# REPlanner: Efficient UAV Trajectory-Planning using Economic Reinforcement Learning

Alvi Ataur Khalil*, Alexander J Byrne*, Mohammad Ashiqur Rahman, and Mohammad Hossein Manshaei
Analytics for Cyber Defense (ACyD) Lab, Florida International University, Miami, USA
{akhal042, abyrn010, marahman, mmanshae}@fiu.edu

***Abstract*—Advances in the unmanned aerial vehicle (UAV) design and capability, as well as decreases in the manufacturing cost, have opened up applications of UAVs in various fields, including surveillance, firefighting, cellular networks, and delivery purposes. The uniqueness of UAVs in systems creates a novel set of trajectory or path planning and coordination problems. Environments include many more points of interest (POIs) than UAVs, with obstacles and no-fly zones. We introduce REPlanner, a novel multi-agent reinforcement learning algorithm inspired by economic transactions to distribute tasks among UAVs. This system revolves around an economic theory, in particular an auction mechanism where UAVs trade assigned POIs. We formulate the path planning problem as a multi-agent economic game, where agents can cooperate and compete for resources. We then translate the problem into a partially observable Markov decision process (POMDP), which is solved using a reinforcement learning (RL) model deployed on each agent. As the system computes task distributions via UAV cooperation, it is highly resilient to any change in the swarm size. Our proposed network and economic game architecture can effectively coordinate the swarm as an emergent phenomenon while maintaining the swarm's operation. Evaluation results prove that REPlanner efficiently outperforms conventional RL-based trajectory search.**

***Index Terms*—Unmanned aerial vehicles, reinforcement learning, path planning, trajectory optimization, swarm robotics**

## I. Introduction

Unmanned aerial vehicles (UAVs) are applicable to a wide-ranging set of problems such as fire fighting, security monitoring, agriculture, edge computing, 3D mapping, and network support [1]. All of these problems can be abstracted to a set of partially observed points and must be traveled to, in the shortest amount of time possible, and then some tasks must be carried out in the vicinity of these points. More generally, swarm surveillance missions are essential in both civilian and military contexts, where solutions must be secure, reliable, and efficient. The problem is computationally expensive as solutions must be provided for each UAV and take into account the paths of other UAVs to avoid collisions and use resources efficiently. Current methods include swarm intelligence optimization (SIO), formal methods, convex optimization, and graph-based methods [2], [3], [4]. Besides SIO, these methods suffer from the glaring issue that all the participating UAVs take commands from a single point, which, if compromised, brings the entire system down. Thus, a framework is needed that considers the UAV swarm's characteristics from the ground up to provide a more distributed, intelligent, and reliable service.

*Khalil and Byrne are the co-first authors of this paper.

The cutting edge of UAV swarm technology is reinforcement learning (RL), which is used to control the agents for executing versatile operations, e.g., simulating indoor environment [5], proving various cellular internet [6] and data harvesting [7] missions, avoiding collisions [8]. Each UAV is controlled by its own agent, acting in its own interest, and attempting to gain the most reward for itself. While all of these papers frame the problem as a cooperative one, it is immediately seen that agents are also competing for reward, as an agent completing a shared point effectively removes the opportunity for another agent to gain that reward and potentially wastes fuel. It is natural to then frame the problem in terms of game theory and agent-to-agent communications as part of an economic game.

To utilize the above mentioned phenomenon, we propose REPLANNER, an **R**L and **E**conomic game-based trajectory **Planner** for UAV swarms. In this framework, the agents enter into bids for actions to take, allowing the swarm as a whole to find the value of actions through pricing. As this emerges from the interactions of agents, it is decentralized and comes at a low computational cost. Furthermore, as we employ RL, the agent can learn which strategies are effective in the context of other agents' strategies and the current configuration of the environment. We evaluate the trajectory planning performance between the standard q-learning and our economic variant. The results show a 33% increase in path distance efficiency, an 18% decrease in time to completion, and a 200% increase in reward gain. According to the reward gain, our model adapts to its directive to act in the environment far better than standard Q-learning. In summary, our contributions are as follows:

- We design and implement REPLANNER, a novel distributed optimization technique, which mimics economic exchange to reassign objects to agents.
- We introduce an auction mechanism for distributed trajectory or path planning system, which anticipates competitive as well as cooperative strategies from agents.
- We evaluate the proposed REPLANNER framework with respect to flight duration, traveled distance, and goal completion. The evaluation results show that REPLANNER was more efficient than standard Q-learning.
- We conduct a case study through the AirSim [9] simulator to validate the performance of the proposed framework. The codes related to the evaluation as well as the simulation are available at [10].

The rest of the paper is organized as follows: We discuss

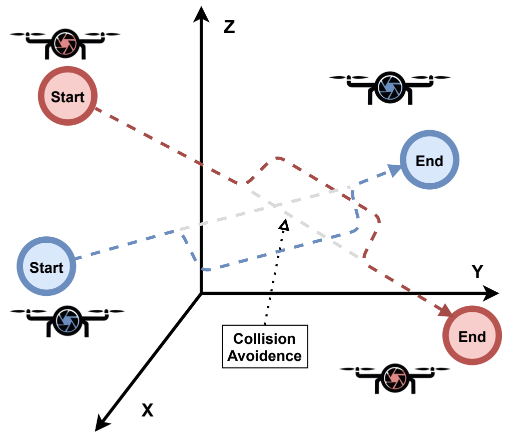


Fig. 1. Path-planning, where two UAVs efficiently avoid a collision.

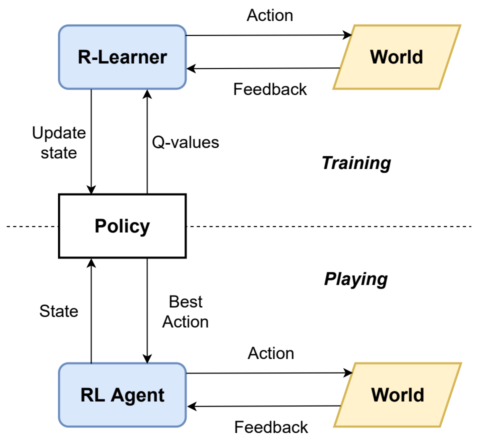


Fig. 2. Reinforcement Learning Architecture

preliminary information in Section II. The related works are discussed in Section III. We introduce our proposed REPLANNER framework in Section IV. In Section V, we discuss the technical details of the frameworks and the complete analysis of our algorithms. In Section VI, we explain the evaluation setup and dataset. The empirical analysis and findings are formulated in Section VII. Finally, we conclude the paper in Section VIII.

## II. BACKGROUND

In this section, we discuss the preliminary information that helps to explain the problem domain and the proposed REPLANNER mechanism.

### A. *UAV path-planing*

Path planning is referred to the mechanism of finding an optimal path between source and destination, and it is one of the most important problems to be explored in the UAVs' domain. The main objective of UAV path planning is to design a cost-effective flight path that meets the UAV performance requirements with minimal probability of being destroyed during the flight [11], including providing a collision-free environment to the UAVs, as presented in Fig. 1. There are numerous path-planning methods for UAVs to navigate in the obstacles-filled environment.

### B. *Reinforcement learning*

Reinforcement learning refers to the field of Machine learning that deals with the mapping of situations to actions in order to maximize the achievement of the actor. In RL, an agent interacts with its environment to learn an optimal policy that maximizes expected cumulative rewards for a given task [12]. The objective of RL is to maximize future rewards. However, since an actor cannot predict the future changes in its environment perfectly, value functions reflect the actor's empirical estimates for its future rewards. The distant future rewards are often discounted temporally so that more immediate rewards exert a stronger influence on the actor's behavior. The general architecture of reinforcement learning is presented in Fig. 2, which has two main properties: learning and playing [13]. In the continuous training phase of the model, initially, the reinforcement learner (R-learner) does not know which action will maximize the gain, as in most other machine learning. It has to discover which actions are most profitable by applying them. So it performs random actions from a state and observes the incurred reward (or penalty) using the feedback of that action from the environment. These rewards (or penalties) are the Q-values for particular state-action pairs, which define the policy. The RL agent, while playing, uses this policy to predict which action is best in a given state and learns through feedback from the environment.

### C. *Decision-Making: Economic vs. Reinforcement Learning*

Decision-making refers to the mechanism that is used by an actor to choose its actions. Economic theories of decision-making attach numerical metrics to alternative actions. In this way, the choices for specific actions can be better understood as actions with maximum value will be selected more often among all possible actions. These hypothetical quantities are called utilities and can be applied to any kind of behavior. An actor will always, by definition, choose the behaviors that maximize the actor's utility [14]. These theories behave agnostically about how these utilities are determined. However, they are presumably constrained by individual experience and evolution [14]. Similar to utilities in economic theories, value functions in RL theory refer to the estimates for the sum of future rewards. Although both of these two approaches have their own unique benefits, one assigns more weight to the immediate reward while the other does it for future rewards. In this work, we consider two types of actions performed by the agent to maximize the reward. The agent will optimize the trajectory by leveraging Q-learning algorithm-based RL and will trade points of interest (POIs) with other agents depending on the reward gain versus resource invested, utilizing the economic decision making.

## III. RELATED WORKS

Originally, the path-planning problems were solved by framing the problem as a convex optimization problem and utilized analytic and numerical techniques [15]. While it was computationally expensive, a path need only be calculated once, and so in effect, this was not a complex problem [15].

For realistic, complex situations, the model-free reinforcement learning methods have become popular as it enables an agent to autonomously learn an optimal policy through trial-and-error interactions with its environment [16]. Similar to [17], we used Q-learning to calculate collision-free paths, but unlike it, we also used Q-learning to generate the initial paths. Li et al. introduced a method for path planning that combines an improved version of the Q-learning algorithm with heuristic searching rules for mobile robots in a dynamic environment [18]. In [19], a Q-learning-based distinct derived learning method with cyclic error correction has been proved effective for mobile robot navigation. The authors in [20] improved the performance of the Q-learning algorithm with an action selection strategy and a Q-function initialization method, which has been applied to UAV path planning in an antagonistic environment.

Although the trading of tasks or goals in multi-objective games is not a new concept, existing works hardly maintain the balance of cooperation and competition. Like in [21], the authors set up agents in an economic game, including trading product or capital and receiving rewards based on performance in the economic game. They utilized Q-learning as opposed to directly solving for the Nash Equilibrium in order to generalize the method. In [22], Schultink et al. utilize a hierarchy of agents at different levels of control, working together to complete a task. This is similar to how our bidding system rewards agents choosing to move closer to their POI and facilitates the exchange of POIs. However, our agents are more autonomous, meaning the bidding process cannot select from a set of agents and allow that agent to act. The authors in [23] used an auction mechanism, in addition to support vector machines to assign UAVs services, optimizing cellular coverage while balancing trust and profit potential. Our work differs in solution technique and architecture by simultaneously solving for path planning and point assignment. An auction system based on profit potential is proposed by Ng et al. in [24], constrained by resource and current objectives to distribute UAVs to cells to complete federated learning tasks. The UAVs were allowed to form cooperative teams, choosing based on profit potential. Whereas, in our work, we consider all the UAVs working together as a group of individual trading agents. In [25], Duan et al. introduced an auctioning system for UAV task assignment, taking into account resource constraints in environments with varying profit potential, risk, and information. A priority-based point selection mechanism is used there before assigning them to the UAVs. On the other hand, we consider visiting each of the points at the earliest possible time, not assigning low priority to any of them.

## IV. Proposed REPlanner Framework

Fig. 3 shows the basic framework of the proposed REPlanner mechanism. Here, the agent uses the Q-learning algorithm [26] to take any kind of decision regarding its movement and leverages greedy economic approach for trading POIs. The agent can take the best actions under the given observed, potentially noisy, and incomplete information state. It does so by querying a table built from the result of these *action and state* pairs and the consequent reward. Operating independently of this table is an auction system that modifies the environment in tandem with the Q-learning algorithm. In effect, our agent acts upon two different perspectives of the environment at the same time. The first perspective of the environment consists of the physical locations that an agent has to travel within the time limit to gain its rewards. The second perspective contains the offers and bids of all the agents in the environment. So effectively, the agent, given that it is in two different but correlated states, must take the two most optimal actions in these two perspectives of the environment.

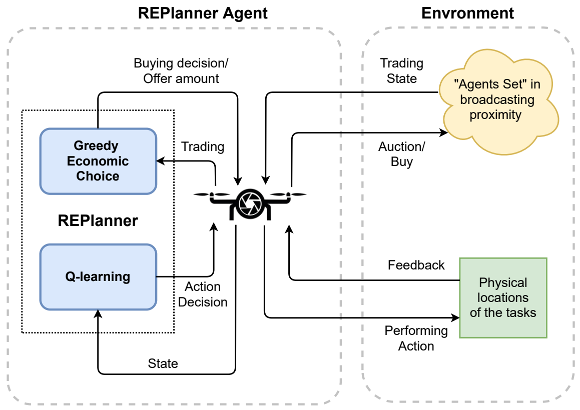


Fig. 3. Proposed REPlanner framework

The agent uses the Q-learning module for trajectory decisions given its goal set. The Q-learning module takes state information of the agent as well as the partial information of the environment and calculates the trajectory incurring the highest amount of reward. The agent takes action according to the output of the Q-learning module. On the other hand, an agent checks the feasibility of the points in its goal set using the greedy economic choice (GEC) module to decide on which points to visit and which to sell-off. The agents broadcast an auction for each of their infeasible points to the other agents that are within the broadcasting proximity. The other agents, listening to this broadcast, use their own GEC module to calculate the feasibility for the point of interest and estimates an offer price for that point. They broadcast the offer price to the selling agent, and after finding the best offer, the selling agent transfers the point to the buying agent's goal set. An agent observes the environment and provides the current position and trading state to the Q-learning module and GEC module, respectively.

## V. Technical Details

We start by discussing the necessary formal models underlying the proposed REPlanner and later, we discuss the model in detail.

### A. Environment Model

We model the environment as two simultaneous partially-observable Markov games. Markov games are a generalization of a Markov decision process to multiple agents. A partially observable game hides a subset of the state from the players, usually other player's states. Formally it consists of a tuple $(S_i, A_i, O_i, Z_i, T_i, R, b_i^{(0)}, \gamma)$, which is explained in the next two sections. There are two primary problems, assignment of POIs to agents and path planning for each agent to each of its assigned POIs.

## B. The Path Planning Game model

In a path planning game, agents must make a sequence of movement decisions in order to navigate from POI to POI, attempting to gather data before all other agents. The agents must avoid no-fly zones and agent-agent collisions. The agents are deployed onto a $W \times L \in \mathbb{N}^2$ grid-world with width $W$, and length $L$. POIs distributed randomly throughout with positions given by $POI := \{(x_1, y_1), (x_2, y_2), ..., (x_j, y_j)\}$ where, $(x, y) \in \mathbb{N}^2$ and $j$ is the number of POIs. Similarly no-fly zones are given by $\{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$ where $m$ is the number of off-limits spaces with $(x, y) \in \mathbb{N}^2$. Agents positions are pairs $(x_i, y_i)$ where $i \in 1, \ldots, n$ and $n$ is the number of agents. A player receives a positive reward inversely proportional to the time elapsed during the game when their coordinates match that of some element of $POI$, $P_l \in POI$. The POI is then removed from all agents' goal sets, and no agent can receive a reward from traveling to it. Formally, a partially observable Markov path planning game is defined as the following:

- $S_i := (p_i, s_i)$ where $p_i \subseteq POI$ are the points the agent must visit and $s_i \in \mathbb{N}^2$ is the agents position.
- $A_i := m(s_i)$ if $s_{i+1} \notin N$ where $m$ adds or subtracts one from the elements of $s_i$ moving the agent to another square and $N$ is the set of girds composing a no fly zone.
- $O_i$ consists of the agent's state, $POI$ and the completeness status of each element of $POI$.
- $Z_i$ is defined implicitly based on the randomly initialized strategies of other players.
- $T_i$ is defined implicitly based on the actions of other players altering $POI$ and their own positions.
- $R_i$ is defined by a function which compares $p_i$ to $s_i$ and a function which compares $p_i$ to $\mathbf{s_{-i}}$.
- $b_i^{(0)}$, the initial belief, is initialized randomly.
- $\gamma$, the discount factor, is set to 0.95.

The grid world is initialized with random positions for the POIs, agents, and NFZs. It is represented by an image in which different colors represent clear spaces, POIs, NFZs, and UAVs. Agents are then allowed to move about in the grid world until all POIs are visited or a set time limit is reached.

## C. The Economic Game

Simple models, such as the laws of supply and demand function to describe the aggregate behavior of self-interested agents cooperating and competing. We attempt to emulate this situation as a Markov game in order to exploit this optimization process. In an economic game, agents trade objects using a virtual currency, attempting to accumulate the greatest amount of privately valuable objects and capital. Each round, agents engage in an auction, putting up an object currently in their possession and bidding a private amount for another object at auction. Fig. 4 represents the auction architecture followed by the UAV agents. Here we use an auction-based on the resources an agent is likely to spend pursuing a point. This first price auction is secured by temporarily locking the offer amount so that there is no incentive in lying about the bid, forcing an agent to honestly bid its valuation of the object while still being able to profit by selling it. As agents have different valuations of each object, this allows for a stable distribution of objects to emerge over successive auctions. Accordingly there is a finite set of objects $O := \{o_j\}$ and

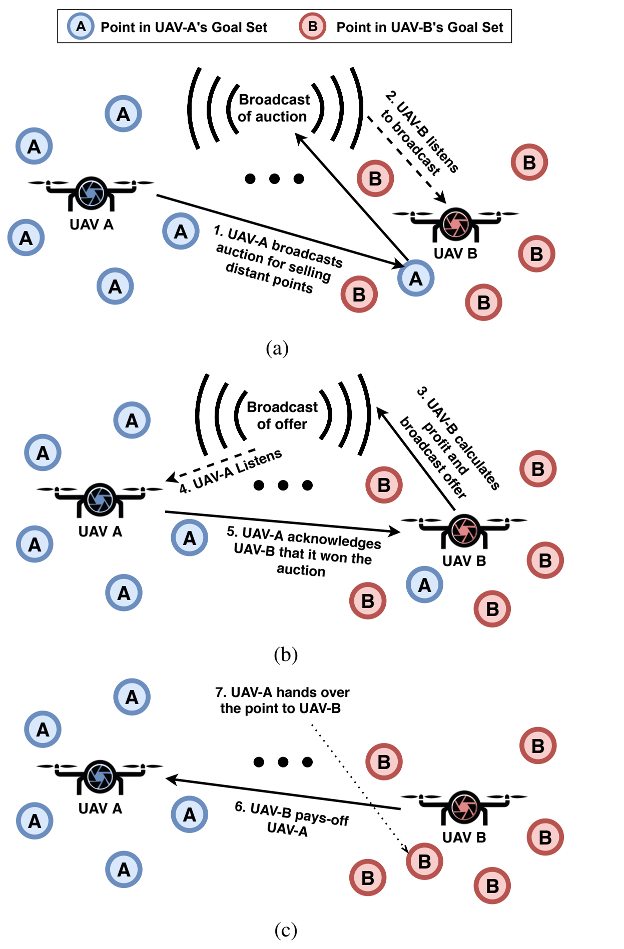

A Point in UAV-A's Goal Set
B Point in UAV-B's Goal Set
Broadcast of auction
2. UAV-B listens to broadcast
UAV A
UAV B
1. UAV-A broadcasts auction for selling distant points
(a)
Broadcast of offer
3. UAV-B calculates profit and broadcast offer
4. UAV-A Listens
5. UAV-A acknowledges UAV-B that it won the auction
(b)
7. UAV-A hands over the point to UAV-B
6. UAV-B pays-off UAV-A
(c)

Fig. 4. Auction architecture: a) UAV-A broadcasts the auction for the distant point in its goal set while other UAVs listen to it and b) UAV-B determines its profit with respect to the expense-reward associated with the point of interest and broadcasts its offer. Later, if it wins the auction, UAV-A acknowledges its victory. Finally, c) UAV-B pays-off UAV-A to buy that point, and accordingly, the point is handed over.

TABLE I
LIST OF NOTATIONS

| Symbol | Definition |
|---|---|
| $U$ | Set of agents in the network |
| $u_i$ | The ith agents in the network |
| $E$ | Environment data |
| $N_t$ | Number of training episodes |
| $P_i$ | Object list of ith agent, $P_i$ = [$p_{i1}$, $p_{i2}$,$p_{i3}$ ... ] |
| $P_{all}$ | List of all "object list"s, $P_{all}$ = [$P_1$, $P_2$, $P_3$ ... ] |
| $B_i$ | List of incoming bids of ith agent |
| $R_i$ | Total reward of ith agent |

a corresponding set $W_i := \{o_k \in 0 | \ o_k$ is owned by some agent $k\}$ , and exchange function $E$ is defined as:

$$E_i(o, Pr, A) := argmax_P(Pr, A) \rightarrow W_a \cup \{o\}, W_i \setminus \{o\}$$

where $o$ is the object for sale, $Pr$ is the set of offered prices, $A$ is the set of agents who offered prices. For example, the first price is offered by the first agent in $A$ and so on. Formally an economic game is defined as the following

- $S_i := (W_i, c_i)$ where $W_i$ is the list of owned objects as defined above, and $c_i \in \mathbb{R}^+$ represents the capital an agent has in trading.
- $A_i := (o_j, o_k, p)$ where $o_j \in O$ is an object selected for purchase, $o_k \in W_i$ is an object owned by the agent selected for sale, and $p \in \mathbb{R}^+$ is the price proposed for $o_j$.
- $O_i$ consists of the agents' state and the objects for sale.
- $Z_i$ is based on the bids of other agents and the objects in their possession which they decide to sell.
- $T_i$ is based on the exchange function $E_i$ defined above, assigning objects to the winners of bids, and exchanging capital between the seller and buyer of the object.
- $R_i$ is proportional to the size of $W_i$ and $c_i$.
- $b_i^{(0)}$, the initial belief, is initialized randomly.
- $\gamma$, the discount factor, is set to 0.95.

**Algorithm 1:** Training of UAV Trajectory Planning

```
initialize $List_{BroadCast}$ = $\Phi$;
for each episode in $N_t$ do
    while $P_{all}$ is not empty do
        for each $u_i$ in $U$ do
            for each object in $P_i$ do
                if object is not completed then
                    $List_{BroadCast}.append(object)$;
        for each $u_i$ in $U$ do
            for each object in $List_{BroadCast}$ do
                $B_i.append(Distance(u_i, p_i), object\ )$
        for each $u_i$ in $U$ do
            choice, distance= $argmin_{distance}(B_i)$;
            Update $P_i$ adding choice;
            Remove choice from $P_{seller}$;
            $R_{seller} + = 10$;
            $R_i - = 10$;
        for each $u_i$ in $U$ do
            Use Q-table to choose object from $P_i$ to take action in
              environment;
            Take action;
            for each object in $P_i$ do
                if object is completed then
                    Check object for $R$;
                    if completed by self then
                        Add to $R_i$;
                    Remove object from $P_{all}$;
                else
                    if Collision or No fly zone detected then
                        Subtract from $R_i$
                    continue;
for each $u_i$ in $U$ do
    update Q-table;
```

## *D. Optimization problem*

Time is used as a proxy to real-world quantities such as battery life and distance traveled, as these metrics are necessarily increasing functions of time. Thus by minimizing mission time, we also minimize distance traveled and battery used. We assume all UAVs are able to communicate with one another at any given time during auction periods. They move to an adjacent grid from their current position if the grid is accessible, according to:

$$s_{i+1} = \begin{cases} M(P_i, U_i) = U_i + d_i, & \text{if } s_{i+1} \notin N \\ s_i, & \text{otherwise} \end{cases}$$

Our solution focuses on maximizing the number of POIs traveled to in the least amount of time subject to redundancy and travel constraints. That is to say $sup\{(\sum_{Contract \in C^*} c - t) - T\}$, where $W$ is the collection of all contracts, $c$ is the contract's completion status, $t$ is the contract's elapsed time, and $T$ is the total mission time.

## *E. Description of the Algorithm*

Finally, we introduce a novel distributed machine learning algorithm in order to allocate targets optimally. Algorithm 1 describes the bidding, moving, and training process of the REPLANNER framework. Table I represents the list of notations used in this algorithm. We note that market economies can be characterized as a distributed computation by market members for the appropriate distribution and production of items and that these computations can be represented as repeated games as described above. Each UAV is controlled by an RL agent. Its state consists of known targets and completion status; its position is organized into a map. Its actions consist of traveling to a point, selling and removing a point from its memory, or buying and adding a point to its memory. When another agent reaches a point, it receives a large reward, and the point is removed from all other agent's memories. Every other UAV is then unable to bid for and gain reward by traveling to the POI. The process of buying and selling points is done via broadcasts and does not require any centralized authority. In the simulation, however, it is necessarily centralized.

## *F. A Case Study*

We initialized an environment with 20 points, set at random locations in a 40x40 grid. Each agent is assigned contracts randomly. Fig. 5 represents the two-dimensional space, where the UAVs, trained with economic Q-learning (Fig. 5(a)) and regular Q-learning (Fig. 5(b)- 5(c)), move around to explore the environment and visit the goal points. Each colored line is a UAV's path throughout the 200 timesteps in the twenty-five thousand-th episode. From Fig. 5(b), we can see a lot of overlapping paths and a lot of distance covered. However, none of the agents are able to complete the whole goal set assigned to them. Only after seventy-five thousand episodes in Fig. 5(c), all the goal sets are successfully completed, but still with overlapping paths. On the other hand, using the economic model in Fig. 5(a), all the agents learned to trade the initially assigned POIs, and they display an excellent optimization of resource utilization. There are no overlapping paths, and they seem to cover just part of the whole map, yet completed all the goals assigned.

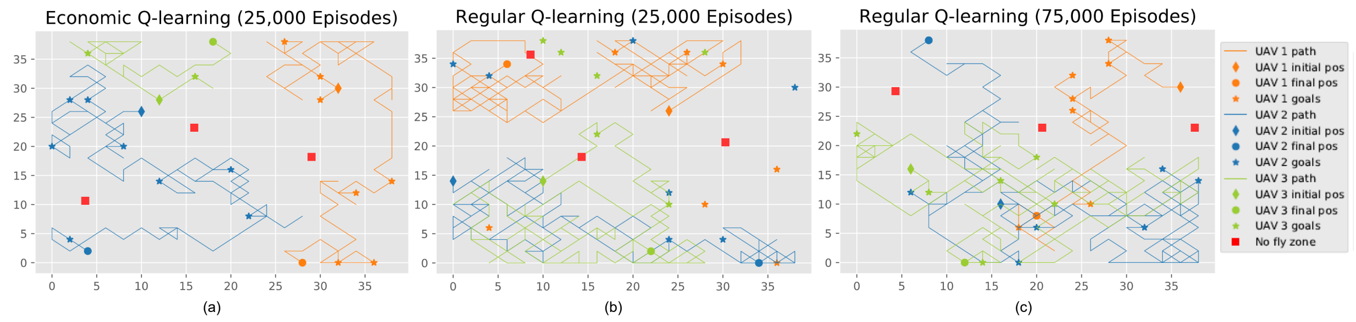


Fig. 5. Case study of (a) Economic Q-learning after 25000 training episodes, (b) Regular Q-learning after 25000 training episodes, and (c) Regular Q-learning after 75000 training episodes with 3 UAV agents and 20 points to visit.

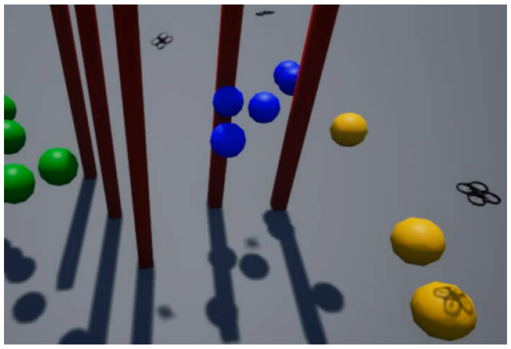


Fig. 6. Simulation of the case study using AirSim.

### G. Simulation

For demonstrating the effectiveness of the REPLANNER framework in a real-world scenario, we run simulations for the *grid world case study* by leveraging the AirSim [9] simulator from Microsoft, which is built on Epic Games' Unreal Engine. The simulator provides a physics engine for simulating real-world phenomena, including wind. In the simulation, we assign fixed heights for the UAVs and POIs, whereas the NFZs are simulated as red pillars (Fig. 6). The intelligent behavior of the UAVs in the simulation (publicly available at [10]) confirms that even without considering the complex environment parameters, the REPLANNER framework generalized the trajectory efficiently and could potentially control real UAVs.

## VI. Evaluation Setup

This section presents the experimental setup and necessary evaluation metrics to assess our proposed REPLANNER framework's performance. We implement the framework considering a simple two-dimensional environment with time constraints for the UAV agents. The experiments are conducted on Dell Precision 7920 Tower workstation with Intel Xeon Silver 4110 CPU @3.0GHz, 64 GB memory, 4 GB NVIDIA Quadro P1000 GPU.

TABLE II
Q-LEARNING MODULE ARCHITECTURE

| Parameter Name | Model hyperparameters |
|---|---|
| *Model* | Q-learning |
| *Exploration paeameter ($\epsilon$)* | 0.5 |
| *Epsilon decay* | 0.9999 |
| *Discount factor* | 0.95 |
| *Episodes (Each iteration)* | 25,000 |
| *Steps (Each episode)* | 200 |
| *Learning rate ($\alpha$)* | 0.1 |

### A. Environment Objects and Actions

The environment contains three types of objects: the UAV agents, the POIs to be visited, and the NFZs to be avoided. Each type of object is initialized in the environment at a unique position to ensure a POI doesn't end up at the same location as an NFZ. The agent can take four diagonal movement actions for visiting the POIs depending on the appropriate (maximum or minimum) q-value for that state-action pair. Also, for exploratory behavior, the agent can randomly take action towards the horizontal or diagonal direction. So a total of 8 actions can be performed for movement. Now, for the trading mechanism, an agent can take two types of actions: sell or buy a POI. The agent decides to sell a point if the monetary value of the resource requirement is more than the potential reward to be achieved. Later, an agent decides to buy a POI if the potential reward is more than the total buying price and resource invested in visiting that point.

### B. Architecture of the Model

Our model begins with defining the various hyper-parameters mentioned in Table II. For the very first episode, the q-table is initialized with random values. For a 50$\times$50 grid environment, the q-table has more than 96 million cells, each representing an action-value pair, which means more than 96 million q-values to be tuned throughout the learning process.

### C. Evaluation Metrics

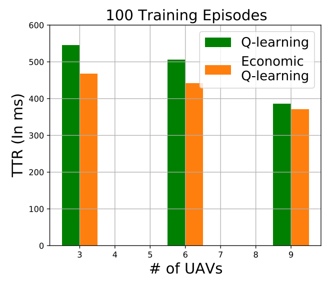
In this subsection, we define the metrics used to evaluate the REPLANNER 's performance.

**Total Time Required (TTR):** The efficiency of the protocol can be measured with respect to the total time required for all the agents to complete their goal-set. This is not the summation of total travel time of all the agents, rather the difference between the starting time and time when all the agents are done with their goal-set. Thus the TTR can be defined as the following:

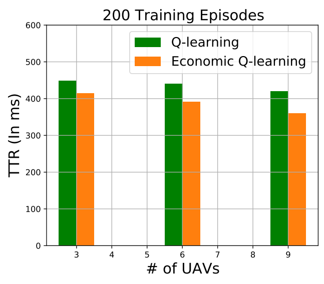
$$TTR = \textit{(final time-stamp)} - \textit{(initial time-stamp)}$$

**Goal Completeness (GC):** GC is defined by the percentage of the goal set points that have already been visited or processed, depending on the context. Thus the GC can be defined as the following:

$$GC = \frac{\textit{\# of points visited}}{\textit{\# of total points in the goal set}} \times 100$$

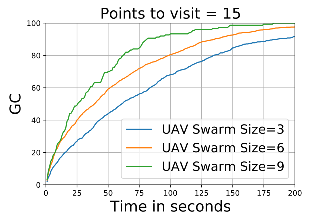


(a)

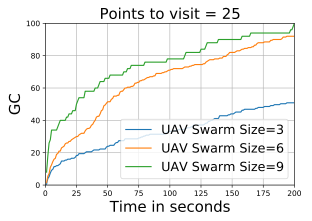


(b)

Fig. 7. Average time required to complete goal set after (a) 100 training episodes and (b) 200 training episodes, for different UAV swarm size with Q-learning and economic Q-learning, visiting 20 POIs.



(a)



(b)

Fig. 8. Percentage of goal set completion with respect to number of steps taken for UAV swarm size of 3, 6 and 9 having (a) 15 points to visit and (b) 25 points to visit, with REPLANNER framework after 1000 training episodes.

**Distance Traveled (DT):** DT is defined by the summation of total distance the agents had to travel to complete their goal sets. Ideally, the smaller is value is, the more efficient the trajectory was. Thus the DT can be defined as the following:

$$DT = \sum_{i \in \mathbb{N}} d_i$$

Here, $d_i$ refers to the distance that i-th agent had to travel to complete its goal set, and $\mathbb{N}$ refers to the total number of agents.

**Episode Average Reward (EAR):** Average of the rewards accumulated by all the UAV agents in a particular episode. The reward increases for a correct decision, whereas wrong decisions incur negative points. Therefore, the reward is a number indicating how good the model's prediction was in an unknown environment.

$$EAR = \frac{1}{\mathbb{N}} \sum_{i \in \mathbb{N}} r_i$$

Here, $r_i$ refers to the incurred reward of the i-th agent at the end of a particular episode, and $\mathbb{N}$ refers to the total number of agents participating in that episode.

## VII. Evaluation Result and Discussion

In this section, we evaluate and analyze the REPLANNER's performance with respect to the four different metrics mentioned in the previous section.

### A. *Time for Completing Goal Set*

In this part, we specifically focus on comparing the economic and non-economic reinforcement learning with respect to the time required for completing the goal of the whole UAV swarm. We try with different number of episodes in a training iteration and show the trend for changing the required time for both models. From Fig. 7, it is evident that economic Q-learning is performing better for each of the swarm sizes and remains better than the normal Q-learning with increasing training episodes. In Fig. 7(a), it is seen that REPLANNER performs exceptionally well with each agent having a larger goal set, that is, way more POIs to visit. Another point to be noticed is, with 200 training episodes in Fig. 7(b), the Q-learning is getting close to the economic one, but still, it is significantly less efficient than the economic Q-learning.

### B. *Percentage of Completion for Different Swarm Sizes*

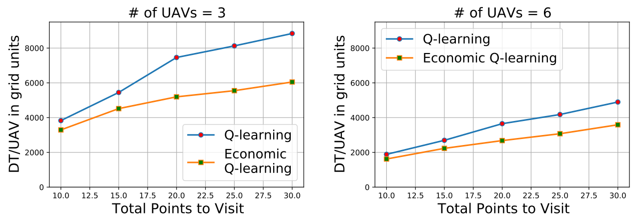
In this part, we evaluate the performance of the REPLANNER model by observing the percentage of completion of the tasks. We necessarily keep the number of training episodes constant, which is 1000, and try with different number of UAVs. We calculate the completion of tasks with respect to the steps needed. Also, we try with different number of points to visit to see how the swarm behavior changes with a larger goal set. From Fig. 8(a), it is seen that after 75 steps, swarm size of 9 almost completed 90 percent of the goal set, where swarm size of 6 and 3 completed close to 70 and 55 percent, respectively. As per Fig. 8(b), it can be observed that with the increase of points in the environment, the completion percentage goes down. Also, in Fig. 8(b), it is seen that during steps 75 through 125, swarm size 6 got really close to the efficiency of swarm size of 9, most likely due to the exploration behavior.

### C. *Optimal Travelling Distance for completing the goal set*

Through Fig. 9, we estimate the efficiency of the REPLANNER framework in optimizing the traveling distance for completing the goal set. Ideally, the more efficient model will require a lesser distance to travel to complete the goal set, effectively optimizing the usage of resources and time. Here, we compare the Q-learning and economic Q-learning with different number of points to visit and show the distance they require to visit those points. Additionally, we try with different number of UAV swarm size in Fig. 9(a), Fig. 9(b) and observe the behavior alterations. Although for all the different swarm sizes, economic Q-learning is doing better than the non-economic one, an improvement trend can be seen for the non-economic Q-learning with a larger swarm size. From this behavior, it can be inferred that the REPLANNER performs way better than the non-economic one with a significantly larger number of points to visit.

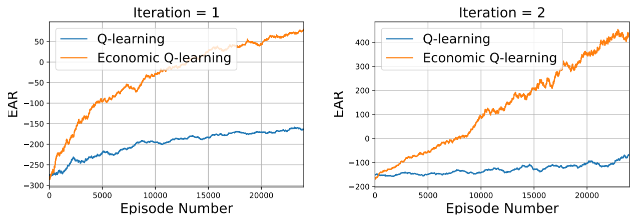
### D. *Reward Accumulation Visuals*

Finally, we evaluate the efficiency of the REPLANNER framework by comparing the attained reward of the economic Q-learning with the achieved reward of the regular Q-learning, as shown in Fig. 10. Theoretically, this metric is the most important one as RL optimizes for the maximum reward. Despite the fact that our algorithm outperforms the standard in UAV-specific metrics, reward accumulation shows the difference in optimizing the power of the two algorithms as seen Fig. 10(a),



(a) (b)

Fig. 9. Average distance required to be travelled to visit all the goal points with swarm size of (a) 3 UAV agents, and (b) 6 UAV agents, trained with regular and economic Q-learning for 100 episodes.



(a) (b)

Fig. 10. EAR incurred by 3 UAV agents to visit 20 POIs in 24000 episodes after (a) 1 training iteration, and (b) 2 training iterations, with Q-learning and economic Q-learning.

10(b). As such, these are the most impressive results of our work, providing evidence that performance can be increased even further.

## VIII. Conclusion

In this paper, we have introduced a reinforcement learning algorithm coupled with an economic trading theory to distribute points and create paths for a multi-agent UAV surveillance problem. Our evaluation results have shown that our approach outperforms the standard Q-learning method, allowing for UAVs to delegate tasks to other agents and learn cooperative strategies in a multi-agent context. More specifically, with as few as three UAVs, we have observed a significant decrease of 33% in the distance needed to travel to all POIs. As the number of POIs has increased relative to that of UAVs, the economic algorithm showed increasingly efficient paths. The overall performance has continued to scale as UAVs are added to the swarm, the best shown by a 200% greater reward than the Q-learning. The UAVs have been able to avoid collisions with the swarm's path plans. Through the auctions, the UAVs have avoided unnecessary competition and been able to focus on sections of the map. This feature shows the swarm strategizing as a whole without direct communication, with the auction operating over the Q-learning algorithm. Our future works will focus on the refinement of the training process and the design of the reward function. The generality of the environment as a Markov game allows for an extension of this method to other multi-agent problems. In addition, we will focus on the use of deep reinforcement learning to enable the integration of broader contexts and further refinement of the auction process.

## References

[1] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges," *Ieee Access*, vol. 7, pp. 48 572–48 634, 2019.

[2] H. Duan and P. Qiao, "Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning," *International journal of intelligent computing and cybernetics*, 2014.

[3] X. Liu, P. Lu, and B. Pan, "Survey of convex optimization for aerospace applications," *Astrodynamics*, vol. 1, no. 1, pp. 23–40, 2017.

[4] M. Naazare, D. Ramos, J. Wildt, and D. Schulz, "Application of graph-based path planning for uavs to avoid restricted areas," in *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2019, pp. 139–144.

[5] O. Walker, F. Vanegas, F. Gonzalez, and S. Koenig, "Multi-uav target-finding in simulated indoor environments using deep reinforcement learning," in *2020 IEEE Aerospace Conference*. IEEE, 2020, pp. 1–9.

[6] J. Hu, H. Zhang, L. Song, Z. Han, and H. V. Poor, "Reinforcement learning for a cellular internet of uavs: Protocol design, trajectory control, and resource management," *IEEE Wireless Communications*, vol. 27, no. 1, pp. 116–123, 2020.

[7] H. Bayerlein, M. Theile, M. Caccamo, and D. Gesbert, "Multi-uav path planning for wireless data harvesting with deep reinforcement learning," *arXiv preprint arXiv:2010.12461*, 2020.

[8] D. Wang, T. Fan, T. Han, and J. Pan, "A two-stage reinforcement learning approach for multi-uav collision avoidance under imperfect sensing," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, 2020.

[9] "Microsoft airsim," https://microsoft.github.io/AirSim/.

[10] "Replanner codes," https://replanner.wixsite.com/website.

[11] S. Aggarwal and N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," *Computer Communications*, vol. 149, pp. 270–299, 2020.

[12] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," *arXiv preprint arXiv:1709.06560*, 2017.

[13] H. Sethya and A. Patelb, "Reinforcement learning approach for real time strategy games like battle city and s3."

[14] D. Lee, H. Seo, and M. W. Jung, "Neural basis of reinforcement learning and decision making," *Annual review of neuroscience*.

[15] A. Ahmadzadeh, A. Jadbabaie, V. Kumar, and G. J. Pappas, "Multi-uav cooperative surveillance with spatio-temporal specifications," in *IEEE Conference on Decision and Control*, 2006, pp. 5293–5298.

[16] C. Yan, X. Xiang, and C. Wang, "Towards real-time path planning through deep reinforcement learning for a uav in dynamic environments," *Journal of Intelligent & Robotic Systems*, pp. 1–13, 2019.

[17] Y.-H. Hsu and R.-H. Gau, "Reinforcement learning-based collision avoidance and optimal trajectory planning in uav communication networks," *IEEE Transactions on Mobile Computing*, 2020.

[18] S. Li, X. Xu, and L. Zuo, "Dynamic path planning of a mobile robot with improved q-learning algorithm," in *2015 IEEE international conference on information and automation*. IEEE, 2015, pp. 409–414.

[19] R. Tang and H. Yuan, "Cyclic error correction based q-learning for mobile robots navigation," *International Journal of Control, Automation and Systems*, vol. 15, no. 4, pp. 1790–1798, 2017.

[20] C. Yan and X. Xiang, "A path planning algorithm for uav based on improved q-learning," in *2018 2nd International Conference on Robotics and Automation Sciences (ICRAS)*. IEEE, 2018, pp. 1–5.

[21] G. Tesauro and J. O. Kephart, "Pricing in agent economies using multi-agent q-learning," *Autonomous agents and multi-agent systems*.

[22] E. Schultink, R. Cavallo, and D. C. Parkes, "Economic hierarchical q-learning," 2008.

[23] A. S. Khan, G. Chen, Y. Rahulamathavan, G. Zheng, B. Assadhan, and S. Lambotharan, "Trusted uav network coverage using blockchain, machine learning, and auction mechanisms," *IEEE Access*, vol. 8, pp. 118 219–118 234, 2020.

[24] J. Ng, W. Lim, H.-N. Dai, Z. Xiong, J. Huang, D. Niyato, X.-S. Hua, C. Leung, and C. Miao, "Joint auction-coalition formation framework for communication-efficient federated learning in uav-enabled internet of vehicles," *IEEE Trans. on Intelligent Transportation Systems*, 2020.

[25] X. Duan, H. Liu, H. Tang, Q. Cai, F. Zhang, and X. Han, "A novel hybrid auction algorithm for multi-uavs dynamic task assignment," *IEEE Access*, vol. 8, pp. 86 207–86 222, 2019.

[26] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.