



UAV swarm path planning with reinforcement learning for field prospecting

Alejandro Puente-Castro¹ · Daniel Rivero¹ · Alejandro Pazos^{1,2} · Enrique Fernandez-Blanco¹

Accepted: 15 January 2022
© The Author(s) 2022

Abstract

There has been steady growth in the adoption of Unmanned Aerial Vehicle (UAV) swarms by operators due to their time and cost benefits. However, this kind of system faces an important problem, which is the calculation of many optimal paths for each UAV. Solving this problem would allow control of many UAVs without human intervention while saving battery between recharges and performing several tasks simultaneously. The main aim is to develop a Reinforcement Learning based system capable of calculating the optimal flight path for a UAV swarm. This method stands out for its ability to learn through trial and error, allowing the model to adjust itself. The aim of these paths is to achieve full coverage of an overflight area for tasks such as field prospecting, regardless of map size and the number of UAVs in the swarm. It is not necessary to establish targets or to have any previous knowledge other than the given map. Experiments have been conducted to determine whether it is optimal to establish a single control for all UAVs in the swarm or a control for each UAV. The results show that it is better to use one control for all UAVs because of the shorter flight time. In addition, the flight time is greatly affected by the size of the map. The results give starting points for future research, such as finding the optimal map size for each situation.

Keywords UAV swarm · Path planning · Reinforcement learning · Q-Learning · Artificial neural network · Agriculture

1 Introduction

New applications of Unmanned Aerial Vehicle (UAV or drones) swarms are developed nearly every day for different problems, such as crop monitoring [1, 2], forestry activities [3], space exploration [4, 5], or military and rescue missions [6]. The main reason for that popularity lies in the advantages offered by UAVs, such as low cost, great maneuverability, safety, and convenient size for certain kinds of maneuvers [7]. However, they also have disadvantages, the main one being battery consumption, which limits flight time. When UAVs are used in a group or swarm, their flight time limitations are reduced. In other words, several UAVs flying simultaneously allows many tasks to be carried out in

less time because flight paths are shorter (Fig. 1). The flight paths of each UAV are shorter when multiple UAVs fly at the same time (Fig. 1d) than if one UAV has to fly over the complete flight environment (Fig. 1c). This minimizes the probability that the UAVs' battery capacities will be insufficient to allow them to fly over the terrain. As a result of the lower energy usage, there is a lower risk of a UAV crashing in the middle of an activity, resulting in less damage.

The use of UAV swarms can also provide fault tolerance. If only one UAV is used and it crashes, the activity must be stopped. However, if there are several UAVs, the surviving UAVs could assume all or part of the duties of the fallen UAV. This ensures that the work is completed to the best of our ability. Suspending a process when a job is urgent, such as in an emergency or a rescue operation, is difficult since time is of the essence. As a solution, even if one of the UAVs fails, the rescue can continue when using swarms of UAVs.

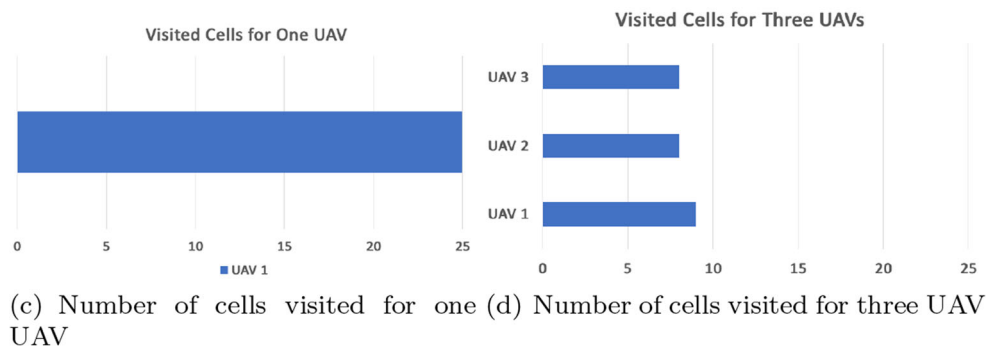
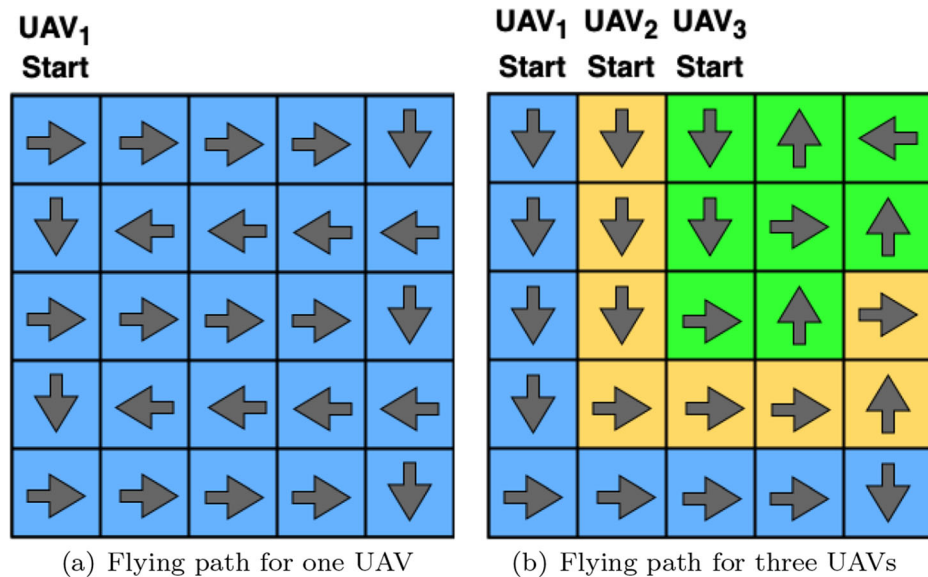
When the first flight tests were conducted, as many operators as UAVs were required, significantly increasing the operational costs. More recently, advances have been registered in the creation of algorithms [8] and telecommunications [9] necessary for the control of the entire swarm with only one user capable of executing the systems. These advances grant better and faster communications between

✉ Alejandro Puente-Castro
a.puentec@udc.es

¹ Faculty of Computer Science, CITIC, University of A Coruna, A Coruna, 15007, Spain

² Biomedical Research Institute of A Coruna (INIBIC), University Hospital Complex of A Coruna (CHUAC), A Coruna, 15006, Spain

Fig. 1 When a swarm of three UAVs is used instead of a single UAV, the number of cells visited drastically changes. When one UAV is used alone, it visits a disproportionately large number of cells compared with the number of cells it would visit if used in conjunction with other UAVs. If the map size is too extensive, the UAV may not be able to visit as many cells



UAVs and grant the fast calculation of collision avoidance paths, so that less human intervention is required if there is any risk. Thus, the operation is less expensive because it requires fewer personnel.

To deal with the complexity of this kind of development, in Swarm Intelligence, different algorithms are proposed that are capable of simultaneously coordinating numerous agents. This coordination is based on a group of individuals that follows common simple rules in a self-organized and robust way [10].

Today, some of these path planning algorithms have military applications. The few civilian applications are usually to follow or reach targets, such as mapping paths through cities [11]. There are few systems oriented to agricultural and forestry use, specially dedicated to the optimization of the field prospecting tasks. Table 1 lists ten publications that demonstrate how different systems solve the Path Planning problem in various scenarios.

The aim of this paper is to use Q-Learning techniques to build a system for solving the Path Planning problem in 2D grid-based maps with different numbers of UAVs. The main contributions of this paper are: 1) a novel system capable of calculating the optimal flight path for UAVs in a swarm

for field coverage in prospecting tasks; 2) a system capable of calculating the flight path of any number of UAVs and with any map size; 3) one of the few systems capable of calculating paths without the need to set targets or provide information other than the actual state of the map; and 4) a study on the difference in the results of using a global ANN for all UAVs and using one ANN per UAV.

This paper has the following structure: in Section 2, there is a brief summary of the current state of the art; in Section 3, an explanation is given of the technical aspects necessary for the development of the proposed algorithm; in Section 4, there is a summary of the results obtained from the experimentation process; in Section 5, the results obtained are discussed; in Section 6, the conclusions reached after reviewing the results obtained are listed; finally, Section 7, lists the possible works and studies into which the problem to be addressed can derive.

2 Background

In the state of the art, there are several approaches, two of which are particularly noteworthy [11]: the first one makes

Table 1 The two primary types of techniques utilized for Path Planning problems with UAVs are summarized in the table below

Technique	Publication	Observations
Reinforcement Learning [12]	Baldazo et al. [13]	Use in emergencies or disasters.
	Yang et al. [14]	Battery energy is considered for path planning.
	Roudneshin et al. [15]	Combine UAVs and ground robots.
	Luo et al. [16]	Requires pretraining phase.
Evolutionary Computation [18]	Speck et al. [17]	Tested on fixed-wing UAVs.
	Duan et al. [19]	Duan et al. [19]
	Zhuo et al. [20]	Path smoothing is required.
	Olson et al. [21]	Focused in maximizing coverage while minimizing flight time.
	Huang et al. [22]	Flight time is considered.
	Perez-Carabaza et al. [23]	Paths should be smoothed.

Each example includes an observation to demonstrate the wide range of approaches to the problem

use of Reinforcement Learning (RL) [12]; while the second one focuses on Evolutionary Computing (EC) [18].

RL algorithms for path planning are the most abundant in the state of the art. For example, Xie et al. use the Q-Learning strategy for three-dimensional path planning [24]. The notion of Heuristic Q-Learning was introduced. This allows a more precise adjustment of the reward depending on the current state and possible actions, leading to faster convergence to the optimal result. Deep Q-Learning is used by Roudneshin et al. to control swarms of UAVs and heterogeneous robots [15]. Rather than using only UAVs, this research incorporates a mix of terrestrial robots into the swarms. However, this is a more challenging problem of swarm path planning than using only UAVs. Due to the differences in restrictions faced by air and ground vehicles, the problem has become more complex to solve. As a result, a land vehicle is more constrained in its mobility and might meet non-geographic impediments.

Others, such as Luo et al., employ the RL algorithm known as SARSA [25], where they tested their Deep-SARSA algorithm in dynamic environments with changing obstacles [16]. They demonstrate how their system behaves in different contexts, demonstrating their utility in the real world. The model requires a pretraining phase, which may limit its application in unfamiliar situations due to the time commitment. When generalizing, Speck et al. integrate object-focused learning with this method in a highly efficient decentralized way [17]. As it was designed for fixed-wing UAVs, this capacity for generalization may be limited, and because these aircraft lack stationary flying capabilities, the configuration of these UAVs restricts the system's use to situations where fixed-wing UAVs are the best option.

On the other hand, there are EC-based methods. For example, Duan et al. combine a genetic algorithm with the VND search algorithm [19]. An initial individual is genera-

ted based on the heuristics of its nearest neighbors and the rest of the initial individuals are configured randomly. The use of the closest neighbors limits the generation of individuals. Especially in the case of many equally close neighbors. In that situation, it is necessary to establish a criterion to determine whether the individual is a member of a group. Recently, Liu et al. employed Genetic Algorithms to adjust ANN for flight path generation [26]. Relying only on the ANN for path computation makes their system dependent on more parameters than weights. Therefore, other parameters, such as learning rates or adjusting the architecture of the ANN should be adjusted.

There are other methods applied to path planning with UAV swarms. For example, Vijayakumari et al. make use of Particle Swarm Optimization for optimal control of multiple UAVs in a decentralized way [27]. They manage to simplify the computation of the problem by means of discretization. They rely on distances for collision avoidance. Although this is a dynamic variable, in certain types of non-stationary flight UAVs, such as fixed-wing UAVs, it does not guarantee collision avoidance. In these cases, a metric that predicts the state of the UAV and the obstacle at future moments in time is of interest and thus makes a decision. Otherwise, the UAV would continue to move forward while the decision is being computed. Li et al. make use of Graph Neural Networks for path computation in robotic systems. Thus, they achieve more capacity for generalization in the face of new cases than other more widely used techniques [28]. Since they are dealing with two ANNs, previous training is necessary in different and very varied cases. Otherwise, ANNs could be overfitted in several flight areas and swarm structures.

As has been shown in the associated literature, systems often require extra map information, such as targets or distance maps. In addition, they use maps with a fixed number of cells. The aim of this work is to propose a system

without the need for extra map information and that works with any map size.

3 Materials and methods

3.1 Problem formulation

Path Planning issues with multiple vehicles are subject to several factors in order to ensure standards of control, cooperation and safe operation while maintaining efficiency and effectiveness. Therefore, it is necessary to be able to solve the problems related to these variables as problems inherent to the main objective.

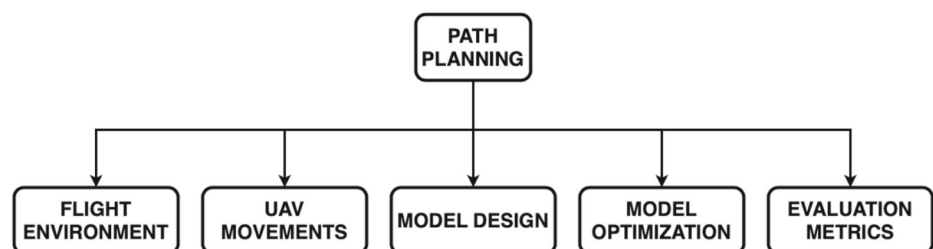
These problems are (Fig. 2): first, to establish the flight environment; second, to define the UAV movements; third, to establish the most appropriate technique for path calculation; fourth, to establish the optimal parameters to solve the problem; and, finally, to define mechanisms to confirm the validity of the proposed model and the satisfaction criteria of the results obtained.

3.2 Flight environments

Despite the existence of well-known tools for flight simulation with UAV swarms, such as AirSim [29], no large datasets are known to be used by most authors. Previously published works, described in Section 2, used fixed squared maps of dimensions between 10×10 and 20×20 . The approach presented in this paper takes a wider point of view allowing the use of arbitrary polygons as maps, e.g. the one presented in Fig. 3a. To do this, the following steps have been followed:

1. The minimum bounding rectangle (MBR) of the map has to be calculated such as in Fig. 3b. The map polygon is surrounded by a rectangle of the smallest possible size based on the combined spatial extent of one or more selected map features [30]. In this case, based on its vertices.
2. The resulting MBR is divided into cell, as shown in Fig. 3c. Cells in the resulting grid have to be labelled as visitable and non-visitable.

Fig. 2 Diagram with the formulation of Path Planning problems. It summarizes all the inherent and necessary problems to guarantee the validity of the final system



3.3 Proposed method

3.3.1 Reinforcement learning

Reinforcement Learning (RL) [12] was chosen as the technique for calculating the optimal path to cover the maps by the UAVs. With this technique, the agents learn the desired behavior based on a trial-and-error scheme of tests executed in an interactive and dynamic environment [12, 31, 32]. The goal is to optimize the behavior of the agent in respect to a reward signal that is provided by the environment. The actions of the agent can also affect the environment, complicating the search for the optimal behavior [33].

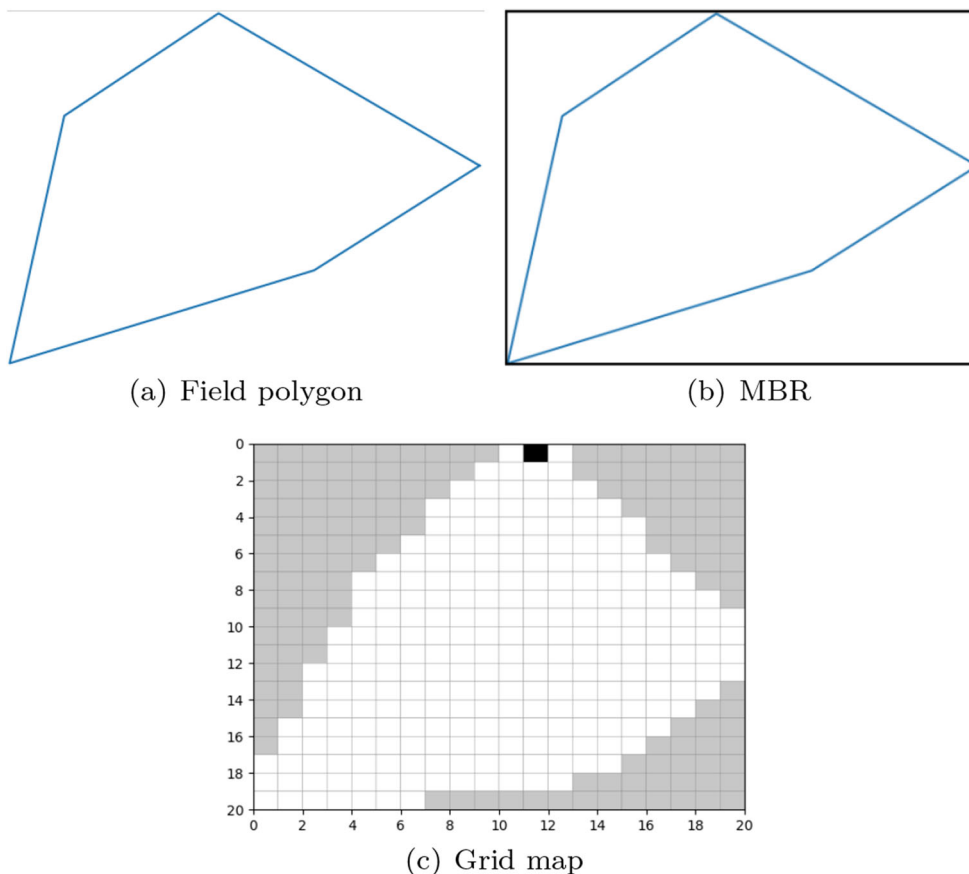
All RL algorithms follow a common structure, the only difference is the learning strategy. There are several types of these strategies which allow the models to deal with different problems. For this paper, it has been decided to use a variant known as Q-Learning [34]. The main motivation is that, unlike other variants, it does not require a model of the environment.

3.3.2 Q-Learning

Classic Q-Learning algorithms [34] are a kind of off-policy RL algorithms, so the agents can use their experience to learn the values of all the policies in parallel, even when they can follow only one policy at a time [35]. It follows a model-free strategy [36], where the agent acquires knowledge by following a policy only by trial-and-error. In this way, Q-Learning convergence towards the optimal solution is greedy, allowing the optimal solution to be reached without being dependent on the decision-making policy. In other words, it makes decisions based purely on the environment surrounding the agent and its interactions with it. In this way, it is guaranteed that the system can work with different types of environments without having to search for the optimal policy that works in all of them. The “Q” in Q-learning stands for quality, which tries to represent how useful a given action is in gaining some future reward.

The most well-known advantage of Q-Learning over other RL techniques is that it can compare the expected utility of various actions without requiring an environment model. The ease with which it generalizes environments

Fig. 3 (a) Example of the polygon representing the area of the indicated field. (b) The same polygon (blue) surrounded by its minimum bounding rectangle (black). The MBR must be as close as possible to the polygon. (c) The MBR divided into cells. The black cell is the starting point of the UAVs. The gray cells are those that cannot be flown over and the white cells are those that can be flown over



without having to model them is the main reason for it having been chosen for this research. In addition, the main difference between these algorithms and other RL algorithms is that they determine the best action based on the values in a table. The table is known as Q-table and the values as Q-values. These values determine how rewarding it would be to perform each action given the current state of the environment. From these values, the action with the highest value for each state is chosen. Typically, models are trained by combining their previous predictions with Bellman’s (1). The equation has different elements: $Q(s, a)$ is the function that calculates the Q-value for the current state (s), of the set of states S , and for the giving action (a), of the set of actions A , r is the reward of the action taken in that state and it is computed by the reward function $R(s, a)$, γ is the discount factor and $\arg \max_{a'}(Q(s', a'))$ is the maximum computed Q-value of the pair (s', a') represented as $Q(s', a')$. The pair (s', a') is a potential next state-action pair. (s' is the next state and it is given by the transition function $T(s, a)$ which returns the state resulting from performance of the selected action. The a' , is each one of the available actions. Through an initial exploration process, the chosen value for γ is 0.91.

$$Q(s, a) \leftarrow r + \gamma \times \arg \max_{a'}(Q(s', a')) \tag{1}$$

Alternatively, in recent years, a modification has arisen called Deep Q-learning. This method differs from the classic Q-Learning [34] in that it seeks to improve the calculation of the Q table through Machine Learning [37] or Deep Learning models [38]. The model is able to abstract enough knowledge to infer the values of the Q table. In this way, it is possible to overcome Bellman’s Equation bias issues in some scenarios [39].

The aim is to improve classical Q-Learning by using small ANNs. In this study, authors chose to use fully connected ANNs with two layers. Using only two-layer learning and decision-making usually takes less time compared with convolutional deep ANNs [40] that other authors propose in their papers. Therefore, the following steps are followed in each Q-Learning experiment:

1. Build the ANN model(s) based on the chosen configuration.
2. Employ the model(s) to determine Q-table values in order to choose the best action for each UAV in the swarm.
3. Train the model(s) according to the consequences of taking each of the selected actions.
4. Select the cases where the flight time required to explore the entire map is lower.

Through empirical experimentation, a network formed by two dense layers [41] has been chosen: the first one with 167 neurons and linear activation function [42] and the second one with 4 neurons and softmax activation function. The chosen optimizer for the ANN was RMSprop [43]. Maps are the only input of the network (Fig. 4). Thus, ANN does not need more information than that included in the maps.

From this point, the system could be used in two different approaches with no clear advantage for either of them. First, a single ANN is developed and used as the control for each of the UAVs. Therefore, all UAVs are going to have the same architecture and weights and their behavior will depend on the current state of the UAV. On the other hand, each UAV can have a different ANN, therefore its response would not only be the result of the state but also of the weights and architecture codified in it.

In all Q-Learning problems, a part of the actions is made randomly with a probability epsilon ($\epsilon = 0.47$), and with probability $1 - \epsilon$ the action with the highest Q-value for that state is taken. The sequence of actions taken by an agent for a given ϵ until it reaches an end condition (task completed, end of time...) is known as an episode. In each episode, the task is restarted from the beginning. As episodes occur during testing, the ϵ value is reduced multiplying it by a reduction factor equal to 0.93. In this way, the choice of actions falls more on the calculated Q-values and less by random selection. To avoid overfitting, ϵ is prevented from reaching a value very close to 0 by setting the minimum value at 0.05. Both values were chosen through a previous exploratory study.

3.3.3 Rewards

In order to prioritize the UAV to move to unvisited areas, the reward must be the highest of all, as shown in Table 2. In addition, it is important that it increases as fewer cells are

Table 2 Assigned rewards to each kind of cell each UAV visits

	Reward
New cell base reward	358.74
Visited cell reward	-31.14
Non-visitable cell	-225.17

The initial values chosen for the rewards by means of a previous random exploration where the best combinations of rewards have been selected

left undiscovered (2). This is known as Hill-Climbing [44]. Another reward is required for cells that have already been visited. Thus, the UAV has a reward in case it is better to fly over an already visited cell to reach an unvisited one than to go around it (for example, when there are spurious cells left unvisited). To prevent UAVs from flying into cells that they cannot visit, they are given the lowest reward. The choice of the selected reward values was made through an initial exploratory process.

$$\text{new cell reward} = \text{new cell base reward} \times \left(1 + \frac{\max(\text{rows}, \text{columns})}{\text{visited cells}} \right) \quad (2)$$

3.3.4 Flying Actions

The possible movements or actions (a) from the set of actions A that UAVs can take were codified. Thus, all possible movements are encoded to a discrete list of values.

Despite the natural complexity of UAV flight, the possible movements have been simplified into straight movements, thereby making it easier to interpret flight paths in a map divided into cells. Otherwise, a UAV could draw a curve passing over the corner of a cell without actually passing through the entire cell. This would create the dilemma of

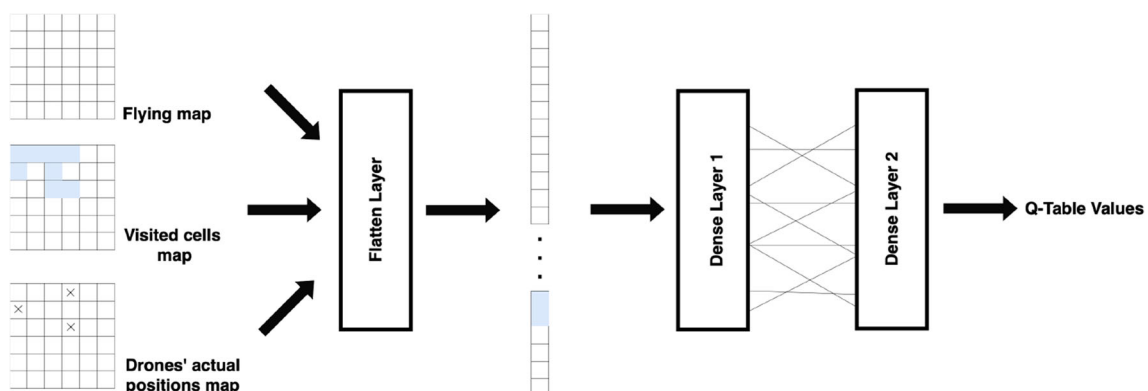
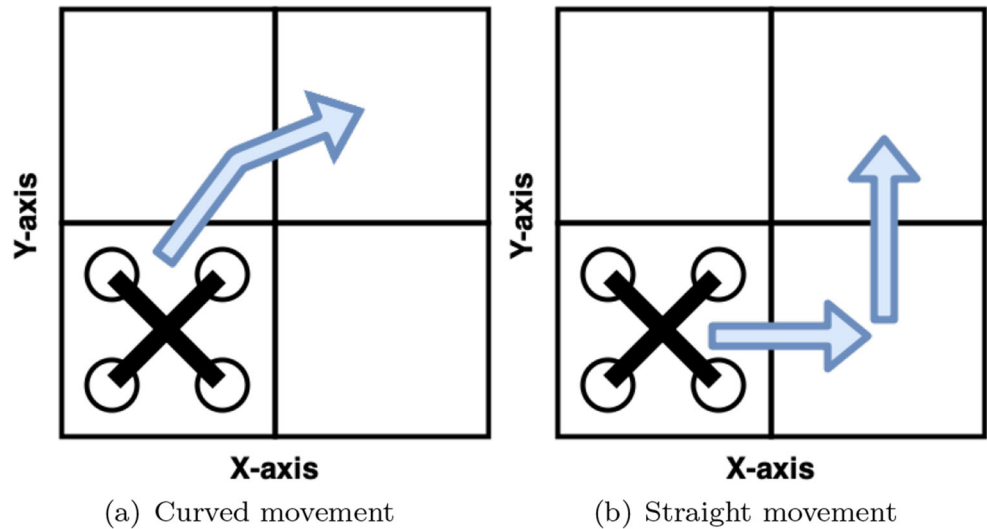


Fig. 4 Diagram showing how data is processed in the ANN in order to get Q-values of the Q-table for a given state of the environment. The maps are combined into a multidimensional matrix and then flattened into vectors. These vectors are used to abstract the knowledge for computing Q-values

Fig. 5 Comparison of curved movements with straight movements. Curved motions produce more easily interpreted flight paths. Moreover, they are limited to the atomicity of the map cells.



whether to mark that cell as visited or not (Fig. 5). Not tracing curves ensures that the graphic data obtained with UAVs always have the same angle and are easier to combine.

3.3.5 Memory replay

In most of the State of the Art, the experience obtained by agents from the environment is reinforced with the Memory Replay technique. Memory Replay is a technique where the model is trained with a set of stored observations called memory. The observations contain a variety of information, such as the actions taken and their reward. It improves sample efficiency by repeatedly reusing experiences and helps to stabilize the training of the model [45]. It is important that the memory contains as many recent observations as possible, but it has a maximum size in order to optimize computational resources. For this reason, the memory follows a First-In-First-Out scheme for eliminating old observations.

Each UAV in the group has its own memory. In its memory, it stores observations with the actions that the UAV

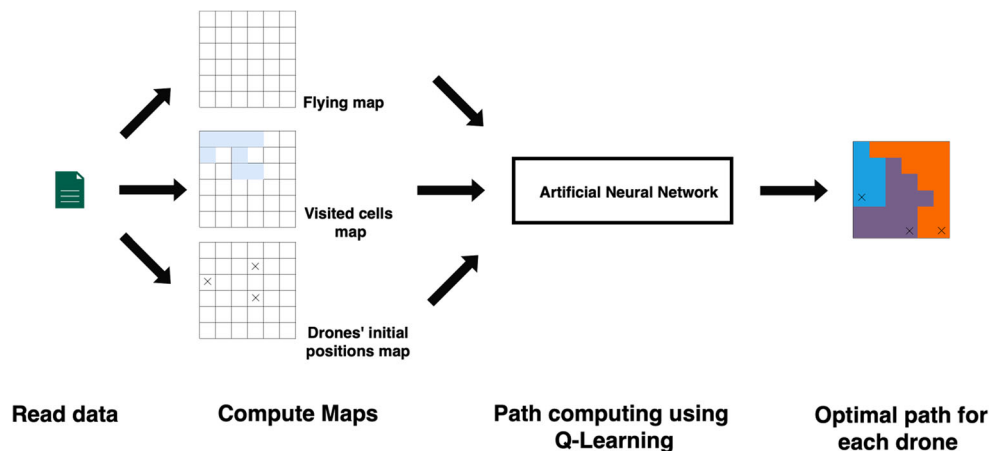
itself takes. At no time the actions of other UAVs are stored. This avoids adding noise to the information. The fact that an action is not correct for one UAV does not imply that it is incorrect for the others since they can be in different positions on the map.

The size of the memory can greatly influence the final results [46]. For this study, a memory size of 60 actions with their respective rewards was chosen after an exploration process. It is important to have a large value with respect to the number of map cells because in the first iterations of the process UAVs make many errors. Thus, learning from most of the errors and training the model multiple times with them will help to avoid them and, thus, achieve a more efficient solution.

3.3.6 Workflow

The scheme shown in Fig. 6 summarizes the main workflow of the proposed method. Starting from the reading of the initial data, that is the vertices of the area to be covered and

Fig. 6 The workflow diagram of the study. The map data and the position of the UAVs are read from a file. The maps that the system will use are constructed from the data read. Using the Q-Learning technique, the best possible path is calculated for each UAV so that the task is completed



the initial positions of the drones, the maps are reconstructed. After that, by using those maps, the ANN is trained with the Q-Learning technique [34]. For part of the experiments a global ANN is used, whereas in another part one ANN is used per UAV. This is going to determine the best action for each UAV.

3.4 Battery estimation

As in the works discussed in Section 2, the authors have not found a standardized method to predict battery power consumption during flight. This is because the consumption depends on the UAV configuration and flight conditions. Normally, most commercial UAVs send to their mobile apps the amount of energy they have left over at regular intervals. On the other hand, there is an increasing number of websites that help to calculate how much battery time is left. The lack of standardization is due to the influence of many variables. The incident wind, the number of direction changes, speed, and many other variables greatly affect the flight time.

As a solution to the problem of battery consumption, the swarm is forced to find a solution in the time corresponding to the minimum remaining battery time among the UAVs of the swarm. In this way, it is expected that in a limited time the UAVs will try to get as close as possible to the desired solution. In this study, it has been assumed that the UAVs all have a maximum energy load that allows them to fly for 30 minutes because no realistic calculation of the remaining battery life has been found.

3.5 Performance measures

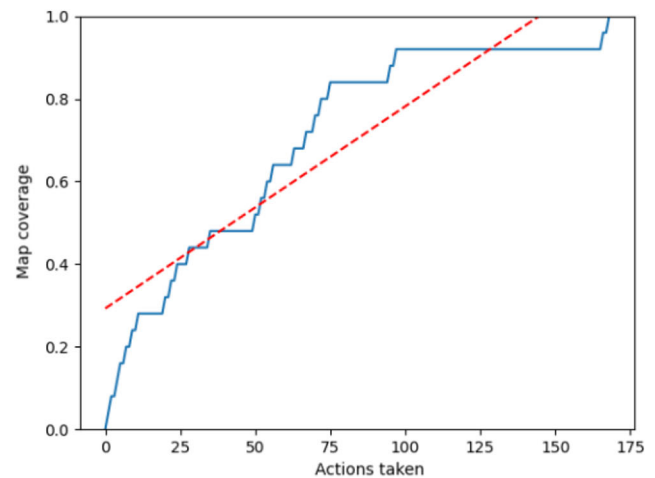
As for performance measures, the most common ones will be taken into account: the time needed to find the solution, the percentage of correct actions out of all actions taken, and the evolution of the map coverage.

It is necessary to find a system that solves the problem as quickly as possible. Thus, it will require less operator time and battery consumption when used in real fields. Low battery consumption indicates that the paths are as short as possible. In addition, due to the charging time of the UAV batteries, low battery consumption might allow the user to do more work without having to stop charging the batteries. This makes it the measure of greatest interest and the most commonly chosen one. For this purpose, we will look for the episodes with the shortest execution time (ET), which is computed as the difference between the actual time when the episode finished or TE_1 and the actual time when the episode started or TE_0 (3). It is important to compute this coverage for each action taken in each individual episode in order to obtain the curve that relates the change in the number of cells discovered by the agents versus the number of total actions that they carry out. The greater the growth of

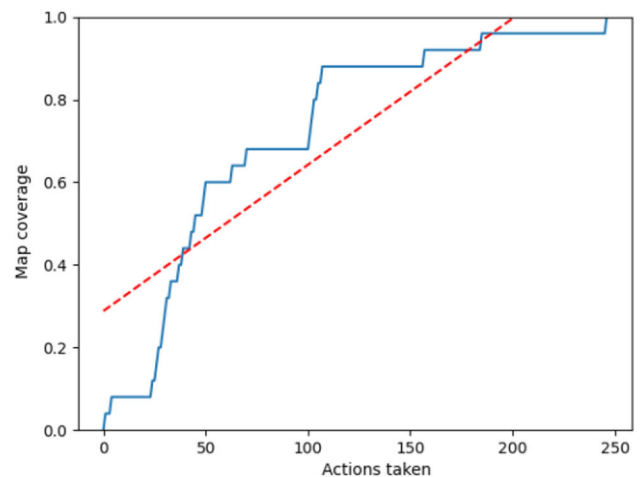
the curve means that fewer movements are needed to reach the solution. This implies that the paths they take have fewer cycles and are therefore more efficient. In Fig. 7 there is an example plot of the curve for one ANN per UAV using two UAVs. By comparison with using the same ANN for all UAVs (a global ANN), its growth is higher, and it reaches total coverage much faster.

$$ET = TE_1 - TE_0 \quad (3)$$

Even though time is used as a measure of performance, it is also needed for calculating the length of the paths UAVs



(a) One ANN per UAV



(b) A global ANN

Fig. 7 Example of curves of the evolution of map coverage (y-axis) as a function of the number of actions taken by two UAVs (x-axis). The dashed line represents the coverage growth trend. Comparing the case of one ANN per UAV with a global ANN it can be seen that the case with a global ANN takes more actions to fly over the whole map

take for each episode. Depending on the configuration of the UAV and the way the UAV flies, the battery consumption can vary greatly. Therefore, it is interesting for the path to be as short as possible. It can be understood as the largest possible number of valid actions to be taken. Valid actions have been defined as those where a new cell is discovered. That is, without loops or passing through cells that cannot be visited. Therefore, it is of interest to know the fraction (PA) of valid actions (VA) out of all the actions taken by the UAVs (TA) (Eq. 4). The closer to 1, the better.

$$PA = \frac{VA}{TA} \tag{4}$$

Knowing how the total map coverage evolves makes it possible to distinguish which methods are better. It is calculated as the fraction of cells that have been visited divided by the total number of cells. Sometimes, the operational resources available (number of UAVs, battery levels, etc.) might not be sufficient to overfly the selected terrain in its entirety. Even so, in such cases, it is important to cover as large an area as possible. That is, a system in which it is able to get closer and closer to 100% map coverage is ideal. The closer to 1, the better.

4 Results

A set of combinations of map sizes and number of UAVs has been defined for conducting the experiments and subsequent analysis of the results. For the analysis of the results obtained, factors such as the evolution of the time required to explore the map and the percentage of actions performed by each UAV have been taken into account.

4.1 Experiment design

To test the capabilities of the system proposed in this paper, 25 experiments have been designed. In each of them, the configuration of the ANNs, the number of UAVs, and the size of the map are different.

The experiments were carried out in a square cell map as in those cited in Section 2.

The aim of this experiment is to identify the best controller for the UAVs. There are two approaches at this point: one ANN per UAV and one ANN for all UAVs (Fig. 8). Both approaches were compared using the same maps and the same UAVs. The results are listed in Table 3. As can be seen in the table, the experiments with one ANN per UAV have been omitted when there is only a single UAV. Using one ANN for only one UAV would be the same as using a global ANN for only one UAV. Therefore, it has been simplified to execute only once with a global ANN for one UAV and it is referred to as baseline (Fig. 9). Thus, it is taken as the starting point of the experimentation taking it as the simplest case, which is to control a single UAV.

Since it is important for the system to operate with any number of UAVs, each selected map type was tested with an increasing number of UAVs. To be more precise, separate experiments have been performed with 1, 2, and 3 UAVs. Thus, it is proved that the system can adapt to a different number of UAVs.

As in the papers mentioned in Section 2, all the maps chosen are grid maps. The experiments were performed with 5×5 , 6×6 , 7×7 , 8×8 , and 9×9 cell grid maps. Having different map sizes provides insight into the capabilities of the system in the face of unfixed map sizes.

In addition, the number of cells in the chosen flight environment is smaller than other cited papers. The cost of flying over large maps is a major constraint. By making one

Fig. 8 Illustrative diagram showing the relationship between the ANN and the UAV in the two proposed approaches: an ANN per UAV and a global ANN. The training process is the same, only the relationship between the model or models and the UAVs changes. In the case of an ANN per UAV, the same ANN architecture is maintained, only the weights change

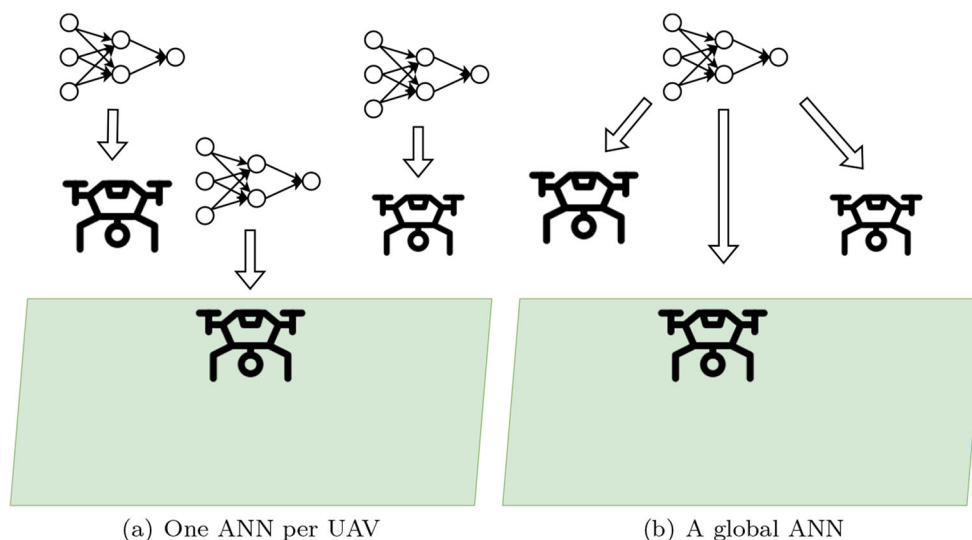


Table 3 Table with the 45 experiments performed

Approach	Map size	Number of UAVs
Baseline	5×5	1 UAV
	6×6	1 UAV
	7×7	1 UAV
	8×8	1 UAV
	9×9	1 UAV
One ANN per UAV	5×5	2 UAVs
		3 UAVs
		4 UAVs
		5 UAVs
		5 UAVs
	6×6	2 UAVs
		3 UAVs
		4 UAVs
		5 UAVs
		5 UAVs
	7×7	2 UAVs
		3 UAVs
		4 UAVs
		5 UAVs
		5 UAVs
8×8	2 UAVs	
	3 UAVs	
	4 UAVs	
	5 UAVs	
	5 UAVs	
9×9	2 UAVs	
	3 UAVs	
	4 UAVs	
	5 UAVs	
	5 UAVs	
Global ANN	5×5	2 UAVs
		3 UAVs
		4 UAVs
		5 UAVs
		5 UAVs
	6×6	2 UAVs
		3 UAVs
		4 UAVs
		5 UAVs
		5 UAVs
	7×7	2 UAVs
		3 UAVs
		4 UAVs
		5 UAVs
		5 UAVs
8×8	2 UAVs	
	3 UAVs	
	4 UAVs	
	5 UAVs	
	5 UAVs	
9×9	2 UAVs	
	3 UAVs	
	4 UAVs	
	5 UAVs	
	5 UAVs	

Each one of them with different configuration. The experiments for an ANN per UAV for a single UAV have been omitted because it is the same as using a global network for a single UAV

stop per cell to photograph the surface of the map each cell contains means that in very large maps the UAVs have to make numerous stops, considerably affecting their battery. Dividing the map into fewer cells reduces the number of stops and starts made by each UAV decreasing their energy consumption.

Another factor to consider is the area of land that each cell represents. The larger, the better, the more information each image contains and the more favorable it is for further processing. These cells must contain an adequate surface area size for each type of activity performed. For example, in tasks such as water stress [47], in which one flies at a height of 12 meters, the area size of the map contained in each cell is enormous.

In many countries the distance from the position to which it flies is limited by the height at which a UAV can fly. For example, in many European countries it is 500 meters or additional measures would have to be taken that not all operators can overcome [48]. Therefore, as it is known that the terrain cannot be too large for the system to be used in a generic way, it is not necessary to use maps with numerous cells. For example, 400-cell maps, such as those used in some papers described in Section 2.

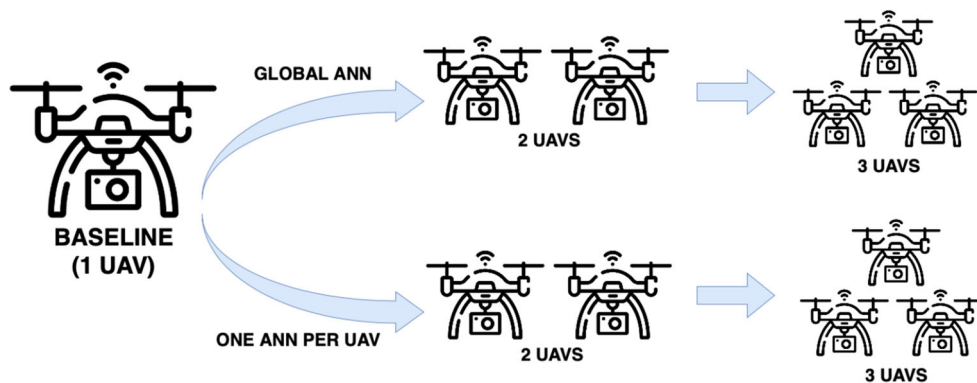
4.2 Experimental results

The results indicate that a global ANN is usually the best choice because usually it has the fastest solutions, specially in larger maps (Table 4). Although it is slower on several cases like 5×5 cell maps for 3 or more UAVs, it is only by a few seconds. When using larger maps, the model needs more time to find a solution regardless of the number of UAVs. This is due to the size of the exploration tree. That is, the larger the map, the more possible path combinations the ANNs have to evaluate. Usually, when using 3 UAVs these exploration times are slower than when using 2 UAVs. It is not a problem since it is often a few seconds and, the more UAVs, the easier it is to reassign the task of a fallen UAV to the others.

In all the experiments conducted the map has been completely covered at least once. The total number of episodes in an experiment where UAVs fly over all cells in the map is usually higher when a global ANN is used. This is indicative of the robustness of the ANN configuration for the problem. Having a greater number of solutions demonstrates that the system is learning and is able to find solutions as the randomness component ϵ is reduced.

Regarding the evolution of the flight paths (Fig. 10), it was found to be highly dependent on the initialization and random component of all Q-Learning problems. In other words, if many wrong decisions are made from the outset, in the long-run this negatively affects future decisions. In the example of Fig. 10, the UAVs keep flying over the cells

Fig. 9 Diagram of how the experiments are conducted for each map. Initially, the system is tested with a given map and a single UAV. If a solution is obtained, the number of UAVs is increased as the system is able to solve the problem for that map. For the experiments, it is differentiated to use a single ANN for all UAVs and a global ANN



at the left of the map. In this case, the UAVs started the flight by taking many actions that resulted in passing through the cells on the left side. An excess of these actions caused the UAVs to barely fly over other cells. In addition, the innermost cells are the least visited, and there are even some cells that have never been visited. This situation leads to understand that the proposed model identifies better the navigable cells of the edges because they have fewer neighbors. It is also worth noting that the cell at the bottom right has been visited when one of its neighbors has not. This reinforces the idea that the proposed model had better performance on cells on the left edge, which causes that when a cell is on the right edge it did not know how to behave.

Having solutions with different map sizes and different numbers of UAVs confirms that the system is generic enough to work under different conditions. To the best of our knowledge, this is the only paper that can do that. Other papers only work with predefined map sizes [49–51].

If we take the times for each configuration of number of UAVs and ANNs we see that they follow a Gaussian distribution according to the Shapiro-Wilk test [52] with a significance level (α) of 5% (Table 5).

Since it was shown that all distributions were Gaussian, we chose the ANOVA statistical significance test [53]. Thus, we can know if the results present significant differences that allow us to decide which option is more appropriate. According to the results listed in Table 6, the solutions of the ANOVA test are significantly different for a 5% significance level, so it can be understood that there are substantial changes when applying some solutions or others. Considering the result of applying the same test to all distributions except Baseline we have also observed that they are significantly different for the same significance level. This reinforces the idea that using a global ANN for all UAVs is significantly better than using a per UAV ANN. Unlike the Baseline case, employing more than one UAV guarantees a level of fault tolerance. Therefore, we can conclude that using a single ANN for the whole swarm is the best con-

figuration and that using more than one UAV reduces the risk during operation.

4.3 Required time evolution

The speed to cover the entire map is also reflected in the time measure. Many episodes have a run time of 30 min. These coincide with the cases in which the entire map is not covered. However, when this is not the case, the time required decreases as the training advances. It can be seen that it is highly dependent on the size of the map and the number of UAVs (Fig. 11). In many cases, once the overall minimum is reached, the results worsen significantly. It is caused by the noise introduced by the random component of the chosen method. Because of this, the sequence of steps that is optimal is that of the episode of the sequence that finds the solution in the shortest time.

The evolution of the time taken by a fixed number of UAVs to find the solution on different maps is highly dependent on the size of the map. In Fig. 12, an example plot with 2 UAVs is shown in which it can be seen that the time curve shows more growth than the curve of the number of cells in a map.

4.4 Taken actions evolution

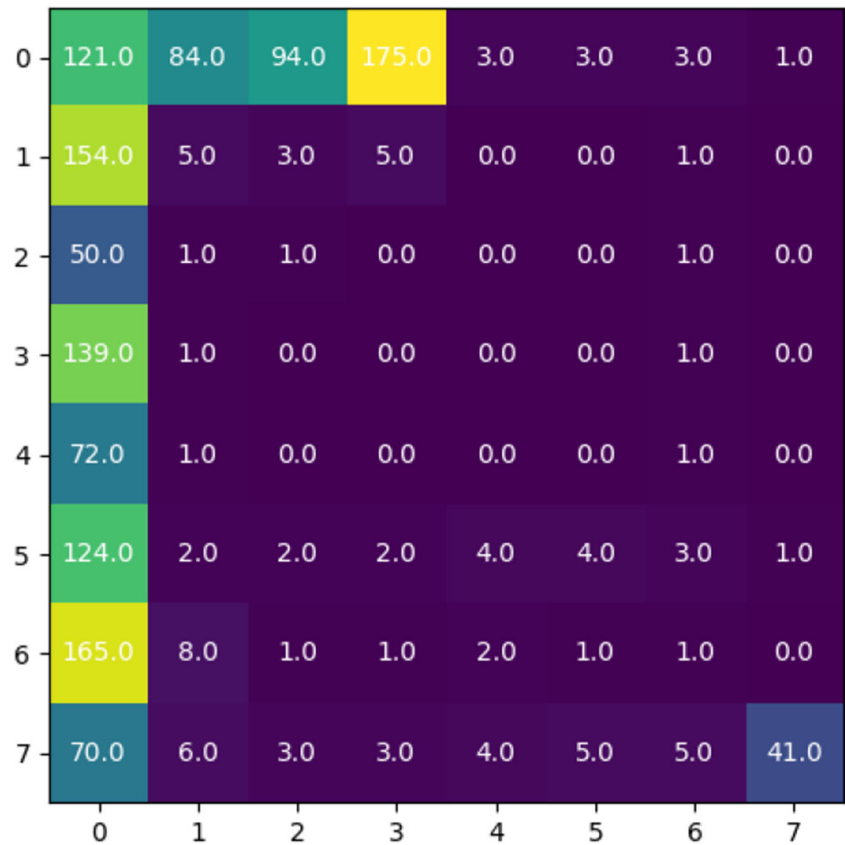
The fraction of the actions where a new cell was discovered, also known as valid actions, among those taken shows different behavior. Therefore, this factor should be taken into account, as it also determines whether the system makes too many errors or takes too many cycles. The more UAVs, the more actions are performed, decreasing the overall percentage of valid actions among all actions taken (Fig. 13). In the initial episodes, the UAVs take a lot of wrong actions. The accumulation of this number of failures impairs the percentage of valid actions taken over the total number of actions. In theory, this is not a problem, since the desired solution is reached faster in the next episode.

Table 4 Summary table with the observed results of the experiments

Map size	Number of UAVs	One ANN per UAV	Global ANN
5×5	1	00:00:40 (28 found solutions out of 30 episodes)	00:00:40 (28 found solutions out of 30 episodes)
	2	00:02:09 (26 found solutions out of 30 episodes)	00:01:32 (29 found solutions out of 30 episodes)
	3	00:01:10 (27 found solutions out of 30 episodes)	00:01:29 (25 found solutions out of 30 episodes)
	4	00:01:07 (23 found solutions out of 30 episodes)	00:01:59 (13 found solutions out of 30 episodes)
	5	00:01:16 (24 found solutions out of 30 episodes)	00:02:21 (16 found solutions out of 30 episodes)
6×6	1	00:02:38 (28 found solutions out of 30 episodes)	00:02:38 (28 found solutions out of 30 episodes)
	2	00:04:23 (7 found solutions out of 30 episodes)	00:02:12 (22 found solutions out of 30 episodes)
	3	00:04:27 (12 found solutions out of 30 episodes)	00:04:39 (14 found solutions out of 30 episodes)
	4	00:02:55 (16 found solutions out of 30 episodes)	00:03:16 (6 found solutions out of 30 episodes)
	5	00:04:32 (13 found solutions out of 30 episodes)	00:03:55 (12 found solutions out of 30 episodes)
7×7	1	00:03:14 (24 found solutions out of 30 episodes)	00:03:14 (24 found solutions out of 30 episodes)
	2	00:06:17 (9 found solutions out of 30 episodes)	00:06:16 (14 found solutions out of 30 episodes)
	3	00:07:01 (12 found solutions out of 30 episodes)	00:06:51 (9 found solutions out of 30 episodes)
	4	00:07:59 (4 found solutions out of 30 episodes)	00:11:47 (4 found solutions out of 30 episodes)
	5	00:04:35 (7 found solutions out of 30 episodes)	00:10:09 (7 found solutions out of 30 episodes)
8×8	1	00:07:37 (9 found solutions out of 30 episodes)	00:07:37 (9 found solutions out of 30 episodes)
	2	00:19:31 (2 found solutions out of 30 episodes)	00:14:14 (5 found solutions out of 30 episodes)
	3	00:16:58 (2 found solutions out of 30 episodes)	00:13:52 (5 found solutions out of 30 episodes)
	4	00:10:16 (3 found solutions out of 30 episodes)	00:08:17 (6 found solutions out of 30 episodes)
	5	00:13:50 (2 found solutions out of 30 episodes)	00:21:18 (2 found solutions out of 30 episodes)
9×9	1	00:12:17 (8 found solutions out of 30 episodes)	00:12:17 (8 found solutions out of 30 episodes)
	2	00:24:45 (1 found solutions out of 30 episodes)	00:16:15 (2 found solutions out of 30 episodes)
	3	00:20:53 (1 found solutions out of 30 episodes)	00:20:27 (1 found solutions out of 30 episodes)
	4	00:17:21 (2 found solutions out of 30 episodes)	00:22:09 (1 found solutions out of 30 episodes)
	5	00:27:56 (1 found solutions out of 30 episodes)	00:18:27 (1 found solutions out of 30 episodes)

Results are displayed with the minimum time in each configuration needed for finding a solution. As in the papers discussed in Section 2, the results shown here are those obtained from a single execution due to the computational and time costs of averaging the results of multiple runs

Fig. 10 Example of a heatmap reflecting the number of times a UAV passes through each cell. In the first actions UAVs take from the beginning, the model took the wrong sequence of initial movements. This caused the UAVs to have a preference for crossing the left edge, thus consuming most of the time. In this way, many cells inside are left unvisited



The improvement of having more errors at the beginning comes from the fact that the map exploration tree is covered faster due to the simultaneous flight of the UAVs. As the exploration tree is covered faster, more information is extracted. Therefore, usually more UAVs in a swarm means that the task is performed faster in future episodes (Table 4). If it is slower, it may be an indicator that more episodes are needed to obtain a better solution. The computational cost is very high considering that it is only a few seconds or minutes slower.

Table 5 Summary table with the p-values resulting from the Saphiro-Wilk test [52]

Distribution	p-value
Baseline	0.5013
Global ANN for 2 UAVs	0.2596
Global ANN for 3 UAVs	0.6719
Global ANN for 4 UAVs	0.4792
Global ANN for 5 UAVs	0.4287
2 UAVs with an ANN per UAV	0.2312
3 UAVs with an ANN per UAV	0.4887
4 UAVs with an ANN per UAV	0.7522
5 UAVs with an ANN per UAV	0.1788

For a significance level of 5%, all distributions are Gaussian

5 Discussion

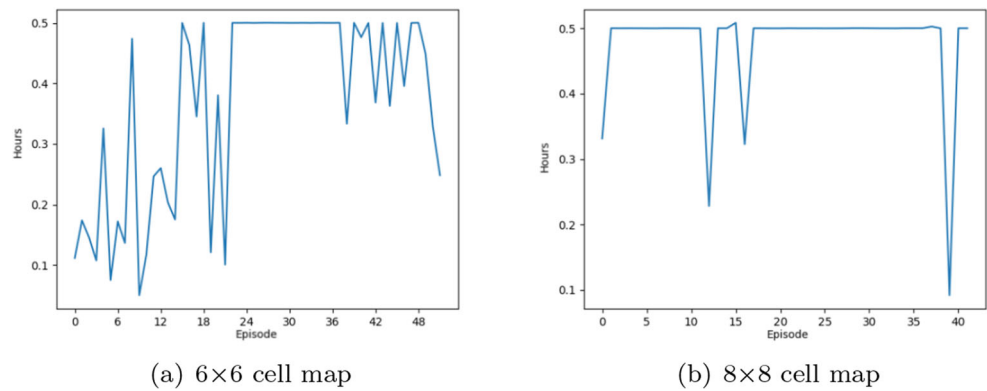
This paper, like the other publications in Section 2, suggests the usage of ANNs based on dense layers. This type of layer has also shown the ability to coordinate groups of UAVs. Some authors, such as Liu et al., have already tested the capabilities of fully connected ANNs in Path Planning problems with UAVs. Moreover, being trained in each case with the memory of each UAV it seems to be able to assign correct actions to the UAVs without extracting spatial information from the map like Convolutional Neural Networks (CNN). As a result, it appears that using networks that need automatic feature extraction, such as CNN, as employed by other authors, is no longer necessary. Because they have already extracted spatial features from the maps, these networks have a slower training time. It may mean that

Table 6 Comparative table of the p-values obtained by performing an ANOVA test [53]

	p-value
All distributions	0.9698
All distributions except baseline	0.9968

Even without the Baseline distribution (1 UAV) the distributions are still significantly different at a 5% significance level

Fig. 11 Example of two plots with the evolution of the hours (y-axis) consumed as the episodes elapse (x-axis) for different maps. In both cases, it can be seen that the solution with the shortest time can be followed by episodes with worse results. The optimal results are those episodes with the shortest duration. The shortest time implies that it is the one with the least number of incorrect actions and the least number of loops



the most important thing may not be the spatial relationship of the map, but the sequence of movements of each UAV without the noise of the actions taken by the other UAVs.

As in other papers in the state of the art, the system has been tested on squared cell maps. Unlike the other papers, it has been tested on different map sizes, not on fixed-size maps [54, 55]. The maps used do not present additional information, like those used in the other papers. That is, it is not necessary to add more information, such as targets or distance maps, so it is not necessary to make previous studies of the map.

Since using a single global ANN for all UAVs usually requires less time than using one ANN per UAV, this indicates that the appropriate configuration is to use a global ANN. This means that paths calculated using a global network have fewer errors and loops, indicating that the paths are as direct as possible. The more direct they are, the shorter they

are, therefore they are more optimal. In other multi-agent problems global ANNs were the best option, like in the paper by Mnih et al. with their A3C algorithm [56]. Despite this, some authors, including Wang et al., have proved the effectiveness of systems that use an ANN per UAV [57].

The overall percentage of correct actions taken decreases with the increase in UAVs. This is due to the fact that in the first episodes too many wrong actions are taken because the agents do not have much knowledge. Despite this, the solution is sometimes reached in less time due to the simultaneity of their flight and, the more UAVs, the more fault tolerance is ensured in case a UAV not being able to continue its flight.

The time taken to explore the entire map is strongly affected by the number of cells on the map. The rise in the time taken exceeds the growth of the number of cells of the maps used in the experiments. This is mainly due to the size

Fig. 12 Comparison of the growth of the curve of the time taken for each map with respect to the growth of the curve of the number of cells contained in each map. The growth is greater in the time curve. Specifically, starting from the map of 8×8 cells

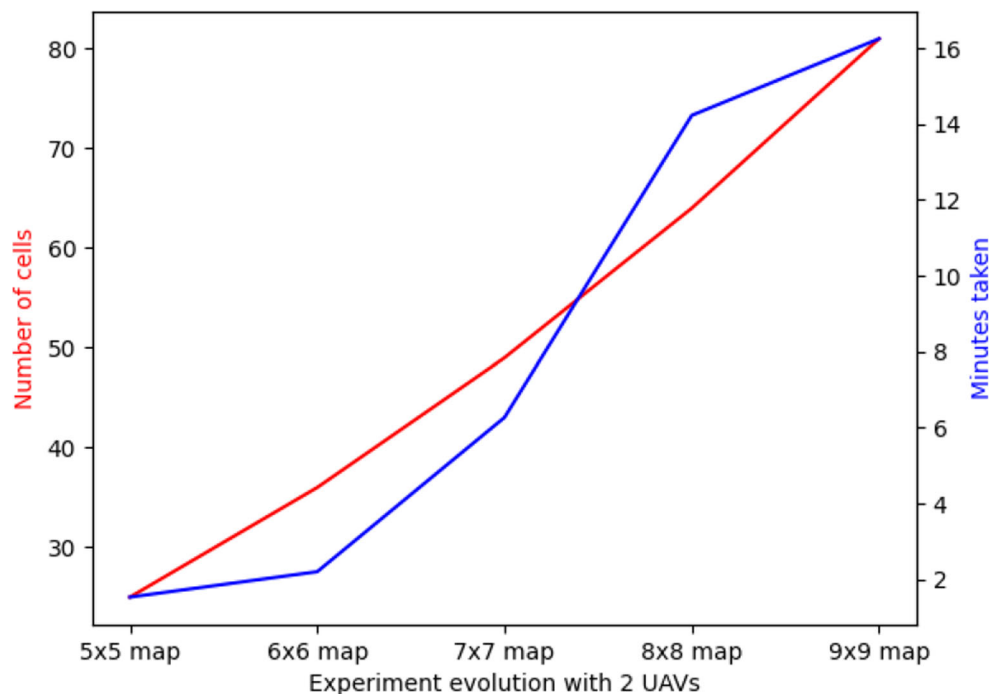
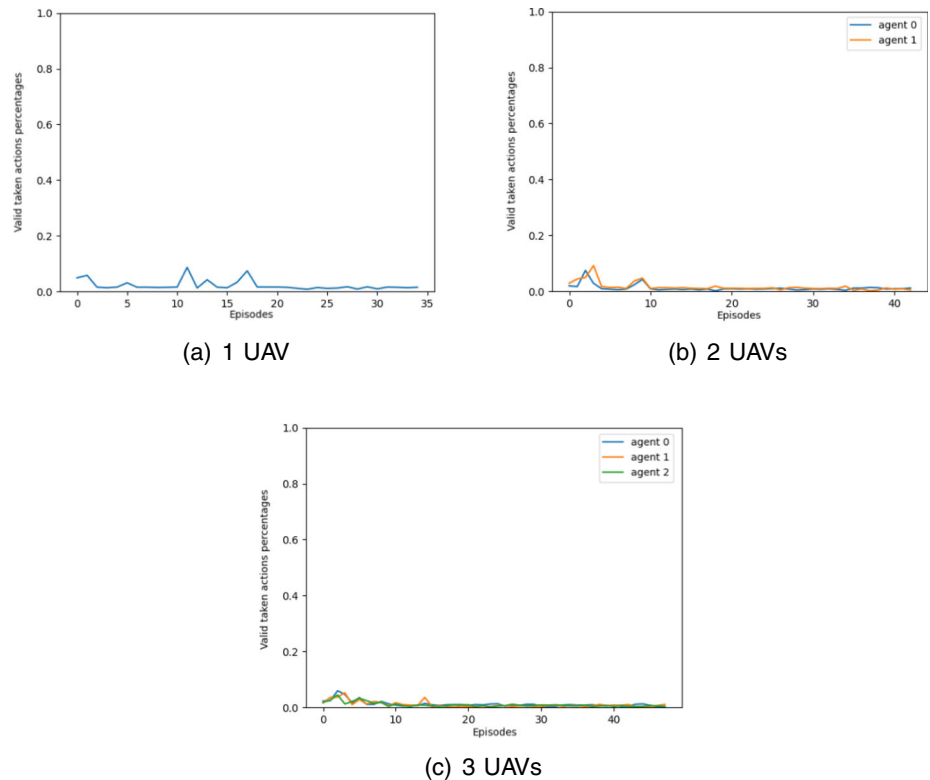


Fig. 13 Percentage of non-error actions (y-axis) taken by each UAV over the course of the episodes (x-axis). Each line of a different color symbolizes a UAV. The more UAVs the lower the percentage. This percentage is affected by the sum of the errors of all UAVs together. When taking the first actions UAVs make many mistakes. This accumulation of errors hurts the percentage of valid actions taken by the UAVs as a whole



of the exploration space that ANN faces in order to find the best paths. The greater the number of cells, the larger the space. In addition, we have to add the sequences of the UAV paths, whether they are one or more. That is, a path has to be a correct sequence of adjacent and fully navigable cells, which adds complexity to the system by having to maintain this consistency as there are more and more cells. This is why the higher the number cells, the more time the ANN requires for obtaining correct pathways.

6 Conclusions

This paper puts forward a system capable of calculating the paths with the shortest flight time for UAV swarms using Q-Learning [35] techniques. To enhance the capabilities of these techniques, decision-making is done with the help of fully connected ANNs. Employing a single global ANN for all UAVs presents more solutions in less time. Finding models that find solutions quickly makes the system more portable to different systems. In this way, users will find it more convenient to use since money does not have to be spent on expensive systems. Typically, the cost savings can be invested in improving UAV features such as battery life by users.

The system is capable of achieving satisfactory results with squared cell maps of different sizes. The evolution of the time required to find a solution increases faster than

the increase in the number of cells in each experiment regardless of the number of UAVs in the swarm. Therefore, it is necessary to adapt the size of the map to the activity to be carried out in order to get the best results possible. Tasks that imply high altitude do not need as many cells because their sensors capture a large part of the terrain in each cell. Reducing the number of cells allows the system to make better and faster decisions due to the smaller exploration space. Other state of the art systems divide the map into a fixed size, which is typically a very large number of cells in the case of a very large map. Because there are more alternative optimal paths to explore for the map, the time it takes to find a solution and the computational cost of processing the maps are higher, and in many cases unnecessary.

It is not necessary to provide additional information on the map to direct the paths. Therefore, the system can calculate the optimal paths using only the information of the cell map, the current position of the UAVs on the map, and the evolution of the flight paths along the map. Using so little information avoids having to know the terrain in advance. If information is to be added to guide the UAV paths, it is necessary to make such a study. Therefore, many users may end up discarding the use of the system due to this additional difficulty. On the other hand, if it is necessary to guide the paths, the system can be biased because user errors can be made that prevent better paths from being found. The disadvantage of not employing targets, as other authors in

the state of the art have done, is that the algorithm is scan-dependent. Because it is so reliant, it is vital to establish algorithm parameters that are as accurate as possible in order to minimize problems with path computations.

The ideal swarm size can also be determined by looking at the change in the time it takes to fly through a map for each swarm size. For example, if a swarm takes less time to do a task than a larger swarm would, it is understood that investing in more UAVs is unnecessary because the task can be solved in less time and at a lower cost with fewer UAVs. In other publications, authors test their systems with a fixed number of UAVs and do not compare this to testing with just one UAV. Furthermore, if the terrain is relatively small, having a large number of UAVs may be excessive and counterproductive. The more UAVs used, the more paths must be computed. If fewer UAVs are used, however, outcomes can be achieved in less time and with less computational resources. Furthermore, using fewer UAVs minimizes the risk of collision.

Due to the atomicity of the movements that each UAV can make, it is not necessary to make a smooth path. Unlike other publications in the state of the art [58], it is not necessary to spend computational time smoothing the path. In addition to this advantage, the movements are simpler if there is no path smoothing, so it is not necessary to compute parameters such as the UAV's yaw angle or tilt angle. On the other hand, paths without smoothing have sharper turns that increase the UAV's battery consumption, so there is no guarantee that consumption is minimized as much as possible.

Finally, the limitations of this system include the fact that the flight height is not considered in the calculation of the paths. In theory, this is not a problem if the working height is high enough to avoid obstacles. On the other hand, if two UAVs flying at the same altitude are in the same cell, the wind thrust or turbulence that one may generate to the other is not considered. These winds or turbulences do not usually distort the paths to a great extent, but they can increase the battery consumption because the UAV needs more power to be able to overcome these environmental disturbances.

7 Future work

This paper is a starting point, laying out some bases for the creation of other systems capable of working on different maps. In this way, generic systems with commercial potential can be obtained.

In the future, efforts will be made to improve these results and a study on the optimal initial distribution of the UAVs on the map should be carried out. Also, models will be trained on cell maps optimally divided according to the resolution of the UAV cameras.

Future developments will include experiments with 3D maps in which more movements, such as pitch and roll, will be possible.

Experiments will be made with maps with obstacles in order to help agents learn how to reduce the risks during the flight. Obstacles can be fixed obstacles (trees, poles, etc.) or dynamic obstacles (birds, other UAVs, etc.).

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. Funding for open access charge: Universidade da Coruña/CISUG. This work is supported by Instituto de Salud Carlos III, grant number PI17/01826 (Collaborative Project in Genomic Data Integration (CICLOGEN) funded by the Instituto de Salud Carlos III from the Spanish National plan for Scientific and Technical Research and Innovation 2013–2016 and the European Regional Development Funds (FEDER)—“A way to build Europe.”. This project was also supported by the General Directorate of Culture, Education and University Management of Xunta de Galicia ED431D 2017/16 and “Drug Discovery Galician Network” Ref. ED431G/01 and the “Galician Network for Colorectal Cancer Research” (Ref. ED431D 2017/23). This work was also funded by the grant for the consolidation and structuring of competitive research units (ED431C 2018/49) from the General Directorate of Culture, Education and University Management of Xunta de Galicia, and the CYTED network (PCI2018_093284) funded by the Spanish Ministry of Ministry of Innovation and Science. This project was also supported by the General Directorate of Culture, Education and University Management of Xunta de Galicia “PRACTICUM DIRECT” Ref. IN845D-2020/03.

Availability of Data and Material Not applicable

Code Availability Source code and a Docker container are available at: https://github.com/TheMVS/uav_swarm_reinforcement_learning https://hub.docker.com/r/themvs/uav_swarm_reinforcement_learning

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Albani D, IJsselmuiden J, Haken R, Trianni V (2017) Monitoring and mapping with robot swarms for agricultural applications. In: 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). IEEE, pp 1–6
2. Huuskonen J, Oksanen T (2018) Soil sampling with drones and augmented reality in precision agriculture. *Comput Electron Agric* 154:25–35

3. Corte APD, Souza DV, Rex FE, Sanquetta CR, Mohan M, Silva CA, Zambrano AMA, Prata G, Alves de Almeida DR, Trautenmüller JW, Klauberg C, de Moraes A, Sanquetta MN, Wilkinson B, Broadbent EN (2020) Forest inventory with high-density uav-lidar: Machine learning approaches for predicting individual tree attributes. *Comput Electron Agric* 179:105815
4. Bocchino R, Canham T, Watney G, Reder L, Levison J (2018) F prime: an open-source framework for small-scale flight software systems
5. Rabinovitch J, Lorenz R, Slimko E, Wang K-SC (2021) Scaling sediment mobilization beneath rotorcraft for titan and mars. *Aeolian Res* 48:100653
6. Liu J, Wang W, Wang T, Shu Z, Li X (2018) A motif-based rescue mission planning method for uav swarms using an improved picea. *IEEE Access* 6:40778–40791
7. Yeaman ML, Yeaman M (1998) Virtual air power: a case for complementing adf air operations with uninhabited aerial vehicles. Air Power Studies Centre
8. Zhao Y, Zheng Z, Liu Y (2018) Survey on computational-intelligence-based uav path planning. *Knowl-Based Syst* 158:54–64
9. Champion M, Ranganathan P, Faruque S (2018) A review and future directions of uav swarm communication architectures. In: 2018 IEEE international conference on electro/information technology (EIT). IEEE, pp 0903–0908
10. Bonabeau E, Meyer C (2001) Swarm intelligence: A whole new way to think about business. *Harvard Bus Rev* 79(5):106–115
11. Puente-Castro A, Rivero D, Pazos A, Fernandez-Blanco E (2021) A review of artificial intelligence applied to path planning in uav swarms. *Neural Comput Appl*:1–18
12. Wiering M, Van Otterlo M (2012) Reinforcement learning. *Adapt Learn Optim* 12:3
13. Baldazo D, Parras J, Zazo S (2019) Decentralized multi-agent deep reinforcement learning in swarms of drones for flood monitoring. In: 2019 27th European signal processing conference (EUSIPCO). IEEE, pp 1–5
14. Yang Q, Jang S-J, Yoo S-J (2020) Q-learning-based fuzzy logic for multi-objective routing algorithm in flying ad hoc networks. *Wirel Pers Commun*:1–24
15. Roudneshin M, Sizkouhi AMM, Aghdam AG (2019) Effective learning algorithms for search and rescue missions in unknown environments. In: 2019 IEEE international conference on wireless for space and extreme environments (WiSEE). IEEE, pp 76–80
16. Luo W, Tang Q, Fu C, Eberhard P (2018) Deep-sarsa based multi-uav path planning and obstacle avoidance in a dynamic environment. In: International conference on sensing and imaging. Springer, pp 102–111
17. Speck C, Bucci DJ (2018) Distributed uav swarm formation control via object-focused, multi-objective sarsa. In: 2018 annual american control conference (ACC). IEEE, pp 6596–6601
18. Davis L (1991) Handbook of genetic algorithms
19. Duan F, Li X, Zhao Y (2018) Express uav swarm path planning with vnd enhanced memetic algorithm. In: Proceedings of the 2018 international conference on computing and data engineering. ACM, pp 93–97
20. Zhou Z, Luo D, Shao J, Xu Y, You Y (2020) Immune genetic algorithm based multi-uav cooperative target search with event-triggered mechanism. *Phys Commun* 41:101103
21. Olson JM, Bidstrup CC, Anderson BK, Parkinson AR, McLain TW (2020) Optimal multi-agent coverage and flight time with genetic path planning. In: 2020 international conference on unmanned aircraft systems (ICUAS). IEEE, pp 228–237
22. Huang T, Wang Y, Cao X, Xu D (2020) Multi-uav mission planning method. In: 2020 3rd international conference on unmanned systems (ICUS). IEEE, pp 325–330
23. Perez-Carabaza S, Besada-Portas E, Lopez-Orozco JA, Jesus M (2018) Ant colony optimization for multi-uav minimum time search in uncertain domains. *Appl Soft Comput* 62:789–806
24. Xie R, Meng Z, Zhou Y, Ma Y, Wu Z (2019) Heuristic q-learning based on experience replay for three-dimensional path planning of the unmanned aerial vehicle. *Sci Prog*:0036850419879024
25. Rummery GA, Niranjan M (1994) On-line q-learning using connectionist systems, vol 37. University of Cambridge, Department of Engineering Cambridge, England
26. Liu C, Xie W, Zhang P, Guo Q, Ding D (2019) Multi-uavs cooperative coverage reconnaissance with neural network and genetic algorithm. In: Proceedings of the 2019 3rd high performance computing and cluster technologies conference. ACM, pp 81–86
27. Vijayakumari DM, Kim S, Suk J, Mo H (2019) Receding-horizon trajectory planning for multiple uavs using particle swarm optimization. In: AIAA Scitech 2019 Forum, p 1165
28. Li Q, Gama F, Ribeiro A, Prorok A (2020) Graph neural networks for decentralized multi-robot path planning. In: 2020 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 11785–11792
29. Shah S, Dey D, Lovett C, Kapoor A (2018) Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In: Field and service robotics. Springer, pp 621–635
30. Chaudhuri D, Samal A (2007) A simple method for fitting of bounding rectangle to closed regions. *Pattern Recogn* 40(7):1981–1989
31. Kaelbling LP, Littman ML, Moore AW (1996) Reinforcement learning: A survey. *J Artif Intell Res* 4:237–285
32. Sutton RS, Barto AG (2018) Reinforcement learning: An introduction. MIT press
33. Van Hasselt H, Wiering MA (2007) Reinforcement learning in continuous action spaces. In: 2007 IEEE international symposium on approximate dynamic programming and reinforcement learning. IEEE, pp 272–279
34. Watkins Christopher JCH, Dayan P (1992) Q-learning. *Mach Learn* 8(3-4):279–292
35. Sutton RS, Precup D, Singh SP (1998) Intra-option learning about temporally abstract actions. In: ICML, vol 98, pp 556–564
36. Gläscher J, Daw N, Dayan P, O’Doherty JP (2010) States versus rewards: dissociable neural prediction error signals underlying model-based and model-free reinforcement learning. *Neuron* 66(4):585–595
37. Michie D, Spiegelhalter DJ, Taylor CC et al (1994) Machine learning. *Neural Stat Classif* 13
38. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521(7553):436–444
39. Fan J, Wang Z, Xie Y, Yang Z (2020) A theoretical analysis of deep q-learning. In: Learning for Dynamics and Control. PMLR, pp 486–489
40. Albawi S, Mohammed TA, Al-Zawi S (2017) Understanding of a convolutional neural network. In: 2017 International Conference on Engineering and Technology (ICET). IEEE, pp 1–6
41. Huang G, Liu Z, Van DML, Weinberger KQ (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4700–4708
42. Marreiros AC, Daunizeau J, Kiebel SJ, Friston KJ (2008) Population dynamics: variance and the sigmoid activation function. *Neuroimage* 42(1):147–157
43. Hinton G, Srivastava N, Swersky K (2012) Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. Cited on 14(8)
44. Kimura H, Yamamura M, Kobayashi S (1995) Reinforcement learning by stochastic hill climbing on discounted reward. In: Machine Learning Proceedings 1995. Elsevier, pp 295–303

45. Foerster J, Nardelli N, Farquhar G, Afouras T, Torr PhilipHS, Kohli P, Whiteson S (2017) Stabilising experience replay for deep multi-agent reinforcement learning. In: International conference on machine learning. PMLR, pp 1146–1155
46. Liu R, Zou J (2018) The effects of memory replay in reinforcement learning. In: 2018 56th annual allerton conference on communication, control, and computing (Allerton). IEEE, pp 478–485
47. Gago J, Douthe C, Coopman RE, Gallego PP, Ribas-Carbo M, Flexas J, Escalona J, Medrano H (2015) Uavs challenge to assess water stress for sustainable agriculture. *Agric Water Manag* 153:9–19
48. (2016). (EASA) EASA. Gtf. https://www.easa.europa.eu/sites/default/files/dfu/GTF-Report_Issue2.pdf. [Online; accessed 19-July-2008]
49. Zhang C, Zhen Z Z, Wang D, Li M (2010) Uav path planning method based on ant colony optimization. In: 2010 Chinese Control and Decision Conference, pp 3790–3792
50. Wang Z, Li G, Ren J (2021) Dynamic path planning for unmanned surface vehicle in complex offshore areas based on hybrid algorithm. *Comput Commun* 166:49–56
51. Li W, Yang B, Song G, Jiang X (2021) Dynamic value iteration networks for the planning of rapidly changing uav swarms. *Front Inf Technol Electron Eng*:1–10
52. Razali NM, Wah YB et al (2011) Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *J Stat Model Anal* 2(1):21–33
53. Hecke TV (2012) Power study of anova versus kruskal-wallis test. *J Stat Manag Syst* 15(2-3):241–247
54. Albani D, Manoni T, Arik A, Nardi D, Trianni V (2019) Field coverage for weed mapping: toward experiments with a uav swarm. In: International conference on bio-inspired information and communication. Springer, pp 132–146
55. Venturini F, Mason F, Pase F, Chiariotti F, Testolin A, Zanella A, Zorzi M (2020) Distributed reinforcement learning for flexible uav swarm control with transfer learning capabilities. In: Proceedings of the 6th ACM workshop on micro aerial vehicle networks, systems, and applications, pp 1–6
56. Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, Silver D, Kavukcuoglu K (2016) Asynchronous methods for deep reinforcement learning. In: International conference on machine learning. PMLR, pp 1928–1937
57. Wang L, Wang K, Pan C, Xu W, Aslam N, Hanzo L (2020) Multi-agent deep reinforcement learning-based trajectory planning for multi-uav assisted mobile edge computing. *IEEE Trans Cogn Commun Netw* 7(1):73–84
58. He W, Qi X, Liu L (2021) A novel hybrid particle swarm optimization for multi-uav cooperate path planning. *Appl Intell*:1–15

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.