Adaptive Multi-Agent Path Planning for Distributed UAV Systems

CS 229 Autumn 2017 Final Project
Category: Theory & Reinforcement Learning

Lloyd Maza
lmaza

Cameron McMillan
cmac12

Introduction

In recent years, unmanned aerial vehicles (UAVs) have become increasingly prevalent in the modern world. These systems are used for a wide variety of applications, from recreation and entertainment to warfare and espionage. UAVs are favored for their low cost and low risk, not to mention their ability to execute tasks in environments which might be inaccessible or dangerous to humans.

In tactical scenarios, there is an increased demand for the use of cooperative, distributed systems of UAVs which enable more complex and larger objectives. Oftentimes, these systems require the use of centralized, human-in-the-loop control approaches to accomplish their goals which may be slow, inaccurate, and computationally intensive. As a result, autonomy and decentralization are desirable traits for any such system of UAVs.

This project makes use of a modified Q-Learning technique to allow UAVs to learn in real time to navigate a contested environment that is initially unknown. Missions contain different surveillance target layouts and the UAVs learn from their observations. The UAVs' path planning is rewarded for balancing danger against path length. The UAVs work cooperatively to explore the unknown environment by sharing information about the risk environment seen by each agent in the system.

The simulation environment for this project is built in MATLAB. In this environment, risks and targets can be programmatically generated. This simulation is able to replicate entire missions, including all the relevant risk, observation, and performance information of the UAVs, threats, and targets.

Related Work

The field of robotic path planning contains many approaches, that do not fully solve the problem of online learning in a multi-agent dynamic environment. The first set of approaches that struggle with this scenario is traditional Q-learning. The work done by Watkins and Dayan works well for single UAV systems, but in a multi-agent scenario, this fails[1]. Because Q-learning requires many simulations to estimate the reward function, this problem becomes exceedingly complex with multiple-agents.

More recent work, such as by Das, et al., doesn't solve this problem either[2]. Many algorithms, such as those by Luna and Bekris, fail as well because they require centralized planning[3]. This approach is inappropriate for this use case because centralized control is unrealistic in many complex UAV missions. Planning using genetic algorithms, such as by Nikolos, Zografos, and Brintaki[4], perform well in static environments, but are difficult to

extend for dynamic environments. Finally, deep Q-learning methods work well for large state spaces, but require millions of training samples, as shown by Mnih, et all[5]. This large simulation requirements excludes this possibility from being used in online learning. Even extended methods, as presented by He, et all[6], require significant training outside the possibility of an onboard UAV computer.

Simulation environment

In the scenarios where the UAVs operate, there are three sets of actors: threats, targets, and UAVs. Each plays a role in how the scenario unfolds. These scenarios are run on a discretized 2D grid representing the state space of the UAVs. All system agents are assumed to lay in this grid.

Threats are static dangers that can represent terrain or anti-UAV systems. All threats are initially unknown to the UAVs and must be discovered by a UAV before they are considered by the path planning algorithm. Before a threat is uncovered, the UAV assumes there is no danger in that region. A threat is found by a UAV when the center of the threat falls inside the observation radius of a UAV. Once discovered by one UAV, the information is shared to all others. In these scenarios, threats are modeled as 2D Gaussian distributions. The mean (location) and covariance (area of effect) may vary between threats.

Targets are the second category of objects in a scenario. They can represent intelligence targets or aerial imagery assigned points. The positions of all targets are known to all UAVs from the start of a scnario. The UAVs can communicate and work together to assign a target to a UAV.

UAVs are the agents in each scenario. They are tasked with moving to as many target locations as they can before they run out of fuel. Each UAV has an observation radius within which it can discover threats. A scenario is finished when the UAVs visit all targets or all run out of fuel. A UAV must pass directly over the target point for the target to marked as visited.

Methods

The principal reinforcement learning technique used in this project was the Cooperative and Geometric Learning Algorithm (CGLA). This algorithm was proposed by a group of researchers primarily based out of Beihang University in China in 2013 [7]. It is an online learning approach which was specifically designed for helping multi-agent UAV systems navigate through uncertain environments, so it is unsurprising that it was deemed a useful starting point for this project.

The algorithm effectively generates a cost matrix, G, for each state in the map, then each UAV runs through the lowest-cost path toward its target. CGLA is model-free, meaning the UAVs do not require a prior estimate of the distribution of threats on the map, and an efficient path can always be found to a target based on what is known. In addition, the cost matrix can be iteratively updated as new threats are encountered, and will continue to encode a memory of all obstacles encountered during the scenario.

The precursor to the cost matrix is a so-called "weight matrix," A, which for a square map of dimension n is an $n^2 \times n^2$ matrix defining the Euclidean distance and integral risk from a state i to every other state j, as seen in Equation 1. Note that the function $F(x, y)$ simply represents the combined PDFs of each known risk distribution evaluated at a point x, y. Initially when no threats are known to any of the UAVs, the risk of traveling between any two states is zero. Gradually as more threats are encountered, the risk becomes a much more significant factor in the weight matrix. Additionally, the parameter K represents the relative weighting of risk and distance in the A matrix such that for larger values of K,

the path planning algorithm will favor less risky paths.

$$A_i, j = d_i, j + K \int_i^j F(x, y)d \tag{1}$$

After computing all values of the weight matrix, the cost matrix is found using the approach outlined in Algorithm 1. Finally, each UAV is set on a path toward its target which results in the minimum total cost. In the event that any UAV encounters a previously unseen threat within its observation radius, the cost matrix A is recomputed to account for the new source of risk, and the entire procedure is repeated. A scenario is considered complete once all target points have been visited.

Algorithm 1 Cooperative and Geometric Learning Algorithm

Input: Weight matrix of the known map A

1: Initialize $G_t^0 = 0$ for the target state t of the given UAV
2: Initialize $G_i^0 = \infty$ for all other states $i \in \left(1, n^2\right) \setminus \{t\}$ on the map
3: **repeat**
4: **for** Each state $i, j \in \left(1, n^2\right)$ **do**
5: $G_i^{k+1} = \min\left\{G_i^k, A_{i,j} + G_j^k\right\}$
6: **end for**
7: **until** $G^{k+1} = G^k$

Output: G^k

It should be noted that this algorithm extends very naturally to a scenario with multiple UAVs and multiple threats. Since the weight matrix A encodes the risk of traveling between any two states, its values are fixed for a given map of known threats. When new threats are discovered by any UAV, this information is shared among all UAVs by updating the underlying shared A matrix. Each UAV then has its own cost matrix G, with the sole distinction between them being the target state associated with that UAV.

Experiments and Results

Various scenarios were run to verify the performance of the algorithm under different test conditions. These scenarios were comprised of different threat distributions, numbers of UAVs, and numbers of targets. In every case the map was discretized into a 20×20 grid, since it was determined that this level of detail offered a fair compromise between map fineness and computational complexity.

Before running any particular simulation, however, it was useful to run a few experiments to gain a sense of how the algorithm's tuning parameters affected performance. The two parameters of interest were identified as the factor K (the relative weighting between path distance and risk) and the observation radius of the UAVs. To this end, a representative scenario with a single UAV and single target was run subject to variable values of K and observation radius, with the path planning outcomes summarized in Tables 1 and 2 below. Note that while the path distance and danger are not assigned units, this is unimportant since the relative values of each are the only relevant takeaway.

Based on the results below, K was set generally between a value of 1000 and 2000. Notice that above this point, the path planning algorithm becomes focused on risk avoidance to the point that it takes overly-circuitous routes and winds up accumulating more risk on its long path. A similar survey of the effect of varying the UAV observation radius is shown in Table 2. Notice the large drop-off in danger between a radius of sizes 2 and 3; based upon this, the radius of 3 was selected for all simulations to balance between generating

Table 1: Relative path distance and danger for various levels of K

K	Distance	Danger
10	23.56	0.2148
100	26.38	0.0669
1000	26.97	0.0592
5000	59.94	0.0684

Table 2: Relative path distance and danger for various observation radii

Radius	Distance	Danger
1	20.38	0.5642
2	26.38	0.2678
3	27.80	0.0582
4	26.97	0.0131

safe paths and not overexposing the UAV to the entire map.
Upon tuning these two parameters, the key features of the algorithm were tested in relevant scenarios. The first and most important of these was the ability to learn to navigate safely to the target in an initially unknown environment. As seen in the bottom right of Figure 1, a map was set up with threats distributed such that the UAV (in red) had to find a small gap which would allow it to travel safely to the target (in blue). Initially, the UAV defines a naive path diagonally across the map as seen in the top-left plot. After progressing along this path, it encountered a threat which completely obstructed its path, and thus generated a new path that routed around the threat. This process repeated two more times until the UAV was able to pass through the gap and safely reach the target.

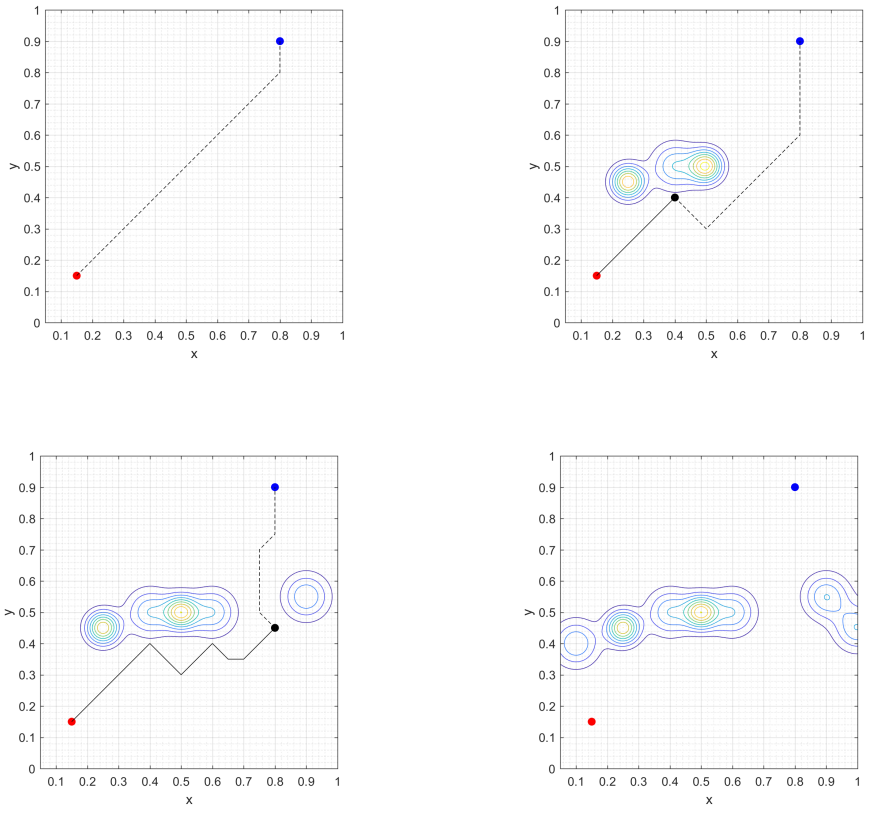

Figure 1: Progression of a single UAV learning an unknown environment (starting from top-left) compared with the true threat distribution (bottom-right)

In a similar vein, it was important to assess the cooperative nature of CGLA by comparing the performance of single- and multi-UAV teams acting on the same map. A scenario was set up with threats on opposite sides of the map such that a single UAV could not be initialized with full knowledge of the threat environment (see Figure 2). In one case, a lone UAV was placed on the map and initialized with knowledge of only one threat (note the

observation radius is depicted by the black circle). It traversed the bulk of the map, but then encountered two new threats near the target and subsequently rerouted around them. In another case, a second UAV was introduced which was initialized near the target-side threats. In this scenario, it can be seen that the first UAV immediately plans a safe route around those further threats without the need to encounter them in its own observation radius.

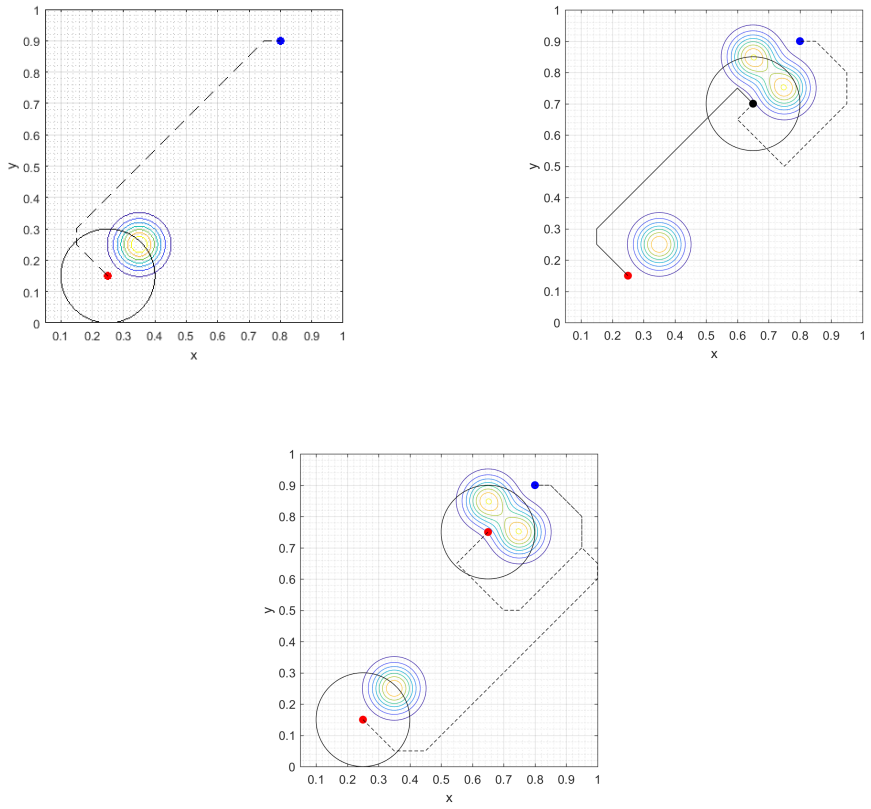

Figure 2: Single UAV path planning without shared information (top) compared with multi-UAV cooperative path planning (bottom)

Future Work

Ultimately, CGLA yielded positive results on the scenarios of interest. While the simulation environment made certain simplifying assumptions (e.g. the UAVs were able to communicate perfectly, UAVs were able to perfectly model threats within their observation radius), the algorithm was still an effective means for implementing online path planning of distributed UAV systems. In the end, the largest drawback was the relatively slow runtime of the algorithm which grows exponentially for very finely discretized state spaces. This is perhaps to be expected given the fact that computation of each A matrix requires $\mathcal{O}(n^4)$ time, and the G matrix requires at least $\mathcal{O}(n^2)$ time.

With that said, the algorithm offers fertile ground for improvements in programmatic efficiency, which would need to be a high priority task for future iterations of this project. It would also be tremendously useful to introduce mobile threats into the simulation environment, since it is perhaps reductive to assume that all tactical threats are stationary on the map. Lastly, this algorithm would need to be implemented on real UAV hardware to truly gain a sense for its ability. Once this is done, a comparison study against more traditional distributed path planning algorithms could offer tremendous insight into the strengths and weaknesses of the CGLA approach.

Contributions

The work on this project has been an even split. Both team members spent equal amounts of time working on literature review, implementation of CGLA, feature development, and assembly of deliverables.

References

[1] Watkins, Christopher JCH, and Peter Dayan. "Q-learning." Machine learning 8.3-4 (1992): 279-292.

[2] Das, Pradipta K., et al. "An improved Q-learning algorithm for path-planning of a mobile robot." International Journal of Computer Applications 51.9 (2012).

[3] Luna, Ryan, and Kostas E. Bekris. "Efficient and complete centralized multi-robot path planning." Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on. IEEE, 2011.

[4] Nikolos, Ioannis, Eleftherios Zografos, and Athina Brintaki. "UAV path planning using evolutionary algorithms." Innovations in Intelligent Machines-1 (2007): 77-111.

[5] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).

[6] He, Frank S., et al. "Learning to play in a day: Faster deep reinforcement learning by optimality tightening." arXiv preprint arXiv:1611.01606 (2016).

[7] Zhang, B., Mao, Z., Liu, W., et. al. "Geometric Reinforcement Learning for Path Planning of UAVs," Journal of Intelligent and Robotic Systems, Volume 77, No. 2, 2015, pp 391–409.

[8] Zhang, B., Mao, Z., Liu, W., et. al. "Cooperative and Geometric Learning Algorithm (CGLA) for path planning of UAVs with limited information," Automatica, Volume 50, No. 3, 2014, pp 809-820.