward the GAN-generated hover point $\mathbf{p}_i^{\text{hover}}$ or another optimal point identified by the MAGRL framework.

At each time step, the GAT-based policy and value networks continuously refine the UAVs' hover point and task allocation decisions, ensuring efficient adaptation to real-time changes in the disaster environment. This integrated approach enables dynamic coordination among UAVs, allowing them to maintain coverage, balance computational loads, and manage energy consumption effectively.

IX. Performance Evaluation

In this section, we evaluate the proposed solution frameworks for UAV swarm optimization in dynamic disaster areas. We begin with the dataset preparation process, followed by the detailed simulation setup.

A. Simulation Setup

The simulations were run on a high-performance system equipped with an NVIDIA T400 GPU (4GB), 512GB SSD, and an Intel(R) Core(TM) i7-14700 CPU (20 cores, 2.10 GHz). Python 3.8 was the programming environment, utilizing libraries such as NumPy, Pandas, Matplotlib, and TensorFlow for implementing the GenAI-GNN and MAGAC frameworks.

The simulation environment models a disaster area sized $D_x \times D_y = 1000 \times 1000$ square meters, containing 100 fixed GTs, each with $f_k = 1 \times 10^9$ cycles per second of computational capacity. The tasks generated at the GTs are categorized as emergency, priority, and routine, with varying task sizes, deadlines, and CPU cycles. Task generation occurs dynamically at a rate of 10 to 40 tasks per minute, utilizing the GenAI model for realism. The simulation environment uses the Telecom Italia Big Data Challenge dataset, representing realistic ground terminal task generation in urban disaster recovery scenarios [46][1].

Fixed NFZs are defined in circular, square, and rectangular patterns, with a buffer distance NFZ_{buffer} around them. The GAN model is used to dynamically generate hover points, ensuring they do not intersect with NFZs and high-risk zones, optimizing UAV positioning and task allocation. The UAV swarm's objective is to maximize task completion, minimize energy consumption, and avoid NFZs, with adaptive reallocation based on real-time changes. The parameters and hyperparameters utilized in the simulations are detailed in

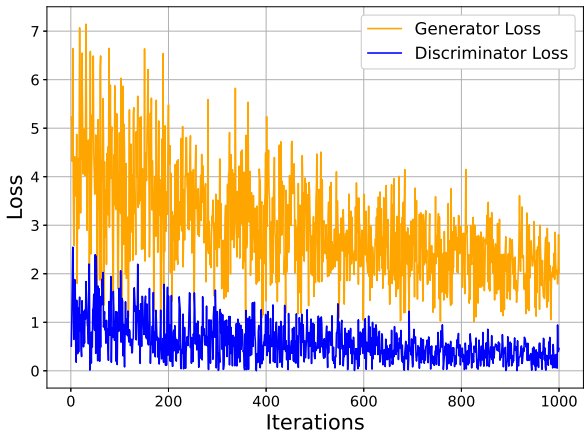
[1]https://www.kaggle.com/datasets/ocanaydin/italian-telecom-data-2013-1week

TABLE I: Parameters

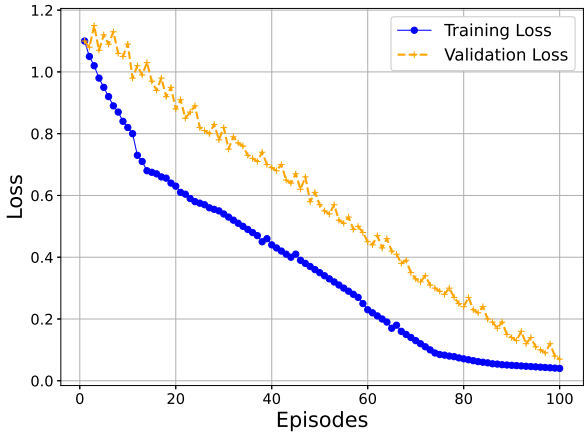
Parameter	Value
D_x	1000 m
D_y	1000 m
h_u	100 m
v_{max}	25 m/s
N_{UAV}	20
N_{GT}	100
NFZ_{buffer}	10 m
R_s	500 m
P_{comp}^{UAV}	4×10^{-9} J/cycle
P_{comp}^{GT}	4×10^{-9} J/cycle
P_{comm}	8×10^{-9} J/bit
P_{move}	30 W/m
P_{hover}	180 W
E_{max}^{UAV}	1×10^{6} J
B	2×10^{7} Hz
P_t	5 W
σ^2	1×10^{-9} W
$tasks/GT$	100
D_k	1×10^{5} to 5×10^{6} bits
f_k	1×10^{5} to 1×10^{6} cycles/bit
τ_k	5 to 30 seconds
κ	1×10^{-10}

TABLE II: Hyperparameters

Hyperparameter	Value	Hyperparameter	Value
z_i	100	GNN Dimension	128
GNN Dropout Rate	0.2	Training Epochs	1000
Batch Size	64	$LR_{\text{generator}}$	0.001
$LR_{\text{discriminator}}$	0.001	LR_{actor}	0.0005
LR_{critic}	0.0005	Attention Head	8
c_v	0.5	c_e	0.01
β	0.99	ϵ	0.01
λ_{plty}	1.0	λ_{dln}	2.0
η	0.1	T_{max}	1000
c_{clip}	0.5	γ_{decay}	0.95



(a) GAN Loss Convergence.



(b) MAGAC Training and Validation Loss.

Fig. 2: GAN and MAGAC Loss.

Tables I and **Table II**, respectively. The learning rates for the generator, discriminator, actor, and critic are denoted as LR_{gen}, LR_{disc}, LR_{actor}, and LR_{critic}.

B. Benchmark Comparisons

We compare our proposed GenAI-GNN-based MAGAC framework with several benchmarks, applying the same GenAI-based task generation method across all approaches for a fair comparison:

1) Multi-Agent Deep Deterministic Policy Gradient (MADDPG): UAVs use MADDPG to make decisions on task allocation and movement. Initial hover points are deployed based on a geometric grid, with each UAV adjusting positions using learned policies in continuous action spaces.

2) Graph Convolutional Reinforcement Learning (GCRL): GCRL uses graph-based Q-learning to optimize task allocation. UAVs start with grid-based hover points, refined using graph convolution layers and Q-learning updates.

3) Particle Swarm Optimization (PSO): UAVs iteratively optimize positions using PSO to balance task completion and energy efficiency. Hover points are adjusted based on particle positions.

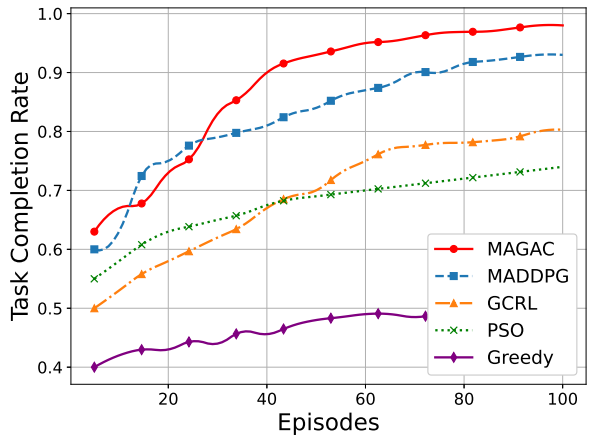
4) Greedy Approach: UAVs make local decisions, always choosing tasks that minimize energy consumption or maximize coverage without considering long-term planning. Hover points are grid-based.

X. Result Analysis

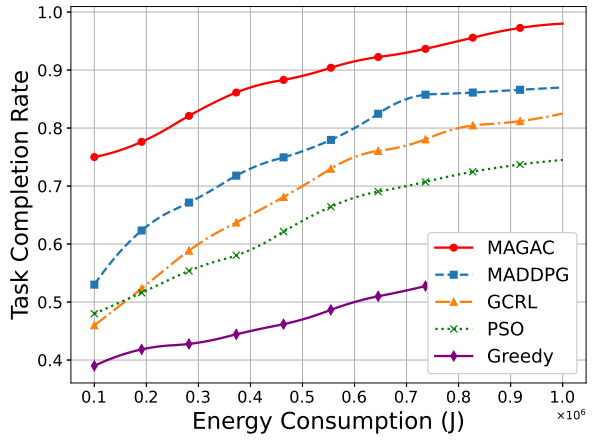
Below we present the simulation results and analysis.

A. Loss

Fig. 2 displays the generator-discriminator and training-validation losses for GAN and MAGAC during the optimization process. In Fig. 2a, the generator loss fluctuates initially but decreases and stabilizes, demonstrating improved performance in generating hover points and task loads. The discriminator loss shows fewer fluctuations, stabilizing faster, indicating its efficiency in distinguishing feasible solutions. Both losses converge, reflecting the effective interaction between the generator and discriminator. In Fig. 2b, the training and validation losses for the MAGAC framework are plotted over episodes. Initially, both losses are high, reflecting early optimization challenges. The training loss steadily decreases, showing effective learning and smooth convergence as episodes progress. The validation loss, while



(a) Task completion rate vs. Episodes.



(b) Task completion rate vs. Energy Consumption.

Fig. 3: Task completion rate analysis.

following a similar downward trend, exhibits more fluctuations, indicating variability in performance on unseen data. Despite the fluctuations, the overall decline in both losses demonstrates successful optimization and generalization of MAGAC.

B. Task Completion Rate

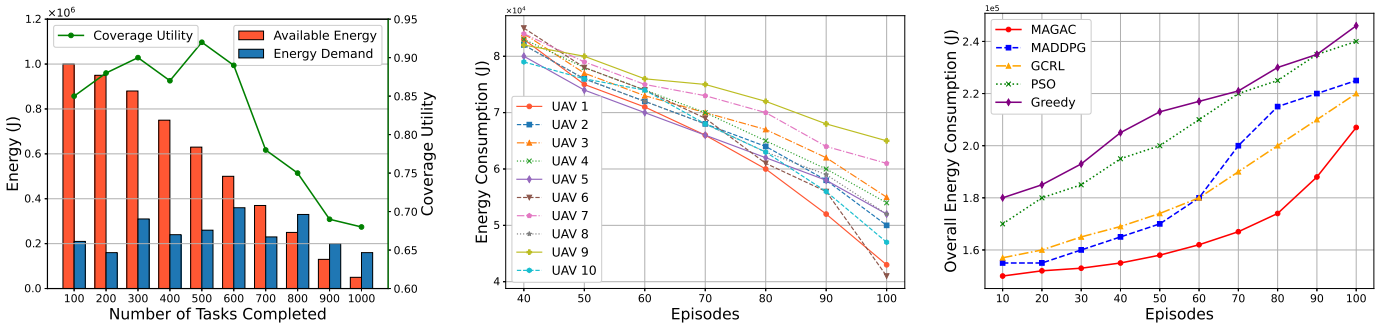
In Fig. 3a, the task completion rate of tasks completed over time. The proposed MAGAC framework consistently achieves the highest task completion rate, reaching 98% after 100 episodes. This performance can be attributed to the integration of GenAI for dynamic hover point selection and the use of GNN in the multi-agent actor-critic framework, allowing UAVs to adapt to real-time task demands and optimize collectively. MADDPG achieves a 93% completion rate, performing well due to decentralized learning, but its lack of explicit inter-agent coordination via GNN limits its overall adaptability compared to MAGAC. GCRL achieves 80.3% task completion, initially lagging due to slower Q-learning convergence but improving with enhanced UAV interactions over time. PSO performs better early on, reaching 74% with fast heuristic optimization but plateaus later, allowing GCRL to surpass it. The Greedy approach performs the worst, with 51% completion, due to its lack of optimization and inefficient task allocation.

In Fig. 3b, the task completion rate is plotted against the total energy consumption by the UAV swarm. Energy consumption is measured as the total energy used by all UAVs during task execution, including hover energy, movement energy, and task processing. The proposed MAGAC framework achieves the highest task completion rate, approximately 98%, while consuming 1×10^6 Joules of energy. This is due to MAGAC's efficient energy management, leveraging GenAI for hover point optimization and GNN to balance energy use among UAVs, allowing them to complete tasks with minimal waste. MADDPG performs slightly below MAGAC, reaching 90% task completion at 1×10^6 Joules. Although it is effective, its independent agent learning results in less efficient energy usage compared to MAGAC, which uses inter-agent coordination. GCRL improves task completion as energy consumption rises, reaching 83% at 1×10^6 Joules, though its slower Q-learning convergence limits efficiency. PSO achieves 75%, benefiting from fast heuristic optimization but lacking adaptability, leading to suboptimal energy use. Greedy performs worst, plateauing at 56% due to poor resource allocation and inefficient energy management.

C. Energy

Fig. 4a illustrates the trade-off between energy consumption and coverage performance in the MAGAC framework. Initially, UAVs maintain high coverage utility as they start with full energy reserves, effectively managing a broad area. However, energy depletion becomes evident as tasks continue, particularly after around 300 and 800 tasks, when high-energy-demand tasks requiring more movement or longer operational times are activated. This depletion is exacerbated when UAVs are reallocated to prioritize emergency tasks in high-risk zones, indicated by $\mathcal{C}_E$ tasks in $\mathcal{Z}_{HR}$, leading to less efficient coverage elsewhere. The decline in coverage utility becomes more pronounced after 700 tasks, marking a critical threshold where remaining energy is insufficient to sustain earlier coverage levels due to cumulative energy expenditure from intensive tasks. This results in a significant drop in coverage utility to approximately 0.68, reflecting the UAVs' struggle to balance energy conservation with effective task completion under constrained conditions.

In Fig. 4b, the energy consumption pattern of 10 UAVs over episodes 40-100 shows variability in depletion rates, reflecting differences in task assignment. UAVs such as UAV 1 and UAV 6 deplete energy faster due to more intensive tasks, while others, like UAV 7 and UAV 9, show slower depletion, suggesting more efficient energy use or less demanding tasks. This highlights the need for better energy distribution to optimize overall performance. Fig. 4c compares total energy consumption across approaches. MAGAC consumes the least energy due to its efficient task and energy management. MADDPG consumes slightly more due to less optimal coordination. GCRL, while initially moderate, increases sharply in later episodes due to slower convergence. PSO and Greedy, lacking optimization, show the steepest



(a) Energy vs. Tasks vs. Coverage.

(b) Energy vs. Episodes.

(c) Total energy vs. Episodes.

Fig. 4: Energy consumption and coverage trends across different tasks, episodes, and system-wide metrics.

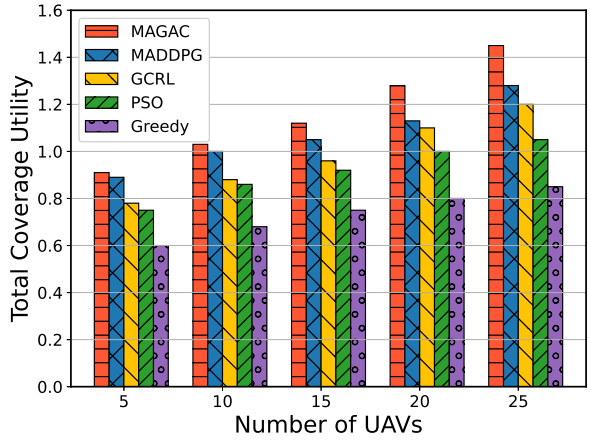


Fig. 5: Coverage utility vs. number of UAVs

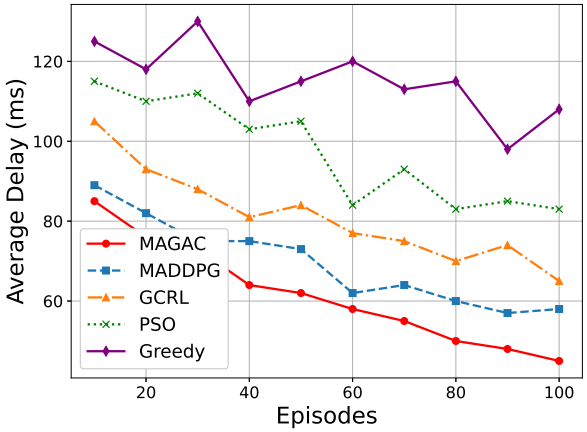


Fig. 6: Average delay per Episode.

energy consumption, with Greedy displaying erratic patterns due to its lack of task allocation strategy.

D. Coverage

Fig. 5, shows the coverage utility as a function of the number of UAVs deployed. MAGAC, leveraging GenAI-generated hover points and GNN coordination, achieves the highest utility, reaching 1.45 with 25 UAVs by minimizing redundant coverage and optimizing communication. MADDPG follows with a utility of 1.28, performing well through decentralized learning, but lacking the inter-agent coordination of MAGAC, leading to slightly more redundant coverage. GCRL reaches 1.20, benefiting from graph-based learning but hindered by slower convergence and limited obstacle handling. PSO achieves 1.05 but plateaus due to its heuristic nature. Greedy, with 0.85, shows minimal improvement as more UAVs are added due to its inability to account for future states and obstacles.

E. Average Delay

Fig. 6 depicts the average delay per episode for various UAV optimization algorithms. The MAGAC framework shows consistent delay reduction, demonstrating effective task and UAV coordination. MADDPG exhibits noticeable fluctuations in delay, which stem from its decentralized learning structure. This approach can lead to suboptimal decisions when UAVs, operating independently, face uneven task distributions or unexpected changes in the environment, affecting their response times. GCRL displays high initial delays that decrease over time but with marked variability. These fluctuations arise from the inherent slow convergence of its graph-based Q-learning process, which struggles with rapid adaptability in dynamic conditions. In contrast, simpler strategies like PSO and Greedy show less consistent delay reduction. PSO's heuristic approach results in erratic performance, while Greedy's lack of a strategic outlook leads to the highest delays due to inefficient decision-making.

F. Total Utility

Fig. 7 shows the overall utility performance of different approaches over episodes. MAGAC demonstrates the best total utility performance because of its ability to optimally balance trade-offs between energy efficiency, coverage maximization, and task offloading. The effective use of GenAI for generating hover points and tasks allows MAGAC to adjust its strategy dynamically, ensuring both high coverage and low energy consumption. This synergy between components explains the steady utility growth with minimal variability, even in complex task scenarios. MADDPG shows a slower utility increase due to its focus on local optimization.

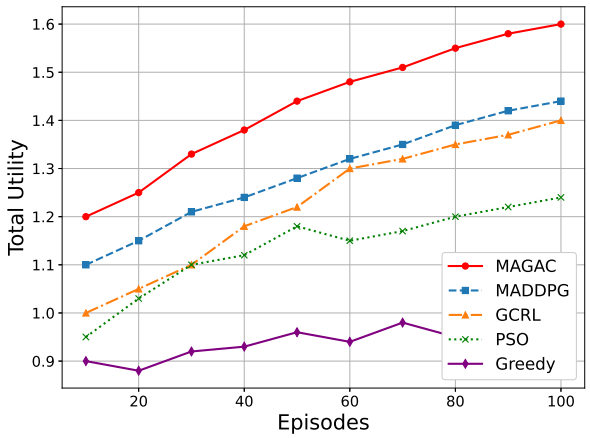


Fig. 7: Total utility vs. Episode.

Its lack of explicit global coordination between agents causes some inefficiencies regarding task distribution and energy usage. However, its decentralized decision-making allows for fairly stable performance, as tasks are managed without overwhelming energy drain. GCRL starts with lower utility due to the slower convergence of its graph-based reinforcement learning approach, which requires more episodes to optimize resource allocation. As the graph structure enables improved information sharing among UAVs, GCRL reduces task redundancies and redundant energy usage, leading to a sharp improvement in utility in later episodes. PSO exhibits erratic utility progression because its heuristic nature fails to adapt effectively to changing task demands and energy constraints. The algorithm's inability to learn or evolve results in temporary improvements but causes an early plateau, as PSO struggles to balance energy use with optimal task completion in dynamic scenarios. Greedy remains the worst performer, as it prioritizes immediate gains without considering future energy or task demands. This short-term focus leads to high energy consumption and inefficient resource utilization, which justifies its low and unstable utility throughout the episodes.

XI. Conclusion and Future Work

In this paper, we proposed an advanced framework for UAV swarm deployment in disaster recovery scenarios, integrating GenAI, GNN, and a MAGRL framework. Our approach effectively addresses key challenges such as energy efficiency, task offloading, real-time adaptability, and safe navigation through dynamic hover point generation and GNN-based collision avoidance. The inclusion of a graph attention mechanism further enhanced UAV coordination, leading to significant improvements across metrics like task completion rates, energy efficiency, and overall system utility. Through extensive simulations and benchmark comparisons with MADDPG, GCRL, PSO, and Greedy approaches, our proposed MAGAC framework demonstrated superior performance, achieving a 98% task completion rate while optimizing energy consumption.

For future work, we aim to address the current assumption of constant risk levels by introducing a dynamic risk-aware framework.

References

[1] J. Dai, W. Pu, J. Yan, Q. Shi, and H. Liu, "Multi-UAV collaborative trajectory optimization for asynchronous 3-D passive multitarget tracking," *IEEE Trans. Geosci. Remote Sens.*, vol. 61, pp. 1–16, 2023.

[2] H. Pan, Y. Liu, G. Sun, J. Fan, S. Liang, and C. Yuen, "Joint power and 3D trajectory optimization for UAV-enabled wireless powered communication networks with obstacles," *IEEE Trans. Commun.*, vol. 71, no. 4, pp. 2364–2380, 2023.

[3] L. Li, X. Wen, Z. Lu, W. Jing, and H. Zhang, "Energy-efficient multi-UAVs deployment and movement for emergency response," *IEEE Commun. Lett.*, vol. 25, no. 5, pp. 1625–1629, 2021.

[4] B. Li, Z. Na, and B. Lin, "UAV trajectory planning from a comprehensive energy efficiency perspective in harsh environments," *IEEE Netw.*, vol. 36, no. 4, pp. 62–68, 2022.

[5] Z. Yu, Y. Gong, S. Gong, and Y. Guo, "Joint task offloading and resource allocation in UAV-enabled mobile edge computing," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3147–3159, 2020.

[6] X. Dai, Z. Xiao, H. Jiang, and J. C. S. Lui, "UAV-assisted task offloading in vehicular edge computing networks," *IEEE Trans. Mob. Comput.*, vol. 23, no. 4, pp. 2520–2534, 2024.

[7] B. Hazarika, K. Singh, C.-P. Li, A. Schmeink, and K. F. Tsang, "RADiT: Resource allocation in digital twin-driven UAV-aided internet of vehicle networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3369–3385, 2023.

[8] H. Sun, X. Zhang, B. Zhang, K. Sha, and W. Shi, "Optimal task offloading and trajectory planning algorithms for collaborative video analytics with UAV-assisted edge in disaster rescue," *IEEE Trans. Veh. Technol.*, vol. 73, no. 5, pp. 6811–6828, 2024.

[9] H. Yang, R. Ruby, Q.-V. Pham, and K. Wu, "Aiding a disaster spot via multi-UAV-based iot networks: Energy and mission completion time-aware trajectory optimization," *IEEE Internet Things J.*, vol. 9, no. 8, pp. 5853–5867, 2022.

[10] J. Wang, Y. Sun, B. Wang, and T. Ushio, "Mission-aware UAV deployment for post-disaster scenarios: A worst-case SAC-based approach," *IEEE Trans. Veh. Technol.*, vol. 73, no. 2, pp. 2712–2727, 2024.

[11] P. Hou, Y. Huang, H. Zhu, Z. Lu, S.-C. Huang, Y. Yang, and H. Chai, "Distributed DRL-based intelligent over-the-air computation in unmanned aerial vehicle swarm-assisted intelligent transportation system," *IEEE Internet Things J.*, pp. 1–1, 2024.

[12] X. Mao, G. Wu, M. Fan, Z. Cao, and W. Pedrycz, "DL-DRL: A double-level deep reinforcement learning approach for large-scale task scheduling of multi-UAV," *IEEE Trans. Autom. Sci. Eng.*, pp. 1–17, 2024.

[13] B. Hazarika, K. Singh, S. Biswas, and C.-P. Li, "DRL-based resource allocation for computation offloading in IoV networks," *IEEE Trans. Ind. Inform.*, vol. 18, no. 11, pp. 8027–8038, 2022.

[14] B. Hazarika, K. Singh, A. Paul, and T. Q. Duong, "Hybrid machine learning approach for resource allocation of digital twin in UAV-aided internet-of-vehicles networks," *IEEE Trans. Intell. Veh.*, vol. 9, no. 1, pp. 2923–2939, 2024.

[15] P. Singh, B. Hazarika, K. Singh, W.-J. Huang, and C.-P. Li, "Augmented multi-agent DRL for multi-incentive task prioritization in vehicular crowdsensing," *IEEE Internet Things J.*, pp. 1–1, 2024.

[16] Y. Hou, J. Zhao, R. Zhang, X. Cheng, and L. Yang, "UAV swarm cooperative target search: A multi-agent reinforcement learning approach," *IEEE Trans. Intell. Veh.*, vol. 9, no. 1, pp. 568–578, 2024.

[17] X. Dai, Z. Lu, X. Chen, X. Xu, and F. Tang, "Multiagent RL-based joint trajectory scheduling and resource allocation in NOMA-assisted UAV swarm network," *IEEE Internet Things J.*, vol. 11, no. 8, pp. 14 153–14 167, 2024.

[18] B. Hazarika, K. Singh, S. Biswas, S. Mumtaz, and C.-P. Li, "Multi-agent DRL-based task offloading in multiple RIS-aided IoV networks," *IEEE Trans. Veh. Technol.*, vol. 73, no. 1, pp. 1175–1190, 2024.

[19] Y. Xu, L. Zheng, X. Wu, Y. Tang, W. Liu, and D. Sun, "Joint resource allocation for UAV-assisted V2X communication with mean field multi-agent reinforcement learning," *IEEE Trans. Veh. Technol.*, pp. 1–15, 2024.

[20] P. Singh, B. Hazarika, K. Singh, C. Pan, W.-J. Huang, and C.-P. Li, "DRL-based federated learning for efficient vehicular caching management," *IEEE Internet Things J.*, pp. 1–1, 2024.

[21] M. Nie, D. Chen, and D. Wang, "Reinforcement learning on graphs: A survey," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 7, no. 4, pp. 1065–1082, 2023.

[22] C.-L. Liu and T.-H. Huang, "Dynamic job-shop scheduling problems using graph neural network and deep reinforcement learning," *IEEE Trans. Syst. Man Cybern.: Syst.*, vol. 53, no. 11, pp. 6836–6848, 2023.

[23] Y. Xue and W. Chen, "Multi-agent deep reinforcement learning for UAVs navigation in unknown complex environment," *IEEE Trans. Intell. Veh.*, vol. 9, no. 1, pp. 2290–2303, 2024.

[24] Z. Liu, J. Zhang, E. Shi, Z. Liu, D. Niyato, B. Ai, and X. S. Shen, "Graph neural network meets multi-agent reinforcement learning: Fundamentals, applications, and future directions," *IEEE Wirel. Commun.*, pp. 1–9, 2024.

[25] S. Munikoti, D. Agarwal, L. Das, M. Halappanavar, and B. Natarajan, "Challenges and opportunities in deep reinforcement learning with graph neural networks: A comprehensive review of algorithms and applications," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–21, 2023.

[26] B. Hazarika, P. Saikia, K. Singh, and C.-P. Li, "Enhancing vehicular networks with hierarchical O-RAN slicing and federated DRL," *IEEE Trans. Green Commun. Netw.*, vol. 8, no. 3, pp. 1099–1117, 2024.

[27] B. Hazarika and K. Singh, "AFL-DMAAC: Integrated resource management and cooperative caching for URLLC-IoV networks," *IEEE Trans. Intell. Veh.*, vol. 9, no. 6, pp. 5101–5117, 2024.

[28] B. Jiang, J. Du, C. Jiang, Z. Han, A. Alhammadi, and M. Debbah, "Over-the-air federated learning in digital twins empowered UAV swarms," *IEEE Trans. Wirel. Commun.*, pp. 1–1, 2024.

[29] B. Hazarika, K. Singh, T. Q. Duong, and O. A. Dobre, "Quantum-driven context-aware federated learning in heterogeneous vehicular metaverse ecosystem," in *Proc. IEEE ICC 2024*, 2024, pp. 1533–1538.

[30] J. Xu, H. Yao, R. Zhang, T. Mai, S. Huang, and S. Guo, "Federated learning powered semantic communication for UAV swarm cooperation," *IEEE Wirel. Commun.*, vol. 31, no. 4, pp. 140–146, 2024.

[31] K. T. Pauu, J. Wu, Y. Fan, and *et al.*, "Differential privacy and blockchain-empowered decentralized graph federated learning-enabled UAVs for disaster response," *IEEE Internet Things J.*, vol. 11, no. 12, pp. 20 930–20 947, 2024.

[32] J. Tang, J. Nie, Y. Zhang, Z. Xiong, W. Jiang, and M. Guizani, "Multi-UAV-assisted federated learning for energy-aware distributed edge training," *IEEE Trans. Netw. Serv. Manag.*, vol. 21, no. 1, pp. 280–294, 2024.

[33] R. Karmakar, G. Kaddoum, and O. Akhrif, "A novel federated learning-based smart power and 3D trajectory control for fairness optimization in secure UAV-assisted MEC services," *IEEE Trans. Mob. Comput.*, vol. 23, no. 5, pp. 4832–4848, 2024.

[34] Y. Liu, H. Du, D. Niyato, J. Kang, Z. Xiong, D. I. Kim, and A. Jamalipour, "Deep generative model and its applications in efficient wireless network management: A tutorial and case study," *IEEE Wirel. Commun.*, vol. 31, no. 4, pp. 199–207, 2024.

[35] J. Wang, H. Du, D. Niyato, J. Kang, Z. Xiong, D. Rajan, S. Mao, and X. Shen, "A unified framework for guiding generative AI with wireless perception in resource constrained mobile edge networks," *IEEE Trans. Mob. Comput.*, pp. 1–17, 2024.

[36] H. Du, D. Niyato, J. Kang, Z. Xiong, P. Zhang, S. Cui, X. Shen, S. Mao, Z. Han, A. Jamalipour, H. Vincent Poor, and D. I. Kim, "The age of generative AI and AI-generated everything," *IEEE Netw.*, pp. 1–1, 2024.

[37] Z. Tao, W. Xu, Y. Huang, X. Wang, and X. You, "Wireless network digital twin for 6G: Generative AI as a key enabler," *IEEE Wirel. Commun.*, vol. 31, no. 4, pp. 24–31, 2024.

[38] X. Huang, X. Yang, Q. Chen, and J. Zhang, "Task offloading optimization for UAV-assisted fog-enabled internet of things networks," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1082–1094, 2022.

[39] C. You and R. Zhang, "Hybrid offline-online design for UAV-enabled data harvesting in probabilistic los channels," *IEEE Trans. Wirel. Commun.*, vol. 19, no. 6, pp. 3753–3768, 2020.

[40] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Efficient deployment of multiple unmanned aerial vehicles for optimal wireless coverage," *IEEE Commun. Lett.*, vol. 20, no. 8, pp. 1647–1650, 2016.

[41] C. Li, Y. Gan, Y. Zhang, and Y. Luo, "A cooperative computation offloading strategy with on-demand deployment of multi-UAVs in UAV-aided mobile edge computing," *IEEE Trans. Netw. Serv. Manag.*, vol. 21, no. 2, pp. 2095–2110, 2024.

[42] Q. Zhou and G. Liu, "UAV path planning based on the combination of A-star algorithm and RRT-star algorithm," in *Proc. IEEE ICUS 2022*, 2022, pp. 146–151.

[43] G. Sun, W. Xie, D. Niyato, H. Du, J. Kang, J. Wu, S. Sun, and P. Zhang, "Generative AI for advanced UAV networking," *arXiv preprint arXiv:2404.10556*, 2024.

[44] K. Wang, C. Gou, Y. Duan, Y. Lin, X. Zheng, and F.-Y. Wang, "Generative adversarial networks: introduction and outlook," *IEEE/CAA J. Autom. Sin.*, vol. 4, no. 4, pp. 588–598, 2017.

[45] Z. Wang, M. Bennis, and Y. Zhou, "Graph attention-based MADRL for access control and resource allocation in wireless networked control systems," *IEEE Trans. Wirel. Commun.*, pp. 1–1, 2024.

[46] G. Barlacchi, M. De Nadai, R. Larcher, A. Casella, C. Chitic, G. Torrisi, F. Antonelli, A. Vespignani, A. Pentland, and B. Lepri, "A multi-source dataset of urban life in the city of milan and the province of trentino," *Sci. data*, vol. 2, no. 1, pp. 1–15, 2015.