Neural Tree Expansion for Prioritized Multi-Agent Path Deconfliction

Khushal Brahmbhatt
Oregon State University
Corvallis, OR
brahmbhk@oregonstate.edu

Andrew Festa
Oregon State University
Corvallis, OR
festaa@oregonstate.edu

Kavinda Senewiratne
Oregon State University
Corvallis, OR
senewiry@oregonstate.edu

ABSTRACT

Coordinating search efforts in high-risk and time sensitive environments has the potential for applications ranging from rescuing lives to exploring previously inaccessible locations. However, this problem is made extremely difficult due to the inherent communication restrictions, real-time requirements, and potential for collisions of any such solution. Previous work has focused on solving path deconfliction in an efficient manner decentralized planner that allows each agent to learn individually how to optimize a global value. Even for approaches that allow for prioritization between different agents, these prioritization schemes generally rely on some characteristic of the environment as a means for the agent to infer its priority. In this paper, we propose a solution to allow for an explicit total ordering among the agents in the system in order to allow for valuing completion of more time-critical tasks before lower critical tasks. This is done by having a priority assigned to a task, allowing for a variable number of agents (with differing priorities) in an agent's field of view, and then teaching each agent when to yield right of way but only in the case of a potential conflict. This allows for a system designer or dispatcher to explicitly prioritize solving important tasks when presented with a large number of tasks that must be completed in parallel.

KEYWORDS

multi-agent, decentralized, Monte-Carlo tree search, priority path planning

1 INTRODUCTION

Due to rising global temperatures, there has been a significant increase in the number of wildland fires in the USA. In a span of 30 years, the number of annual wildfires have doubled from approximately 500 to 1000 [2]. There were about 1.3 million acres burned by wildland fires in 1983, and this number grew to approximately 10 million acres by 2020 [3]. In 2021, the Bootleg Fire which started in Southern Oregon, burned almost 500,000 acres of land [1]. This is almost half the number of acres totally burned in 1983 across the entire USA. In 2018, the estimated insured losses totaled to about $10Bn in the USA, and the Federal Government spent $3Bn for wildfire suppression. More importantly, there were countless lives (civilian and firefighters) lost [4]. If we can better fight these fires and predict how they are going to evolve over time, these major losses may be able to be mitigated.

Fighting these fires requires fast decision making under extreme conditions, and a single, wrong action could result in losing lives or damaging valuable assets. Data and information are vital for any decision making process, and fighting wildland fires requires a lot of real-time information as there are many constantly changing factors (e.g. weather conditions, fuel type, fuel moisture, etc), which affect the rate of spread. Specifically, observing how the boundaries of the fire are growing provides a lot of information useful for predicting the future path of the fires. These boundaries, called fire fronts, are where the fires are actively burning, and any given wildland fire usually has several fire fronts. Thus, monitoring these boundaries is a difficult task due to the highly dynamic and time-sensitive nature of the data collected.

There are many methods used to monitor the behavior of the fire and conditions around it. A primary method is satellite imagery. Generally, it is used for initially detecting a wildland fire. However, due to the low temporal resolution of the imagery they provide, it is difficult to rely solely on this data as the fronts are constantly changing. Stationary, terrestrial sensors could also be used. However, in a wildland fire, communication is extremely limited. This could be due to infrastructure that has been damaged by the fire or that the fire is taking place is an extremely remote location where there is no ground communication infrastructure in place.

All of this amounts to needing a real-time monitoring solution that is capable of operating in areas of low visibility without relying on communication to a centralized location. Unmanned Aerial Vehicles (UAVs) are ideal for the task of reconnaissance, but are limited in air time due to the current limits of battery technology. Due to this, any autonomous system used, would need to plan efficiently to make the most out of the resources available to them. Using a large number of UAVs would allow the task to be spread over several agents and thus help alleviate the battery limitation. Drone swarms have been proposed in helping with monitoring and suppressing wildland fires [5]. However, as the size of the swarm increases and the communication remains tenuous, it quickly becomes intractable for the drones to plan non-conflicting paths.

This work contributes to the problem of finding near-optimal prioritized non-conflicting paths among multiple agents with limited communication by defining an explicit objective-based prioritization scheme that can be applied to the work outlined in [10]. In this way, different aspect of a meta-problem can be addressed with high importance even when the number of agents and objectives in the system becomes intractable for directly computing an optimal solution.

We use a Neural Tree Expansion (NTE) algorithm [10] that leverages Monte-Carlo Tree Search and policy and value networks to learn optimal paths for each drone from a starting position to a given goal position on the fire front. To plan for collision-free paths between the drones, we need a way to deconflict colliding paths. To this end, we use a priority-based reward function to determine

ROB 537, , October 2021, Corvallis, OR, USA. © Learning-Based Controls

which drone gets priority along its planned path and which one has to wait or find a different path. The priorities are assigned to the goal positions and change with time to represent the criticality of that part of the fire front. This priority is used in a shared system reward that is passed to each NTE model associated with locally planning each drone's trajectory.

2 BACKGROUND

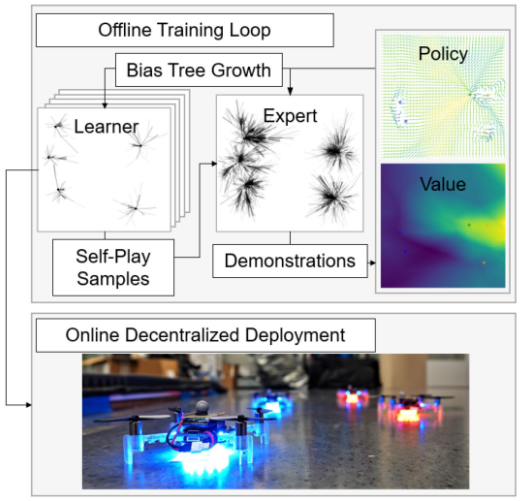
In multi-agent path planning, multiple agents attempt to find the shortest paths to reach their individual goals. However, in order to avoid colliding, each agent may have to follow a sub-optimal path compared to if they were the only agent in the system. The problem thus becomes how to search a set of non-conflicting paths in order to jointly minimize the time each agent takes to reach its goal and the total throughput of all the agents in the system. To add to the challenge of the problem, there often exists a priority among the agents. That is, it would be desired that a certain agent, or subset of agents, reach their targets faster than those with a lower priority. At the extreme ends, solving this optimization problem is often done in one of two ways: centralized and decentralized. In a completely centralized approach, each agent is able to communicate with the other agents in the system, while in a completely decentralized solution, each agent only performs its own computations or a local search without additional knowledge from other components in the system. There is an inherent trade-off in the two approaches in terms of optimality and resource cost, and so most approaches fall somewhere in between [15].

The centralized approach is able to compute the optimal solution that would jointly minimize the multiple objectives, but it comes at the cost of intractability in the number of agents in the system [6]. In fact, this problem has been shown to be NP-hard [8] and thus cannot be reliably computed for any scenario with a sizeable number of agents. Alternatively, in a decentralized approach, each agent only searches a local space and has no global information regarding the other agents in the system. The issue comes in that multiple agents may make local decisions which conflict with the local decisions of other nearby agents.

2.1 Prior Work

In order to combat the intractability inherent in computing a global, optimal solution, many approaches construct heuristics that seek to efficiently find near-optimal paths. A*, and variations on it, have been shown to yield decent results [12, 14], and conflict-based search (CBS) [11] is widely used and extended in order to cut down on the number of path conflicts that must be searched through. The general idea that CBS introduces is searching at two levels: a global tree with lower resolution and a low level search for each agent. The high-level search is meant to find the points in a path where the agents may collide. The agents themselves are responsible for searching locally in order to resolve those path conflicts. In doing this, the total system is able to search only over the parts of the paths that give rise to the conflicts, and thus it greatly cuts down on the search space. However, this approach falls short when there may be multiple objectives or each agent may have a unique objective [9]. Several attempts have been made toward this particular challenge, [6, 9, 13], but they fail to consider the situation where a priority

Figure 1: Neural Tree Expansion Training Loop



may exist between the agents in the systems. Instead, they seek to minimize the total time it takes for all of the agents to reach their objectives rather than a weighted average for each individual agent, which is a necessary consideration in many situations, as discussed previously. In the case of UAVs tracking wildfire spread, the agents may not all be able to reliably communicate and different points along the boundary may be more critical due to proximity to urban centers. Thus, a fully centralized approach is not suitable, and a prioritization scheme is necessary in order to ensure that critical parts of the overall problem are addressed first.

Some work has been proposed to address this specific gap. In [7], the path planning problem is formulated as a mixed integer problem defined using temporal logic and uses a search optimization method to find the optimal collision-free paths based on a fairness function. In [16], the authors use a prioritization heuristic based on the number of path choices available to an agent to deconflict paths. However, this relies on knowing the robot's path options upfront and assign the priority based on characteristics of the agent and environment rather than the task of the agent(s).

2.2 Neural Tree Expansion

The approach in [10] uses a dual Monte-Carlo tree search to both search the global space and the local space of each agent. The global search, depicted as the Expert in figure 1, is only used for training each local agent and is not used when the agents are deployed.

This work extends the idea proposed in CBS of searching along conflicting paths to apply the sequential search to the agents rather than the paths, and it outlines two core loops, as depicted in figure 1: an offline training loop, where each agent is trained to imitate an expert, and an online decentralized deployment loop, where the trained agents are deployed to the real world. The training loop contains both the learning agents, all of which only have partial information, and an expert oracle, that has complete and global information. Both the learning agents and the oracle use Monte Carlo Tree Search (MCTS) to explore the possible local paths to a

particular depth and is further biased by a value function (during the selection step of MCTS) and a policy function (during the expansion step of MCTS). The learners explore various states (which are all labeled by the expert) and are then used to train the policy and value networks. This loop is repeated until the networks have converged or achieved an acceptably low loss.

Inherently, this creates a partial ordering among the agents as the agents which are searched first will have fewer constraints limiting their potential paths than those agents which search after other agents have already decided on a path. While this makes the solution non-exponential in the search space, the rigidity of the ordering does not allow for maximizing a throughput metric with respect to the relative ordering of the agents. Additionally, the prioritization is not explicitly defined. The proposed research focuses on this aspect of the problem: exploring methods for applying a dynamic priority based on the task, the agent, and the environment.

3 APPROACH

This work is an extension of the solution proposed in [10]. Specifically, it explicitly assigns a priority to an agent (or group of agents) based on the priority of the task. This allows the system designer to account for task-dependent priorities rather than computing the priority on the basis of an environment factor, as was done in [16]. However, when designing the reward function to allow for this capability, two other key features had to be considered.

First, the high-level goal of the agents in the system is to reach their assigned goal as quickly as possible without crashing. This is the basis of optimizing non-conflicting paths.

Second, the agent is assumed to not have complete knowledge about its environment or the other agents in the system. That is, it does not have any knowledge about the map it is operating in, and more importantly, it does not have a priori knowledge of how many other agents are in the system or their relative priorities. It only gains this information once another agent has been observed. To this end, one of the limitations placed on the agents is an individual sensing radius. Each agent is only able to observe obstacles and agents that are within a certain distance away from the agent.

3.1 State Representation

A key challenge of representing the state arises from the unknown number of agents in the system. It is desired that the state contains information of the other agents, especially the ones nearby as those are more likely to give rise to conflicting paths in the near future. However, an agent cannot simply include the position (and priority) of each other agent. Instead, there must be some mechanism for being able to account for a variable number of agents. Additionally, this variability must be conducive to the limited sensing range of each agent. At a high level, the state representation must effectively compress the full information describing the agent's surroundings into a fixed length vector, but this fixed length representation must be able to take into account a variable number of observed obstacles and agents.

3.1.1 Initial State Representation. One solution is to base the state representation on the assumption that each agent knows its own sensing capabilities. For example, since each agent knows that it can see all obstacles (and agents) within 5 meters of itself, then this circle around the agent with a radius of 5 meters could be discretized into a fixed number of regions.

However, this solution quickly gives rise to several problems. The first is a simple matter of inefficiency. Most of the regions surrounding the agent are likely to be empty, leading to a very sparse state space. The agent would see these all as separate states even though many of these states would be highly correlated, even if they aren't exactly the same state.

A second issue is with respect to how the regions are computed. They can either be a static size or each region is divided into a static number of regions (in the form of a Cartesian grid) and each gives rise to a separate issue.

If each sub-region is a static size, then the number of regions per agent would be dependent on the sensing range of the individual agent. It would thus be very difficult (if at all possible) to transfer the learned policy to a different agent, and each agent could not leverage information learned by another agent at the end of each episode. In this way, the system would be losing a lot of the power that the expert learner is meant to provide.

If instead the sub-regions are divided into a static number of regions, then the information contained in each region is substantially less for larger sensing regions compared to smaller sensing regions. Following along this implication, this would mean that providing the agent with a greater sensing capability would likely lead to a degradation in performance. Fundamentally, this runs counter to how performance would want to be improved. Increasing a sensing range should lead the agent to have more global information and so should increase the performance of the agent, not decrease it. Digging into why the performance is likely to decrease, the performance of the sensing capability of the agent is no longer dependent on its individual sensing capability. Rather, it would be dependent on the agent in the system with the least sensing capability (ie the weakest link).

3.1.2 Refined State Representation. Moving beyond the initial state representation is based on two observations. First is the notion that the area around each agent does not have to be divided based on a Cartesian grid. If the regions are divided based on a polar representation, with the agent at the center, then the aforementioned issue of a static number of regions would not be as pronounced as the regions all have a common intersection, and that point of intersection has importance with respect to the agent. That is, the point of intersection is the location of the agent.

Additionally, this issue of a reduced resolution for larger sensing ranges can be remedied by the second observation: the obstacles (and agents) in the system that are most likely to alter the immediate path of the agent in order to avoid a conflict are the obstacles that are closest to the agent. For example, if an agent is moving at 3 meters per second and it sees another agent 8 meters away in a specific direction and another that is 2 meters away (in the same direction), then only the one that is 2 meters away would alter the immediate path of the agent.

Using these insights, the important information for an agent can be captured in a fixed length vector that essentially is composed of two types of information: information about an agent and information about the agent's surroundings. For the information about

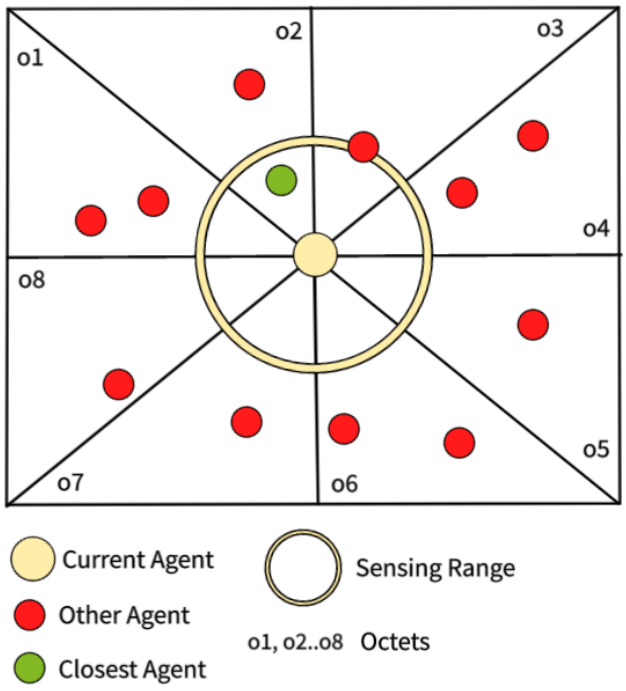


Figure 2: State Representation

the agent, this can be represented as the agent's position, it's priority, and it's goal location. For the information about the agent's surroundings, this is the distance to (and priority of) the closest agent in a particular sub-region, where each sub-region is a polar region with the agent at the origin, and the boundaries between each region occurs every $\frac{\pi}{4}$ radians. This effectively divides the area around the agent into 8 sub-regions.

Figure 2 shows an example configuration where the agent only senses another agent in octant o2. In this case, the observation of the agent would be the distance to the agent in o2 and its priority. For all the other regions, the "distance to the closest" agent would be set to the maximum sensing distance, and the priority of that region would be set to the minimum priority. This can be thought of as a signal to the agent that they have free reign to move around in that region.

More concretely, the state s of robot r_i at time t is a $1x21$ vector composed of the following elements:

$$s_t =< x_{ri}, y_{ri}, p_{ri}, x_{gi}, y_{gi}, [d_1 ... d_8], [p_1 ... p_8] >$$

where:

- x_{ri} is the x-coordinate of robot i
- y_{ri} is the x-coordinate of robot i
- p_{ri} is the priority of robot i
- x_{gi} is the x-coordinate of the goal of robot i
- y_{gi} is the y-coordinate of the goal of robot i
- $[d_1 ... d_8]$ is the distance to the closest agent in each octant $[o1 ... o8]$
- $[p_1 ... p_8]$ is the priority of the closest agent in each octant $[o1 ... o8]$

3.2 Reward Function

Our major contribution to the NTE approach for planning is the allowance for including an explicit prioritization between agents in a system, and this contribution is reached through the formulation of the reward function. As the reward is dependent on priorities, when agents are close to each other, the agent with a higher priority should be encouraged (given right of way) to continue along its optimal path towards its goal, while an agent with lower priority is discouraged from continuing along its optimal path which would conflict with the higher priority agent's path. Thus, when defining the reward function, it should satisfy certain criteria.

(1) Drive the agent towards reaching its goal
(2) Encourage agents not to collide
(3) Cause an agent to yield its path to a nearby agent of a higher priority

These criteria can all be satisfied by equation 1, which shows the reward R for agent x

$$R_x = \frac{1}{d_g} + \sum_{i=1}^{8} (P_x - P_i) \left(1 - \frac{d_{xi}}{r_x}\right) \tag{1}$$

where:

- R_x is reward for agent x
- d_g is the distance between the agent x and its goal
- P_x is priority for agent x
- P_i is priority for closest agent in octant i
- d_{xi} is the distance between agent x and the closest agent in octant i
- r_x is the sensing range of agent x

The effect of each parameter can be individually examined to ensure it causes the agent to exhibit the desired behavior. The first term $\frac{1}{d_g}$ increases as the agent gets closer to its goal. Additionally, this growth is non-linear, and so an agent that is closer to its goal is more incentivized to continue on its path. This is a desired behavior based on the idea of freedom of movement. As an agent gets closer to its target, it has fewer options about how it may be able to avoid other agents while moving closer to its goal location.

The second term can effectively be viewed as a regularization term in that it restricts the possible actions an agent may be able to take in order to achieve its goal. At a high level, the agent is encouraged, or discouraged, from entering a particular octant i based on the presence (or non-presence) of an agent in that octant. The desired behavior of this term is that an octant with a closer agent should contribute more heavily to discouraging the agent from moving in that direction, and an octant with a far (or no) agent should encourage the agent to explore that octant. Additionally, if the closest agent in that octant is a higher priority than the priority of agent x, then the agent should yield way for that other agent. However, if the priority of agent x is higher, it should have right of way in continuing along its current trajectory. This characteristic is captured by the difference between the two agents' priorities: $(P_x - P_i)$.

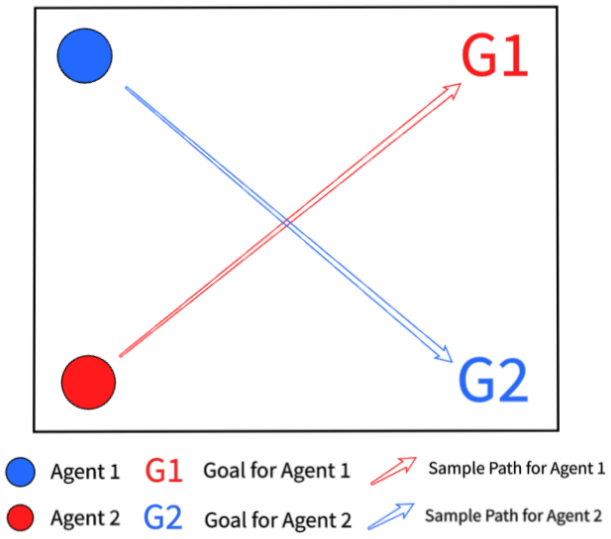


Figure 3: Simple reward test with no obstacles

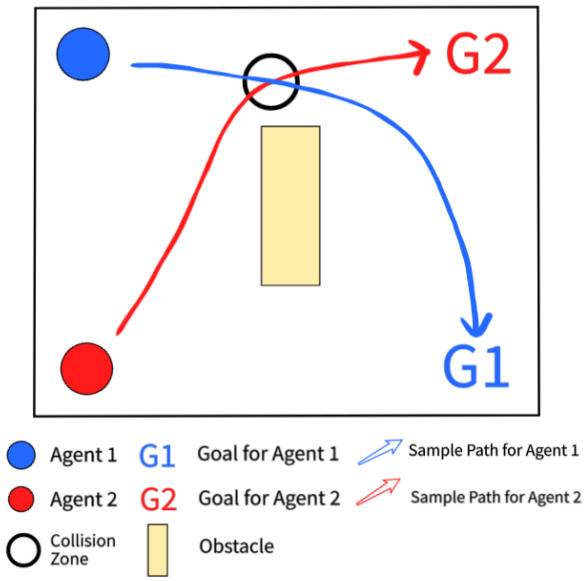


Figure 4: Simple reward test with an obstacle

4 EXPERIMENTS

The experiments for the system is dependent on two aspects: the performance of any single agent in the system and the performance of the system as a whole. That is, any single agent should not take an exorbitant amount of time to reach its goal, but an agent of lower priority should yield right of way to an agent of higher priority in the case of a potential path conflict. Additionally, the cumulative time for each agent to reach its goal should be jointly minimized.

To this end, different types of experiments were designed. The first are aimed at showing the correct behavior of the interaction between nearby (and far away) agents in the system. That is, it evaluates how much an agent's path deviates when encountering a single agent with a higher (and lower) priority.

The second type of experiments are scaled up versions of the two agent-two goal test. They contain increasing number of agents and goals in an attempt to evaluate the point where the system is no longer able to learn to complete the task. The performance of these tests can also be compared against the results found in [10] in terms of throughput of the system as the different in problem formulation is an allowance for explicitly controlling the priority of the agents in the system. This allowance should not (overly) reduce the time for any given agent to reach its goal, and so the cumulative time for all of the agents should still be minimized while allowing for this additional functionality.

4.1 Agent Interaction Experiments

The first type of experiment was implemented in two forms: with no obstacle and with a single obstacle, as shown in figures 3 and 4 respectively. As mentioned above, the purpose of these experiments is to evaluate how each individual agent responds in the case of a local conflict. To this end, the two experiments are designed to force a collision when two agents are attempting to reach their individual goal.

For figure 3, the path of each agent is forced to collide by simply placing each agent equidistant from their respective goals and having their paths cross. Note that each agent is assumed to be moving at the same speed.

The second test, shown in figure 3, is similar to the first except that there is an obstacle in the middle of the map forcing the agents to move around the obstacle. The trouble for these agents comes in two forms. It is assumed that the obstacles will make it more difficult for each agent to reach its goal, thus adding an additional level of complexity as the agent must solve two tasks: reach its goal and avoid the other agent.

The second challenge comes in the form of how information is presented to the agents. The state representation has no concept of an obstacle. Rather, the information about the world (not including the agent itself) does not include information about an obstacle. It only contains information about other agents. As mentioned previously, the cases of no agents or an obstacle in an octant is handled as a special type of agent. This test is meant to test how the agent responds to that special case of viewing an obstacle as an agent with an infinitely large priority.

Also note that for this test, the velocities of each agent is not the same. Rather, agent 1 moves significantly slower than agent 2. In doing this, it is also testing how an agent responds to another agent with a lower priority that is incapable to moving out of the way. This could be thought of as a scenario where another agent has a faulty sensor and does not see the agent with a higher priority.

4.2 System Throughput Experiments

For evaluating the throughput of the system, as the results are to be compared against the results found in [10], the same tests were used as provided along with two additional tests that use their provided code and simulator.

The first test, lovingly named "example4", is described as "3d double integrator, multi robot uncooperative target", and for their experiment, both the state and action spaces are continuous. However, for our formulation, the action space is discrete as it only allows for the agent to move in one of 8 directions. In this example, there are only two agents in the system on a fixed size grid.

The modification of "example4" to make it increasingly more difficult is to increase the number of agents in the system. By not changing the environment size, it is assumed that more potential collisions would occur as the environment is more congested. To this end, the system was tested on increasing number of agents until each agent was unable to reach it's goal in less than three times the time it would take without any other agents in the system. That is, if a single agent would take 15 seconds to reach its goal if there were no other agents, then the number of agents was increased until this agent was not able to reach its goal in 45 seconds.

Additionally, as previously stated, multi-agent path deconfliction is NP-hard in the number of agents in the system. Thus, we can expect the optimal solution to take exponentially longer for each additional agent, and so an important evaluation criteria is how long it takes for the agents to converge on learning a policy whereby they are able to solve the experiment without crashing.

5 RESULTS

Something happened.

5.1 Analysis

Here's how it is different from the NTE paper and why it happened.

6 CONCLUSION

This section will be completed after we are able to finish our experiments.

7 FUTURE WORK

In a real wildland fire, there are many dynamic obstacles, which UAVs will have to consider when planning. These include other manned aerial vehicles, trees and even fire. In our test cases, it was assumed that all obstacles are stationary. This does not mimic an actual wildland fire environment thus, we would have to modify our reward function to fairly assign reward when the environment is dynamic.

8 RESPONSIBILITIES

	Andrew (%)	Kavinda (%)	Khushal (%)
Organization	60	20	20
Technical	33	33	33
Coding	33	33	33
Writing	60	20	20

REFERENCES

[1] [n.d.]. Bootleg Fire. https://inciweb.nwcg.gov/incident/7609/. Accessed: 2021-10-25.

[2] [n.d.]. Infographic: Wildfires and Climate Change. https://www.ucsusa.org/resources/infographic-wildfires-and-climate-change. Accessed: 2021-10-25.

[3] [n.d.]. Wildfires and Acres. https://www.nifc.gov/fire-information/statistics/wildfires. Accessed: 2021-10-25.

[4] [n.d.]. Wildfires in the United States 101: Context and Consequences. https://www.rff.org/publications/explainers/wildfires-in-the-united-states-101-context-and-consequences/. Accessed: 2021-10-25.

[5] Elena Ausonio, Patrizia Bagnerini, and Marco Ghio. 2021. Drone Swarms in Fire Suppression Activities: A Conceptual Framework. *Drones* 5, 1 (2021), 17.

[6] Vishnu R Desaraju and Jonathan P How. 2011. Decentralized path planning for multi-agent teams in complex environments using rapidly-exploring random trees. In *2011 IEEE International Conference on Robotics and Automation*. IEEE, 4956–4961.

[7] Connor Kurtz and Houssam Abbas. 2020. FairFly: A Fair Motion Planner for Fleets of Autonomous UAVs in Urban Airspace. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 1–6.

[8] Bernhard Nebel. 2020. On the Computational Complexity of Multi-Agent Pathfinding on Directed Graphs. In *ICAPS*.

[9] Zhongqiang Ren, Sivakumar Rathinam, and Howie Choset. 2021. Multi-objective Conflict-based Search for Multi-agent Path Finding. *arXiv preprint arXiv:2101.03805* (2021).

[10] Benjamin Riviere, Wolfgang Hoenig, Matthew Anderson, and Soon-Jo Chung. 2021. Neural Tree Expansion for Multi-Robot Planning in Non-Cooperative Environments. *arXiv preprint arXiv:2104.09705* (2021).

[11] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence* 219 (2015), 40–66.

[12] David Silver. 2005. Cooperative Pathfinding. *Aiide* 1 (2005), 117–122.

[13] Jur P Van Den Berg and Mark H Overmars. 2005. Prioritized motion planning for multiple robots. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 430–435.

[14] Glenn Wagner and Howie Choset. 2015. Subdimensional expansion for multirobot path planning. *Artificial Intelligence* 219 (2015), 1–24. https://doi.org/10.1016/j.artint.2014.11.001

[15] Koping Wang and Adi Botea. 2011. MAPP: a Scalable Multi-Agent Path Planning Algorithm with Tractability and Completeness Guarantees. *J. Artif. Intell. Res.* 42 (2011), 55–90.

[16] Wenying Wu, Subhrajit Bhattacharya, and Amanda Prorok. 2020. Multi-robot path deconfliction through prioritization by path prospects. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 9809–9815.