Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2021) 29: 2561 – 2579

doi:10.3906/elk-2012-41

Multiagent Q-learning based UAV trajectory planning for effective situational awareness

Erdal AKIN[1,*], **Kubilay DEMİR**[2], **Halil YETGİN**[2,3]
[1]Department of Computer Engineering, Bitlis Eren University, Bitlis, Turkey
[2]Department of Electrical and Electronics Engineering, Bitlis Eren University, Bitlis, Turkey,
[3]Department of Communication Systems, Jožef Stefan Institute, Ljubljana, Slovenia

Received: 08.12.2020 • **Accepted/Published Online:** 14.06.2021 • **Final Version:** 23.09.2021

Abstract: In the event of a natural disaster, arrival time of the search and rescue (SAR) teams to the affected areas is of vital importance to save the life of the victims. In particular, when an earthquake occurs in a geographically large area, reconnaissance of the debris within a short-time is critical for conducting successful SAR missions. An effective and quick situational awareness in postdisaster scenarios can be provided via the help of unmanned aerial vehicles (UAVs). However, off-the-shelf UAVs suffer from the limited communication range as well as the limited airborne duration due to battery constraints. If telecommunication infrastructure is destroyed in such a disaster, maximum coverage to be monitored by a ground station (GS) using UAVs is limited to a single UAV's wireless coverage regardless of how many UAVs are deployed. Additionally, performing a blind search within the affected area could induce significant delays in SAR missions and thus leading to inefficient use of the limited battery energy. To address these issues, we develop a multiagent Q-learning based trajectory planning algorithm that maintains all-time connectivity towards the GS in a multihop manner and enables UAVs to observe as many critical areas (highly populated areas) as possible. The comprehensive experimental results demonstrate that the proposed multiagent Q-learning algorithm is capable of attaining UAV trajectories that can cover significantly larger portions of the critical areas summing up to 43% than that of the existing algorithms, such as the extended versions of Monte Carlo, greedy and random algorithms.

Key words: Reinforcement learning, postdisaster recovery, trajectory planning, flying ad-hoc networks

1. Introduction

Today, hardly a week passes by without news and scenes of natural disasters disrupting and destroying people's lives around the world [1–3]. Although many countries have significantly devoted efforts with a view to reducing the severe impact of natural disasters, technical and scientific solutions are still in the long queue of debate. With this in mind, industry, governments, and academia have been heavily dedicating their resources to disaster management processes, from mitigation, preparedness, and response to recovery and reconstruction, one of which is building critical applications using unmanned aerial vehicles (UAVs) [4].

Owing to their low cost, flexible maneuverability, and high vantage point capability, UAVs have been extensively used in disaster recovery applications to strengthen the assessment of a particular disaster [3]. However, commercial off-the-shelf UAVs suffer from the restricted communication range and battery capacity in the use of surveillance over a relatively large area. In particular, if the telecommunication infrastructure is

*Correspondence: e.akin@beu.edu.tr

disrupted in a postdisaster scenario, this further prevents the use of the UAVs by the search and rescue (SAR) team from fulfilling their missions.

Many of the existing works about the multi-UAV based surveillance focus on how to design optimum trajectories for the UAV networks considering the constraints, i.e. battery energy and connectivity [5–7]. Other researchers also aim at designing optimum paths considering these constraints but rather for constructing an aerial base station network providing an emergency call service for people within the disaster area [8]. Moreover, to design optimal trajectories for the UAVs, while many existing works attempt to exploit mathematical optimization tools [5, 7], others take advantage of the reinforcement learning techniques, such as Q-learning [8, 9]. None of the existing works focus their attention on how to provide coverage for prioritized targets with optimal UAV trajectories accounting for the energy and connectivity constrains, except [10]. However, Zhang et al. [10] stress on the access to the prioritized areas using a single UAV within a shorter time period without considering the aforementioned constraints.

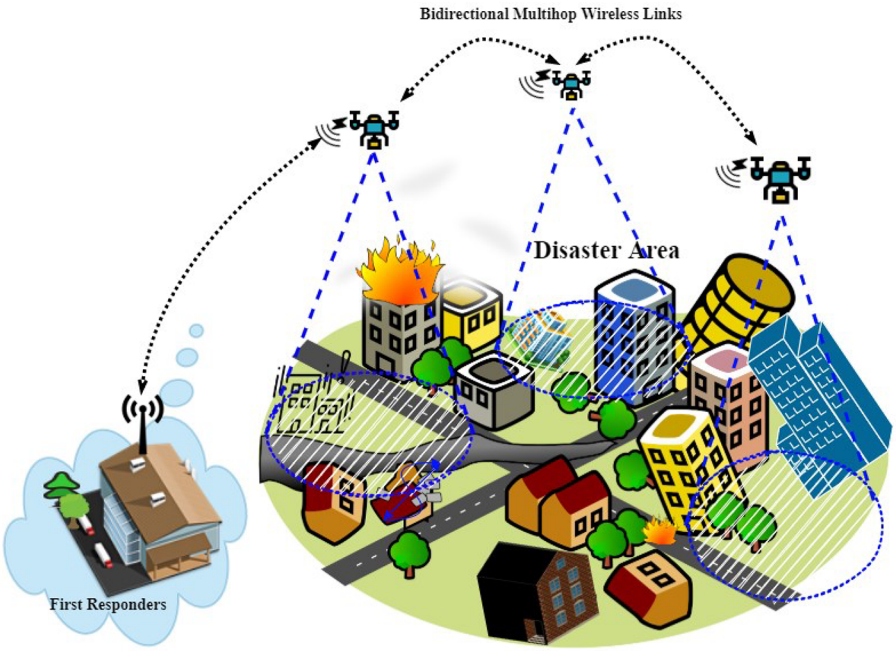

Figure 1. An example of postdisaster scenario leveraging UAVs.

To narrow down this gap, we focus on a research question that is how we can design multiple UAV trajectories covering a maximum number of higher priority areas by considering both the connectivity and energy constrains for the UAVs flying over the affected area for the sake of a quicker and more efficient situational awareness.

To address this issue, first we propose a UAV-based situational awareness framework, as portrayed in Figure 1, where a predetermined number of UAVs are leveraged for the observation of the affected area by means of relaying the gathered critical information through the multihop wireless communication towards the first responders. This approach also paves the way for covering more areas than their standard capacities. Second, we propose a multiagent Q-learning based UAV trajectory planning algorithm, namely MQUTP, which provides trajectories for the UAVs to maximize the number of discovered critical areas, i.e. highly populated areas (higher priority), during their first flight (Before the first battery replenishment) in a postdisaster scenario.

Additionally, during the flight of the UAVs, the proposed Q-learning algorithm ensures that each UAV remains connected to the ground station (GS), either directly or via relaying over a multihop communication. This is to ensure that all the gleaned critical information is communicated towards the first responders for a quick assessment of the disaster so as to respond with the most suitable SAR mission [3, 4].

Finally, we perform a comprehensive experiment to assess the performance of the trajectories provided by the proposed algorithm with regards to the coverage of the critical areas in a prioritized map. The performance of the proposed algorithm is also compared to extended versions of Monte Carlo, random and greedy algorithms, each of which is implemented to provide trajectories for the UAVs while maintaining all-time connectivity. The results demonstrate that the proposed UAV trajectory design algorithm, the so-called MQUTP, provides optimum UAV trajectories visiting significantly higher number of the critical areas up to 43% in comparison to the existing algorithms in their first flight. The main contributions of this paper are outlined as follows.

- We propose a novel framework for UAV-based situational awareness, where the UAVs are able to fly beyond their communication range with the aid of multihop communication towards the GS exploiting a population based prioritized map, which helps the first responders to observe more of the prioritized areas than the existing off-the-shelf UAVs.
- We develop a multiagent Q-learning algorithm for designing efficient UAV trajectories so as to observe maximum number of critical areas taking into account the communication range and battery limitations.
- The effectiveness of the proposed multiagent Q-learning based algorithm has been extensively studied by means of benchmarking against several existing algorithms.

The remainder of this paper is structured as follows. In Section 2, we provide the related work for Q-learning based UAV trajectory design. Section 3 describes the UAV network model and the assumptions, while the proposed multiagent Q-learning algorithm is elaborated in Section 4. Simulation details and performance analyses are provided in Section 5, while the paper is concluded in Section 6.

2. Related work

A paucity of contributions on the trajectory design of multiple UAVs leveraging multiagent reinforcement learning (RL) has been proposed within the scope of several applications, such as resource allocation including sum transmit rate maximization, and the selection strategies of user, power level and subchannel [8, 11, 12], efficient spectrum management [1], fast and accurate localization [2, 13] and maximum data collection [14].

For resource allocation, Liu et al. [8, 15] propose a framework for jointly optimizing the trajectory and power control of multiple UAVs relying on user's mobility information, where a multiagent Q-learning-based algorithm is developed for identifying the most beneficial positions of UAVs according to the initial position of users. Within this framework, a throughput gain of about 17% is achieved. Similarly, Liu et al. [16, 17] propose a novel UAV-based communication framework for maximizing mobile user experience by optimizing 3D deployment and dynamic movement of multiple UAVs, where first user-based cell partitioning is realized, then RL-based deployment of UAVs is performed, followed by a RL-based UAV movement algorithm when users are roaming. The findings introduce useful guidelines for the deployment and movement of UAVs in the interest of increased mobile user satisfaction. Another attempt of a multi-UAV resource allocation problem is formulated in [11, 12] focusing on the maximization of long-term awards jointly providing user, power level and subchannel selection strategies taking into account uncertainty and dynamic nature of the environment, where

each UAV acts as a learning agent and each iterative solution is an action performed by the respective UAVs. Their proposed multiagent RL framework solely based on the local information provides reasonable performance when compared to the scenario considering global information exchange among UAVs.

A spectrum sharing mechanism is proposed in [1] to satisfy the demand for high-throughput real-time data transmission for disaster monitoring applications. Authors apportioned UAVs into relaying (spectrum owners) and sensing (performing disaster relief missions) segments on which the tasks are assigned according to remaining resources. UAV positions are determined by the level of priority of the impacted areas, where UAVs autonomously adapt their position with a RL algorithm with the goal of maximized throughput and lifetime. Similarly, authors of [18] devise an effective trajectory for multiple UAVs using RL algorithm so as to fairly distribute the limited spectrum resources in a decentralized manner, which is built upon a sense-and-send protocol. Compared to its single- and multiagent counterparts, the proposed algorithm converges faster and is more beneficial to UAVs while performing their tasks.

In [13] UAV swarms are leveraged for the localization and tracking of an radio frequency-based mobile target using a multiagent RL algorithm, which is capable of eliminating redundant trajectory paths considering the uncertainty of channel estimate. Higher localization probability and shorter searching time performances are exhibited against single agent RL algorithm. Similarly, Ebrahimi et al. [2] propose a RL-based trajectory design strategy enabling UAVs to autonomously adapt their positions so as to improve the localization accuracy of multiple target objects with the shortest time and path length, and the lowest energy dissipation possible.

Cui et al. [14] formulated a RL problem for solving optimal trajectory design with the goal of maximized collected data in the presence of uncertainty factors under flight-time constraints. Finally, a compelling application of UAVs performing SAR missions in a postdisaster scenario can be observed in [19], where a UAV is navigated towards a wireless signal source, based on the received signal strength (RSS) levels fed to RL algorithm, that is attached to the victim without relying on GPS coordinates. Their RSS-based RL algorithm outperforms the location-based counterpart in terms of convergence speed, required steps per episode and total length of the final trajectory.

None of the abovementioned studies have investigated the research question that is how to design trajectories providing a maximum coverage of critical areas considering the energy and connectivity constraints in the UAVs' first flight for performing situational awareness. Therefore, in this study, we address this research question by developing a UAV-based situational awareness framework and a multiagent Q-learning based trajectory design algorithm.

3. System model

In this section, we first introduce the system model, and then identify and formulate the problem.

3.1. System description

We consider a network composed of a ground station (GS) and $N = \{N_1, N_2, ..., N_i\}$ UAVs, which horizontally fly at a certain altitude with the purpose of providing situational awareness in a postdisaster scenario. We assume that each municipality can leverage a few commercial UAVs for situational awareness, whilst it may be impractical for each municipality to acquire a plethora of UAVs maintaining the uninterrupted connectivity over a large target area. Therefore, by establishing a UAV mesh network towards the GS and maintaining the connectivity between them, GS can monitor much larger and farther areas in an energy efficient manner owing

to multihop communication. Restricted communication range of UAVs is depicted by D, where each UAV is required to communicate to either GS or any other UAVs by keeping the distance within D in order to ensure ongoing connectivity. Moreover, to incorporate battery-capacity constraint of UAVs, without loss of generality, a UAV N_i can fly maximum K_i number of units in the grid, whose Euclidean distance is proportional to its distance to GS. Furthermore, we consider a slotted-time monitoring of the units, i.e. $T = \{t_1, ..., t_i\}$, while remaining airborne at a certain altitude. For the sake of simplicity and tractability, we assume that each UAV contains the same battery-energy and that consumption rate of the energy during its flight is fixed to be the same for each time-slot t_n.

Moreover, we consider that each UAV can fly up to a certain time, i.e. $T_{max} = 100\ t$, due to the battery limitation. During the training process, the proposed algorithm allows the last UAV to travel from and return back to the GS within T_{max}. To this end, we limit the number of the grid cells that a UAV can fly, using the formulation $K_i = T_{max}\ /\ 6\ (N\ +1\ -\ i)$, which allows UAVs to fly certain numbers of grid cells proportional to flying order of UAVs. By doing so, we implicitly enforce the UAVs to maintain their connection with corresponding UAV(s) during their travels. However, during their travel, instantaneous communication failures may occur due to the instant violation of communication range. The main reason for this issue is due to the training of the UAVs in a sequential way. However, the UAVs autonomously fly to their destination without requiring telemetry connection. They require the communication to instantly deliver the monitoring data to the ground station. Therefore, the monitoring data that is missed during the instantaneous communication failures can be delivered from the records stored on UAVs once connection is recovered. Furthermore, after UAVs returning back to the GS (when T_{max} is reached), their batteries are replenished for the second flight.

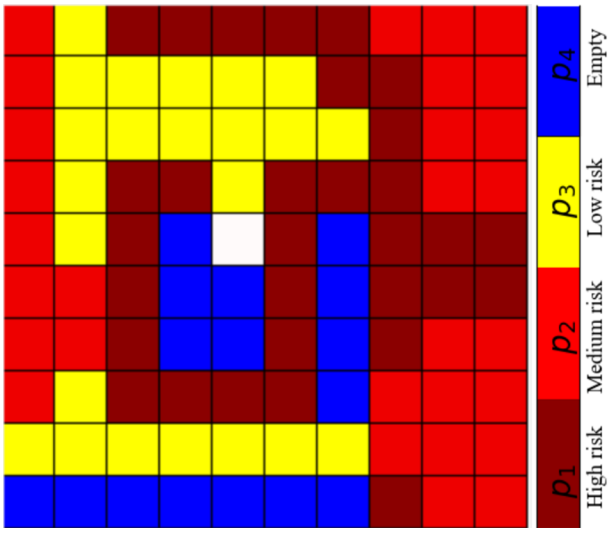

Figure 2. An exemplifying grid-based map utilized for our postdisaster scenario.

Given that the cities include areas with diverse populations, devising intelligent trajectory for UAVs against arbitrary or sub-optimal paths could increase the likelihood of finding more victims under debris. To this end, we build a priority areas map as in [20, 21] by taking into account the population of a part of İstanbul (Maltepe-Pendik-Sultanbeyli) obtained from ArcGIS website [1]. The priority map is split into M fine-grained

[1] ArcGIS (2021). Population Density Map of Istanbul [online]. Website https://www.arcgis.com/apps/View/index.html [accessed 23 November 2020]

units.

In order to fulfil the task using the Q-learning algorithm, we convert the said real-world population density map into grid-based priority areas allotting corresponding rewards to cells of the grid based on the density of population. Then, the task is converted into a game, in which each agent (UAV) is expected to conduct a jointly-decided energy-efficient trajectory and gain maximum reward while sticking to the objectives providing uninterrupted connectivity, and ensuring to observe as many priority areas as possible.

3.2. Problem formulation

The task of the UAVs in this game is to observe as much areas, with high priority owing to their high population density, as possible within their first-time battery usage without replenishing. However, while performing this task, the agents should obey the restrictions listed below in order to gain maximum reward, indicating the maximum number of visits over the highest priority grids potentially without repetitions due to battery-energy constraint.

- Minimizing the number of visits for the areas already visited by either other UAVs or itself.
- A certain transmission range between a given UAV and any other UAV (or GS) cannot be violated to ensure the desired ongoing connectivity.
- The maximum UAV flight time, thus the maximum number of cells that can be visited, is constrained by the battery-capacity of the UAVs.

In the priority map, as exemplified in Figure 2, we divide the grid cells into four different priority classes, i.e. high p_1, medium p_2, low p_3 risk areas, and empty p_4 areas. We can simply generalize the task with the following equation;

$$Y_i^e(R) \leftarrow \sum_K max(R(s')), \tag{1}$$

where $Y_i^e(R)$ keeps maximum total reward earned for i_{th} UAV N_i for K steps in each episode (e). Here, $s' \in S$ represents the next state where the UAV moves from the current state $s \in S$. $R(\cdot)$ indicates the reward function that calculates a reward for a given state.

The simple problem, introduced in Eq. (1), can be solved by leveraging dynamic programming techniques. However, UAVs do not have access to the full state transition and reward model in a simple form. In other words, when the grid cell is already observed by a UAV, then the priority of that particular grid cell is decreased a certain point for other UAVs planning a visit to the grid cell. Moreover, since each agent struggles to maximize its reward individually, the total reward of the agents (UAVs) might be not maximum in most cases and is suboptimal. In particular, this case is likely to happen when UAVs maintain the connectivity among each other, and the closest UAV to GS moves towards an undesired direction (albeit it may be a better decision for this individual UAV), the remaining UAVs could follow that particular UAV at the expense of the total reward. To circumvent this issue, the solution algorithm must generate the paths of UAVs covering as high priority grid cells as possible with the aid of collaborative decisions of UAVs rather than an individual one.

4. Q-learning based UAV trajectory design

In this section, we first introduce Q-learning preliminaries and then the description of how we model the problem as a game. Finally, we elaborate on how a single agent and multiagent UAV network play this game.

4.1. Q-learning preliminaries

Reinforcement learning (RL) is a goal-driven technique by which an agent is expected to learn for a successful completion of a specific task within a dynamic environment, whereby a numerical reward mechanism is employed through trial-and-error interactions with no prior knowledge [16]. The core motivation behind RL is to enable an agent to efficiently decide on a series of actions in order to discover a way of maximizing the overall reward value, which represents the success level of an action when having completed a requested task. Generally speaking, each agent decides on actions according to the reward received from its environmental observations, which is then followed by a representative form of the environment's state. Therefore, an agent includes two main components; i) a policy which is a mapping approach to select the appropriate actions with respect to outputs of the environment, and ii) a learning algorithm that is responsible for finding an optimal policy based upon the reward using actions and states.

In the literature, there are subsequent learning algorithms under the umbrella of RL, one of which is the Q-learning algorithm that seeks for the best action to be taken based on the current state. Q-learning is considered as an off-policy since it learns from actions that are not within the current policy, e.g., arbitrary actions. More explicitly, it estimates the reward for future actions given the current state and appends the learned-value to the new state without obeying any greedy policy, which is obtained based on an equation, in particular using Bellman equation [12]. Q-learning relies on an agent that learns from an environment through episodes without relying on any prior knowledge. The agent learns with the aid of a table, the so-called Q[S, A], where S indicates the set of states, whereas A records the set of actions [12]. Q-learning algorithm iterates with the help of corresponding values in $Q(s,a)$ table, namely Q-table, representing the anticipated reward when an agent carries out an action a in a given state s. The Q-table is updated after each action a of the agent in a state s, i.e. $Q(s,a)$ using the following formula [22, 23];

$$Q_{t+1}(s_t,a_t) = Q_t(s_t,a_t) + \alpha(r_t + \gamma max_a Q_t(s_{t+1},a_t) - Q_t(s_t,a_t)), \tag{2}$$

where r_t is the reward granted while moving from state s_t to the state s_{t+1}. The maximization operator is utilized for the action value a which then can be used for the forthcoming state s_{t+1}, and $\gamma, \alpha \in (0,1]$ is the discount factor and the learning rate, respectively. Learning rate can simply be described as the fitness of the new value compared to the previous value. The discount factor is used to balance immediate and future rewards [23] [2].

To explorer the environment, the agent exploits an epsilon greedy method rather than exploiting the Q-function equation above for the action selection process. At the start of this method, the epsilon rates are expected to be higher, as the environment is unknown. The agent will gradually explore the environment with arbitrary actions. As soon as the agent learns the environment, the epsilon rate declines and the agent attempts to leverage the Q-function for the action selection process [1].

4.2. Game formulation

In this paper, our main goal is to attain a quick situational awareness within a municipal area in the event of a disaster by means of exploiting multiple UAVs having constrained resources, such as battery-energy capacity while maintaining all-time connectivity. The real-world population density map of the target area is split into

[2]Toward Data Science (2021). Simple Reinforcement Learning: Q-learning [online]. Website https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56 [accessed 10 November 2020]

M grid cells with their respective priority scores based on the population density determining the critical level of the concerned areas as depicted in Figure 2.

To model our problem using game theory, we use the following notations:

- A UAV in the group of UAVs N is represented with N_i.
- S states represent M cell in the game, $S = \{s_1 ... s_M\}$.
- a is an action that an UAV can perform in a state s and $a \in A$, where A includes four actions, i.e. up, down, left, and right.
- $r(s, a)$ denotes the immediate reward when the agent fulfil the action a in the s.
- K_i is the number of the cell that the UAV can move on from GS due to battery-capacity constriction.
- D is the limited communication range that a UAV is allowed to move away from either GS or any other UAV.

The game modeling our problem can be summarized as: the agents/UAVs, N, simultaneously take off from the GS. Each agent moves from the current cell s to the next cell s', $(s, s' \in S)$ with an action a, $(a \in A)$ that provides maximum reward, based on $R(s')$. However, the agent must obey the restriction rules, i.e. the number of cells K_i to move on and the maximum distance D to move away.

4.3. Single agent Q-learning algorithm

Our game modeling the problem includes multiple agents and requires a multiagent Q-learning algorithm to successfully win the game. However, to better explain how the proposed multiagent Q-learning algorithm works, we commence with a single agent based Q-learning algorithm.

In the single agent-based Q-learning algorithm, since the agent does not contain information about the environment (grid-based priority map), first it explores the environment with random steps. Q-learning algorithms save the environment information in a Q-table updating the Q-values each time after the agent fulfils an action a in a state s. The Q-value is updated based on the Q-function $Q_{t+1}(s', a)$ (Eq. (5)). The reward function $r(s, a)$ (Eq. (3)) provides the immediate reward for the Q-function based on the priority score of the grid-based map. By learning the grid environment, Q-learning algorithm builds a policy that pushes the agent in a given state s to perform an action a with the purpose of maximizing the total reward. For obtaining an optimal policy, the algorithm requires a great number of repetition of the game, namely episodes. Obtaining an optimal policy is possible with the convergence of the total reward value. To attain the convergence at a lower complexity, Q-learning algorithm needs to follow a strategy that is composed of a certain amount of random steps and policy based steps. These strategy are called as the epsilon greedy strategy, whose value shows the likelihood of a random step throughout the game episodes.

4.4. The proposed multiagent Q-learning algorithm, MQUTP

The proposed MQUTP algorithm is a centralized approach, where the computation of the trajectories for all UAVs are maintained by a back-end computer. Since the population density map is already known and there is nothing to change a UAV's decision while performing its task during the flight, the centralized approach could help to make more reliable decisions.

In the real scenario, all UAVs take off from the GS, simultaneously. However, in the training process of the algorithm, in each episode, each UAV traces its trajectory in turn, namely after N_i arrived its terminal cell, N_{i+1} can start to trace its trajectory towards its terminal cell. In each episode, a grid cell become a terminal cell for a UAV when it violates either communication range D or battery-capacity K_i constraints. The terminal cell is unknown at the beginning of each episode. The flow chart of path computation process for each UAV is depicted in Figure 3.

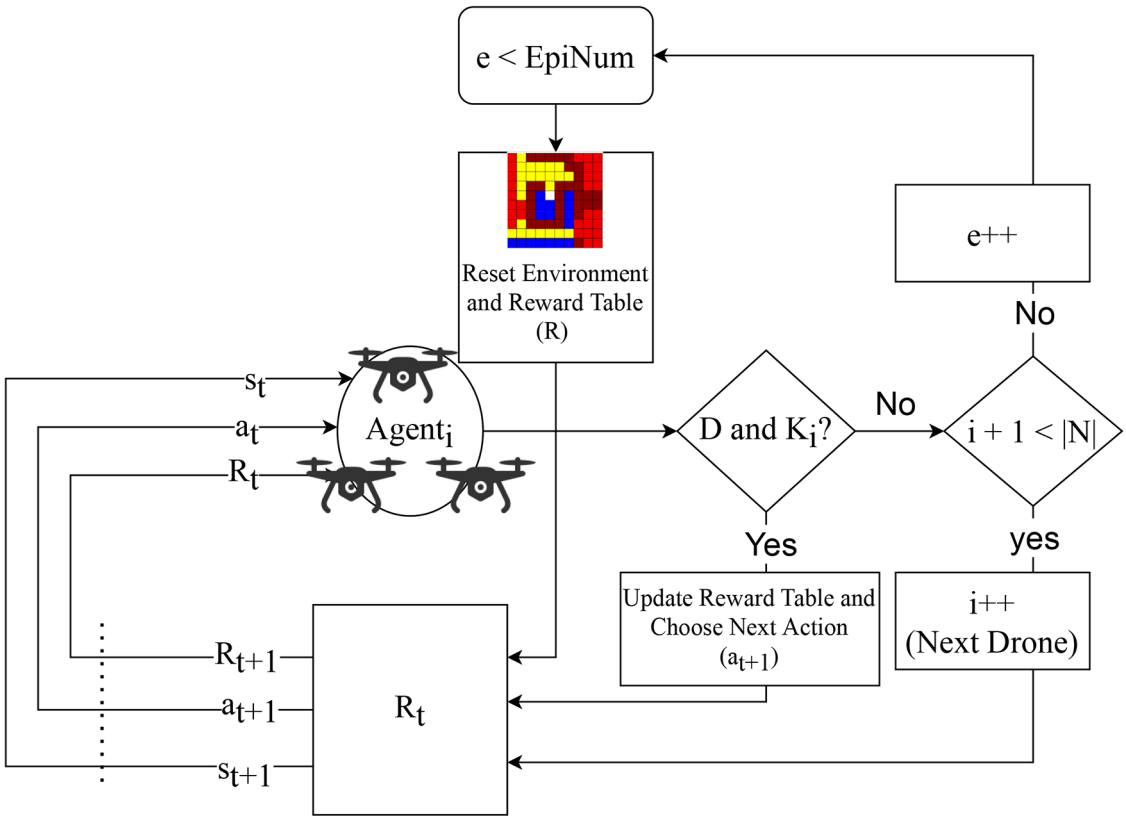

Figure 3. Flow chart of the trajectory design for UAVs.

In the proposed multiagent Q-learning algorithm, all agents use and update a single Q-table versus multiple Q-tables in order to accelerate the convergence of the UAVs. However, using single Q-table could cause all N UAVs taking off from the same point to follow the same path on the grid. Therefore, the proposed algorithm also updates the reward table based on the activities of the UAVs. For example, when UAV N_i observes a cell/state s (regardless of its action a), the algorithm decrease the priority score of that state/cell in the reward table. By doing so, the algorithm struggles to send all N UAVs to distinct areas to discover more cells/states. However, in each new episode, whereas the reward table is reset, Q-table is saved. As a result of this, the proposed algorithm carries out a trade-off between rapid convergence and diversity of the trajectories, which is, to the best of our knowledge, the first attempt in Q-learning algorithms.

When UAV N_i (agent) aims to visit a s', the reward function $R(s)$ of the proposed Q-learning algorithm considers two main parameters in the calculation of the reward. These parameters are the priority score of the

state p_j and the number of visits in the state/cell s'. The reward function is then formulated as follows:

$$R(s') = \begin{cases} P_j(s') - V \cdot \theta, & \text{if } P_j(s') - V \cdot \theta < \Phi. \\ \Phi, & \text{otherwise,} \end{cases} \tag{3}$$

where s' shows the next state, and $P_j(s')$ indicates the priority score p_j of that grid, while V denotes the number of the visits at the grid of the other UAVs and θ is a constant factor to adjust the influence of the other UAVs' visits at a cell. Moreover, we fix the minimum reward to a constant value, Φ, (if $\Phi < 0$, than it is a punishment).

In the proposed multiagent Q-learning algorithm, we do not train each UAV individually, but rather in a collaborative manner. Subsequently, the algorithm finds the best trajectories for all UAVs, which provides the maximum total reward over the observed critical areas. To this end, for the training purpose, we develop a total reward formula R^*, as in Eq. (4), which calculates the total reward of all UAVs N in a single episode, where each UAV can move on maximum K cells and s_j^i denotes jth step of a given UAV N_i.

$$R^* = \sum_{i=0}^{|N|} \sum_{j=0}^{K} R(s_j^i) \tag{4}$$

In the training process, each UAV decides its action based on the Q-value of the Q-table or randomly, which is called as the epsilon-greedy approach. In our experiment, we observe that when the epsilon value (randomness rate) starts with a high value, e.g., 1 and is gradually reduced down to 0.0001, the algorithm produces much better trajectories (total reward).

Each agent updates Q-values of the same Q-table using the following equation that calculates the expected future reward when the agent aims at moving onto a state s' with an action a.

$$Q_{t+1}(s', a) = Q_t(s, a)(1 - \alpha) + \alpha(R(s) + \gamma max_a Q_{t+1}(s', a)) \tag{5}$$

Algorithm 1 demonstrates how the proposed multiagent Q-learning algorithm works. At the beginning, the algorithm initializes a Q-table ($|S|$ x $|A|$) with zero values and the reward table R is constructed based on the priority scores of the map. The proposed multiagent Q-learning algorithm trains the agents collaboratively by enabling the agents to perform their tasks in turn in each episode. To trace their trajectories in each episode, based on the current epsilon value, agents take their actions either randomly or based on the Q-table in their turn. When an agent visits a state, the corresponding reward table value and Q-table value are updated using Eqs. (3) and (5), respectively. In each episode, each agent would have a trajectory from starting location (GS) to its own termination state. If an agent takes more steps than K_i or goes further than D from the closest UAV (which is obtained by getCommDist() modular), the last state becomes the agent's termination state. Therefore, agents obtain a trajectory for each episode. At the end of the simulation, each agent is able to trace a trajectory which provides the maximum total reward for the total observation. Agents calculate and record total reward with respect to Eq. (4).

5. Performance evaluation

In this section, we first describe our simulation settings and then compare the proposed MQUTP algorithm against extended versions of three existing algorithms, followed by result discussions.

Algorithm 1: The proposed multiagent Q-learning based UAV trajectory planning algorithm (MQUTP).

Initialization: $Q_t[s,a] = 0$, Reward table R is built based on the priority map, D, K_i, ϵ and $EpiNum$ (episode number).
foreach $EpiNum$ **do**
  Reset the reward table R
  Reset Environment
  foreach $|N|$ **do**
    actionNumber = 0
    while *True* **do**
      actionNumber++
      if $random < \epsilon$ **then**
        select action randomly
      else
        Choose an action a = $argmax_a(Q(s',a))$
        if $actionNumber > K$ **then**
          break
        end
        if *getCommDist()* $> D$ **then**
          break
        end
        Update Q-table using Eq. (5)
        Update reward table R based on Eq. (3)
      end
    end
  end
end

5.1. Simulation settings

In our simulation, we set the area affected by the earthquake event as a square with a size of $10x10$ units of square each of which has separated by distinct colors, as illustrated in Figure 2. Four different colors represent the density level of the population in respective grid cells as follows:

- Dark red cell is a high-risk area and its priority value is p_1.
- Red cell is a medium-risk area and its priority value is p_2.
- Yellow is a low-risk area and its priority value is p_3.
- Blue is empty area and its priority value is p_4.

We initialize our reward table ($R(s)$) based on the priority score (level of risk). Since observing risky areas is vitally important, we assign the rewards (priority score) from highest to the lowest for the corresponding risk areas, i.e. high, medium, low and empty. In this experiment, we assign 150, 125, 25, 0 for p_1, p_2, p_3 and p_4, respectively. During the training process, the proposed Q-learning algorithm employs Eq. (3), where $s \in S$ depicts all possible states on the map. In addition, we define a constant factor (θ) of 150 and minimum reward (punishment) value (Φ) of -150.

We train the proposed multiagent Q-learning based UAV trajectory planning algorithm (MQUTP) in 5000 episodes (E). For each episode, each UAV N_i operates until it faces termination conditions, i.e. $D = 5$ square units and $K_i = T_{max}$ / 6 (N +1 - i) square units, where i is index of a respective UAV. The hyperparameters used in simulation are summarized in Table . We base our work on an off-the-shelf UAV specification to solidify

our realistic application scenario. With this in mind, we consider Bayraktar Mini UAV[3] whose endurance time is around 1 h and maximum flight speed is around 60 km/h. If the time unit t is 3 min, than the velocity of UAVs is set to 3 km per grid cell. In the (10 × 10) grid used in our experiments, we consider one edge of a grid cell to be 3 km. We developed our simulations with Python 3.7.1 version using Numpy and pandas libraries. We used Tkinter Canvas Widget for graphical presentations, such as map, UAV movement and terminal positions[4].

Table . Main hyperparameters for MQUTP.

Parameter	Value
Episodes E	5000
Discount rate γ	0.9
Number of UAVs N_i	$1, 2, 3, 4, 5$
Learning rate α	$0.5, 0.10, 0.15$
Epsilon e	$1 : 0.0005 : 0.0001$
K_i value	$T_{max}/6(N+1\text{-}i)$
Priority values p_1, p_2, p_3, p_4	$150, 125, 25, 0$
Punishment Φ	-150
Limited communication range D	5 cells

Having provided the simulation details, now we evaluate the performance of the proposed MQUTP algorithm against the extended existing algorithms, which we named extended-greedy (eGreedy), extended-random (eRandom), and extended-Monte Carlo (eMonteCarlo) algorithms:

- eGreedy: This is an extended version of the greedy algorithm, where each drone maximizes its own instantaneous reward by selecting the high-risk areas in each step. In each step, it also records the observed units in order to prevent selecting the same unit again. Therefore, it always aims at selecting actions with the highest rewards to maximize reward. eGreedy is trained in 5000 episodes.
- eRandom: This is an extended version of the random algorithm. In each step, it selects an action randomly and records the observed areas with the purpose of the avoidance of repeatedly selection of the observed ones similar to eGreedy. eRandom does not include a training process with episodes, as it represents the random movement of the UAVs.
- eMonteCarlo: This is an extended version of the Monte Carlo method. As a further extension to Monte Carlo's randomness approach, in each step of an episode, previously observed areas are avoided. At the end of each episode, eMonteCarlo algorithm picks the trajectory that maximizes the cumulative reward. eMonteCarlo is trained in 5000 episodes.

5.2. Performance evaluation

For a fair performance evaluation, we use the following metrics to compare the algorithms:

- Ratio of observed area to the entire grid (ROA) (%): This is the ratio of the number of observed area to the entire grid. Since our grid has 100 units, it is basically the total number of observed units.

[3]BAYKAR (2021). IHA-16 [online]. Website https://www.baykarsavunma.com/iha-16.html [accessed 26 February 2021].
[4]Github, Inc. (2021). QL-MUTO simulation codes [online]. Website https://github.com/erdalakin-cs/QL-MUTO [accessed 26 Februray 2021].

- Ratio of observed risky areas (RORA) (%): This is the ratio of the obtained score for the observed risky areas to the maximum available score. The maximum available score and the obtained score are calculated based on priority values of observed units.

- Cumulative total score (CTS): This is the total score of the designed trajectories at the end of each episode, where the scores are calculated based on the priority values of the covered units of the grid. We use this metric in our algorithms to demonstrate their convergence performance.

- Cumulative reward (CR): This is the total reward of the UAVs provided by Q-learning algorithm at the end of each episode. We present convergence performance of the algorithm based on the obtained rewards in each episode.

5.2.1. Impact of learning rate parameter (α)

The learning rate (α) is a hyperparameter that aids in identifying to what extent the amount of information obtained in the current state overrides old information obtained the previous state. Hence, it is important to tune the α parameter in order to optimize algorithm performance. To this end, we commence with the analyses for the effect of the learning rate (α) on our proposed algorithm MQUTP, as portrayed in Figure 4, where we demonstrate the performance of the proposed Q-learning algorithm w.r.t ROA and RORA metrics for different set of UAVs, i.e. $N = \{1, 2, 3, 4, 5\}$ by means of adopting the learning rates of $\alpha = \{0.05, 0.10, 0.15, 0.20\}$.

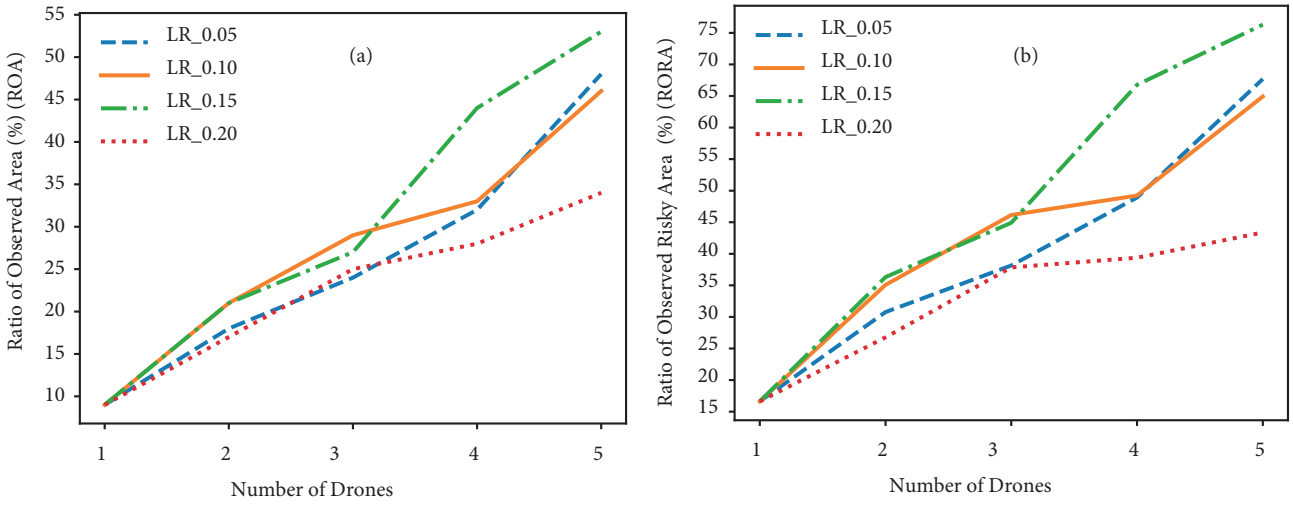

Figure 4. The performance of MQUTP algorithm w.r.t: (a) ROA and (b) RORA metrics vs. the learning rate parameter (α).

Figure 4a shows that the learning rate does not influence the results, when the number of UAV is rather small. However, when the number of UAV increases, MQUTP provides a better performance, particularly for $\alpha = 0.15$. Our well-grounded reason behind this is because if the number of UAV increases, the acquired knowledge begins to change the Q-table values, and thus provoking the selection of an appropriate action for each UAV. Owing to the importance of the trade-off between prior and current knowledge, learning rate α should be carefully tuned, where in our case, $\alpha = 0.15$ provides a superior performance compared to that of $\alpha = \{0.05, 0.10, 0.20\}$ in terms of ROA, as illustrated in Figure 4a.

Figure 4b exhibits the effect of α over RORA. Similar to Figure 4a, the proposed algorithm provides better results when α is set to 0.15. Therefore, we opt for this exact value for the following performance evaluations.

5.2.2. Training efficiency of the proposed algorithm

In this section, we carry out the convergence analysis of the proposed algorithm using cumulative total score (CTS), and cumulative reward (CR) metrics in the case of $N = 3$ and $N = 5$.

Figure 5a reveals that CTS increases while episode number is rising. The convergence starts at around 2000th episode. Thereafter, CTS does not change any further for both $N = 3$ and $N = 5$ scenarios, which confirms the convergence of the algorithm. On the contrary, the episode reward is fluctuating considerably until the 2000th episode. The main reason behind this could be that using an evolving reward table and a shared Q-table complicates the convergence.

The results for the CR metric are similar to the CTS, as illustrated in Figure 5b. However, as we can see in Figure 5b, CR values rapidly falls below zero near convergence. The reason behind this is that since the first agents (UAVs) start to learn optimal actions selecting the units providing maximum rewards in a greedy manner, the following agents (UAVs) gain the negative rewards, according to Eq. (3), when they choose an action through the already computed trajectories.

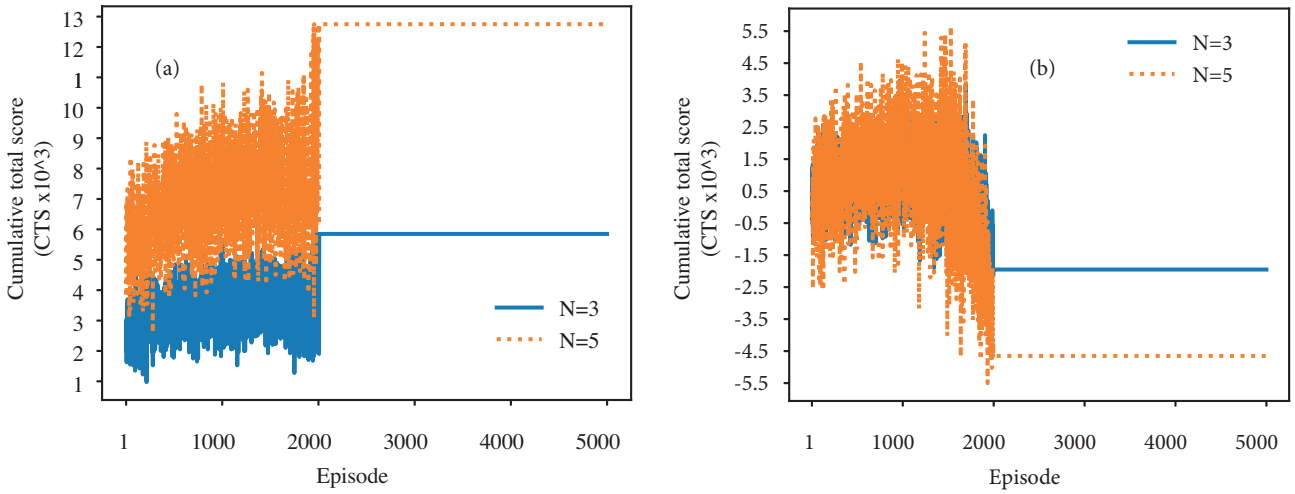

Figure 5. Convergence of MQUTP algorithm vs. the number of UAVs considering metrics: (a) cumulative total score (CTS), (b) cumulative reward (CR).

5.2.3. Performance comparison with the extended algorithms

We now compare the abovementioned extended version of the existing algorithms with the proposed MQUTP algorithm considering $N = \{1, 2, 3, 4, 5\}$ with respect to ROA and RORA metrics.

In Figure 6a, MQUTP yields a better ROA for the set of $N = \{1, 2, 4, 5\}$ UAVs apart from the set $N = \{3\}$. For $N = 3$, eGreedy provides a slightly better performance (ROA) than MQUTP, albeit in terms of RORA, which is in this particular scenario vitally important to observe, MQUTP outperforms eGreedy with about 36% improvement, as portrayed in Figure 6b. Furthermore, as illustrated in Figure 6b, MQUTP algorithm consistently outperforms other algorithms for all N with at least $1\%, 16\%, 22\%, 33\%$ and 43% performance

improvements against their nearest baseline performances, respectively, which are obtained by the eGreedy algorithm.

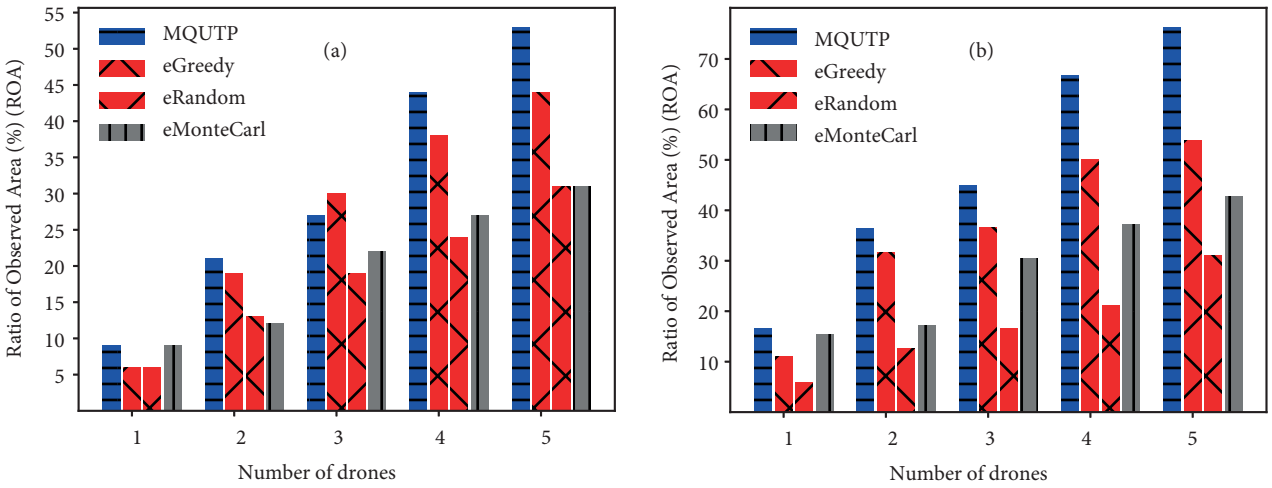

Figure 6. Evaluation of all algorithms in terms of: (a) ROA and (b) RORA performances vs. the number of UAVs.

To gain further evaluation insights, in Figure 7, we also compare the algorithms with respect to their observation performances for the different risky areas by fixing the number of UAVs to 5. In Figure 7, we observe that proposed algorithm MQUTP outperforms the remaining algorithms in terms of their coverage performances of high- and medium-risky areas. eGreedy, eMonteCarlo and eRandom cover around 77%, 55%, and 39% of the high-risky (dark red) areas, respectively, while MQUTP covers around 94% of the high-risky areas. That corresponds to 22%, 70% and 140% of coverage improvements against eGreedy, eMonteCarlo and eRandom algorithms, respectively. Since high-risky areas are of more significance, the algorithm is less concerned for the coverage of the medium-risky (red) areas. However, comparing the others, the results are even better for the proposed MQUTP algorithm, which beats the others with at least 93.5% improvement versus eMonteCarlo algorithm. For empty area, both MQUTP and eGreedy yield a better performance compared to other two algorithms.

5.2.4. Visual comparison with the other approaches

In this section, we compare the algorithms visually over the risk (priority) map. Accordingly, initial map is given in Figure 2, where all UAVs are situated at the ground station (GS), which is depicted by white color in the middle of the grid. When the simulation is terminated for $N = 5$ UAVs, the snapshot of the obtained trajectories and terminal positions of the UAVs are illustrated in Figure 8. Grey colors represent the trajectories of the UAVs and black colors refer to the terminal positions of the UAVs.

As seen in Figure 8, the proposed MQUTP outperforms all the compared algorithms under each performance evaluation scenarios. It can be also readily seen that the worst observation performance is attained by the eRandom algorithm, since it arbitrarily flies over the areas which are mostly empty. eMonteCarlo provides better performance than eRandom. The reason for this is that while eMonteCarlo selects the trajectories with maximum rewards after training in many episodes, eRandom generates the trajectories without training in many episodes.

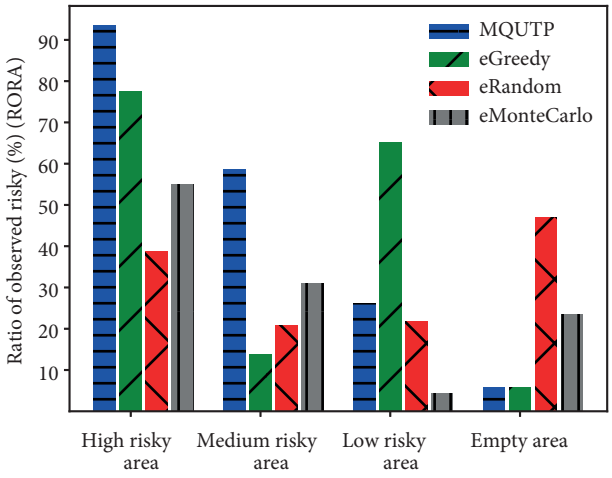

Figure 7. RORA performance comparison of all algorithms for priority areas using $N = 5$ UAVs.

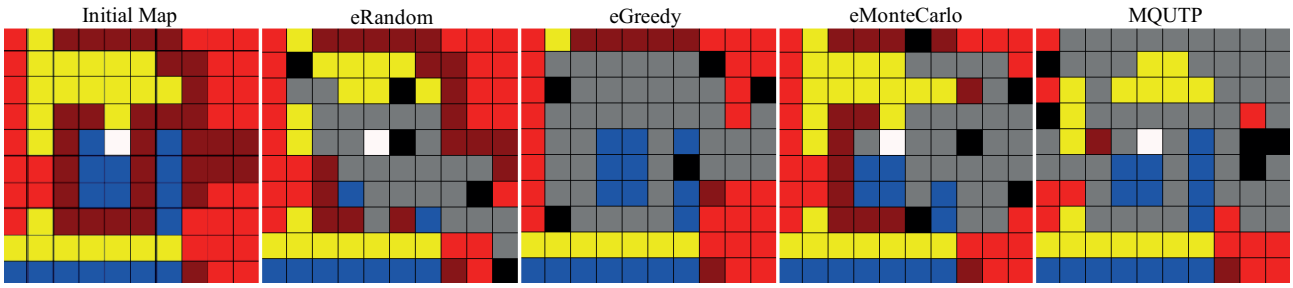

Figure 8. Portraying the final state of five UAVs (dark) for all considered algorithms, which are ordered (from left to the right) based on the coverage performances, where observed areas are delineated in gray color.

5.2.5. Synopsis

In this paper, we aim to maximize the number of observed risk-areas having the largest population with the aid of a set of UAVs. The UAVs have to create a multihop relay to stay connected to the GS in order to dispatch critical information to GS for a quick assessment of the disaster. Accordingly, we proposed a multiagent Q-learning based UAV trajectory planning algorithm (MQUTP). We performed a comprehensive experiment using the prioritized map (Figure 2) to show the performance of MQUTP compared to other extended versions of the existing algorithms, i.e. Monte Carlo, random, and greedy, namely eMonteCarlo, eRandom and eGreedy, respectively.

First of all, we evaluate the proposed MQUTP algorithm for a diverse learning rates $\alpha = 0.05, 0.10, 0.15$ and 0.20 in order to gain insights on which learning parameter value provides a better performance in terms of the coverage performance for the risk-areas. As readily seen in Figures 4a and 4b, $\alpha = 0.15$ is an optimal learning rate for *MQUTP*, particularly when the number of UAVs is larger.

Moreover, Figures 5a and 5b denote the convergence of the proposed MQUTP algorithm, where the UAVs fulfil their tasks in each episode and learn from their mistakes and rewards while moving towards the convergence point. The findings reveal that the convergence point is accomplished around 2000 episodes with a near-optimal solution.

Finally, we compare MQUTP algorithm with eMonteCarlo, eRandom and eGreedy for different set of

$N = \{1, 2, 3, 4, 5\}$ UAVs with respect to ROA and RORA. Figures 6a and 6b demonstrate that the proposed MQUTP algorithm consistently outperforms the other algorithms in terms of ROA and RORA metrics while the number of UAVs is increased from $N = 1$ to $N = 5$. Moreover, in Figure 7, we clearly see that MQUTP algorithm covers more high- and medium-risk areas than that of the other algorithms.

Since our trajectory design is based on a centralized Q-learning algorithm, the trajectories of each drone are calculated at the GS. As the drones fly at the same velocity and altitude, they can exactly follow the trajectories assigned for them. However, during their simultaneous flying, negligible instantaneous communication failures may occur due to the instant violation of communication range. The main reason for this issue is due to the training of the UAVs in a sequential way. However, the UAVs autonomously fly to their destination without requiring telemetry connection. They require the communication to instantly deliver the monitoring data to the ground station. Therefore, the monitoring data that is missed during the instantaneous communication failures can be delivered from the records stored on UAVs once connection is recovered. Furthermore, after UAVs returning back to the GS (when T_{max} is reached), their batteries are replenished for the second flight.

In a nutshell, the results demonstrate that the proposed MQUTP algorithm is an efficient RL-based trajectory design algorithm considering multiple UAVs, since it is capable of covering more risk-areas than the extended versions of the existing algorithms. Additionally, its convergence is proven with a near-optimal solution after conducting a certain number of episodes.

6. Conclusion

In the event of a disaster, such as earthquakes, the first 72 h is of a great importance to save the life of victims. A rapid and easy way to provide situational awareness can be maintained by the aid of UAVs. However, it may be infeasible for every municipality to obtain sophisticated UAVs, but relatively cheap UAVs with the restricted battery-capacity and communication range. For such a scenario, in this study, we focus our attention on the observation of as much further areas as possible prioritizing the critical areas in the map. To address this issue, we propose a multiagent Q-learning based trajectory planning algorithm, which considers devising the best paths for each UAVs by considering two restrictions; i) each UAV can fly a certain amount of distance considering its limited battery capacity, and ii) each UAV has to maintain its connectivity with either GS or any other UAV. By doing so, a relatively large territory in the early stage of a disaster can be monitored using UAVs with the least possible interruptions. To evaluate the performance of the proposed algorithm, we perform comprehensive experiments while comparing to three other existing algorithms, i.e. Monte Carlo, greedy and random. The results demonstrate the effectiveness of the proposed multiagent Q-learning algorithm over the aforementioned algorithms in terms of the ratio of observation of the risk-areas. However, from the demonstrated results, we also infer that training multi-UAVs for trajectory planning may provide suboptimal results especially due to the connectivity constraint. Therefore, for our future work, we plan to extend this work to simultaneous training using more advanced multiagent RL algorithms, such as Deep Q-Learning (DQN).

Acknowledgment

Halil Yetgin was funded by the Slovenian Research Agency (Grant no. P2-0016).

References

[1] Shamsoshoara A, Afghah F, Razi A, Mousavi S, Ashdown J et al. An autonomous spectrum management scheme for unmanned aerial vehicle networks in disaster relief operations. IEEE Access 2020; 2: 58064-58079. doi: 10.1109/AC-

CESS.2020.2982932

[2] Ebrahimi D, Sharafeddine S, Ho P, Assi C. Autonomous UAV Trajectory for Localizing Ground Objects: A Reinforcement Learning Approach. IEEE Transactions on Mobile Computing 2020; 20 (4): 1-1. doi: 10.1109/TMC.2020.2966989

[3] Erdelj M, Natalizio E, Chowdhury KR, Akyildiz IF. Help from the sky: leveraging UAVs for disaster management. IEEE Pervasive Computing 2017; 16 (1): 24-32. doi: 10.1109/MPRV.2017.11

[4] Bekmezci İ, Sahingoz OK, Temel S. Flying ad-hoc networks (FANETs): a survey. Ad Hoc Networks 2013; 11 (3): 1254-1270. doi: 10.1016/j.adhoc.2012.12.004

[5] Hayat S, Yanmaz E, Bettstetter C, Brown TX. Multi-objective drone path planning for search and rescue with quality-of-service requirements. Autonomous Robots 2020; 44 (7): 1183-1198. doi: 10.1007/s10514-020-09926-9

[6] Yanmaz E, Yahyanejad S, Rinner B, Hellwagner H, Bettstetter C. Drone networks: communications, coordination, and sensing. Ad Hoc Networks 2018; 68: 1-15. doi: 10.1016/j.adhoc.2017.09.001

[7] Scherer J, Rinner B. Persistent multi-uav surveillance with energy and communication constraints. In: IEEE 2016 International Conference on Automation Science and Engineering (CASE); Fort Worth, TX, USA; 2016. pp. 1225-1230.

[8] Liu X, Liu Y, Chen Y, Hanzo L. Trajectory design and power control for multi-UAV assisted wireless networks: a machine learning approach. IEEE Transactions on Vehicular Technology 2019; 68 (8): 7957-7969. doi: 10.1109/TVT.2019.2920284

[9] Zhang B, Liu CH, Tang J, Xu Z, Ma J et al. Learning-based energy-efficient data collection by unmanned vehicles in smart cities. IEEE Transactions on Industrial Informatics 2018; 14 (4): 1666-1676. doi: 10.1109/TII.2017.2783439

[10] Deng L, Yuan H, Huang L, Yan S, Lai Y. Post-earthquake search via an autonomous UAV: hybrid algorithm and 3d path planning. In: IEEE 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD); Huangshan, China; 2018. pp. 1329-1334.

[11] Cui J, Liu Y, Nallanathan A. The application of multi-agent reinforcement learning in UAV networks. In: IEEE International Conference on Communications Workshops (ICC Workshops); Shanghai, China; 2019. pp. 1-6.

[12] Cui J, Liu Y, Nallanathan A. Multi-agent reinforcement learning-based resource allocation for UAV networks. IEEE Transactions on Wireless Communications 2020; 19 (2): 729-743. doi: 10.1109/TWC.2019.2935201

[13] Chen YJ, Chang DK, Zhang C. Autonomous tracking using a swarm of UAVs: a constrained multiagent reinforcement learning approach. IEEE Transactions on Vehicular Technology 2020; 69 (11): 13702-13717. doi: 10.1109/TVT.2020.3023733

[14] Cui J, Liu Y, Nallanathan A, Hanzo L. Adaptive UAV-trajectory optimization under quality of service constraints: a model-free solution. IEEE Access 2020; 8: 112253-112265. doi: 10.1109/ACCESS.2020.3001752

[15] Liu X, Liu Y, Chen Y, Wang L, Lu Z. Machine learning aided trajectory design and power control of multi-UAV. In: IEEE Global Communications Conference (GLOBECOM'19); Waikoloa, HI, USA; December 2019. pp. 1-6.

[16] Liu X, Liu Y, Chen Y. Reinforcement learning in multiple-UAV networks: deployment and movement design. IEEE Transactions on Vehicular Technology 2019; 68 (8): 8036-8049. doi: 10.1109/TVT.2019.2922849

[17] Liu X, Liu Y, Chen Y. Deployment and movement for multiple aerial base stations by reinforcement learning. In: IEEE 2018 Globecom Workshops (GC Wkshps); Abu Dhabi, United Arab Emirates; 2018. pp. 1-6.

[18] Hu J, Zhang H, Song L. Reinforcement learning for decentralized trajectory design in cellular UAV networks with sense-and-send protocol. IEEE Internet of Things Journal 2019; 6 (4): 6177-6189. doi: 10.1109/JIOT.2018.2876513.

[19] Chowdhury MMU, Erden F, Guvenc I. RSS-based Q-learning for indoor UAV navigation. In: IEEE Military Communications Conference (MILCOM'19); Norfolk, VA, USA; November 2019. pp. 121-126.

[20] Langford M, Unwin DJ. Generating and mapping population density surfaces within a geographical information system. The Cartographic Journal 1994; 31 (1): 21-26. doi: 10.1179/000870494787073718

[21] Tenerelli P, Gallego JF, Ehrlich D. Population density modelling in support of disaster risk assessment. International Journal of Disaster Risk Reduction 2015; 13: 334-341. doi: 10.1016/j.ijdrr.2015.07.015

[22] Watkins CJCH. Learning from delayed rewards. PhD, University of Cambridge, Cambridge, England, 1989.

[23] Kusy M, Zajdel R. Stateless Q-learning algorithm for training of radial basis function based neural networks in medical data classification. In: Korbicz J, Kowal M (editors). Intelligent Systems in Technical and Medical Diagnostics. Berlin, Germany: Springer, 2014, pp. 267-278.