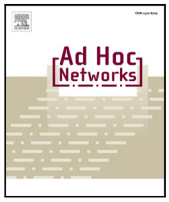
Ad Hoc Networks 160 (2024) 103519

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc



Optimizing UAV-UGV coalition operations: A hybrid clustering and multi-agent reinforcement learning approach for path planning in obstructed environment

Shamyo Brotee [a,1], Farhan Kabir [a,1], Md. Abdur Razzaque [a], Palash Roy [b], Md. Mamun-Or-Rashid [a], Md. Rafiul Hassan [c], Mohammad Mehedi Hassan [d,*]

[a] Green Networking Research Group, Department of Computer Science and Engineering, University of Dhaka, Bangladesh
[b] Department of Computer Science and Engineering, Green University of Bangladesh, Dhaka, Bangladesh
[c] College of Arts and Sciences, University of Maine at Presque Isle, ME 04769, USA
[d] Department of Information Systems, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia

ARTICLE INFO

Keywords:
UAV-UGV coalition
Path planning
Multi-agent deep reinforcement learning
Mean-shift clustering
Obstructed environment

ABSTRACT

One of the most critical applications undertaken by Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs) is reaching predefined targets by following the most time-efficient routes while avoiding collisions. Unfortunately, UAVs are hampered by limited battery life, and UGVs face challenges in reachability due to obstacles and elevation variations, which is why a coalition of UAVs and UGVs can be highly effective. Existing literature primarily focuses on one-to-one coalitions, which constrains the efficiency of reaching targets. In this work, we introduce a novel approach for a UAV-UGV coalition with a variable number of vehicles, employing a modified mean-shift clustering algorithm (MEANCRFT) to segment targets into multiple zones. This approach of assigning targets to various circular zones based on density and range significantly reduces the time required to reach these targets. Moreover, introducing variability in the number of UAVs and UGVs in a coalition enhances task efficiency by enabling simultaneous multi-target engagement. In our approach, each vehicle of the coalitions is trained using two advanced deep reinforcement learning algorithms in two separate experiments, namely Multi-agent Deep Deterministic Policy Gradient (MADDPG) and Multi-agent Proximal Policy Optimization (MAPPO). The results of our experimental evaluation demonstrate that the proposed MEANCRFT-MADDPG method substantially surpasses current state-of-the-art techniques, nearly doubling efficiency in terms of target navigation time and task completion rate.

1. Introduction

The potential of using Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs) in various scenarios has garnered significant attention from researchers in recent years. Regarding revenue, the global autonomous vehicle market for UGVs generated USD 2.24 billion in 2022 and is projected to reach USD 4.08 billion by 2030 [1]. Simultaneously, the market for UAVs is expected to nearly double in value (from USD 24.9 billion to USD 52.3 billion) over eight years, from 2022 to 2030 [2]. Currently, these vehicles are employed in diverse domains, including surveillance, search and rescue, inspection, inventory checking, mapping unknown terrains, and planetary exploration [3]. This has led researchers to explore deploying various UAV-UGV combinations for complex tasks like navigation in environments with obstructions. The collaborative UAV-UGV systems have shown promise in intelligence surveillance, reconnaissance, object localization, and crucially, path planning in obstructed environments.

UAVs offer the advantage of high altitude and a broad line of sight, while UGVs have longer battery duration and can execute precise operations on ground objects. However, both UAVs and UGVs face significant limitations. UGVs are constrained by limited vertical reach, and UAVs are hampered by a limited power supply, leading to restricted operational range and duration [4,5]. Collaborations between UAVs and UGVs can significantly enhance task execution efficiency by overcoming these individual limitations. In collaborative systems,

* Corresponding author.
E-mail addresses: shamyo-2017214977@cs.du.ac.bd (S. Brotee), farhan-2017414975@cs.du.ac.bd (F. Kabir), razzaque@du.ac.bd (M.A. Razzaque), palash@cse.green.edu.bd (P. Roy), mamun@cse.univdhaka.edu (M. Mamun-Or-Rashid), md.hassan@maine.edu (M.R. Hassan), mmhassan@ksu.edu.sa (M.M. Hassan).
[1] Authors contributed equally.

https://doi.org/10.1016/j.adhoc.2024.103519
Received 1 December 2023; Received in revised form 18 March 2024; Accepted 14 April 2024
Available online 17 April 2024

UAVs can access destinations beyond the reach of UGVs. Conversely, UGVs can transport UAVs to conserve their battery life and are better equipped to handle ground objects with greater payload capacity. A homogeneous swarm of UAVs or UGVs often struggles with distant and time-intensive missions, particularly in mission-critical applications. In complex, obstructed environments, a robust UGV can autonomously transport a UAV beyond its flight range, supplying power and substantially increasing its operational range. In turn, the UAV assists the UGV by accessing high-altitude destinations with fewer collision risks. However, integrating multiple UAVs and UGVs for efficient path planning in obstructed terrains poses significant engineering challenges.

Recent research has explored deploying multiple UAVs or UGVs for specific complex tasks. A study proposed a multi-UAV homogeneous system utilizing the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm for optimized target assignment and collision-free path planning [6]. However, this approach does not incorporate UGVs, and many recent works focus on such homogeneous systems. Zhang et al. introduced an Imitation Augmented Deep Reinforcement Learning (IADRL) approach for a UGV-UAV coalition, where both vehicle types complement each other in performing difficult tasks [7]. Yet, their methodology is limited to one UAV and one UGV system while addressing UAV-UGV coalitions in terrains with obstacles. Although a collaboration of UAVs and UGVs solves the problems of limited reachability of UGVs and limited battery of UAVs, in general, such a coalition comes with certain challenges. The huge computational cost and the lack of a proper framework to establish communication and collaboration between those agents are some of the major problems. Existing works on such coalitions are usually not tailored to create a suitable framework for the UAV-UGV coalition to carry out complex path-planning tasks that reduce computational costs and ensure proper communication among multiple agents. They are also usually inflexible regarding the number of UAVs and UGVs in a coalition since only one UAV and one UGV comprise the coalitions in most cases. In path-planning tasks with randomly distributed predefined targets, reaching all the targets with minimal collisions and time is essential. These observations have motivated us to develop an efficient framework to effectively enhance collaboration among multiple UAVs and UGVs in obstructed environments by developing a novel approach to mitigate these problems.

In this paper, we introduce a novel approach, **MEANCRFT**, for one-to-many collaboration among UGVs and UAVs, utilizing **MEAN**-Shift clustering to deploy UAV-UGV **C**oalitions using **R**einforcement learning **F**or reaching aerial and ground **T**argets. The MEANCRFT employs heuristic methods to segment target locations into zones, followed by individualized training for vehicles to navigate efficiently in an obstructed environment. This model enables the variability in the count of UAVs and UGVs in a single coalition. This approach mainly aims to find fast and collision-free paths toward multiple ground and aerial targets, exploiting the collaborative capabilities of UAVs and UGVs. The use of heuristics in the MEANCRFT simplifies the zonal segmentation of target locations and the complexity of the path-planning task. To address the challenges of reaching all targets in an obstructed environment while minimizing time steps, avoiding collisions and ensuring task completion, we have employed different Multi-Agent Deep Reinforcement Learning or MADRL frameworks to train UAV-UGV coalitions for efficient path planning. Ultimately, the goal of the approach is to reach all the targets within a limited time delay and minimal number of collisions among unmanned aerial or ground vehicles.

Our work introduces novel contributions to the field of path planning through the innovative dimension of a UAV-UGV coalition. The primary contributions are:

- We have developed a modified mean-shift clustering algorithm that assigns multiple targets to different zones based on target density. The clustering also assigns multiple UAVs and UGVs within each zone to address complex path-planning challenges. Using target density as a decisive factor enables our approach to reduce coalition travel time.
- The variability of the UAV-UGV coalition has been enhanced through extended training governed by advanced multi-agent deep reinforcement learning algorithms, which in turn helps to avoid obstacles and increase system robustness.
- Extensive numerical analysis on a custom environment created using OpenAI's gym platform, shows that the developed MEANCRFT-MADDPG significantly reduces the number of steps to complete tasks and increases task completion rates compared to state-of-the-art models.

This paper is systematically organized: Section 2 reviews relevant literature on the UAV-UGV coalition, Section 3 discusses the system model and assumptions, Section 4 details the proposed approach, Section 5 describes the experiment and its results, and Section 6 concludes the paper with a summary and future research directions.

2. Related work

UAVs and UGVs have seen an unprecedented rise in a multitude of fields, notably path planning, and search and rescue missions. However, due to the limited battery of UAVs and the limited reachability of UGVs, both types of vehicles are restrained in their movement in many path planning and target-reaching scenarios. Many researchers have proposed several approaches to solve this complex path-planning problem for a smart implementation of a functional UAV-UGV coalition.

Bellingham et al. [8] have developed a novel mathematical approach based on Mixed-Integer Linear Programming (MILP) that designs nearly optimal routes toward a goal for multiple UAVs with constraints like no-fly zones and the highest speed and turning rate of the UAVs. Based on genetic algorithms, Shima et al. [9] solved this problem, having special considerations including task prioritization, coordination, time restraints, and flying trajectories. An approach based on particle swarm optimization (PSO) principles was presented by Cruz et al. [10]. In recent years, Babel et al. tackled the issue of cooperative flight route planning while minimizing the overall mission time, provided that the UAVs arrive at their destinations simultaneously or sequentially with predetermined time delays [11].

A computationally effective mathematical model based on cell-winning suppression (CWS) was developed by J. Liu et al. for multi-target task assignment in a complex battlefield environment with attacking and defensive UAVs on two sides [12]. Yan et al. [13] proposed a path-planning algorithm for the UAVs based on improved Q-learning in an antagonistic environment. A multi-UAV-enabled homogeneous system to optimize target assignment and path planning has been proposed in [6]. The authors in [14] have dealt with the path planning problem for a ground target tracking scenario through an obstructed environment by using the line of sight and artificial potential field to construct their reward. Another group of authors in [15] introduced a method to implement local online path planning in unknown rough terrain by a UGV. However, the aforementioned works in literature have only considered homogeneous vehicles (Only UAVs or only UGVs) in their respective field. A collaborative approach to heterogeneous vehicles has been ignored in these studies.

To tackle these multiple-agent target assignment and path planning problems, different deep reinforcement learning (DRL) frameworks have been used in the existing literature. Reinforcement learning refers to goal-oriented algorithms that help to attain objectives along a particular dimension over repetitive steps by being rewarded or punished for good and bad actions, respectively. Usually, a Q-table is maintained to store the state–action pairs in the Q-learning, a basic form of reinforcement learning. When neural networks are associated with reinforcement learning, it is called DRL [16]. Deep Q-Network (DQN) is an early implementation of DRL, that approximates a state-value function in a Q-Learning framework (Basic reinforcement learning) with a neural network [17]. Consequently, various improvements have come in recent years in the field of DRL [18]. Deep Deterministic Policy Gradient (DDPG), Multi-agent DDPG (MADDPG), Proximal Policy

Optimization (PPO), and Multi-agent PPO (MAPPO) are the state-of-the-art additions in this field. DDPG is an actor-critic, model-free algorithm based on the deterministic policy gradient that can operate over continuous action spaces. It combines the actor-critic approach with insights from DQN. MADDPG can be considered just the multi-agent version of DDPG [19], and it is considered a state-of-the-art algorithm in the multi-agent reinforcement learning field. On the other hand, PPO is a policy gradient method for reinforcement learning [20], whereas, MAPPO is essentially the multi-agent version of this [21–23].

Qie et al. [6] have proposed a multi-UAV-enabled homogeneous system to optimize target assignment and path planning using a MADDPG algorithm, which has achieved excellent performance. However, their developed system is only applicable to homogeneous UAVs. Another group of authors in [24] have used a modified K-means clustering algorithm and MAPPO for trajectory design in disaster areas with homogeneous UAVs only. Li et al. [25] have also used PSO-based clustering along with a MAPPO-based algorithm to design a trajectory planning algorithm for situations that require urgent communication. In path planning problems, using a clustering method to manage the targets or destination areas better is a great way to improve performance. Chen et al. [26,27] have exploited density-based clustering methods to get improved performances in coverage path planning problems with UAVs. These studies also do not consider UGVs as a coalition approach to solving path-planning problems.

Extensive research on coalitions of UAVs and UGVs is a relatively new venture. Ropero et al. designed a UAV-UGV collaborative system for planetary exploration where the goal is to reach a set of target points while minimizing the traveling distance [28]. In a UAV-UGV coalition, UAVs play a major part in planning the path and guiding the collaborative system. Lee et al. proposed a solution to a problem where a UAV shepherds a swarm of UGVs [29]. Zhang et al. used Imitation Augmented Deep Reinforcement Learning (IADRL), which is a mix of Imitation Learning and DRL, to address the issue of the enormous complexity of UAV-UGV collaboration [7]. In their approach, a UGV carries a UAV up to the point where it is unable to navigate vertically to reach the target. The UAV can then be launched to do the assignment and arrive at the destination. To increase the operating range of the UAV, the UGV also serves as a charging station. In their work, they assumed their environment without any obstacles and assigned one UAV to a single UGV.

Although the above-discussed works have helped improve and overcome a lot of challenges in combined hybrid UAV-UGV systems, there are still some limitations that need to be addressed. Most of the existing works have not considered a collaboration of multiple UAVs and UGVs in an obstructed environment, which enhances the variability of vehicles and the robustness of the system. These observations have motivated us to develop a framework, namely MEANCRFT, to enhance collaboration between multiple UAVs and UGVs in an obstructed environment.

3. System model of MEANCRFT

This section provides information about the proposed system along with a detailed depiction of the terms and variables used in it. Fig. 1(a) depicts the 2D model of different components and their relationship to the developed model along with a 3D representation in Fig. 1(b).

In the proposed 2D environment, UAVs and UGVs navigate to reach the targets by avoiding obstacles, where the set of UAVs and UGVs is denoted by $\mathcal{A}$ and $\mathcal{G}$, respectively. Each aerial target is represented by $p \in \mathcal{P}$ and each grounded target is represented by $w \in \mathcal{W}$, where, $\mathcal{P}$ and $\mathcal{W}$ are the set of the positions of all aerial and grounded targets within a zone. The position of each aerial and grounded target can be denoted by (x_p, y_p), and (x_w, y_w), respectively. The position of a UAV at any time step t is represented by $(x_a(t), y_a(t))$, where, $a \in \mathcal{A}$ and the position of a UGV at any time step t is $(x_g(t), y_g(t))$, where, $g \in \mathcal{G}$. The set of obstacles is denoted by B and each obstacle position is represented by (x_b, y_b), where, $b \in B$. Let $F_a = \{(x_a(0), y_a(0)), (x_a(1), y_a(1)), \ldots, (x_a(n), y_a(n))\}$ be the path of a UAV in a zone, and $L_g = \{(x_g(0), y_g(0)), (x_g(1), y_g(1)), \ldots, (x_g(n), y_g(n))\}$ be the path of a UGV in a zone, where, n is the number of time steps in an episode.

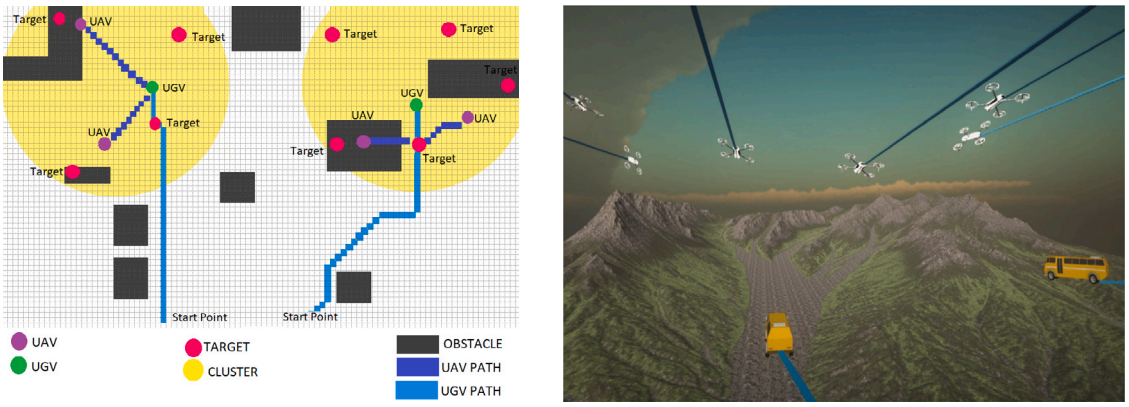
Table 1
Notations and their meaning.

Notation	Description
$\mathcal{A}$	The set of center points of the unmanned aerial vehicles (UAVs)
$\mathcal{G}$	The set of center points of the unmanned ground vehicles (UGVs)
$\mathcal{B}$	The set of center points of the obstacles
$\mathcal{P}$	The set of center points of the aerial targets
$\mathcal{W}$	The set of center points of the grounded targets
F_a	The set of grid points traveled by a UAV $a \in \mathcal{A}$ to reach a target
L_g	The set of grid points traveled by a UGV $g \in \mathcal{G}$ to reach a target
μ_i	Deterministic policy of agent i
θ	Set of policy parameters
$\nabla_{\theta_i} J(\mu_i)$	The gradient of μ_i
$\mathcal{L}(\theta_i)$	The loss function
$\mathcal{D}$	Experience Replay

As depicted in Fig. 1(a), the developed framework, MEANCRFT, has two main functional parts. At first, the entire area of deployment is divided into zones based on the density of the targets exploiting a modified mean-shift clustering algorithm. In Fig. 1(a), we can see that the targets on the map are divided into two zones or clusters. We assume that UAVs and UGVs are deployed in the environment to reach targets. Two different coalitions are sent to each of the zones. After reaching a zone, UGVs deploy the UAVs that have been trained to reach the assigned aerial targets while the UGVs get to ground targets. We need the UAVs to be within a certain range of the UGVs so that they can return to the UGV in due time. For this purpose, we set the zones' radius R based on the UAVs' flying range V. They navigate in the environment avoiding collisions with obstacles and other vehicles. After reaching, the UAVs return to their nearest UGV and the coalitions move to the next zone when the assigned zone has been cleared. In this way, the developed model can cover all the targets receiving minimal damage within a short amount of time. A widespread application of our work can be a post-earthquake rescue mission in a collapsed building, where all the means of ground entry are blocked. In this scenario, the developed model can efficiently plan a path and reach the location of inhabitants who require help. In such an application, high task efficiency and minimal time are of utmost importance, which our model achieves. We have also assumed that the positions of the targets were unchanging. Within a zone, the agents are informed about the position of their reachable targets, the position of the obstacles, and the position of other agents. Moreover, in the 2D environment, the UAVs do not collide with the UGVs as they operate at a higher altitude. Table 1 summarizes the major notations used in this paper.

4. Design details of MEANCRFT

In this section, we provide a comprehensive explanation of our coalition approach to reach targets in an obstructed environment. At first, we will delve into a detailed explanation of the methodology for dividing the targets into multiple circular zones using a heuristic based on a modified version of mean-shift clustering to deploy a specific number of coalitions into those zones for planning a safe route to reach all those targets. In the latter phase, we briefly discuss the two different MADRL models, namely MADDPG and MAPPO, that we used to train our coalitions for efficient path planning and obstacle avoidance.



(a) 2D grid

(b) 3D representation

Fig. 1. Visual representation of the relationships between key elements of UAV-UGV enabled network.

4.1. Zone division: Assignment of targets into zones

The first step of our approach is to divide the targets P into suitable zones. For the selection of zones, we have designed a modified version of the mean-shift clustering algorithm that divides the overall area into circular zones based on target density. Coalitions travel from zone to zone, clearing the targets in them. Algorithm 1 gives a formal depiction of the proposed algorithm for the division and assignment of the targets into zones. It takes as input a radius value R, which is the flight range of the UAVs in our system, and the set of points P in the environment. As output, it gives the set of center points of all the circular zones it calculated, C.

4.1.1. Mean-shift clustering

The generic Mean-Shift clustering [30] is a centroid-based sliding window clustering algorithm focused on finding out the center point of each cluster. It has been implemented in Algorithm 1. Mean-shift clustering suits our purpose for mainly two reasons: (1) It is a density-based approach since we also want to detect the areas with the highest target density to create the zones, and (2) It creates a zone with a certain radius. This algorithm chooses an initial point and creates a sliding window with a specific diameter. This diameter in the model is a function of the UAV flight range. It works by updating candidates for center points to be the mean of the points within the sliding window.

By moving the center point to the average of the window's points at each iteration, the sliding window is moved toward areas with higher densities. The number of points inside the sliding window determines how dense it is. Naturally, it will eventually gravitate toward locations with larger point densities by adjusting to the window's mean of the points. The sliding window is moved following the mean of the points inside it and it keeps moving until there is no longer any direction in which a shift can accommodate any center point for the window. This process is continued until all the points fall inside a cluster. As Mean-Shift Clustering is used for identifying clusters, there can be situations where multiple windows overlap. In that case, the sliding window with the most points is kept. The data points are then assigned based on the sliding window that contains them.

4.1.2. Modified mean-shift clustering

The developed Algorithm 1 is based on a modified version of the aforementioned Mean-Shift clustering algorithm. Firstly, we take the radius of the zones R and the list of the targets' coordinates P in the environment as inputs. The radius of the zones is a preset value that

Algorithm 1 Modified Mean-Shift Clustering

Input: $R \longleftarrow$ Radius of UAV Zones, $P \longleftarrow$ Set of points in the environment
Output: $C \longleftarrow$ Center of Zones

$C \longleftarrow \emptyset$
while there are points in P **do**
  $Q = P$
  $T \longleftarrow$ Constant
  $k \longleftarrow$ Constant
  while k **do**
    $U \longleftarrow \emptyset$
    for $q \in Q$ **do**
      $Z \longleftarrow \emptyset$
      for $q' \in Q$ **do**
        if $distance(q, q') \leq R$ **then**
          $Z \leftarrow Z \cup q'$
        end if
      end for
      $V = mean(Z)$
      $U \leftarrow U \cup V$
    end for
    $M = mean(U - Q)$
    if $M \leq T$ **then**
      break
    end if
    $Q = U$
    $k \leftarrow k + 1$
  end while
  Initialize G as empty set
  $G \leftarrow G \cup q, \forall q \in Q$
  $C \leftarrow C \cup \{g\}, \forall g \in G$
  for each $p \in P$ **do**
    for each $c \in C$ **do**
      if $distance(c, p) \leq R$ **then**
        $P.delete(p)$
      end if
    end for
  end for
end while
return C

is dependent on the functional range of the drones. Here, T and k are constant values that are set to accept cluster centers within a specific range and Q is a copy of the list of data points on which it will perform the clustering. We use Mean-Shift clustering from the lines 3 to 26 which returns the centers of the densest zones in the environment that can form a cluster.

First, an empty list U is taken which will be used to store the center of the windows. Next, we take a random point as the center of a window and determine the points inside it. We assign these points in a list Z and then determine the centroid of Z. This is the current center of the window which is saved in U. This process is continued till all the points are inside a window, the center of which is in U. Q is a list that is used to contain the center of the window in which a point resides. Initially, it is the point itself, and as the process continues, its value is updated (line 22). This process is repeated at most k times to constantly update Q for each point in the given dataset. In each update, the value of the window center can change. The magnitude of this change is determined in line 19. If this value is more than the threshold value T, it means that the window has moved significantly. Otherwise, the shift is not too significant and the current centers are used as the final cluster centers. Finally, after finishing, the non-repeating center values are used as the cluster centers. We add them to a list C in line 27 and use it to find out all the coordinates that are inside a zone by comparing their distances with the center points stored in C. Then in line 31, we delete the points from the coordinate list P that are within R distance from each element in C since they are already assigned to a zone. We keep repeating this process of Mean-Shift clustering and assigning the points into zones until all the points in P are assigned properly into a zone. This calculates new center points and the process continues until no points are left unassigned and all the center points are in the list C. This is how our proposed heuristic works, based on a densest-area-first approach.

In a typical scenario, when a significant number of targets are available, going from one target to another for a coalition becomes much less efficient when we consider the number of times a UAV has to leave a UGV, reach a target, and return to the UGV. As the number of aerial targets can vary, selecting the nearest aerial target and moving on to the next one requires a lot of time and battery. Our Modified Mean-Shift Clustering approach saves a huge amount of time and battery by dividing the available targets into zones based on density, serving as a great advantage to our proposed solution. Each of these zones has a radius R based on the flight range V of a UAV. This allows the coalitions to reach specific zones and deploy UAVs where they can freely move around reaching available targets without unnecessarily returning to UGVs. This increases the number of targets that are reached while simultaneously decreasing the time required to finish the tasks.

We have designed this zoning algorithm because we want to use the battery capacity and flight range V of the UAV as decisive factors for choosing our zones. As every zone has to be covered by the coalitions to reach all available targets in the environment, dividing the targets based on density reduces the overall zone count. Again, as the number of UAVs in a coalition can be variable, we ensure maximum target-reaching efficiency within a zone by sending maximum available agents in it to reduce overall time.

After dividing the target areas into zones, one or multiple coalitions are sent to each zone based on the requirements. When a coalition enters a zone, the UAVs can be deployed to reach the aerial targets by avoiding obstacles and other agents. In the meantime, the UGV in the coalition can navigate through the obstructed terrains and reach the ground targets. When the battery of a UAV reaches a critical level, the target assignment for the UAV will stop and it will immediately land on the nearest available UGV. This ensures maximum utilization of the battery life of the UAV and also clears the zones rapidly.

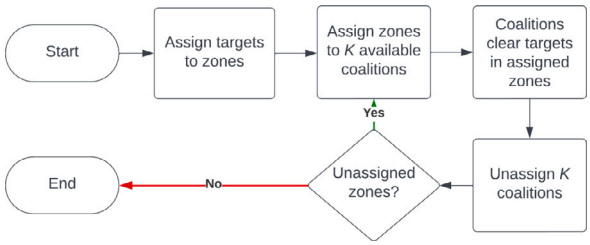


Fig. 2. Flowchart of the proposed solution framework.

4.2. Assignment of coalition to zones

After grouping all the targets into various circular zones, we deploy MADRL-trained coalitions in each of those zones to reach the targets while avoiding collisions. The sequential steps for our solution to assign the coalitions into zones are depicted in Fig. 2 and the step-by-step procedure can be described as follows,

Step 1: At first, the number of coalitions, K are sent to a zone.
Step 2: After that, the trained models of UAVs and UGVs are deployed to cover the targets. A coalition is considered available only when all the deployed UAVs have landed on the available UGVs after clearing the zone.
Step 3: When a zone is cleared, the previously assigned K coalitions are available, which are then assigned to the next target zone.
Step 4: Steps 2 and 3 are repeated until all the zones have been cleared, and subsequently all the targets are reached.

The coalitions will follow a specific set of steps when they reach the zone that they are assigned to: the UGVs will navigate to reach the ground targets after deploying UAVs. The UAVs will also roam to reach the aerial targets. During this process, if a UAV needs to land, it will signal its nearest UGV. The UGV will pause its current activity, wait for the UAV to land, and after landing, resume its function.

It should be noted that we have trained the UGVs first as they require less complex inputs for training. This is because the only function a UGV has is to reach its assigned target. For actions such as UAV landing, it only needs to pause its function which does not require training. On the other hand, UAVs need to reach their assigned targets and return to their nearest UGV. This makes the task more intricate and also requires the information of the nearest UGV in their observation space. This is why, we first train the UGVs and use this trained model to get some UGV coordinates in an environment. Next, in the UAV training scenario, we used these demo coordinates as input in the observation space of the UAV. This way, the first half of the UAV training is to reach the assigned target. The next half is to "land" on the nearest UGV, the position of which is provided by our demo data. In this way, we divided our training into two parts and used our UGV-trained model to train our UAVs. Ultimately, these two separate training models are combined in the full simulation environment.

There, the available targets are first divided into zones. Then only the trained models for UGVs are used to move coalitions to their assigned zones. After reaching the zones, the UGVs deploy UAVs, and together, they move in the environment to reach all the targets inside the zone. The UGVs will navigate in the environment to reach their assigned targets, avoiding obstacles and other UGVs, while at the same time, UAVs will also move around the environment, avoiding obstacles and other UAVs to reach assigned aerial targets. When UAVs are done reaching their respective targets, they move to their nearest UGV location. The selected UGV for landing pauses its model until the UAV lands. After that, it resumes its model to reach ground targets.

4.3. Constraints for MADRL training

From Table 1, we know that F_a and L_g are the path lengths of UAV $a \in \mathcal{A}$ and UGV $g \in \mathcal{G}$, respectively. The goal of this work is to minimize these path lengths while keeping the journey of the vehicle safe and free from collisions, which can be denoted as,

$$min(\sum_{a\in\mathcal{A}} F_a + \sum_{g\in\mathcal{G}} L_g). \tag{1}$$

There are certain constraints that we need to follow while achieving the sole objective. The following constraints are set that need to be satisfied:

$$(x_p, y_p) \in F_a, \quad \forall p \in \mathcal{P}, \quad \exists a \in \mathcal{A} \tag{2}$$

$$(x_w, y_w) \in L_g, \quad \forall w \in \mathcal{W}, \quad \exists g \in \mathcal{G} \tag{3}$$

$$(x_b, y_b) \notin F_a, \quad \forall b \in \mathcal{B}, \quad \forall a \in \mathcal{A} \tag{4}$$

$$(x_b, y_b) \notin L_g, \quad \forall b \in \mathcal{B}, \quad \forall g \in \mathcal{G} \tag{5}$$

$$(x_a(t), y_a(t)) \neq (x_{a'}(t), x_{a'}(t)), \quad \forall a, a' \in \mathcal{A} \tag{6}$$

$$(x_g(t), y_g(t)) \neq (x_{g'}(t), x_{g'}(t)), \quad \forall g, g' \in \mathcal{G} \tag{7}$$

$$\forall a, (x_a(n), y_a(n)) = (x_g(n), y_g(n)) \tag{8}$$

The goal of this work is to minimize the sum of all the UAVs' path length F_a, and all UGV's path length L_g, as depicted in Eq. (1). At a given time, each UAV and UGV is assigned one aerial target and one grounded target, respectively. Constraint in Eq. (2) specifies that each aerial target's position (x_p, y_p) is covered by a UAV's path F_a, and similarly, Eq. (3) signifies that every ground target (x_p, y_p) is covered by a UGV's path L_g. Moreover, the path of a UAV and a UGV, denoted by F_a and L_g, respectively, is free from collisions with any obstacle (x_b, y_b), as denoted in constraints of Eq. (4) and (5). Constraints (6) and (7) denote that each UAV $a \in \mathcal{A}$ must not collide with another UAV $a' \in \mathcal{A}$ and each UGV $g \in \mathcal{G}$ must not collide with another UGV $g' \in \mathcal{G}$, respectively. The constraint in Eq. (8) specifies that a UAV must come back to its nearest UGV and have the same position as the UGV at the nth step, which is the final step of an episode.

4.4. MADDPG framework

In reinforcement learning, the dynamic problem is defined by the Markov Decision Process (MDP). An MDP model is formally defined with five components: state, action, reward, transition probability, and discount factor. When an agent acts in a single-agent scenario, it receives a reward and traverses to a new state from the environment which is determined by a transition probability. The environment is typically modeled as an MDP, whose solution means identifying a policy that maps from the state space to a distribution across the action space to maximize the cumulative reward. A key assumption in this situation is a stationary environment. Since each agent's reward and state transitions are influenced by the actions of other agents, this premise cannot be applied to multi-agent frameworks. In addition, every agent must maximize its cumulative reward, which is determined by the other agents' policies. Hence, multi-agent versions of the popular algorithms are required. Deep Q-Network (DQN) is an early implementation of deep reinforcement learning (DRL), that approximates a state-value function in a Q-Learning framework with a neural network. Consequently, DDPG and Multi-agent DDPG (MADDPG) are two of the state-of-the-art additions in this field. The DDPG is an actor-critic, model-free algorithm based on the deterministic policy gradient that can operate over continuous action spaces. It combines the actor-critic approach with insights from DQN.

A MADDPG algorithm can be used to solve target assignment and path planning problems and it is considered as a multi-agent variation of DDPG [19] in the multi-agent reinforcement learning field. Its core concept is to centralize training while decentralizing execution (CTDE). Because they do not utilize the information of other agents, the deep Q-learning network (DQN) and DDPG perform poorly in multi-agent systems. The MADDPG method gets around this problem by relying on the observations and actions of other agents. There are two networks for each agent: an actor network and a critic network. The actor network calculates the action to be taken based on the agent's current state, whereas the critic network assesses the action taken by the actor network in order to improve the actor network's performance. An experience replay buffer is employed to retain a set amount of training experience and random reads are performed on it to update the network. This is done to break correlations in the training data and avoid divergence. The actor-network only gets observation information during the training phase, however, the critic network gets information from other agents' actions and observations. The critic network is only involved in the training phase, and after training, every agent uses only the actor-network. As the actors are not connected to each other and guide the agents individually in the execution phase, decentralized execution is achieved.

4.4.1. State action representation

In the developed system, we consider each of the vehicles to be an agent. The agents in each of the training phases have an action space and a state space. The action space of each UGV and UAV consists of its velocities along both the X and Y axes in a geometric system. Therefore, the action space of UAV $a \in \mathcal{A}$ can be defined as $\xi_a = (u_a, v_a)$; and the action space of UGV $g \in \mathcal{G}$ as $\xi_g = (u_g, v_g)$. The state space of each UGV and each UAV is denoted by o_g and o_a, respectively. The state space of a specific UGV agent o_g consists of the relative position of all the grounded targets in the environment, the relative position of the obstacles in the environment, the agent's own velocity as well as its own position, and the relative position of other UGVs in respect to the agent. Similarly, in the case of UAVs, the state space of a specific UAV agent o_a consists of the relative position of all the targets and obstacles, the relative positions of other UAVs with respect to the agent, and the agent's own velocity and position. However, as UAVs require a position to land after reaching targets; therefore, the relative position of its nearest UGV is also added, along with a binary variable that indicates whether the UAV has reached its assigned target or not. This value is 0 when the target has not been reached and it is 1 when the UAV reaches the target. This informs the UAV that it has reached its assigned target and now it can start the process of landing on a UGV. Both of these are additions to a UAV agent's state space. These inputs are used to train the models in separate environments. After the training phases are done, the trained models are deployed in a combined environment where both the UGVs and UAVs collaborate to reach the targets in the obstructed environment.

4.4.2. Training process

In each training scenario, there are N agents. Each agent has its own deterministic policy μ_i. The policies are parameterized by $\theta = \{\theta_1, \theta_2, \dots, \theta_N\}$. For an agent i, the gradient of the deterministic policy μ_i can be written as

$$\nabla_{\theta_i} J(\mu_i) = \mathbb{E}_{x,a\sim D}[\nabla_{\theta_i} \mu_i(a_i|o_i) \nabla_{a_i} Q_i^{\mu}(x, a_i, a_2, \dots, a_N)|_{a_i=\mu_i(o_i)}], \tag{9}$$

where, x is the state space which is equal to $\{o_1, o_2, \dots, o_N\}$, $Q_I^{\mu}(x, a_1, a_2, \dots, a_N)|_{a_i=\mu_i(o_i)}$ is the $Q - value$ function, a_i is the action of the agent i, o_i is the observation of each agent i, D is the experience replay where each tuple is $(x, x', a_1, \dots, a_N, r_1, \dots, r_N)$ as a record of the experiences of all agents. After the actions of all agents have been executed, x' is used to represent the new state of the environment. r_i is the reward of agent i. The loss function $\mathcal{L}(\theta_i)$ is given by

$$\mathcal{L}(\theta_i) = \mathbb{E}_{x,a,r,x'}[(Q_i^{\mu}(x, a_i, a_2, \dots, a_N) - y)^2], \tag{10}$$

Algorithm 2 MADDPG pseudo-code

$E \longleftarrow$ Number of episodes
$T \longleftarrow$ Number of time steps
$N \longleftarrow$ Number of agents
Initialize replay buffer D
for $Episode = 1$ to E **do**
  Initialize a random process $\mathcal{M}$ for action exploration
  Receive initial state x
  for $t = 0$ to T **do**
    for $i = 1$ to N **do**
      select action $a_i = \mu_\theta(o) + \mathcal{M}_t$
    end for
  end for
  Execute actions $a = (a_1, \ldots, a_N)$ and observe reward r and new state x'
  Store (x, a, r, x') in replay buffer D
  for $i = 1$ to N **do**
    Sample a random minibatch of S samples $(x, a, r, x^{'})$ from D
    Set $y = r_i + \gamma Q_i^{\mu'}(x', a'_1, \ldots, a'_N)|_{a'_k = \mu'_k(q_k)}$
    Update critic of agent i by minimizing the loss using Eq. (10)
    Update actor of agent i using the sampled policy gradient via Eq. (9)
  end for
  Update target network parameters for agent i:
$$\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$$
end for

where,

$$y = r_i + \gamma Q_i^{\mu'}(x', a'_i, \ldots, a'_N)|_{a'_j = \mu'_j(o_j)}. \tag{11}$$

The value of $\mathcal{L}(\theta_i)$ is used to update the critic network, Q_i^{μ}, and it is in turn helping to train the model.

4.4.3. Reward calculation

For training the vehicles in their respective environments, a multi-agent deep reinforcement learning setup has been exploited. In such setups, appropriate reward structures are necessary in order to get acceptable behavior from agents, i.e., the UAVs and UGVs in their respective environments. In the developed UAV and UGV environments, multiple homogeneous agents are trained to reach the available targets in an acceptable manner. To attain such outputs, we have to come up with a reward structure that encourages multiple agents to reach their targets efficiently. At the same time, the agents also need to avoid any obstacle collisions while on their way to the targets. For this, the assigned reward value for each agent is divided into some components which are affected by some parameters of the environment and are different for each environment type. For encouraging the agents to reach targets, the distance between agents and targets is considered. For collision avoidance, the agent-agent distance as well as the agent-obstacle distance is also considered. It should be noted that for various reward components, a function $T_i(.)$ is used which accepts distance as input and returns a reward value.

4.4.3.1. UGV reward components. The detailed calculation of different reward component of the UGV reward structure can be determined as follows,

(a) Reward Based on Distance to Targets: In each type of environment, there can be multiple targets with multiple agents where the agents together need to find a way to reach all the available targets efficiently. To do so, a reward component r_1 is assigned based on the distance between all targets and agents. Specifically, the minimum distance between available targets and agents is taken into consideration. To assign such a reward, at first, the distance between all UGV agents to each ground target w is calculated and stored in a set d_w. From all these values, the minimum distance, $\min(d_w)$ of each ground target w to its closest UGV is taken. After calculating, we sum the minimum values and assign this negative reward component to each agent.

$$r_1 = r_1 + T_1(-min(d_w)), \quad \forall w \in \mathcal{W} \tag{12}$$

(b) Reward Based on Agent-Agent Collision: While navigating the environment, the agents must avoid colliding with other agents. For this, a reward component, r_2 has been derived, where the agents are punished for being within collision distance of each other.
Here, we have considered an area around each agent and named it **Danger Zone**, whose width is σ_V and δ_V is the maximum distance between agents during which collision happens. Fig. 3 shows an example scenario of the danger zones for agents. Whenever the danger zones of two agents overlap or collide, i.e., the distance between agents, $d_V < (\delta_V + \sigma_V)$, we call it fake collision. A negative reward is given to the fake colliding agents based on the size of the overlap area. The more the overlapping area, the more the magnitude of the negative reward. If agents actually collide, the agents get the maximum negative reward value and the collision information is recorded for evaluation. The reward value can be estimated as follows,

$$r_2 = r_2 - T_2(\delta_V + \sigma_V - d_V) \tag{13}$$

(c) Reward Based on Vehicle-Obstacle Collision: Another reward component, r_3 is taken that considers the collision between agents and obstacles. It works similarly to the agent-agent collision component where, like agents, obstacles also have a danger zone area σ_O around them, as depicted in Fig. 4. The maximum distance between an agent and an obstacle when a collision happens is δ_O. When danger zones overlap, i.e. the distance between an agent and obstacle, $d_O < (\delta_O + \sigma_O)$, a negative reward is added to the agent colliding. This value also increases as the length of the overlapping area increases. This reward value. r_3 can be determined as follows,

$$r_3 = r_3 - T_3(\delta_O + \sigma_O - d_O) \tag{14}$$

After calculating each of the reward components for each UGV agent, their sum is the final guiding reward for navigating the agents through obstructed terrain to reach targets. The reward function for the UGV, r_G can be estimated as follows,

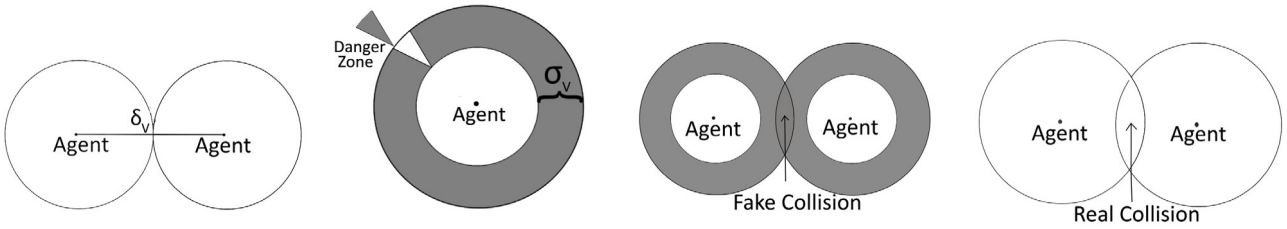
$$r_G = r_1 + r_2 + r_3 \tag{15}$$

This reward is assigned to each UGV using the MADDPG framework.

4.4.3.2. UAV reward components. The training environments are divided based on the type of vehicle that is being trained. However, due to the similarities in the training environment of UAVs and UGVs, the reward structure for the UAVs is almost similar to the UGV's reward structure. The UAV agent reward is divided into some components. The description and the method of calculation of these components are given below.

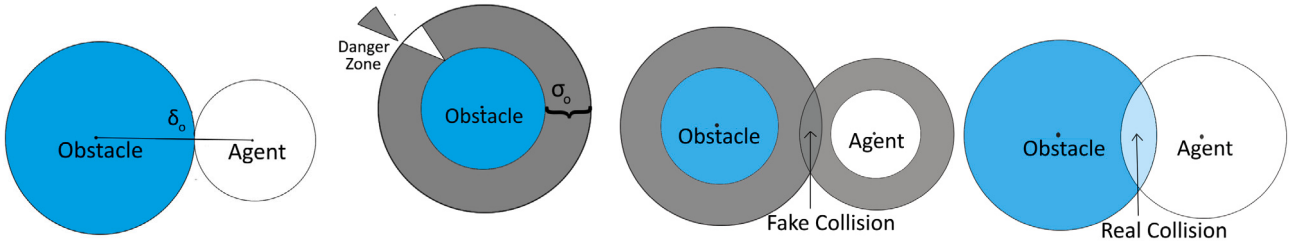
(a) Reward Based on Distance to Targets: Similar to the UGV environment, we have a target encouragement reward component r_4 which sums the distance of each target, $p \in \mathcal{P}$ to its closest agent. This distance value is converted to a negative reward which forms the reward component. It should be noted that for the UAV scenario, when a UAV reaches its assigned target, r_4 becomes zero and the rest of the reward components still remain active.

$$r_4 = r_4 + T_1(-min(d_p)), \quad \forall p \in \mathcal{P} \tag{16}$$



(a) δ_V (b) σ_V (c) Fake collision (d) Real collision

Fig. 3. Agent-agent collision.



(a) δ_O (b) σ_O (c) Fake collision (d) Real collision

Fig. 4. Agent-obstacle collision.

(b) Reward Based on Agent-Agent Collision: Agent-agent collision is more severe for aerial vehicles as they are more sensitive to collision than ground vehicles. For this reason, the same danger zone concept is adopted from UGVs, where each agent has a danger zone, σ_V around it, and when two danger zones overlap or cause a fake collision, a negative reward component r_5 is added to the agent reward. This component helps the UAVs learn to avoid colliding with each other while navigating. The reward component, r_5 can be determined as follows,

$$r_5 = r_5 - T_2(\delta_V + \sigma_V - d_V) \tag{17}$$

(c) Reward Based on Agent-Obstacle Collision: In the obstructed environment, the UAVs also need to navigate through obstacles to reach the targets safely. For this, a reward component r_6 has been used similar to the UGV environment of danger zone σ_O and a negative reward based on the overlapping area of agents and obstacles is provided, which can be calculated as follows,

$$r_6 = r_6 - T_3(\delta_O + \sigma_O - d_O) \tag{18}$$

(d) Reward for Returning to Nearest UGV: Although both of the environments have very similar functions, a key difference in the UAV environment is that the UAVs need to return to their nearest UGV after reaching their targets. To encourage such behavior, another reward component r_7 is added for the UAV. Here, if a UAV has not reached its assigned target, it receives a negative reward of magnitude r_T. After reaching the target, this reward gets replaced with a reward value based on the distance d_h between the UAV and its nearest UGV.

$$r_7 = r_7 - T_1(d_h) \tag{19}$$

The final UAV reward is calculated by adding each of the reward components together and the reward function r_A for each agent can be determined as follows,

$$r_A = r_4 + r_5 + r_6 + r_7 \tag{20}$$

After calculating, we use this reward to train the UAVs.

4.5. MAPPO framework

As a form of reinforcement learning method, Deep Q-Network or DQN became very popular. It was a simple and easy approach to train models to run in simulations in an effective way. But, it also has a fair amount of drawbacks. Notably, DQN tends to be unstable. Being an offline method and policy iteration method, DQN is very data efficient. But because of the way it does this, it is unstable and it cannot learn some difficult environment. One of the alternative solutions to this was the application of policy gradient methods. These methods work by calculating an estimator of the policy gradient and using the value in a stochastic gradient ascent algorithm. So, if you have a policy π_θ, based on the reward signal, some direct loss L and propagate it back directly to the network. To overcome the instability, multiple steps of optimization on L using the same sample or trajectory were suggested. However, this was not justified and it often caused large policy updates which were based on a small amount of data. Trust Region Policy Optimization, or TRPO, was an alternative approach that tried to deal with the issue of instability while not sacrificing data efficiency.

$\pi_{\theta_{old}}$ is the old policy and after updating, we get the new policy, π_θ. The ratio of the new policy is maximized by the old policy and is multiplied by the advantages $\hat{A}_t$ which basically means how much a certain action is than the actual on-policy action. This is essentially trying to change the network in a way such that, the policy tries to maximize the advantage. TRPO worked really well in a lot of situations, but its main drawback is that it is very complicated. It was difficult to implement and not compatible with architectures that included noise or parameter sharing. This led to the proposition of Proximal Policy Optimization or PPO which had the data efficiency of TRPO, while being easier and quicker to understand and implement.

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{21}$$

$r_t(\theta)$ in Eq. (21) is called the probability ratio which is used to simplify the policy ratio. TRPO tries to maximize the objective:

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right] \tag{22}$$

Algorithm 3 MAPPO pseudo-code

Initialize θ, the parameters for policy π and ϕ, parameters for critic V
Set learning rate α
while $step \leq step_{max}$ **do**
  Set data buffers D as empty list
  for $i = 1 \rightarrow batch\_size$ **do**
    Set τ as empty list
    Initialize $h_{0,\pi}^{(1)}, \ldots h_{0,\pi}^{(n)}$ actor RNN states
    Initialize $h_{0,V}^{(1)}, \ldots h_{0,V}^{(n)}$ critic RNN states
    for $t = 1 \rightarrow T$ **do**
      for all agents a **do**
        $P_t^{(a)}, h_{t,\pi}^{(a)} = \pi(o_t^{(a)}, h_{t-1,\pi}^{(a)}; \theta)$
        $u_t^{(a)} \sim p_t^{(a)}$
        $v_t^{(a)}, h_{t,V}^{(a)} = V(s_t^{(a)}, h_{t-1,V}^{(a)}; \phi)$
      end for
      Execute actions u_t, observer r_t, s_{t+1}, o_{t+1}
      $\tau += [s_t, o_t, h_{t,\pi}, h_{t,V}, u_t, r_t, s_{t+1}, o_{t+1}]$
    end for
    Compute advantage estimate $\hat{A}$ via GAE on τ, using PopArt
    Compute reward-to-go $\hat{R}$ on τ and normalize with PopArt
    Split trajectory τ into chunks of length L
    for $l = 0 \rightarrow T/L$ **do**
      $D = D \cup (\tau[l : l + T], \hat{A}[l : l + L], \hat{R}[l : l + L])$
    end for
  end for
  for mini-batch $k = 1 \rightarrow K$ **do**
    $b \leftarrow$random mini-batch from D with all agent data
    for each data chunk c in the min-batch b **do**
      update RNN hidden states for π and V from first hidden state in data chunk
    end for
  end for
  Adam update θ on $L(\theta)$ with data b
  Adam update ϕ on $L(\phi)$ with data b
end while

Maximizing L^{CPI} in Eq. (22) without constraints would create excessively large policy updates. To penalize policy changes that move $r_t(\theta)$ away from 1, Eq. (23) was proposed as the main objective:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right]. \tag{23}$$

The main term in Eq. (23) has two parts. The first term is the Eq. (22), while the second term is the $clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t$ which modifies the objective by clipping the probability ratio. This removes the incentive for the policy to move beyond the factor of ϵ, not letting it move too far and cause instability. The next part is the minimum function. What it does is take the lower bound of what we know is possible. As we can have multiple values for which the network can learn, if anything but the lower bound of the possible values is selected, there is a higher chance for the learning to collapse or be unstable. Therefore, selecting the absolute minimum value ensures the safest form of update possible.

4.5.1. State action representation

Both MAPPO and MADDPG are deep reinforcement learning algorithms where they use the actor-critic network to train agents to take proper actions based on their observation. Since they also use the same environment for training, the state action representation for MAPPO is similar to the MADDPG representation in Section 4.4.1.

4.5.2. Training process

The form of MAPPO used in this experiment is very similar to the structure of the single-agent PPO but modified for multiple agents. It works by learning a policy π_θ and a value function $V_\phi(s)$. Separate neural networks are used to represent these functions where variance reduction is performed using $V_\phi(s)$. As it is only used during training, it can use global information not observable to the agent to allow PPO in multi-agent domains to follow the centralized training decentralized execution structure. Here, PPO with a centralized value function is regarded as MAPPO, and in our experiment, MAPPO operates in settings where agents share a common reward for cooperative settings. As our individual training environments are comprised of homogeneous agents, parameter sharing is used to improve the efficiency of learning. PPO has also utilized the decentralized partially observable Markov decision processes (DEC-POMDP) with shared rewards. This is defined by $\langle S, A, O, R, P, n, \gamma \rangle$. $o_i = O(s : i)$ is the local observation for agent i at global state s. The transition probability from s to s' is denoted by $P(s'|s, A)$ for the joint action $A = (a_1, \ldots, a_n)$ for n agents. Action a_i is produced from local observation o_i by agent using policy $\pi_\theta(a_i|o_i)$, parameterized by θ and jointly optimize the discounted accumulated reward,

$$J(\theta) = \mathbb{E}_{A^t, s^t}[\sum_t \gamma^t R(s^t, A^t)] \tag{24}$$

4.5.3. Reward calculation

As both MAPPO and MADDPG use the same environment, the reward components used in MAPPO are the same components used in MADDPG which can be found in the MADDPG Reward Component Section 4.4.3.

4.6. Complexity of MEANCRFT

Note that, the MEANCRFT combines the Modified Mean-Shift Clustering with a MADRL algorithm. Firstly, the complexity of Mean Shift Clustering is $O(tn^2)$, where t is the time for processing each data point and n is the number of data points. Therefore, the complexity of the Modified Mean Shift Clustering is $O(tn^3)$. This algorithm runs once before deploying the coalitions in the environment. The second part, the MADRL algorithm, has variable complexity. In this work, we deployed the MADDPG algorithm that has a computational complexity of $O(2N(\sum_{l^a=1}^{D^a} \zeta_{l^a-1}\zeta_{l^a} + \sum_{l^c=1}^{D^c} \zeta_{l^c-1}\zeta_{l^c}))$ where D^a and D^c are respectively the depth for the neural networks in the actor and the critic [31]. For MAPPO, the overall time complexity can be expressed as $O(s \times b \times m \times n \times K \times T^2/L)$, where s is the max step, b is the batch size, m is the number of data chunks in a mini-batch, n is the number of agents, K is the number of mini-batches, T is the number of time steps within a trajectory and L is the length of trajectory chunk. It should be noted that, for DRL, the actual efficiency and runtime can be influenced by various factors, including the specifics of the neural network architecture, the size of the environment, and the computing resources available.

5. Experimental evaluation

This section describes the details of the simulation platform, the training parameters used for training, and the performance metrics used for evaluation along with simulation results and analysis.

5.1. Simulation platform

We have created the simulation training environment based on OpenAI's Gym platform. It has been created to simulate an environment with a fixed number of obstacles where a collaborative coalition of UAV-UGV will be deployed to plan a reasonably safe and fast path toward some targets. We have assumed that the area of the developed customized environment is 2000×2000 m^2. The position of agents, targets, and obstacles are distributed randomly following uniform random distribution at the start of each episode to ensure the robustness of the work in varying environments. For the zone radius, we have used 25m,

Table 2
Parameters used in simulation environment.

Parameters	Values
Environment Size	2000 m ×2000 m
Number of Obstacles	1 to 6
Number of UGVs	1 to 4
Number of UAVs	2 to 16
Number of Targets	3 to 20
Speed	1 ms^{-1}
Zone Radius R	10 m to 25 m
UAV Flight Range V	50 m

since it is a convenient distance for the size of the vehicles and the obstacles we chose.

Essentially, in every episode, a new map is given to the training, which ensures the randomization of the training process. UGVs navigate on the ground surface to reach their destination. They only require motion along the 2 axes to move on the surface they are on. For this, we have selected a 2D environment for training the UGV agents. To create this environment, we have used the environment structure provided by the Multi-agent Particle Environments (MPE) [32] package and created our own custom scenario. It uses the geometric coordinate system where the center of the environment is the origin and the environment extends from −1 to +1 on both axes. The agents and landmark components provided by the package are used to build the environment. Agents are used to represent UGVs while landmarks are used to represent targets, obstacles, and rough terrains. A similar environment is used for UAV training. Table 2 summarizes the major simulation parameter values.

5.2. Training parameters

For training the models in their respective environments using MADRL, some hyperparameters for each type of training need to be set. At first, the hyperparameters of the MADDPG and MAPPO algorithms are set similarly to [19,22], respectively. However, We had to tune these parameters based on our environment for better performance. Tables 3 and 4 show the effects of various parameters on the performance metrics of our custom environment. For example, in Table 3, we see that changing the batch size for MADDPG significantly affects the convergence rate as well as the performance. With a low batch size of 256, the algorithm never converges. From our experiments, it was still learning after $100K$ episodes. Again, with a batch size of 512, it stopped learning at $20K$ episodes, but its performance was extremely poor, only completing 20% of all the given tasks. This improved to 80% with a batch size of 768, but it took $80K$ episodes to converge. With a 1024 batch size, we completed the training at $40K$ episodes and the model completed 95% of the tasks. Similarly, Table 4 gives us an idea about the effects of hyperparameters on the MEANCRFT-MAPPO model. For example, changing the epoch value affects the convergence time of our model. When the value is 5, it converges slowly and as it goes higher, the convergence time decreases. However, having a high epoch of 15 decreases the average reward of the model. A lower reward means that the model is performing worse than it can at a different epoch value. Therefore, the best epoch value is accepted as 10. Again, not using value normalization in MAPPO training can slow down the convergence and also yield a low reward value. Hence, it was applied in our training to get a better performance.

After conducting numerous experiments to set the suitable values for the hyperparameters, we have used similar values for both UAV and UGV training. Here, we use 0.01 as the learning rate for both the actor and critic network, and a batch size of 1024 is considered for sampling the memory buffer. The discount factor γ is set to 0.99 and 0.01 is used as the value for τ. Table 5 lists the training parameter values of the developed MEANCRFT-MADDG and MEANCRFT-MAPPO systems.

Table 3
Performance at different hyperparameter values for 2 agents in MADDPG.

Learning rate	Batch size	Convergence	Rate of task
0.01	256	No	/
0.01	512	Fast	20%
0.01	768	Slow	80%
0.01	1024	Moderate	95%
0.1	1024	No	/
0.5	1024	No	/
0.9	1024	No	/

Table 4
Performance at different hyperparameter values for 2 agents in MAPPO.

Epochs	Value normalization	Convergence	Avg reward
5	Yes	Slow	High
10	Yes	Fast	High
15	Yes	Fast	Low
10	No	Slow	Low

Table 5
Training parameters used in MEANCRFT-MADDPG and MEANCRFT-MAPPO.

Parameters	MADDPG values	MAPPO values
α	0.01	/
β	0.01	/
γ	0.99	0.99
τ	0.01	0.01
Activation	/	Tanh
GAE λ	/	0.95
Gain	/	0.01
Batch Size	1024	25
Memory Size	10^6	/
Nodes of 1st Hidden Layer	64	64
Nodes of 2nd Hidden Layer	64	64

5.3. Baseline and performance metrics

To compare the performance of the proposed framework, we selected the following approaches to compare with MADDPG and MAPPO

-

- **COordinated Policy Optimization (COPO)** is a state-of-the-art MADRL approach that uses bi-level coordination of agents to improve collective performance. Here, In the process of centralized training, at first, each agent performs individual learning to maintain local coordination among themselves through a mechanism that each agent is primarily influenced by its proximate counterparts through local coordination. Next, the local coordination process is optimized, which leads to global coordination that can improve the collective performance of the population [33].
- **Multi-agent Distributional Reward Estimation (DRE-MARL)** is a state-of-the-art approach that works by modeling reward distributions for all action branches. Here, along with the environmental rewards, the potential rewards on other action branches are also considered to perform more stable critic updating to achieve better performance [34].
- **Single Coalition (Simplified IADRL)** is a state-of-the-art DRL approach, which considers only single UAV-UGV coalition [7].
- **CRFT without Zones** This approach does not apply modified mean-shift algorithm-based zoning heuristics.

To evaluate the performance of the studied systems, the following performance metrics have been used:

- **Completion Rate of Tasks**: In our scenario, when all the available targets in the environment have been reached and the UAVs have reached their closest UGV, we call this a completed task. Task completion rate is the summation of the total number of

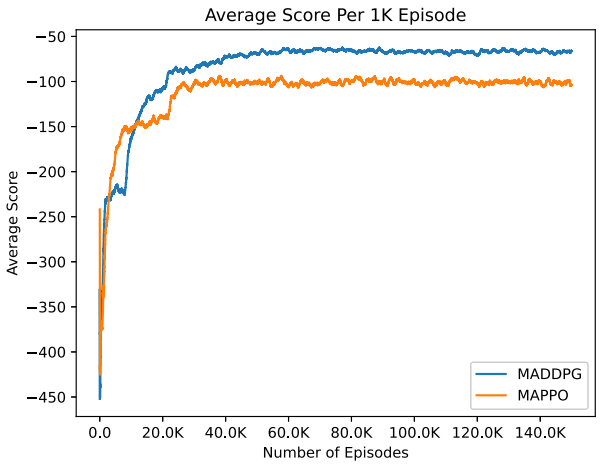


Fig. 5. Average score per 1000 episodes for 2 UGVs.

completed tasks in the combined scenario divided by the total number of evaluated episodes, which can be determined as follows,

$$\frac{\sum_{e=1}^{E} T_e}{E}, \tag{25}$$

where, T_e is task completion value of an episode e and E is the maximum number of episodes. The task completion value of an episode is set to 1 when an agent performs its task; otherwise, 0.

- **Collisions Per 1K Episode**: This indicates the total number of agent-agent and agent-obstacle collisions per 1K episodes, which can be estimated as follows,

$$\sum_{e=1}^{1000} (\alpha_e + \beta_e), \tag{26}$$

where, α_e and β_e denote the number of agent-agent and agent-obstacle collisions for an episode e, respectively,

- **Number of Steps per Episode**: The average of the number of steps required to finish an episode.
- **Accuracy**: The percentage of targets reached out of the total number of targets in the environment per episode.

We evaluated our combined model for 1,000 episodes in different target clusters with different vehicle combinations. For example, 1 UGV with 2 UAVs, 1 UGV with 3 UAVs etc.

5.4. Simulation results and analysis

This section illustrates the graphical collections of the results we have collected from our experiment. We have analyzed the results and explained the behaviors of the coalitions both in training and combined model testing. To investigate the performance of the developed models, we randomly vary the numbers of target count (randomly distributed in the environment), coalition combination, and zone radius along the x-axis, and show the result of our performance metrics on the y-axis.

5.4.1. Performance analysis for training

The average reward value per episode is used in RL algorithms to identify the model's learning. This is because the main objective of the model is to have the highest reward possible by exploring and exploiting its actions. As the entire process is driven by the reward, the overall state of the trained model can be understood by observing the average reward per episode after training. If this reward value changes after each episode, it means that the model is still learning. Again, if the average reward stabilizes after some episodes, it means that the model has converged and reached a saturation point, and continued training will not improve the model. Fig. 5 shows the average reward scores of 2 agents for MEANCRFT-MADDPG and MEANCRFT-MAPPO frameworks as the number of training episodes increases. From the graph, we can conclude that the training process functioned appropriately since the reward converges for both of these cases after around $40K$ episodes. Here, we notice that MADDPG's score converged below the score of -50, whereas MAPPO converged below the score of -100. Since both the algorithm environments are the same and use the same reward structure, the MADDPG framework seems to provide a better reward, which implies that the MADDPG framework can find shorter or better paths for reaching the target safely.

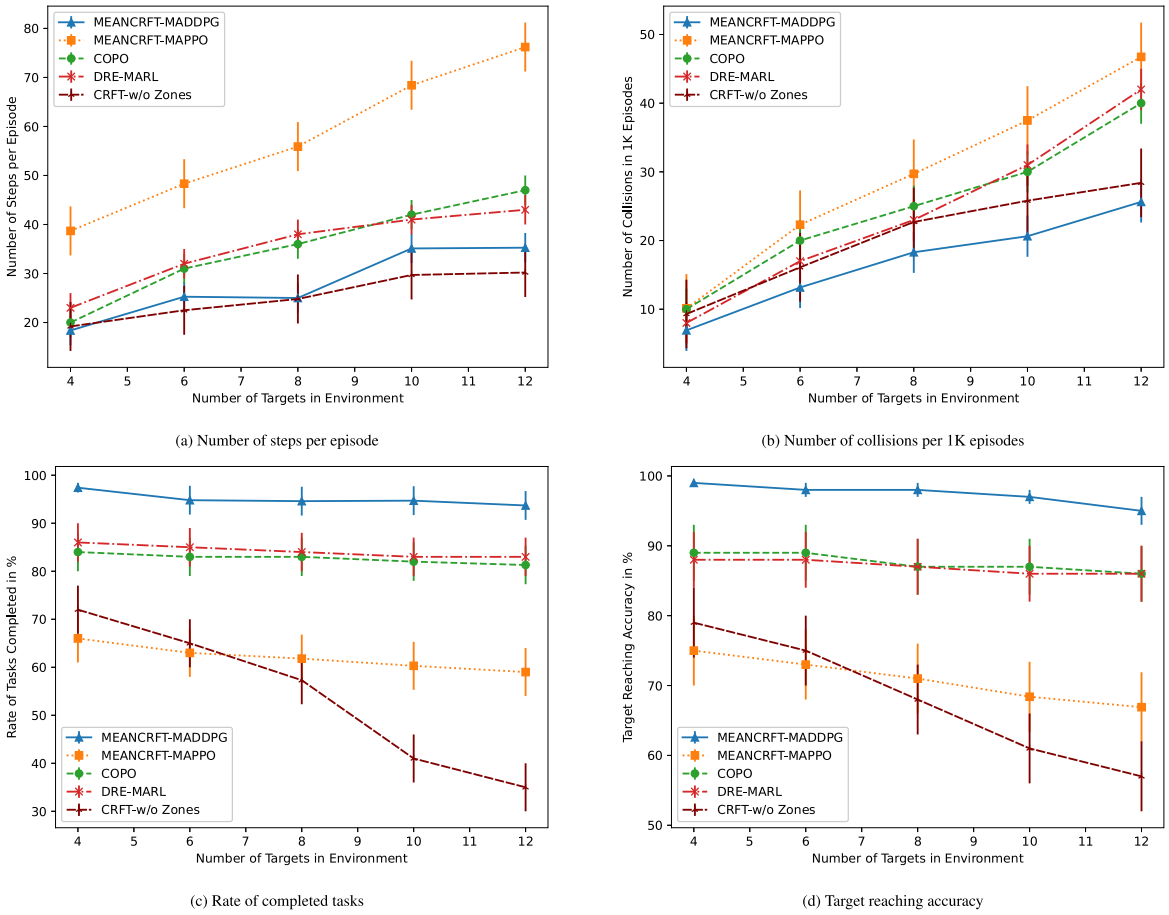
5.4.2. Impacts of varying number of targets

Fig. 6 shows the effects of changing the number of targets in the environment. In this experiment, the number of UAVs and UGVs is fixed at 8 and 2, respectively, whereas, the radius of each zone is set to 25 m. We only changed the available targets in the environment from 4 to 12 to observe the changes in the environment. Error bars are included in all of the graphs of Fig. 6, where we put the lowest and highest value of all simulation results as the lowest and highest point, respectively of the error bar of a metric. The results depict that no value exceeded more than 5% in each direction from the mean, which verifies the robustness of our results.

Fig. 6(a) shows the number of steps required to reach the targets when the number of targets varies in the environment. Here, we see that the MEANCRFT-MADDPG outperforms COPO, DRE-MARL, and MAPPO. Moreover, the graph of CRFT without zoning seems to work just as well as the MEANCRFT-MADDPG one. This is because, to calculate the number of steps per episode, the agents need to complete the episode by reaching all the targets. CRFT without zoning tends to reach all the targets in an episode only when the randomly generated targets are very close to each other and the targets are minimal in number. In that case, the battery limitation issue of UAVs does not harm the performance. In all other cases, the agents fail to reach all the available targets because of the limited step counts, which represent the limited battery of a UAV.

Fig. 6(b) demonstrates that the number of collisions per 1000 episodes increases with the increasing number of targets in the environment, and this happens because when the number of targets is minimal, most of the UAVs are not deployed in action since the number of aerial targets is less than the number of available UAVs. Although the number of UAVs and UGVs in the environment is constant, the increase in the number of targets enables the UGVs to deploy a higher number of UAVs to reduce the overall time of the episode. This results in a higher number of collisions with the increase in the number of agents. MADDPG also beats the other methods comfortably in terms of collisions.

Fig. 6(c) illustrates the rate of tasks completed in percentage, which takes into account only those episodes where all the targets are reached and all the UAVs have come back to the stationary UGV, and only then it is deemed as a completed task. Here, the without zoning approach fails miserably, since with the increase in the number of targets, the method fails to determine how to get the deployed UAVs back to their respective UGV which might be very far, which results in the UAVs losing battery and failing to complete the task. The MEANCRFT-MADDPG works exceptionally well, with an almost perfect rate of success for the lower number of targets, which decreases a bit, but still retains an impressive value when the target count gets much larger. DRE-MARL and COPO maintain a decent performance and perform almost similarly regarding completing tasks. The MEANCRFT-MAPPO fails substantially as the number of targets increases. In our experiments, we realized that MEANCRFT-MAPPO's on-policy approach works well when the environment is straightforward. As the agent count and complexity of the environment increases, it starts to perform worse. In the developed system, we have tried to fine-tune the environment



(a) Number of steps per episode

(b) Number of collisions per 1K episodes

(c) Rate of completed tasks

(d) Target reaching accuracy

Fig. 6. Impacts of varying the number of targets.

and other parameters; however, it is not significant enough compared to MEANCRFT-MADDPG.

Fig. 6(d) depicts the target reaching accuracy, which is very similar to the previous metric. The difference is that here we consider reaching all the individual targets in an episode, not completed tasks like the previous metric. We count the ratio of the number of targets reached by a method to the total number of targets in an episode. The MEANCRFT-MADDPG always maintains a higher accuracy and this is much more noticeable as the target count increases. Its performance does go down slightly as more targets are added, but it still maintains an accuracy above ninety-five percent. This shows that the developed MEANCRFT-MADDPG approach is robust against changes in target counts compared to conventional approaches. Because of the higher complexity of the environment, MEANCRFT-MAPPO underperforms, and its accuracy also decreases as the target count increases. CRFT without Zones starts with an eighty percent accuracy, but because of its limited reachability, its performance drastically falls as it fails to reach most of the targets. The two other baseline algorithms, COPO and DRE-MARL, perform almost similarly but still fall short in comparison with MADDPG.

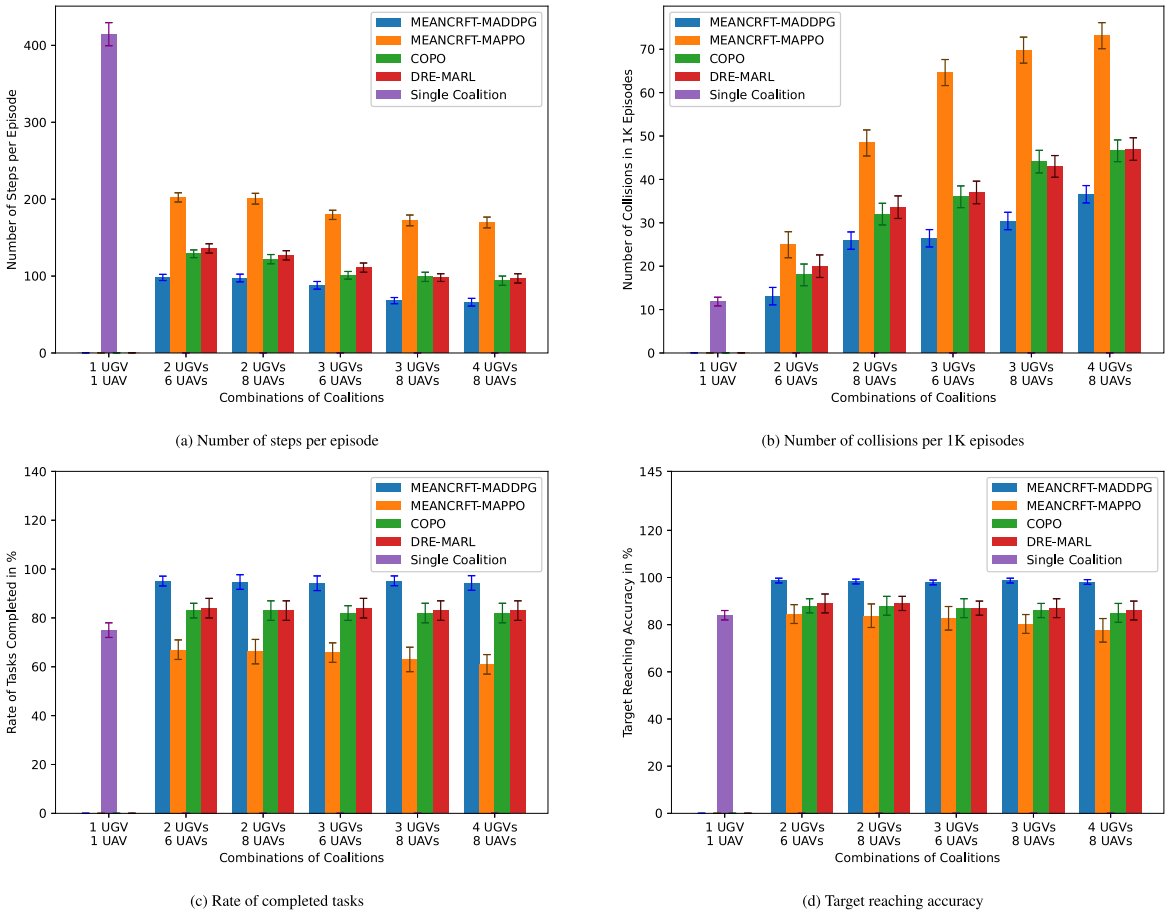
5.4.3. *Impacts of varying the combination of UAV-UGV coalitions*

Fig. 7 varies the combination of UAV-UGV coalitions and measures the performance of the metrics, keeping all the other values in the environment constant. Here, the radius of the zone is chosen as 25 m, the target count is set to 10, and the obstacle count can vary. The combinations of coalitions tested in the experiment are 1–1, 2–6, 2–8, 3–6, 3–8, and 4–8, where the first digit indicates the number of UGVs and the second digit is the number of UAVs.

Fig. 7(a) shows the steps required for each method when the number of coalitions varies in the environment. Here, the general trend is that as agent count increases, the number of time steps decreases, which is self-explanatory as more agents naturally get the job done faster. The developed MEANCRFT-MADDPG model seems to work much better than the single coalition or MEANCRFT-MAPPO model in all cases. COPO comes close to MADDPG when the agent count increases, but neither COPO nor DRE-MARL can quite reach the performance level of MADDPG. Since both MEANCRFT-MADDPG and MEANCRFT-MAPPO use more than a single UAV-UGV pair, they easily outperform the single coalition method. However, because of the increased complexity of the UAVs in the custom environment, MEANCRFT-MAPPO is not as effective as MEANCRFT-MADDPG in reducing the number of steps needed to reach the targets.

In Fig. 7(b), we demonstrate that the number of collisions per 1000 episodes increases with the increasing number of vehicles in the environment. The single coalition faces few collisions since it has only one UGV and UAV. However, in multi-coalition scenarios, this value increases with the increase in coalition count. The MEANCRFT-MADDPG and MEANCRFT-MAPPO both have an increasing number of collisions with the increasing number of agents in the environment. COPO and DRE-MARL follow a similar trend of increasing collision counts as the agent number increases as expected. However, the increase in MEANCRFT-MADDPG is much more forgiving compared to others because of its robustness against complex environments.

Fig. 7(c) illustrates the rate of tasks completed with various combinations of UAV-UGV collision. Within the given experiment, the rate of completion remains almost constant as the number of coalitions increases, which means that for the given target count, all the combinations tested have worked reasonably well. Here, MEANCRFT-MADDPG outperforms the other methods by maintaining a rate above 90%. Fig. 7(d) shows similar trends we found in the previous graph. The



(a) Number of steps per episode

(b) Number of collisions per 1K episodes

(c) Rate of completed tasks

(d) Target reaching accuracy

Fig. 7. Impacts of varying the combination of UAV-UGV coalitions.

target reaching accuracy in MEANCRFT-MADDPG is almost 10% better than the single coalition method and is also consistent as the number of coalitions increases. The MEANCRFT-MAPPO maintains its accuracy with a single coalition method while trying to maintain consistency as the coalition count increases. This shows that both of the multi-agent approaches stay robust with an increase in agent count, and sometimes may even perform better with a higher agent count. In fact, MEANCRFT-MAPPO, COPO, and DRE-MARL perform similarly in this metric and stay quite close to the performance of MADDPG.

5.4.4. *Impacts of varying the radius of the zone*

In the experiment of Fig. 8, we only vary the radius of zones from 10 m to 25 m to measure and observe the effects of this change on the considered metrics. Here, the target count is fixed at 10, 2 UGVs and 8 UAVs are deployed in the environment and the obstacle count can vary.

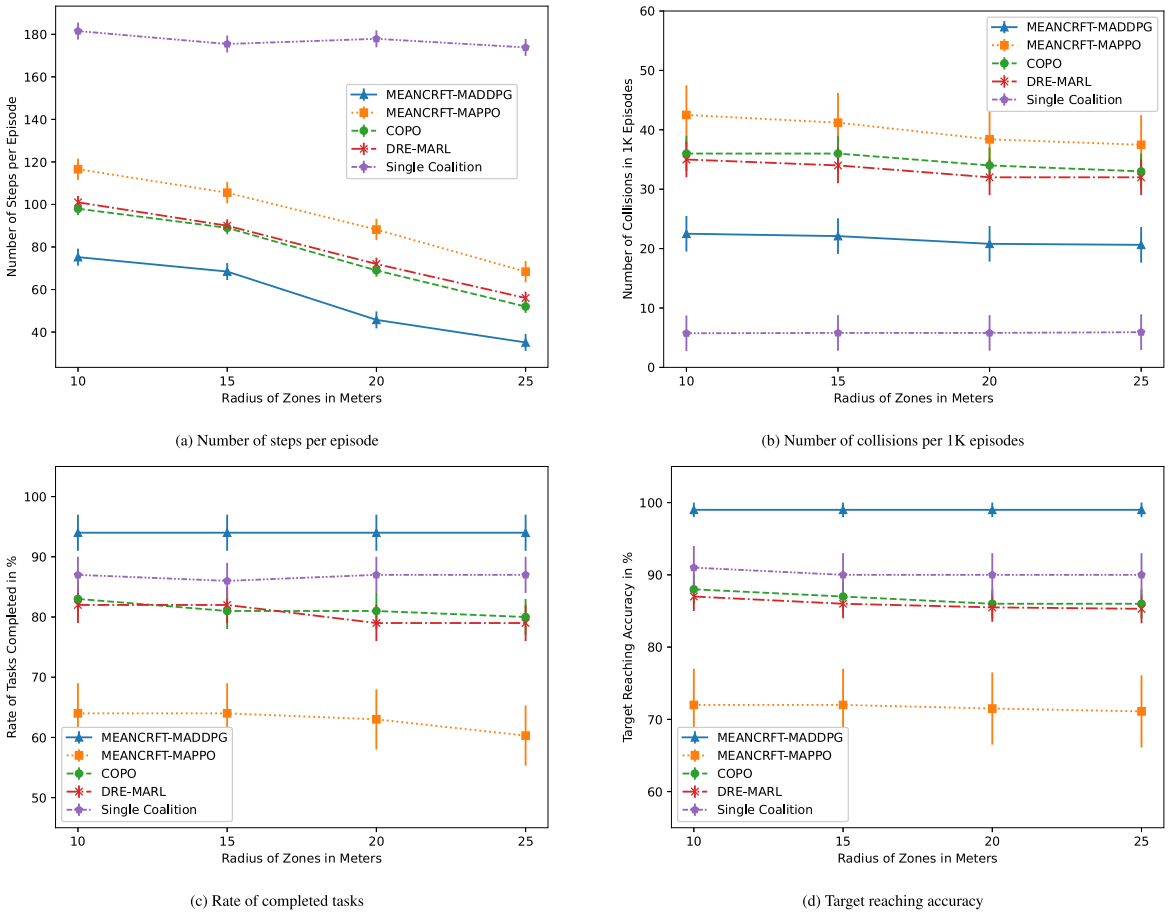
Fig. 8(a) shows the required number of steps for each of these methods with the varying radii of the zones. When the radius is low, dynamic coalitions work similarly to the single coalition method where the entire coalition moves to targets individually, or at least as far as possible until the UGVs face obstacles, and then deploy UAVs. These UAVs then reach the targets and return to the closest UGV. All multi-coalition approaches work better than the single coalition method because there are multiple UAVs and UGVs involved, and multiple targets are being selected at a time. As we increase the radius size, all multi-coalition approaches start to get significantly better. Compared to all the other methods, MEANCRFT-MADDPG has the least number of steps, and its improvement over the change of radius is much more prominent. In Fig. 8(b), we demonstrate the number of collisions per 1000 episodes. Here, the single coalition has the least collisions because it has only 1 UGV and 1 UAV. In our multi-coalition methods, we do notice a small downward trend as the radius increases. The collision count is increased with the increasing radius as more agents are deployed per zone. However, as there are a limited number of targets in the environment, the coalitions require fewer steps to reach all the targets with the increase in radius size. As a result, the collision count did not increase significantly. Comparatively, MAPPO faces the most collisions, while MADDPG makes the least and COPO and DRE-MARL maintain their average standard as before.

Fig. 8(c) illustrates the rate of tasks completed in percentage. We note that the rate of completed tasks does not change much with the change in radius for MEANCRFT-MADDPG. This shows that our approach is strong against changes in the radius of the zones. COPO and DRE-MARL slightly underperform since the single coalition method seems to work better than them in this regard. MEANCRFT-MAPPO does not perform well because of the increased complexity of the environment. As the number of targets is already high in the environment, it starts to struggle from the beginning and maintains a similar performance. We show the target reaching accuracy in Fig. 8(d), which shows similar trend as the previous graph Fig. 8(c). Here, MEANCRFT-MADDPG performs at least 10% better than the other algorithms and maintains its superior performance as the radius increases.

From the charts and graphs above, we can conclude that while different metrics show different trends for different configurations, one of our proposed methods MEANCRFT-MADDPG works much better than the baseline, the non-zoning method, and MEANCRFT-MAPPO.

6. Conclusions

In this work, we proposed a unique method for one-to-many collaboration between a UGV and a few UAVs. Our methodology used



(a) Number of steps per episode

(b) Number of collisions per 1K episodes

(c) Rate of completed tasks

(d) Target reaching accuracy

Fig. 8. Graphs varying the radius of zones.

customized training for UGVs and UAVs to develop an efficient coalition needed to navigate through an environment filled with obstacles. The key contribution of our work is extending the flexibility of the number of UAVs and UGVs in a coalition and the quick reaching of various target points while avoiding collisions. A distinctive feature of our method is the division of the targets into various circular zones based on density and range by utilizing a modified mean-shift clustering, which was a heuristic we needed to develop to enhance our approach. The result of our numerical experiments showed that, in terms of target navigation time, collision avoidance rate, and task completion rate, the suggested approach performed better or within an acceptable range when compared to the existing solutions. Our future work plan includes implementing this using the MLAgents package of Unity, since Unity has much better built-in properties that mimic the physical world really well, and models trained there tend to be much more realistic.

CRediT authorship contribution statement

Shamyo Brotee: Methodology, Investigation, Data curation, Conceptualization, Software. **Farhan Kabir:** Methodology, Investigation, Data curation, Conceptualization, Software. **Md. Abdur Razzaque:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Investigation. **Palash Roy:** Writing – review & editing, Writing – original draft, Software, Methodology, Conceptualization. **Md. Mamun-Or-Rashid:** Writing – original draft, Writing – review & editing. **Md. Rafiul Hassan:** Writing – original draft, Writing – review & editing. **Mohammad Mehedi Hassan:** Software, Supervision, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the King Saud University, Riyadh, Saudi Arabia, through the Researchers Supporting Project under Grant RSP2024R18.

References

[1] Next Move Strategy Consulting, Unmanned ground vehicle market, 2023, URL https://www.nextmsc.com/report/unmanned-ground-vehicle-market.

[2] Vantage Market Research, Unmanned aerial vehicle (UAV) market, 2023, URL https://www.vantagemarketresearch.com/industry-report/unmanned-aerial-vehicle-uav-market-2086.

[3] S.A.H. Mohsan, N.Q.H. Othman, Y. Li, M. Alsharif, M. Khan, Unmanned aerial vehicles (UAVs): Practical aspects, applications, open challenges, security issues, and future trends, Intell. Service Robot. (2023).

[4] Y. Wang, J. Wu, X. Hua, C.H. Liu, G. Li, J. Zhao, Y. Yuan, G. Wang, Air-ground spatial crowdsourcing with UAV carriers by geometric graph convolutional multi-agent deep reinforcement learning, in: 2023 IEEE 39th International Conference on Data Engineering, ICDE, IEEE, 2023, pp. 1790–1802.

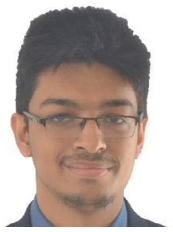
[5] S. Murshed, A.S. Nibir, M.A. Razzaque, P. Roy, A.Z. Elhendi, M.R. Hassan, M.M. Hassan, Weighted fair energy transfer in a UAV network: A multi-agent deep reinforcement learning approach, Energy 292 (2024) 130527.

[6] H. Qie, D. Shi, T. Shen, X. Xu, Y. Li, L. Wang, Joint optimization of multi-UAV target assignment and path planning based on multi-agent reinforcement learning, IEEE access 7 (2019) 146264–146272.
[7] J. Zhang, Z. Yu, S. Mao, S.C. Periaswamy, J. Patton, X. Xia, IADRL: Imitation augmented deep reinforcement learning enabled UGV-uav coalition for tasking in complex environments, IEEE Access 8 (2020) 102335–102347.
[8] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C.J. Taylor, V. Kumar, Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments, IEEE Robot. Autom. Lett. 2 (3) (2017) 1688–1695.
[9] Y. Chen, D. Yang, J. Yu, Multi-UAV task assignment with parameter and time-sensitive uncertainties using modified two-part wolf pack search algorithm, IEEE Trans. Aerosp. Electron. Syst. 54 (6) (2018) 2853–2872.
[10] A. Sharma, S. Shoval, A. Sharma, J.K. Pandey, Path planning for multiple targets interception by the swarm of uavs based on swarm intelligence algorithms: A review, IETE Tech. Rev. 39 (3) (2022) 675–697.
[11] L. Babel, Coordinated target assignment and UAV path planning with timing constraints, J. Intell. Robot. Syst. 94 (3–4) (2019) 857–869.
[12] J. Liu, Z. Shi, Y. Zhang, A new method of UAVs multi-target task assignment, DEStech Trans. Eng. Technol. Res (2018) 388–394.
[13] C. Yan, X. Xiang, A path planning algorithm for UAV based on improved Q-learning, in: 2018 2nd International Conference on Robotics and Automation Sciences, ICRAS, IEEE, 2018, pp. 1–5.
[14] B. Li, Y. Wu, Path planning for UAV ground target tracking via deep reinforcement learning, IEEE Access 8 (2020) 29064–29074.
[15] S. Josef, A. Degani, Deep reinforcement learning for safe local planning of a ground vehicle in unknown rough terrain, IEEE Robot. Autom. Lett. 5 (4) (2020) 6748–6755.
[16] A. Razzak, M.T. Islam, P. Roy, M.A. Razzaque, M.R. Hassan, M.M. Hassan, Leveraging deep Q-Learning to maximize consumer quality of experience in smart grid, Energy 290 (2024) 130165.
[17] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[18] Y. Li, L. Feng, Y. Yang, W. Li, GAN-powered heterogeneous multi-agent reinforcement learning for UAV-assisted task offloading, Ad Hoc Netw. 153 (2024) 103341.
[19] R. Lowe, Y.I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, Adv. Neural Inf. Process. Syst. 30 (2017).
[20] J. Queeney, Y. Paschalidis, C.G. Cassandras, Generalized proximal policy optimization with sample reuse, Adv. Neural Inf. Process. Syst. 34 (2021) 11909–11919.
[21] B. Fang, Z. Peng, H. Sun, Q. Zhang, Meta proximal policy optimization for cooperative multi-agent continuous control, in: 2022 International Joint Conference on Neural Networks, IJCNN, IEEE, 2022, pp. 1–8.
[22] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, Y. Wu, The surprising effectiveness of ppo in cooperative multi-agent games, Adv. Neural Inf. Process. Syst. 35 (2022) 24611–24624.
[23] Z. Wu, C. Yu, D. Ye, J. Zhang, H.H. Zhuo, et al., Coordinated proximal policy optimization, Adv. Neural Inf. Process. Syst. 34 (2021) 26437–26448.
[24] Y. Guan, S. Zou, H. Peng, W. Ni, Y. Sun, H. Gao, Cooperative UAV trajectory design for disaster area emergency communications: A multiagent PPO method, IEEE Internet Things J. 11 (5) (2024) 8848–8859.
[25] J. Li, S. Cao, X. Liu, R. Yu, X. Wang, Trans-UTPA: PSO and MADDPG based multi-UAVs trajectory planning algorithm for emergency communication, Front. Neurorobot. 16 (2023).
[26] J. Chen, C. Du, Y. Zhang, P. Han, W. Wei, A clustering-based coverage path planning method for autonomous heterogeneous UAVs, IEEE Trans. Intell. Transp. Syst. 23 (12) (2022) 25546–25556.
[27] J. Chen, Y. Zhang, L. Wu, T. You, X. Ning, An adaptive clustering-based algorithm for automatic path planning of heterogeneous UAVs, IEEE Trans. Intell. Transp. Syst. 23 (9) (2022) 16842–16853.
[28] F. Ropero, P. Muñoz, M.D. R-Moreno, TERRA: a path planning algorithm for cooperative UGV–UAV exploration, Eng. Appl. Artif. Intell. 78 (2019) 260–272.
[29] W. Lee, D. Kim, Autonomous shepherding behaviors of multiple target steering robots, Sensors 17 (12) (2017) 2729.
[30] D. Tang, J. Man, L. Tang, Y. Feng, Q. Yang, WEDMS: an advanced mean shift clustering algorithm for ldos attacks detection, Ad Hoc Netw. 102 (2020) 102145.
[31] D. Fan, H. Shen, L. Dong, Multi-agent distributed deep deterministic policy gradient for partially observable tracking, in: Actuators, 10, (10) MDPI, 2021, p. 268.
[32] OpenAI, Multi-agent particle environment, 2018, URL https://github.com/openai/multiagent-particle-envs.
[33] Z. Peng, Q. Li, K.M. Hui, C. Liu, B. Zhou, Learning to simulate self-driven particles system with coordinated policy optimization, Adv. Neural Inf. Process. Syst. 34 (2021) 10784–10797.
[34] J. Hu, Y. Sun, H. Chen, S. Huang, Y. Chang, L. Sun, et al., Distributional reward estimation for effective multi-agent deep reinforcement learning, Adv. Neural Inf. Process. Syst. 35 (2022) 12619–12632.

Shamyo Brotee received his B.Sc degree from the Department of Computer Science and Engineering, University of Dhaka, Bangladesh. In addition to conducting his research work, he is also currently employed as a software engineer. His research interests include Multi-Agent Deep Reinforcement Learning and AI applications in path planning and networking.

Farhan Kabir has completed his B.Sc degree from the Department of Computer Science and Engineering, University of Dhaka, Dhaka, Bangladesh. His research interests include multi-agent reinforcement learning and AI-based approaches in path planning and networking.

Md. Abdur Razzaque is a Professor of Dept. of Computer Science and Engineering, University of Dhaka. He is chairing 2021 Executive Committee for IEEE Computer Society Bangladesh Chapter. He received his BS in Applied Physics and Electronics and MS in Computer Science from University of Dhaka, Bangladesh, in 1995 and 1996, respectively. He obtained a Ph.D. in Computer Engineering from Kyung Hee University, South Korea in 2009. He worked for Green University of Bangladesh for different periods as Pro-Vice-Chancellor, Dean of Faculty of Science and Engineering and Chairperson, Dept of CSE during 2016–2021. He was a research professor at Kyung Hee University, South Korea, during 2010–2011. He worked as a visiting professor at Stratford University, Virginia, USA, in 2017. He is the director of Green Networking Research Group (http://cse.du.ac.bd/gnr) of the Dept. of CSE, DU. He has been promoting Outcome Based Education for Science and Engineering Faculties of the country's leading universities. He led many national and international-funded research projects. His research interest is in the area of modeling, analysis and optimization of wireless networking protocols and architectures, Mobile Crowdsourcing, Sensor Data Clouds, Internet of Things, Edge Computing, etc. He has published 140+ research papers in peer-reviewed conferences and journals. He is offering editorial services to reputed journals and contributing to numerous international conferences at different capacities. He is a senior member of IEEE, a member of IEEE Computer Society, Internet Society (ISOC), etc.



Palash Roy received B.Sc. and M.Sc. degree from Dept. of Computer Science and Engineering, University of Dhaka in 2019 and 2022, respectively. He is currently working as a lecturer in, the Department of Computer Science and Engineering, Green University of Bangladesh. His research interests include mobile device cloud, network function virtualization, mobile edge computing, AI-based approaches for network applications design, etc. He is an active member of the IEEE Computer Society.

Mamun-Or-Rashid received his B.S. and M.S. degrees from the University of Dhaka, Bangladesh in 1998 and 1999, respectively. He obtained his Ph.D. degree in Wireless Networking from the Department of Computer Engineering, School of Electronics and Information, Kyung Hee University, South Korea in 2008. He was a postdoctoral fellow at the same university during the 2008–2009 Institute of Multimedia Technology. He worked as a Technical Support Team Consultant for the Bangladesh Research and Education

Network (BdREN) component of the Higher Education Quality Enhancement Project (HEQEP) during 2013–2016. Which has been implemented by the University Grants Commission under the Ministry of Education (MoE) and financed by the World Bank. Later he served as a Distance/Hybrid Learning Consultant for the same organization in 2018. He closely works in different committees of the Information and Communication Division to contribute to innovation evaluation and innovation fund policy making. He is a technical committee member for the National Data Center, the first Tier IV Data Center (4TDC) in Bangladesh, Smart Metering for Northern Electric Supply Company (NESCO), and many others. He is now working as a Professor in the Department of Computer Science and Engineering, University of Dhaka, Bangladesh. His research interest includes modeling, analysis, and optimization of wireless networking protocols and architectures, wireless sensor networks, wireless mesh networks, Internet of Things (IoT), etc. He has published a good number of research papers in international conferences and journals.

Dr Md Rafiul Hassan is working as an Associate Professor of Computer Science in University of Maine at Presque Isle (UMPI), USA. Before joining at UMPI Dr Hassan worked as an Associate Professor in Department of Computer Science and Information Science at King Fahd University of Petroleum & Minerals for more than ten years. He worked in the department of Computer Science and Software Engineering as Research Fellow (Lecturer-B) from 2008 to 2010. He received a Ph.D. in Computer Science and Software Engineering from the University of Melbourne, Australia in 2007. His research interests include Artificial Intelligence, Machine Learning and Computational Intelligence with a focus on developing new data mining and machine learning techniques for big data analysis, cyber-security and IoT. He is the author of around 50 papers published in recognized international journals and conference proceedings. He is a member of Australian Society of Operations Research (ASOR), and IEEE Computer Society; and is involved in several Program Committees of international conferences. He also serves as the reviewer of a number of renowned journals such as BMC Breast Cancer, IEEE Transactions on Fuzzy Systems, Neurocomputing, Applied Soft Computing, Knowledge and Information Systems, Current Bioinformatics, Information Science, Digital Signal Processing, IEEE Transactions on industrial electronics and Computer Communications.

Mohammad Mehedi Hassan received the Ph.D. degree in computer engineering from Kyung Hee University, South Korea, in February 2011. He is currently a Professor with the Information Systems Department, College of Computer and Information Sciences (CCIS), King Saud University (KSU), Riyadh, Saudi Arabia. He has authored and co-authored more than 365+ publications including refereed journals (333 SCI/ISI-Indexed Journal papers, 42 conference papers, 1 book, and 2 book chapters. His research interests include cloud computing, edge computing, the Internet of Things, body sensor networks,big data, deep learning, mobile cloud, smart computing, wireless sensor networks, 5G networks,and social networks. He has served as the Chair and a Technical Program Committee Member in numerous reputed international conferences/workshops, such as IEEE CCNC, ACM BodyNets, and IEEE HPCC. He was a recipient of a number of awards, including 2021 Outstanding Editors Award from Future Generation Computer Systems journal, Distinguished Research Award from College of Computer and Information Sciences, KSU 2020, Best Conference Paper Award from IEEE Int'l Conf on Sustainable Technologies for Industry 4.0 (STI) 2020, Best Journal Paper Award from IEEE Systems Journal in 2018, Best Conference Paper Award from CloudComp in 2014 conference and the Excellence in Research Award from College of Computer and Information Sciences, KSU (2015 and 2016). He is one of the top 2% Scientists of the world in Networking and Telecommunication field. He is one of the top computer scientists in Saudi Arabia as well. Recently, his six publications have been recognized as the ESI Highly Cited Papers.