# Multi-UAV Collaborative Deep Learning and Consensus to Maximize Coverage Over Continuous and Dynamic Wildfire Environment

Gaurav Srikar, Hung Manh La, *IEEE Senior Member*

***Abstract*— We all are aware of consequences of the wildfire, especially when controlling the spread is challenging. Firefighters try to understand the environment in planning strategies to get the wildfire under control. To analyze the environment, gathering information is crucial. People around the world came up with their way of approaches to achieve this task. Deploying UAVs are one of the best ways for tracking, covering, and gathering information about wildfire. A team of UAVs has additional advantages too. There were many approaches discussed regarding the use of multiple UAVs to track and cover the fire region. Very few were discussed about the communication between the UAVs with respect to their performance. In this paper, we discuss 4 different variations of communications within the team of UAVs and compare them based on a set of performance measures like not just the reward collection but also the coverage, the duration of tracking, and other environmental metrics.**

## I. INTRODUCTION

Firefighters risk their lives while bringing control over the wildfire [1]. They use technology, plan a strategy, prepare their equipment, and execute the planned operations [2]–[4]. To draft a better strategy, they must monitor the environment from different perspectives and analyze the impact. For monitoring, coverage of the environment is very important and becomes tricky as the environment behaves dynamically.

So far, the coverage is so limited, the firefighters installed a network of cameras sited on top of mountains and use them for monitoring at the time of an event [5]. This approach has a lot of limitations, a few significant reasons are due to the lack of mobility. The camera sites are fixed providing less coverage quality, and single-sided view and in worst cases, the camera equipment at the site would be damaged as the wildfire passes through it.

From the technological side, the firefighters had adapted to the advanced equipment for monitoring and analysis purposes. One such advancement is the use of unmanned aerial vehicles (UAV) [6] [7]. Though the applicability of UAVs has its own limitations, the mobility of the UAVs has overcome all the limitations of establishing a camera network.

Casbeer et al. [8] discussed using multiple small UAVs to monitor the wildfire. They demonstrated a path-planning algorithm in a wildfire-simulated environment to track the fire autonomously and transmit the information to the base station. Pham et al. [9] proposed a multi-agent reinforcement learning algorithm for a team of UAVs to optimize the coverage of an unknown field of interest. The team of UAVs cooperates with each other to avoid overlapping and cover maximum region. Seraj et at. [10] proposed a predictive framework to prioritize the dynamically propagating region and make decisions to distribute the resources based on their needs. This enables the coverage to be case-specific and then tracks based on a probabilistic performance guarantee. Phan et al. [11] developed a hierarchical platform, in which a centralized top-level controller collects feed from UAVs/UGVs in the field, merges, analyzes, plans the mission, and passes the decisions to the agents in the field.

Pham et al. [12], [13] designed a distributed control framework to monitor the wildfire and maximize the coverage by avoiding in-flight collisions. A potential field-based method was used to track the fire and maintain a safe distance from the ground and other UAVs. Haksar et at. [14] used deep reinforcement learning to generate a decentralized control strategy over a network of aerial robots. Shrestha et al. [15], [16] compare different methods of reinforcement learning over maximizing the coverage in a dynamic wildfire environment. The author compared Q learning with and without experience replay buffer, and deep Q-network with and without state estimator in terms of maximizing rewards and coverage.

Most of the approaches were focused on tracking and covering in a static and dynamic environment. Very few papers discussed the communication among the UAVs in a network. One such paper is by Siedler et al. [17], with emphasizing the importance of communication and collaboration among a team of autonomous drones to achieve the goal of reforestation. They enabled collaboration among agents using a communication mechanism based on graph neural networks.

In this paper, we discussed tracking and coverage of wildfires using multiple UAVs, primarily focusing on the need for communication. We proposed 4 methods, each varying in the way of communication among the UAVs. In all methods, a set of UAVs are deployed, and their goal is to track the wildfire and cover as much fire region as possible. In the first method, the UAVs do not communicate with each other, even their experiences. In the second method, the UAVs do communicate their experiences via centralized memory. In the third method, a UAV communicates its position along with its experience with the rest of the UAVs. Finally, in the fourth method, we form a sub-network of UAVs and search for the aid of fire or any neighbor UAV, which had already

Gaurav Srikar and Hung Manh La are with the Advanced Robotics and Automation Lab, Computer Science and Engineering, University of Nevada, Reno, Nevada, USA. Emails: gauravsrikar@nevada.unr.edu, hla@unr.edu

*Source code for all methods are available at https://github.com/aralab-unr/Wildfire-tracking-DDPG

discovered fire. We used ranking and consensus concept to follow an ideal UAV's location based on past and current track records. This communication helps other UAVs, which are still searching in the wild for fire to a real location of the fire, making the coverage faster and more efficient. We compare all 4 methods in terms of the received rewards, region coverage ratio, time to track the fire, time for complete coverage, the fallout of the environment boundaries, and fallout from the fire region.

The nature of wildfire is based on Huygen's principle of wave propagation [18]. There are other natural phenomena following Huygen's principle like an oil leak over the ocean, where the leak spreads dynamically with respect to wind and tides. Another great example is the pattern of garbage floating in the ocean, causing the great pacific garbage patch.

## II. CONTRIBUTION

### A. Continuous Environment

One of our goals is to build the entire environment in a continuous way. So far, the simulated data used to map the environment is discrete. The data is a discrete coordinate of fire region, each represented as a circle of radius 1 point over the map, and the region within the radius is considered as a fire point. The environment that we built using this data handles continuous observation and action spaces. To elaborate, the positions of the UAV and the action performed to move their positions are continuous. Meanwhile, rewards are collected by counting the number of discrete fire points under the field of view (FoV) of the UAV using a reward function and returned as a discrete score. The dimensions of the UAV's FoV depend upon the altitude of the UAV, which is continuous, but the information perceived inside FoV is discrete. The environment works as expected and can work even more efficiently as environment dimensions get larger. To handle the continuous space, we used Deep Deterministic Policy Gradient [19] (DDPG) as the agent, which is proven to perform best in a continuous environment.

### B. Variations in Communication

All 4 methods that we used to understand the importance of communication, vary by the information they share with each other. The information can be expressed in terms of experience, positions, fire alerts, or none. We pretend to show the variations from zero communication to a fully connected network with consensus-based communication. In case of fire alerts, all the UAVs inside the environment form a network and search for the fire. If any UAV discovers fire, then all other UAVs connected in the network receive an alert and collectively navigate toward the source.

### C. Target Navigation

In method 4: consensus-based communication, the last three input space of an agent is the target coordinates. By default, at the start of an episode, the target coordinates are set random, and the UAV agent learns to navigate towards the target coordinates. The target coordinates get reset once the UAV approaches close to the target. This enables the agent to train the model to navigate in the direction of the target. This target navigation practice helps the agent at the time of fire discovery to make a reward-rich decision. The results show that the use of random target coordinates helps the UAV agent to navigate accordingly.

### D. Ranking and Consensus

To illustrate the whole process, we form a dynamic network of UAVs to discover the presence of fire. Once the network discovers fire, the targets of all UAVs get a consensus based on a rank equation. And the target is passed as an input to the deep reinforcement learning algorithm. The algorithm predicts an action, which is performed over the continuous environment space. The UAVs in the new coordinates, scan their FoV for fire points, applies the reward function, and update the experience in the memory. Frequently, the experiences in the memory are sampled at random to train the algorithm to predict the actions better in the future.

The rest of the paper is organized as follows: Section III, the background required to understand the basics. In Section IV, an in-detailed description of the methodology includes the environment, the data, and all four methods. Section V shows the experimental results obtained from all 4 methods over 6 metrics. Finally, Section VI covers the conclusion and the possible future works that we planned, followed by acknowledgment and references.

## III. BACKGROUND

### A. Policy Gradient

To handle the continuous environment, the Q-learning will get under-fit easily because of its deterministic nature. To overcome this problem, a gradient-based policy was proposed where the weights for a certain state and action pair are adjusted automatically to maximize the reward outcome. This way the policy gradient [20] will learn stochastic policies.

### B. Actor-Critic Model

There are two entities, the actor, the policy function which predicts an action given a set of states, and the critic, which estimates the value function for the predicted action, in other terms the Q-value for the state and action pair. The actor-critic [21] belongs to temporal difference learning methods [22] meaning an estimated reward is generated for the new state and is compared to the actual reward gained. Based on the difference, the actor is trained to update the policy.

### C. Deep Deterministic Policy Gradient

The deterministic policy gradient is a special case of stochastic policy gradient, which is proved to be a simpler and more efficient version. In deep deterministic policy gradient (DDPG) [19], the deep learning concepts were adapted to enhance the performance. DDPG consists of two pairs of actor-critic entities, one pair updates its weights frequently, and the other pair copies the first pair's weights periodically to avoid instability. DDPG uses off-policy data and the bellman equation to train the critic network and uses

the Q-value produced by the critic to train the actor-network. DDPG performs better in a continuous action setting, so we used DDPG in all proposed methods.

## IV. Methodology

### A. Environment

The environment consists of 30x30 discrete space, renders no-fire data points as black, fire data points as white, UAVs as blue, and its field of view as yellow. As a 30x30 matrix representation, black points are assigned 0.0 and white points as 2.0 initially. Upon the UAVs hovering over the five points, the score reduces from 2.0 to 1.0 but the color remains white here, 2.0 represents undiscovered fire by UAV, and 1.0 represents already discovered fire by UAV.

The action space of the environment is 3 valued vectors, each representing the direction of a UAV's movement. Assume a 3-dimensional space with 6 axes (+X, -X, +Y, -Y, +Z, -Z) each represents a direction. If the values in the action vector are positive then the UAV's new position moves towards +X, +Y, +Z, and vice versa with the negative values. This rule applies to any variations of positive and negative values. For example, assume an agent at position [2.0, 1.0, 4.0], and received action [+1.267, -0.003, -1.042], then the new position will be towards [3.267, 0.007, 2.938]. We assume X in terms of breadth, Y in terms of length, and Z as the height of the environment.

Each episode length is 1000 and sets no conditions to terminate an episode.

Each agent maintains an individual 30x30 matrix map representing its position and the points it covers. The coverage of the points depends upon the altitude of the UAV. If the UAV is higher, it covers a larger area, and vice versa. The altitude is limited to 5 points.

To perform the reward function, the environment's map is overlapped with the agent's map. Finally, each point is classified and assigned a reward.

### B. Data

To generate the data based on Huygen's principle of wave propagation, we use the FARSITE simulator [23], [24]. Though the simulator provides data regarding wildfires, we can pre-process and use the data for general wave propagation. The data contains 22,796 iterations, each iteration consists of a series of x and y coordinates along with fire intensity over the wave. Over the iterations, the wave starts to spread from the source in all directions as shown in Fig. 1. After certain iterations, the discrete points on the wave become the source and perform individual and random wave propagation. We can also set a little randomness to propagate the wave in a particular direction. This biased randomness, in real-time can be interpreted as wind speed, which influences the direction of the oil or fire spread.

In pre-processing data stage, we discarded the fire intensity and assigned x and y data points as 2.0 over a 30x30 grid. Here, 2.0 represents undiscovered points. While training, once any UAV hovers over these data points, 2.0 will be updated to 1.0, marking this data point as visited/discovered.

### C. Method 1: Independent UAVs Without Any Communication

In this method, we assume all UAVs explore the environment without knowing the presence of other UAVs of any sort. Each agent (UAV) contains its own neural network (DDPG) and memory (Experience Replay Buffer). This method is to understand how fast and accurately all agents learn the environment independently. Each agent passes 6 inputs to the neural network:

i. - iii. $x$, $y$, $z$ – coordinates;
iv. New fire indicator: Represents True if this agent hovers over undiscovered fire points, else False;
v. Old fire indicator: Represents True if this agent hovers over already discovered fire points, else False;
vi. Timestep;

And expects 3 actions to perform movement over X, Y, Z-axis, and rewards are collected. After performing an action, each agent stores the old position, action, reward, new position, and terminal status in its memory. The reward function for this method is designed in a way to contain all agents within the boundaries of the map. Upon crossing the boundary, that agent receives -100 for that time step. The agent's field of view is retrieved, and each fire point is classified among new fire, old fire, and no fire classes. For all retrieved fire points, the classes are incremented and multiplied with +1, -4, and -10, based on the classes new fire, old fire, and no fire, respectively. All calculated rewards are summed up and returned to the agent. The method 1 approach is defined as Algorithm 1.

**Algorithm 1:** Independent UAVs Without Any Communication

**Input :** number of UAVs $n$, sample batch size $k$
$DDPG_n \leftarrow$ Initialize
$L_n \leftarrow \emptyset$
**for** *each episode* $e$ **do**
  **for** *each time step* $t$ **do**
    **for** *each agent* $i$ **do**
      Randomly initialize position for each agent $A_{i,[x,y,z]}$
      $S_{i,t}$ = [$A_{i,[x,y,z]}$, $newfire$=False, $oldfire$=False, $timestep\ t$]
      $a_t$ = Predict action based on $S_{i,t}$
      Obtain $S_{i,t+1}$, $R_t$
      Store experience into replay buffer $L_i$
      $L_i \leftarrow (S_{i,t}, a_t, R_t, S_{i,t+1})$
      Retrieve $k$ replays from $L_i$ for $DDPG_i.learn()$
    **end**
  **end**
**end**

### D. Method 2: Independent UAVs With Memory Sharing Only

In this method, all agents explore the environment and share their experience in a centralized experience replay

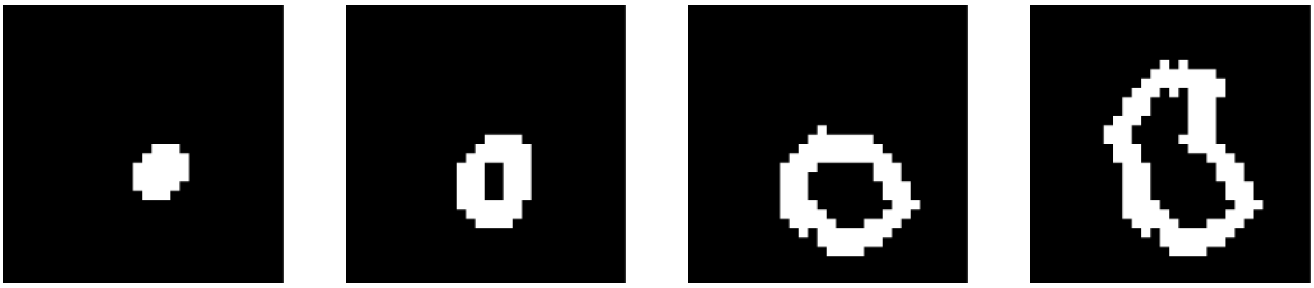

Fig. 1. The timelapse of the wildfire environment over episodes 100, 1000, 6000 & 7000.

buffer. We assume, by sharing the experience with other agents, the agents will learn the environment faster and navigate better. Each agent contains an individual neural network but a common memory.

While learning, the agent retrieves a batch sample from the memory, which contains the experience of all other agents at random and is shuffled further to break the correlations. This helps the agent to navigate the locations that it never visited before. All other operations performed by the agents, and reward functions are similar to method 1. The difference between methods 1 and 2 is the placement of Experience Replay Buffer. In algorithm 1, $L_n$ & $L_i$ is replaced with $L$, and the rest of the algorithm is same.

### *E. Method 3: Dependent Agents With Positional And Memory Sharing*

In this method, the agents contain their own neural network, and the memory is centralized. Each agent collects the positional coordinates of all other agents, clubs them, and passes them as input to the neural network. For example, if we have three agents, the number of inputs is 18 (3 sets of 6 parameters).

We assume that if an agent knows the whereabouts of other agents, in case any of the agents discover fire then other agents can navigate towards the discovered agent.

For any agent the parameters of all agents are appended and passed through its neural network to predict an action of 3 spaces. These 3 action spaces along with the agent's index are passed through the environment, where the agent's particular parameters are spliced based on the index and performed action over the spliced parameters. The updated spliced parameters are appended back with other parameters and are returned as new observations to the agent along with the rewards and termination status.

For the next agent to perform an action, it receives an updated observation of all agent's parameters every time. All other operations performed by the agents and reward functions are similar to method 1. The method 3 approach is defined as Algorithm 2.

**Algorithm 2:** Dependent Agents With Positional And Memory Sharing

**Input :** number of UAVs $n$, sample batch size $k$

$DDPG_n \leftarrow$ Initialize
$L \leftarrow \emptyset$
**for** *each episode* $e$ **do**
  $S \leftarrow \emptyset$
  **for** *each time step* $t$ **do**
    **for** *each agent* $i$ **do**
      Randomly initialize position for each agent $A_{i,[x,y,z]}$
      $S_t$ = [$S_t$, $A_{i,[x,y,z]}$, $newfire$=False, $oldfire$=False, $timestep$ $t$]
      $a_t$ = Predict action based on $S_t$
      Obtain $S_{t+1}$, $R_t$
      Store experience into replay buffer $L$
      $L \leftarrow (S_t, a_t, R_t, S_{t+1})$
      Retrieve $k$ replays from $L$ for $DDPG_i.learn()$
    **end**
  **end**
**end**

### *F. Method 4: Consensus-Based Agent Network*

In the previous method, the agents though communicate by sharing their locations, there is no proper communication network is established. Especially when an agent discovers fire and fails to pass an alert to other agents or even the agents within close distance.

In this method, we try to solve the problem by establishing a communication based on a consensus filter where a set of agents within a distance connect with each other, become neighbors, and form a network as shown in Fig.2. Any agent in the network who discovers a fire will alert all other agents in the network, and guide them to navigate toward the fire. We classify all agents among,

i. Discoverer: Agents who discovered the fire and still exploring to discover more. Meanwhile, passing the fire coordinates with its neighbor agents in a network.
ii. Seeker: Non-discoverer agents within a network, where one or more agents in the network turn out to be a discoverer.
iii. Wanderer: Agent/network of agents which is still searching for fire randomly without any discoverer's guidance.

The classification is performed based on the agent's new fire input status. If an agent's new fire is true then the agent is classified as the discoverer, else if any agent's new fire input in a network is true then the discovered agent is classified

**Algorithm 3:** Consensus-Based Agent Network

**Input :** number of UAVs $n$, sample batch size $k$

$DDPG_n \leftarrow$ Initialize
$L \leftarrow \emptyset$
**for** *each episode* $e$ **do**
  **for** *each time step* $t$ **do**
    Randomly initialize position for all agent $A_{n,[x,y,z]}$
    $S_{n,t}$ = [$A_{n,[x,y,z]}$, $new\_fire$=0.0, $old\_fire$=0.0, $timestep$=$t$, $x_{target}$=-1.0, $y_{target}$=-1.0, $z_{target}$=-1.0]
    $streak\_info_n \leftarrow$ [$avg\_alt$=0, $streak$=0, $fire\_scores = 0$]
    $sub\_net$ = {}
    **for** *each agent* $i$ **do**
      **if** $S_{i,t}$ *discovered new fire points* **then**
        Take the mean of all newly discovered fire coordinates in tgt_x, tgt_y
        Set $S_{i,t}[x_{target}, y_{target}]$ = $[tgt\_x, tgt\_y]$
        Update $streak\_info_i$
      **end**
      **else**
        $sub\_net_i$ = Neighbor indices of agent i
        Reset $streak\_info_i$
      **end**
    **end**
    $follow$ = {}
    **for** *each agent* $i$ **do**
      **for** *each neighbor in* $sub\_net_i$ **do**
        Rank each neighbor
        Assign Max ranked neighbor to $follow_i$
      **end**
    **end**
    **for** *each consensus iteration* $c$ **do**
      **for** *each agent* $j$ *in* $sub\_net$ **do**
        $follow_i$ = $MAX_{rank}$ ($follow_{Neighbor} \bigcup$follow$_i$)
      **end**
    **end**
    Update $S_{n,t}$ Targets according to $follow$ **for** *each agent* $i$ **do**
      $a_t$ = Predict action based on $S_{i,t}$
      Obtain $S_{i,t+1}$, $R_t$
      Store experience into replay buffer $L$
      $L \leftarrow (S_{i,t}, a_t, R_t, S_{i,t+1})$
      Retrieve $k$ replays from $L$ for $DDPG_i.learn()$
    **end**
  **end**
**end**

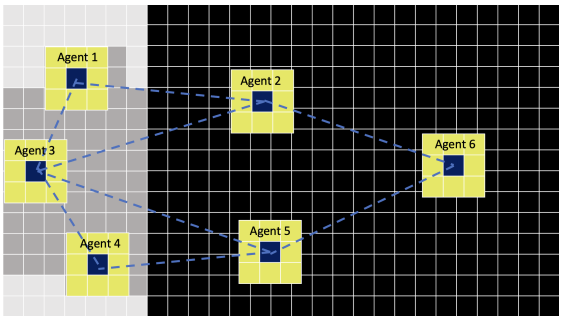


Fig. 2. White region is undiscovered fire points and grey region is already discovered fire points. Agents 1-6 form a network. Agents 1 and 4 are classified as discoverers, and the rest are classified as seekers.

as a discoverer, and the rest of the agents are classified as seekers. Or else none of the agent's fire inputs are true then is classified as Wanderers. The neighborhood of a network is based on the Euclidean distance between any 2 agents being less than D=10. We store the neighborhood data in a dictionary of lists.

The number of inputs to the actor-network is 9 and are as follows: x, y, z – coordinates, new & old fire status, timestep, x, y, z – target coordinates. By default, and for Wanderers x, y, z – target coordinates will be random coordinates within the environment's limits.

Konda et. al. [25] proposed a multi-robot system architecture for predator avoidance. A network of robots flocks around the environment to avoid a predator. Each robot in the front line detects the presence of the predator, consensus the direction among the network and as a predator approaches closer, all robots predict a consensus direction towards a safe region using function-approximated reinforcement learning and flock away.

A set of sub-networks whose agents are not discoverers are formed. The goal of the agents in the sub-network is either to search for fire points or check whether the neighborhood agent outside the sub-network exists as a discoverer. In case any agent discovers fire points, then the alert is passed among the sub-network and all agents within the network would set their target coordinates as the newly discovered agent's coordinates. In another case, if any agent within the sub-network identifies an external neighbor as a discoverer or randomly navigates towards any discoverer and connects as a neighbor, then all the agents within the sub-network receive the alert and update their target coordinates as the discoverer coordinates.

If a sub-network found the discoverers more than one, then the best discoverer neighbor has ranked among all discoverers and all agents within the sub-network will consensus to the best-ranked neighbor/discoverer and navigate towards it. The ranking equation is as follows:

$$Rank = \left(\frac{Fire\_Score}{Avg\_Alt. \times Streak}\right) \cdot \left(\frac{Current\ Fire\ Disc.}{Current\ Alt.}\right)$$

- Streak: Number of timesteps an agent continuously discovering new fire.
- Fire_Score: Number of fire points discovered over a streak.

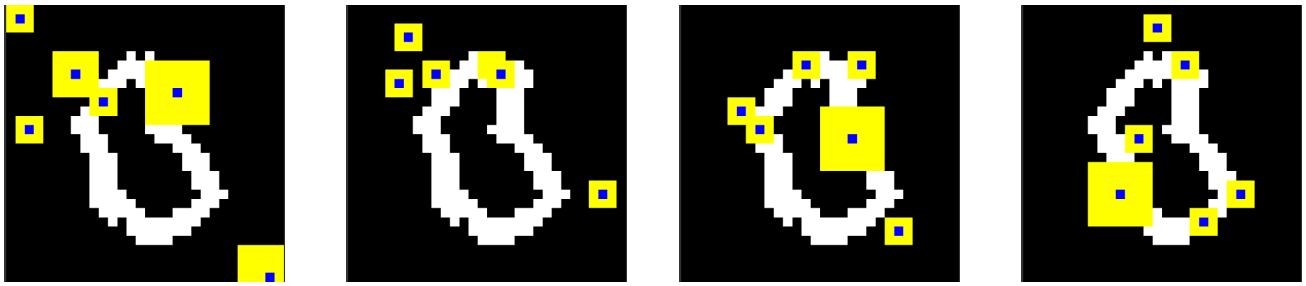

Fig. 3. Timelapse of multi-UAVs forming a sub-network (Top-Left corner), consensus the target coordinates, and navigating towards them in a single episode. Meanwhile, the discoverers (the agents over the fire region) try to discover as many fire points as possible.

- Avg._Alt.: Average altitude maintained by an agent over a streak.
- Current Fire Disc.: At the current timestep, the number of fire points discovered by an agent.
- Current Alt.: At the current timestep, the altitude of an agent.

By performing ranking and navigating towards the discoverer, we assumed that there might be abandoned fire points near the discoverer. The navigation example is shown in Fig.3. The ranking equation is described in 2 parts: the first part is the track record of a particular discoverer so far, and the second part is the abundance of fire points at the current time step.

Any agent after discovering new fire points becomes a discoverer. The goal of the discoverer is to maintain the streak of new fire discoveries. To navigate through new fire points, the agent must set a target and learn to predict the actions to achieve higher rewards as shown in Fig.4. To identify the target coordinates, the agent lists all new fire point coordinates in the current timestep and takes the mean of the list. These averaged coordinates are set as a target.

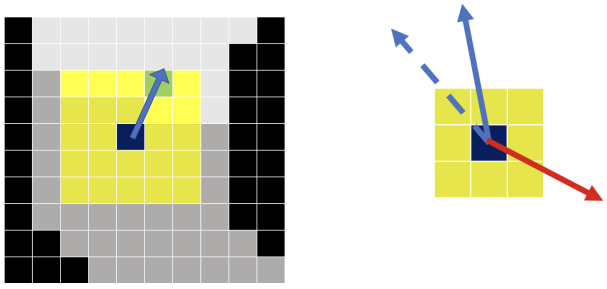

Fig. 4. The discoverer agent predicts the direction to maximize the future rewards. In the right image, the dashed line represents the target angle, agent moving towards.

Compared to other methods, this environment's reward function has an additional feature. The number of input parameters is 9 instead of 6, and the last 3 parameters are considered target coordinates. We calculate the angular difference between the target direction and the predicted direction and sum the inverse of the difference to the reward.

$$Reward = \left( \frac{180 - \|(\|Angle_{target}\| - \|Angle_{predict}\|)\|}{10} \right)$$

- $Angle_{target}$: The angle towards which the agent has to head.
- $Angle_{predict}$: The angle towards which an agent has headed.

We assume that if the difference is higher, the rewards received will be lower and vice versa, indeed helping the neural networks to learn this feature. The method 4 approach is defined as Algorithm 3.

## V. Experimental Results

This section represents the results obtained by simulation of all 4 methods. The results are the performance comparison of all 4 methods over reward collected, the area covered, the speed of coverage, the speed of tracking, the average number of UAVs falling out of fire region as well as the environment. To summarize the methods, method 1 represents independent UAVs without any communication, and the detailed procedure is described in algorithm 1. Method 2 represents independent UAVs with memory sharing only. The algorithm for method 2 was not presented in this paper, but by centralizing the memory in algorithm 1, thus algorithm for method 2 can be interpreted. Method 3 represents dependent agents with positional and memory sharing, and the detailed procedure is described in algorithm 2. Finally, method 4 represents a consensus-based agent network, and the detailed procedure is described in algorithm 3.

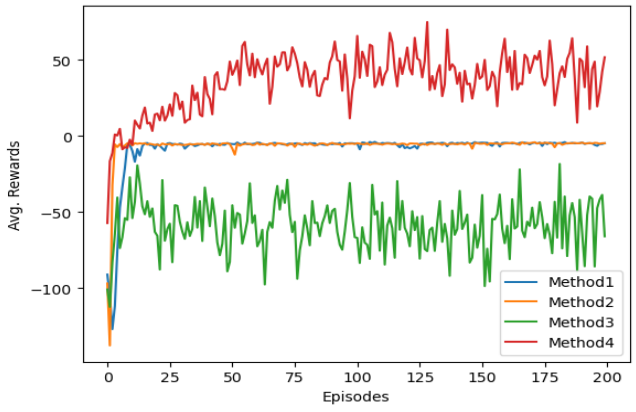


Fig. 5. Averaged rewards collected by all UAVs at each episode.

Our simulation was performed over 200 episodes, each episode lasting over 1000 timesteps. As there are no termination conditions, each episode will complete till the last timestep. We deployed 6 UAVs in a 30 x 30 x 5 grid

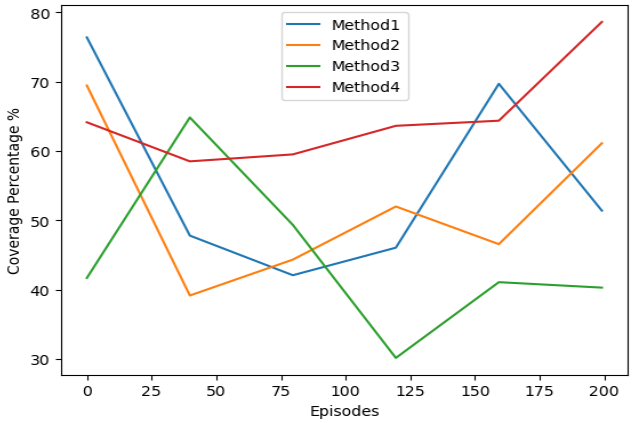


Fig. 6. The ratio of fire region covered by the UAVs over total wildfire out in the environment.

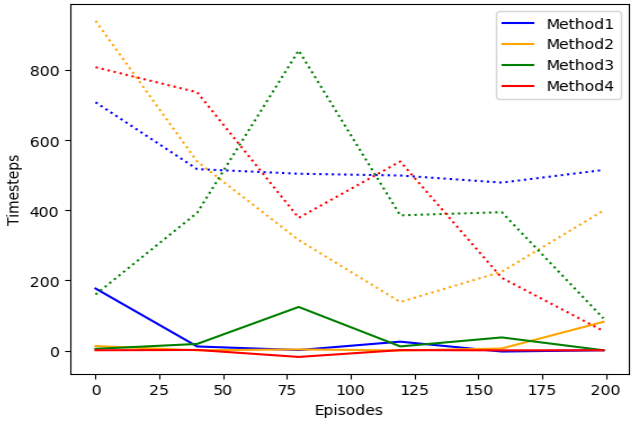


Fig. 7. The fastest and slowest time taken by the UAVs to track the fire at each episode. Fastest represented with a Solid line and Slowest represented with a Dotted line.

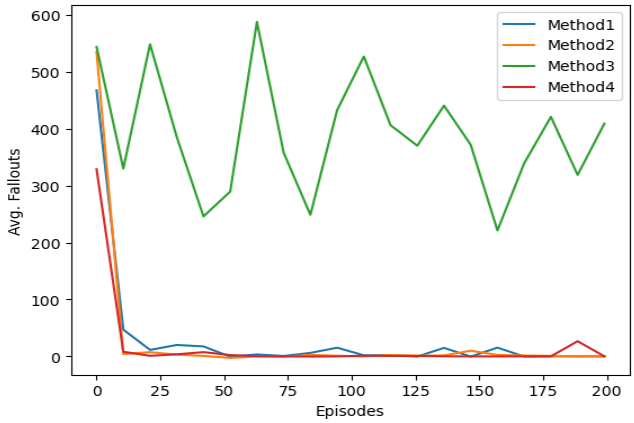


Fig. 8. The average number of times, all UAVs in each episode attempted to cross boundaries.

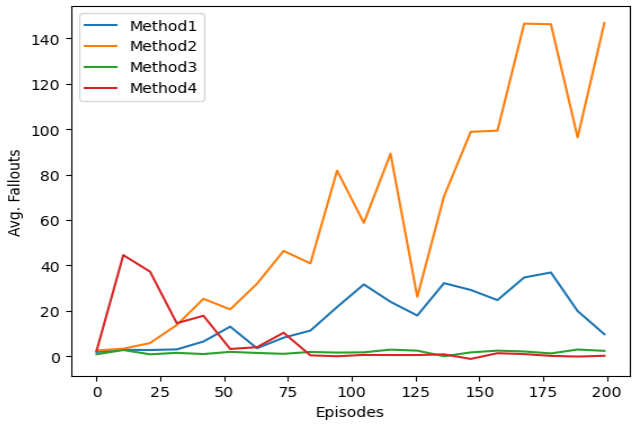


Fig. 9. The average number of times, all UAVs in each episode, once after discovering the fire, fell out of the fire region.

environment. The field of view angle for all UAVs is 30 degrees.

### A. Rewards Scored

Fig. 5 shows the rewards scored by all UAVs collectively for all methods. The collective score was averaged over an episode and is again averaged over the number of UAVs. The final average score is plotted against the episodes for all four methods. The steeper the slope of the curve, the faster the models will train. On the contrary the jagged lines, more time is required for the model to learn. By tuning reward function variables, we can optimize the jaggedness of the lines. Method 4 performs much better, in terms of steepest curve, and reward score than methods 1 and 2. But methods 1 and 2 were clearly steady as there was a very simple ruled environment to follow, learn and claim rewards.

### B. Coverage Ratio

Fig. 6 shows the total fire points covered by the whole UAVs within an episode by overall fire points existing in the environment. The tricky part is as the environment is dynamic, there might be last-minute fire points generated, which the UAVs might fail to learn. The plot represents the percentage of fire points covered against the episodes. The method performs the best and maintains consistency. This may be due to the navigation of sub-networks, where once they discovered fire and communicates with the rest of the sub-network. This shows one of the reasons that consensus and collective navigation would perform best in a dynamic environment. On a side note, the fire points which were missed by a discoverer can be discovered by its followers.

### C. Duration of Overall Coverage

We also tried to plot the time taken to cover all the fire points, but no method covered the environment fully. This metric can be useful in the long run with a larger number of episodes and timesteps.

### D. Duration of Overall Tracking

Fig. 7 shows all 4 methods whose models have been trained to search for fire and navigate to them as fast as possible. The plot is illustrated between timesteps and episodes. For each method, each episode, maximum and minimum timesteps taken by the UAVs are recorded. In the plot, a solid line represented the US minimum and dotted as maximum timesteps were taken at each episode. All methods performed well in the case of minimum, but in the case of maximum, the only method for steady learning was to reduce

the search time. This emphasizes as episodes run longer, method 4's model learns faster ways to build a network and navigate toward the fire.

### *E. Environment Fallout*

Fig. 8 shows the frequency of the UAVs attempting to escape the environment boundaries (30 x 30 x 5). This metric helps us understand how well the models are trained to stay within the environment. The mean count of escapes performed by a UAV within a mean of time steps for an episode is plotted. All methods except the 3rd perform very well in training the model to predict actions within the environment boundaries. This was made possible by imposing a high penalty for the crossing.

### *F. Fire Fallout*

Fig. 9 shows how frequently the UAVs after discovering and hovering over the fire, navigate away from the fire region. This metric is mostly similar to the previous metric. In this metric, method 4 performed well but there are some inconsistencies. Assume that a UAV discovered fire, hovers over it for a while, falls out of the fire region, and navigates the subway, the recorded count will be 1. As for this metric, the least the count is, the better the model is. To solve this error, the plot data needs to be filtered and pre-processed.

## VI. Conclusion And Future Works

In this paper, we discussed 4 different variations in terms of communication within the UAVs to emphasize the need for communication in a continuous and dynamic environment. **On an overall scale, though method 4 took some additional episodes to train when compared to other methods, it achieved 1.3x more fire coverage, 1.4x faster tracking, and 2x lesser environment cross-boundary cases than the next best method.** Clearly, method 4 is using consensus communication to outperform any other method. In the continuous world, this is the first step in building a continuous environment over discrete data. In the future, we plan to include a 2D heat equation, a concept from partial differential equations to fully convert the environment to continuous. Further, we also plan to use the convolutional Q-learning technique to visualize the input in real-time.

## VII. Acknowledgement

The authors would thank Revanth Konda for his suggestion for reward function and consensus communication. His support and guidance are appreciated.

## References

[1] K. Navarro, "Working in smoke:: Wildfire impacts on the health of firefighters and outdoor workers and mitigation strategies," *Clinics in chest medicine*, vol. 41, no. 4, pp. 763–769, 2020.

[2] M. Castellnou, N. Prat-Guitart, E. Arilla, A. Larrañaga, E. Nebot, X. Castellarnau, J. Vendrell, J. Pallàs, J. Herrera, M. Monturiol, *et al.*, "Empowering strategic decision-making for wildfire management: Avoiding the fear trap and creating a resilient landscape," *Fire Ecology*, vol. 15, pp. 1–17, 2019.

[3] M. A. Reams, T. K. Haines, C. R. Renner, M. W. Wascom, and H. Kingre, "Goals, obstacles and effective strategies of wildfire mitigation programs in the wildland–urban interface," *Forest policy and economics*, vol. 7, no. 5, pp. 818–826, 2005.

[4] M. A. Moritz, E. Batllori, R. A. Bradstock, A. M. Gill, J. Handmer, P. F. Hessburg, J. Leonard, S. McCaffrey, D. C. Odion, T. Schoennagel, *et al.*, "Learning to coexist with wildfire," *Nature*, vol. 515, no. 7525, pp. 58–66, 2014.

[5] "Alert wildfire - fire cameras," https://www.alertwildfire.org/, last accessed: February 28, 2023.

[6] J. Jeyavel, A. A. Prasad, K. M. Shelke, P. D. Sargade, and U. V. Thoke, "Survey on fire fighting techniques using unmanned aerial vehicles," in *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. IEEE, 2021, pp. 239–241.

[7] "How firefighters are effectively utilizing uavs to combat wildfires," https://www.forbes.com/sites/forbestechcouncil/2020/09/10/how-firefighters-are-effectively-utilizing-uavs-to-combat-wildfires/?sh=bc6065fb4f42, last accessed: February 28, 2023.

[8] D. W. Casbeer, R. W. Beard, T. W. McLain, S.-M. Li, and R. K. Mehra, "Forest fire monitoring with multiple small uavs," in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 3530–3535.

[9] H. X. Pham, H. M. La, D. Feil-Seifer, and A. Nefian, "Cooperative and distributed reinforcement learning of drones for field coverage," *arXiv preprint arXiv:1803.07250*, 2018.

[10] E. Seraj, A. Silva, and M. Gombolay, "Multi-uav planning for cooperative wildfire coverage and tracking with quality-of-service guarantees," *Autonomous Agents and Multi-Agent Systems*, vol. 36, no. 2, p. 39, 2022.

[11] C. Phan and H. H. Liu, "A cooperative uav/ugv platform for wildfire detection and fighting," in *2008 Asia Simulation Conference-7th International Conference on System Simulation and Scientific Computing*. IEEE, 2008, pp. 494–498.

[12] H. X. Pham, H. M. La, D. Feil-Seifer, and M. Deans, "A distributed control framework for a team of unmanned aerial vehicles for dynamic wildfire tracking," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 6648–6653.

[13] H. X. Pham, H. M. La, D. Feil-Seifer, and M. C. Deans, "A distributed control framework of multiple unmanned aerial vehicles for dynamic wildfire tracking," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 4, pp. 1537–1548, 2018.

[14] R. N. Haksar and M. Schwager, "Distributed deep reinforcement learning for fighting forest fires with a network of aerial robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1067–1074.

[15] K. Shrestha, H. M. La, and H.-J. Yoon, "A distributed deep learning approach for a team of unmanned aerial vehicles for wildfire tracking and coverage," in *2022 Sixth IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2022, pp. 312–319.

[16] K. Shrestha, R. Dubey, A. Singandhupe, S. Louis, and H. La, "Multi objective uav network deployment for dynamic fire coverage," in *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2021, pp. 1280–1287.

[17] P. D. Siedler, "Dynamic collaborative multi-agent reinforcement learning communication for autonomous drone reforestation," *arXiv preprint arXiv:2211.15414*, 2022.

[18] G. D. Papadopoulos and F.-N. Pavlidou, "A comparative review on wildfire simulators," *IEEE systems Journal*, vol. 5, no. 2, pp. 233–243, 2011.

[19] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015. [Online]. Available: https://arxiv.org/abs/1509.02971

[20] S. M. Kakade, "A natural policy gradient," *Advances in neural information processing systems*, vol. 14, 2001.

[21] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

[22] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[23] M. A. Finney, *FARSITE, Fire Area Simulator–model development and evaluation*. US Department of Agriculture, Forest Service, Rocky Mountain Research Station, 1998, no. 4.

[24] T. M. Williams, B. J. Williams, and B. Song, "Modeling a historic forest fire using gis and farsite." *Mathematical & Computational Forestry & Natural Resource Sciences*, vol. 6, no. 2, 2014.

[25] R. Konda, H. M. La, and J. Zhang, "Decentralized function approximated q-learning in multi-robot systems for predator avoidance," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6342–6349, 2020.