KIET IJCE KIET International Journal of Communications & Electronics
Volume. No. 7, Issue No. 1, Jan-June 2019, ISSN: 2320 – 8996

Multi-Agent UAV Path Planning

Shubham Shukla[#1], Atul Kumar[*2], Akshay Singh[$3]
[1,2]Department of Electronics & Communication
[1.2]University Of Allahabad
[1]Shukla.shubham1989@gmail.com,
[3]Atulkashyap59@gmail.com

ABSTRACT

This paper introduces a simulation designed to test real-time path planning done by single and multiple agents. The components of the simulation include a road network, several Uninhabited Aerial Vehicles (UAVs) with electro-optic sensors, a target and an Uninhabited Ground Vehicle (UGV). The target and UGV are located on the road network. Random blockages are placed on the road network, possibly preventing UGV traversal to the target. These blockages can be detected by the UAVs. Figure (i) outlines the components in the simulation.The task of the UAVs is to scan the road network and find the optimal clear path for the UGV. The simulation ends when the optimal path is found or no clear path exists. An online path planning algorithm extended from A* called Online A*, is used in the agents which control the UAVs. The algorithm is heuristic based and finds the optimal path in a dynamically changing environment. Results from single-agent trials in the simulation showed that in each case, the agent found the optimal path through the road network or determined that no clear path existed.

Keywords*: Path planning; agent; unmanned aerial vehicle; simulation; dynamic programming.*

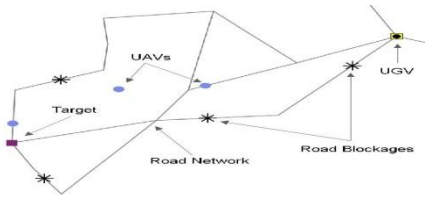

Figure (i). Components of the simulation

INTRODUCTION

Multiple uninhabited (or unmanned) aerial vehicles (UAVs) working cooperatively have significant advantages over single UAV platforms when applied to various military applications. Advantages include increased speed of task completion as well as improved robustness in the system. A military application applicable to multi-UAV platforms is that of path planning. This paper introduces a simulation designed to test techniques and strategies for multiple UAVs that have the task of planning a path for an uninhabited ground vehicle (UGV). The problem presented in this paper is different to previous research, which focuses primarily on robots performing their own path planning. UGVs navigating in an environment have difficulty detecting holes or ditches (Stentz *etal* 2002). Having UAVs perform path planning forUGVs is therefore advantageous.

In 1994, Stentz introduced the Dynamic A* (D*) path planning algorithm. The D* algorithm efficiently replans a path for a robot in a dynamically changing environment. Stentz (1995) further optimized D* to produce the Focussed D* algorithm. Other solutions to the path planning problem include D* Lite (Koenig and Likhachev 2002), and more recently the Delayed D* algorithm (Ferguson and Stentz 2005), all of which perform an initial (offline) path search before the robot sets out. Every time an obstacle or discrepancy in the path is detected, the algorithms efficiently replan a whole new path either from the start to the goal or from the agents' location to the goal. While

this technique is appropriate for robots performing their own path planning, it is unsuitable for a team of UAVs performing path planning for an UGV.
The path planning algorithm presented in this paper takes advantage of the fact that no initial path search is required for the UAVs. In fact, it is not crucial for the UAVs to find the optimal path, in one go. A reasonable amount of "back tracking" is allowed, as long as they find the optimal path. The algorithm presented is a simple extension to the heuristic search technique A* (Nilsson 1980), transforming it to an online algorithm able to adapt to a changing environment.

In the second half of the paper, a stochastic dynamic programming formulation (Bertsekas 2001) of the path-planning problem is outlined. Here, instead of searching road paths, as does the previous work, critical intersections or nodes of the network are searched. From the stochastic dynamic program the optimal policy is calculated via valueiteration. Finally, Monte Carlo simulations are conducted to test strategies for two UAVs collaboratively path planning.

SIMULATION

Outline

The agents run in a simulation called Tempest Seer. The main feature of Tempest Seer is that it can run trials hundreds of times faster than real-time. The components used in the simulation include a road network, several UAVs with electro-optic sensors, a target and an UGV. Figure 1 outlines the components in the simulation. Both the target and UGV are located on a road network. A road in the network can be blocked for any reason. These blockages on the roads can be detected by the UAVs. The UAVs job is to search the road network for an unblocked optimal path from the target to the UGV. The simulation ends when either an optimal path is found or no clear path exists.

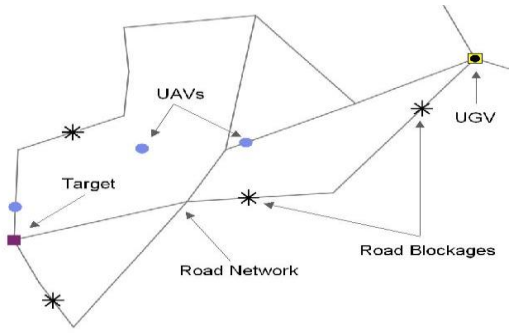

Figure 1. Components of the simulation

On initialization, the team of UAVs has knowledge of the topology of the road network and knows the location of the UGV and target. The road network is represented as a graph, with intersections representing vertices and roads representing edges. A pre-check is done to make sure the graph is connected (i.e. there exists a possible path from the target to the UGV). The simulation produces a random road network, and places the target and UGV at random locations. Blockages can occur on roads and intersections, possibly preventing UGV traversal. Blockages were placed at random locations on the network. During a run of the simulation, the blockages are stationary and no new blockages are added. The locations of the blockages are initially unknown to the team of UAVs.

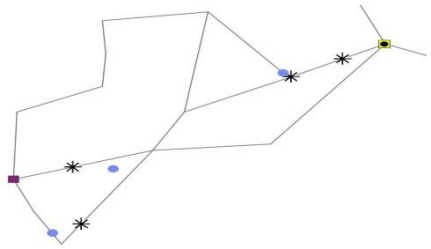

Figure 2. Blockage encountered

Single UAV Path Planning

Separate threaded agents control the UAVs in the simulation. The UAV agents use an extended version of the A* algorithm (Nilsson 1980) to search the road network for an optimal path from the target to the UGV. A* is an optimal, complete and computationally efficient search technique (Nilsson 1980). In A*, the nodes that can be searched are stored in an "open list". Each node in the open list has a back pointer to its parent node, and an associated value: $f(n)$.$f(n)$ equals the

sum of the cost of getting to node n from the start node, plus an estimate of the cost of getting from to the goal. At each iteration, A* chooses the node with the smallest $f(n)$ to search, and adds

this node to a "closed list". The search ends when the node chosen with the smallest $f(n)$ is the goal

node. At this point, A* follows the back pointers of the nodes contained in the closed list starting at the goal node, to produce an optimal path from the start to the goal. For a more detailed explanation of A*, refer to Nilsson (1980). In this paper the A* algorithm has been extended to create a true online algorithm called Online A*. Similar to D* (Stentz 1995), the online A* algorithm finds optimal paths in a dynamically changing environment. Online algorithms run while the agent is traversing the path, allowing the agent to dynamically replan if an unanticipated situation is encountered.

Online A* is similar to A* but has two key differences. Firstly, it is not run in entirety before the agent sets out, but runs while the UAV is in motion. That is, when the algorithm selects the best road for the UAV to search, it pauses and waits till the UAV has finished scanning the road. The second difference is that while the UAV is scanning the road, if a blockage is found the UAV informs the algorithm, which in turn penalizes that road. Only when a road has been fully traversed by the UAV and no blockages are found that the road is added to the closed list in the algorithm. Below is the pseudo-code for the online A* algorithm:

P Run A* to select next best road.
Q Pause while UAV scans road.

R If road is clear, add road to closed list, return to (1).
Else if road is blocked, penalize road, and leave on open list, return to (1).

In the simulation, once the UAV has located the target it starts the path search from there. The online A* algorithm chooses the next best road to take and the agent commands the UAV to fly along the road to the next intersection. While the UAV is flying above the road it constantly scans the road for blockages. If a blockage is found, the search algorithm heavily penalizes the road by adding to its score a maximum value, and finds the next best road to search. The search ends when either the UAV arrives at the UGV, or if the UAV has no more paths left to check. If the UAV arrives at the UGV, the algorithm creates a path from the roads in the closed list, and coverts the path to a set of waypoints for the UGV to traverse. If the UAV has no more paths to check, it implies that each possible route from the UGV to the target is blocked.

The major difference between online A* and other path replanning algorithms is that at the end of the online A* algorithm only one path has been created. D* and the other path planning algorithms mentioned, replan and create a whole new path either from the start to the goal, or from the agents location to the goal, every time a blockage is found in the road network. The online A* algorithm is thus computationally efficient, and it is extremely simple, as it is just a small extension to A*. Trials were run in the simulation with different road networks and different locations and numbers of blockages, in each case the agent found the optimal path through the network or determined that no clear path existed.

Multiple UAV Path Planning

While there has been extensive research done on single agent path planning, multi-agent path planning has received less attention. Market economy (Zlot *et al* 2002) and auction architectures are examples of robust distributed multi-agent frameworks, but rely heavily on inter-agent communication. When dealing with UAVs, keeping inter-communication low is desirable. A simpler robust architecture is required.

Different variations of an online A* multi-agent path planning technique were tested in the simulation. Initially a centralized architecture was

developed. The architecture consisted of a single online A* algorithm which was run on a central UAV agent. All new information on the network found by the agents would be returned back to the central agent. Similarly all agents would request their next best road to check from the central agent. Once a road has been chosen by an agent to scan, no other agent may choose it. All agents in the simulation started their search at the target, resulting in a "breadth-heavy" search (Dijkstra 1959). Alternatively it is possible for some agents to start at the target and others to start at the UGV. This would result in a "depth -heavy" search (Dijkstra 1959). Figure 2 shows a simulation with3 UAV agents collaboratively searching, and one agent detecting a blockage.

While the centralized architecture ran well in the simulation, it is not robust enough for real-world applications. For example, if the central UAV in this architecture fails for any reason, the other UAVs cannot continue their task. A distributed architecture was then developed. In this architecture each agent runs its own online A* algorithm. However, when an agent starts searching a road, it broadcasts this road to all other agents in its team, alerting them to add this road to their open list if it doesn't already exist, and to mark the road as "being searched". Thus, an agent won't start scanning a road already being searched by another agent. Similarly when an agent adds a new road to its closed list, it broadcasts this road to all other agents in its team. An agent receiving this broadcast initially checks to see if the road is in its open list, and if so removes it. The agent then adds the transmitted road to its own closed list. Also, when an agent adds new roads to its open list it broadcasts this information to the other agents, acting as a form of information sharing amongst the agents. There is a chance an UAV may fail midway through scanning a road. In this situation the remaining UAV agents will find a road still on their open list but marked as "being searched", which will be false. The solution is to let the agents now scan this road marked as "being searched"; as long as it has no other roads to check, or all other remaining roads in its open list have values greater than the maximum value. The task is complete when at least one of the agents holds a complete path from the target to the UGV in its closed list, or if all agents have no road in their open list with a value less than the maximum value.

The distributed architecture has many advantages. Being decentralized, if one UAV fails, the task is only slightly hindered, giving the effect of gracefuldegradation. There is a replication of information among the agents. If an UAV fails, it is likely that it had the chance to communicate at least some of its discoveries. Thus a total loss of information is avoided. In the worst case where no communication between the agents is possible, each agent would perform the entire search themselves, leading to a less efficient but satisfactory result. Trials were conducted in the simulation to determine whether applying more UAVs to the path search task would result in a linear reduction of completion times.

2.4. Simulation Results

Sets of trials of the simulation were run with one UAV, and teams of two, three and four UAVs. They were run on random road networks with ten, one hundred, and one thousand roads. For simulations with ten roads, trials were run with zero, one, two and three blockages placed at random locations. With one hundred roads, simulations were run with zero, ten, twenty and thirty blockages. For one thousand road simulations, zero, one hundred, two hundred and three hundred blockages were tested. Figure 3 shows a histogram for one hundred thousand trials conducted for one UAV scanning random networks with one thousand roads and with one hundred blockages placed randomly on the network. The x-axis is the simulation time taken to find the optimal path or determine if no clear path exists. The y-axis is the frequency of the time taken. In Figure 3, the distribution was cut at 2000 milliseconds for picture clarity.

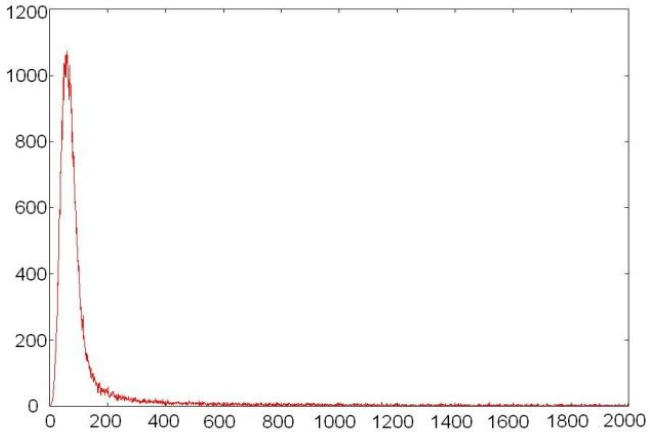

Figure 3. Frequency of time taken for one UAV tofind a path in networks of 1000 roads (y-axis). Simulation time taken (x-axis).

The distribution in Figure 3 starts with a large spike followed by an extremely long tail, and can be fitted to a gamma distribution (Law and Kelton 1991) with $\alpha = 3.48$, and $\beta = 23.27$. The distributions for trials of networks with one hundred road has a similar spike, but with a shorter tail. While the distributions for trials of ten roadnetworks has a similar spike but with only a few outliers. Because of this long tail, it is meaningless to describe the results in terms of mean and variance. A quantile analysis was performed and the results are shown in Tables 1, 2 and 3.

Table 1. Percentage of 10 roads trials completedwithin the simulation time shown.

Blockages	UAVs	75%	90%	100%
0	1	100	338	1478
0	2	80	243	895
0	3	63	152	530
0	4	61	135	407
10	1	130	483	1788
10	2	96	291	1048
10	3	80	191	709
10	4	64	155	641
20	1	156	559	1678
20	2	108	360	1170
20	3	95	230	738
20	4	73	165	502
30	1	225	666	1647
30	2	130	325	829
30	3	84	243	908
30	4	86	217	795

Blockages	UAVs	75%	90%	100%
0	1	247	4991	24513
0	2	146	3125	16728
0	3	125	1982	7910
0	4	126	1659	6408
100	1	239	8202	27793
100	2	166	3323	24726
100	3	143	2407	11895
100	4	137	1827	6683
200	1	459	10134	27693
200	2	270	4122	19176
200	3	243	2933	9185
200	4	150	2150	8151
300	1	836	11378	25981
300	2	285	4867	13739
300	3	279	3448	10842
300	4	144	2175	6917

Table 3. Percentage of 1000 roads trials completedwithin the simulation time shown.

Blockage	UAVs	75%	90%	100%
0	1	45	59	150
0	2	36	49	99
0	3	34	45	93
0	4	30	41	79
1	1	47	70	221
1	2	37	50	122
1	3	35	44	101
1	4	32	43	91
2	1	47	66	187
2	2	38	51	120
2	3	34	44	102
2	4	33	44	106
3	1	51	74	206
3	2	39	52	117
3	3	35	48	89
3	4	34	44	87

Table 2. Percentage of 100 roads trials completedwithin the simulation time shown.

Tables 1, 2, and 3 show for each set of trials, the percentage of trials completed within the simulation time shown. Analysis of the results illustrate that the majority (75%) of the trials were completed quickly, as indicated by the spike in the distributions. The remainder of the simulations took a lot longer to finish. An example of this would be when each road immediately connecting the goal is blocked while the rest of network is relatively clear, the UAV(s) would then spend a lot of time
scanning the whole road network, in hope of finding a clear path to the goal. The results indicate in general that adding more UAVs to the task will reduce the completion time. The greatest
benefit of applying multiple UAVs to the path search task was in trials containing one thousand roads, where a team of four UAVs completed the task roughly four times as fast as a single UAV.

DYNAMIC PROGRAMMINGFORMULATION

In this section a stochastic dynamic programming formulation of the path planning problem is presented. The goal of this work was to find the optimal policy for two UAVs collaboratively path planning, and then to evaluate various strategies for the UAVs via Monte Carlo simulations.

3.1. Outline

In this simulation, critical intersections or nodes of the network are searched, instead of the roads as per the previous work. Implicit in this formulation is the assumption that all edges or roads are passable, as the UGV can only be blocked at the nodes or intersections. We start with a road network containing 10 nodes, and the nodes are labeled 0 to 9. A target is located at node 0, while an UGV is located at node 9. Two UAVs search the network, traveling from node to adjacent node, to find an admissible path for the UGV.

3.2. Transition Probabilities

At the start of the simulation each node is labeled as unknown (U). As each UAV enters a node it determines whether it is blocked (B) or clear (C), and changes the label of the node. Below are the transition probabilities forming the Markov Chain for the process. The transition probabilities specify the probability whether a node is determined to be

(s_n, s_{net}, x_1, x_2) . Where s_n is defined as the state of the nodes: $s_n \in a_0, a_1, \text{L}, a_9$, and where $a_i \in C, B, U$ stands for a clear, blocked or Uncertain node. s_{net} is defined as a mapping from

the probabilities that a transition is made from state s_n to s_n' , given UAV_1 can move from node $x_1$to $x_1'$and $\text{UAV}_2$from node $x_2$to x_2'.Finally, $V(s_n', f(s_n'), x_1', x_2')$ refers the value of the state of individual nodes onto the network state as a whole, either as passable, blocked or uncertain:

$$f : S_n \rightarrow \{Passable, Blocked, Uncertain\}. \quad (3)$$

For example, if certain nodes are blocked (such as node 0) the whole network is blocked and no clear path exists from the target to UGV. Or if all the nodes are uncertain, the whole network is

determined to be uncertain. The current position of
the UAVs is denoted as x_1 and x_2, where

$x_i \in 0,1,L,9$ and represents which node each

Table 4. Average number of moves for strategiesagainst three transition probabilities.

Strategy	0.8	0.5	0.2
Both start at Target	4.066	3.776	2.682
One at target, one at UGV	3.463	3.811	3.306
Both start at UGV	5.076	4.599	3.549
Both start at random nodes	3.703	3.956	2.998

UAV is currently at. If the whole state of the network is either passable or blocked there is no more work for the UAVs to do, they simply return the result to the UGV. Therefore the states (s_n, P, x_1, x_2) and (s_n, B, x_1, x_2) are considered to be the end or goal states, which return rewards to the value function. The value function for a blocked network is:

$$V(s_n, B, x_1, x_2)1. \qquad (4)$$

The reward for a passable network is more specific. In particular we would like one UAV to complete its search at the starting point of the UGV, so the UAV may search again ahead of the UGV, as it heads towards the target. The value function is defined below:

being in state $(s_n', f(s_n'), x_1', x_2')$.

3.4. Monte Carlo Simulation and Results

The value function was solved via value iteration to produce the optimal policy (Bertsekas 2001). Monte Carlo simulations were run using the optimal policy on different starting locations for each UAV. The starting strategies evaluated were:

[2] Both UAVs start at target.

[3] One starts at target, the other at UGV.

[4] Both UAVs start at UGV.

[5] Both UAVs start at random nodes.

Sets of trials of each strategy were run for three different transition probabilities: 0.8 (i.e. the probability a node is clear equals 0.8, and 0.2 if it is blocked), 0.5 and 0.2. From the simulations, the average number of moves taken to reach a goal state (i.e. if the network was blocked, or a path existed) was found. Table 4 outlines the results.

FUTURE WORK

It is the intent of future work to test other strategies using Monte Carlo simulations, as well as to modify the dynamic program to search roads instead of critical intersections, and take into account road length. Extending the dynamic program to more complicated networks is planned, but would require either to divide the network into sub networks to solve the optimal policy or to take advantage of obvious symmetries in the problem to reduce the state space.

Future work also includes additional experimental testing of the multi-agent online A* algorithm to determine the optimal number of UAVs for a path planning task with road networks of varying sizes and topologies. Further analysis will also be conducted to determine whether there are common topological features, that exist in road networks in which the multi-agent online A* algorithm takes the longest time to compute.

CONCLUSIONS

This paper presented a simulation designed to test single and multiple agents performing a path planning task. An online search algorithm extended from A* was applied to a single agent in

the simulation. A multi-agent version of the extended A* algorithm was outlined, both as a centralized and a more robust distributed architecture. Results from trials of the simulation showed that adding more UAVs to the path planning task generally reduced completion time, however it was not a linear reduction. The benefit of applying more UAVs to the task varied between networks of different sizes and topologies.

A stochastic dynamic programming formulation of the problem was also presented. The optimal policy for a ten-node network with two UAVs performing node search collaboratively wascomputed. Monte Carlo simulations were run, testing the strategies for the optimal starting locations for each UAV. Results showed that if the road network tested was likely to be clear, the best strategy is to start one UAV at the target and one UAV at the UGV. Alternatively, if the road network tested was expected to be heavily blocked the best strategy is to start both UAVs at the target.

REFERENCES

[1] Bertsekas, D. P. (2001), *Dynamic Programmingand Optimal Control*, Volume 2, 2nd Ed.,Athena Scientific, Belmont, Massachusetts.

[2]Dijkstra, E. W. (1959), A note on two problems in connection with graphs, *NumerischeMathematik*, Volume 1, 269-271.

[3] Ferguson, D., Stentz, A. (2005), The delayed D* algorithm for efficient path replanning, In *Proceedings of the IEEE International Conference on Robotics and Automation*,April 2005.

[4] Koenig, S., and Likhachev, M. (2002), D* Lite, In *Proceedings of the National Conference on Artificial Intelligence*, 476-483.

[5] Law, A. M., Kelton, W. D. (1991), *SimulationModeling and Analysis*, 2nd Ed. McGraw-Hill,Inc.

[6] Nilsson, N. J. (1980), *Principles of ArtificialIntelligence*, Tioga Publishing Company, 72-88.

[7] Stentz, A. (1994), Optimal and efficient path planning for partially-known environments, In *Proceedings of the IEEE International Conference on Robotics and Automation*,3310-3317, May 1994.

[8] Stentz, A. (1995), The focussed D* algorithm for real-time replanning, In *Proceedings of theInternational Joint Conference on Artificial Intelligence*, 1652-1659, August 1995.