# Extended Abstract

**Motivation** Climate change and poor land management practices are contributing to increasingly severe wildfires that are threatening lives, infrastructure, and ecological balances. Unmanned aerial vehicles (UAVs) offer a safe and scalable method for real-time wildfire suppression. UAVs are well-suited for reinforcement learning (RL) because individual agents take actions with a quantifiable objective in a changing environment. However, RL's application to UAV fire suppression remains underexplored. Instead, most previous work has focused on human-based wildfire mitigation.

**Method** In our environment, UAVs suppress wildfire spread in a grid-based forest. Our adapted environment is a 2D grid with trees represented as cells in one of four states: healthy (green), on fire (red), burnt (brown), and extinguished (white). Fire propagates from a burning tree to a healthy tree with probably $P_{ignite}$ and becomes burnt with probability $P_{burnt}$, which is higher when an UAV is above the tree. A UAV agent can choose from five actions: move left, right, up, down, or stay still.

We want fire suppression to scale with the size of the fire, so we used single-agent training across multiple agents. This allows new UAVs to be added easily, with each following its own policy. We trained policies from four algorithms: PPO, DQN, A2C, and RPPO. We tested eight reward functions that were different combinations of four objectives: minimize total fire area, minimize distance to fire edge, maximize fire extinguishing, and maximize agent movement. We tested our RL policies against a policy where agents take random actions and a heuristic where agents move to the nearest fire edge.

**Implementation** We built upon the wildfire-environment v0.1.9 (2024) simulation in Python and trained our models using stable-baselines3 developed by Raffin et al. (2019). We designed custom reward functions, custom grids, and adapted the environment to fit our application. We first tested our seven reward functions on PPO and DQN on a small grid with fixed fire initialization. We then selected the two highest performing reward functions and trained policies with these functions and A2C and RPPO. We selected the two best algorithms and trained the resulting four models on a small random-initialization grid, large fixed-initialization grid, and large random-initialization grid.

**Results** PPO with reward functions Minimize Fires and Dense Combo consistently outperformed the other RL models across all experiments. However, the heuristic still outperformed this model. The random policy behaved poorly. Most policies performed well on random initializations of small size but struggled on larger initializations. PPO outperformed A2C in all metrics for each of the reward functions. PPO + Dense Combo was consistently the highest performer and demonstrated potential for generalization by excelling on the large randomly initialized grid.

**Discussion** The Minimize Fires reward function performed similarly on fixed vs random initializations on the small grid, but Dense Combo generalized better in the larger random environment for both PPO and A2C. This may be because Dense Combo incorporates multiple objectives such as distance to fire edge, which forces agents to move toward fire, creating a denser reward environment and more feedback relative to Minimize Fires. Policies likely struggle in larger environments due to sparse rewards and limited training examples where agents reach the fire. Alternatively, performance may suffer from too few training steps to explore the expanded state space. Furthermore, pairs of agents exhibited non-identical behavior on the same grid, indicating that training was incorporating relational/geographic information.

**Conclusion** Our results suggest that RL is a promising approach to UAV wildfire suppression, particularly when using multi-objective reward functions. By combining objectives to minimize fire spread, reduce distance to fire, and encourage movement, we transformed a sparse environment into a denser one, which enabled agents to learn more effective and generalizable policies. Our best model, PPO with a dense reward combination, performed well across varied settings. However, it was still outperformed by a simple heuristic, showing the importance of comparing learned policies against strong, interpretable baselines in structured domains.

Future work should train more agents and on a more complex and realistic environment that better captures wildfire dynamics. Future work can also explore imitation learning from human-controlled drones. Additional training and hyperparameter tuning may also improve performance in this complex setting.

# Reinforcement Learning for Practical Wildfire Suppression with UAV Agents

**Amy Guan**
Department of Computer Science
Stanford University
`amyguan@stanford.edu`

**Emily Tianshi**
Department of Computer Science
Stanford University
`etianshi@stanford.edu`

**Ryne Reger**
Department of Computer Science
Stanford University
`rreger@stanford.edu`

## Abstract

Unmanned aerial vehicles (UAVs) offer a safe and scalable method for real-time wildfire suppression. Although previous work has focused on human-based wildfire mitigation, UAVs are well-suited for reinforcement learning (RL) because individual agents take actions with a quantifiable objective in a changing environment. We build upon the wildfire-environment simulation where UAVs suppress wildfire spread in a grid-based forest. We train multiple single-agents across various reward functions and RL algorithm combinations on different grid initializations and tested performance against random and heuristic baselines. PPO with a multi-objective reward function consistently outperformed the other RL models across all experiments. We showed that reward engineering enables the RL agents to learn intuitive strategies, though it is still outperformed by the heuristic, which minimizes the distance to the fire edge, especially on random and larger environments. Our results show that RL is a promising approach to UAV training for wildfire suppression, and this application should continue to be explored.

## 1 Introduction

Climate change and poor land-management practices are contributing to increasingly severe wildfires that are threatening lives, infrastructure, and ecological balances. Unmanned aerial vehicles (UAVs) offer a safe and scalable method for real-time wildfire suppression. The method is well-suited for dangerous terrain or nighttime settings that would otherwise be extremely dangerous for firefighers. UAVs can gather more information about the fire spread that can aid firefighting response. They can also better target where to extinguish fire, which could conserve limited water and retardant resources. Even more, they can be deployed faster than people and help ensure small fires do not become more devastating.

UAVs are well-suited for reinforcement learning (RL) because individual agents take actions with clear objectives in a changing environment. However, RL's application to UAV fire suppression remains underexplored as previous work has focused on human-based wildfire mitigation.

We will consider each UAV to be an individual agent with its own policy. This allows for real-time scaling of adding more UAVs to combat increasingly large wildfires. We want to determine whether RL can be applied to this setting. Specifically, we want to understand what reward objectives incentivize learning, along with what RL algorithms are well suited for this task and why.

## 2 Related Work

Most previous work of applying reinforcement learning to wildfires has dealt with human-based wildfire mitigation. Tapley et al. (2023) created the SimFire and SimHarness simulation environments that enable RL training for human-based wildfire suppression where the action space includes mitigation tactics such as placing a fireline. They model agents with DQN but seek better performance. Similarly, Murray et al. (2024) demonstrate the potential of deep reinforcement learning with models from DQN, DDQN, and DDDQN, but this is again focused on human-based tactics such as creating a firebreak. Altamimi et al. (2022) present the potential of actor-critic algorithms and experiment replay in preemptive mitigation rather than real-time firefighting. All three papers demonstrate the potential of reinforcement learning in combatting wildfires, yet all three are focused on human-based firefighting or preemptive mitigation. We want to train UAVs to provide a safer alternative to firefighting. Although we will incorporate the learning from this literature, specifically the potential of actor-critic methods and deep RL along with MDP fire dynamics, we are trying to tackle a problem with a completely different action space.

Fortunately, there is a nascent but growing literature of training UAV agents for wildfire suppression, to which we hope to contribute. Shami Tanha et al. (2023) uses the multi-agent reinforcement learning (MARL) algorithm MADDPG, which is an actor-critic method where the input for each agent includes the total observation and action space across all agents. Their reward function incorporates incentives to be on top of fire, to be near fire, and to be far from other agents. They demonstrate high performance. Haksar and Schwager (2018) use the MARL algorithm MADQN and a reward function that incentivizes moving toward the fire boundary, controlling the boundary, staying away from other agents, and moving clockwise. They also demonstrate high performance.

This previous literature works well for the specific setting in which a fleet of UAVs can communicate perfectly with each other. However, wildfire environments and spread are unpredictable and thrive in hazardous conditions, so we cannot assume that agents can communicate with each other. Furthermore, wildfires can spread so dramatically that we may need to increase the number of UAV firefighters in real time. As a result, multi-agent reinforcement learning may not be well suited for this application. Instead, we hope to train individual policies that do not require information from others and allow UAVs to operate independently of each other. We will experiment with DQN and actor-critic methods based on this previous literature and tailor our environment to model similar MDP dynamics. We will also model reward functions with similar objectives to theirs: moving toward the fire, controlling the fire boundary, and staying above the fire.

## 3 Method

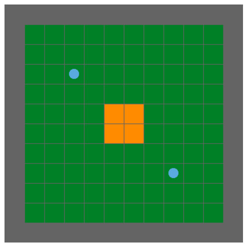
We build upon the wildfire-environment v0.1.9 (2024) simulation in which UAVs suppress wildfire spread in a grid-based forest. After our adaptations, the environment is a 2D-grid with trees as cells in one of four states: healthy (green), on fire (red), burnt (brown), extinguished (white). UAV agents move on the grid and drop fire retardant. Each agent can choose from one of five actions: stay still, move left, move right, move up, and move down. Due to our setup with $P_{\text{burnt}}$, each agent effectively always drops fire retardant when above a fire.

We model this problem as a Markov Decision Process (MDP) defined by the tuple $(S, A, P, R, \gamma)$:

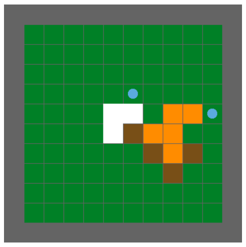
- **State space** $S$: A grid of size $12 \times 12$ for the small size or $17 \times 17$ for the large size, where each cell is labeled with a tree state and UAV positions.
- **Action space** $A$: Each UAV selects from the five actions: $\{\text{stay}, \text{up}, \text{down}, \text{left}, \text{right}\}$.
- **Transition function** $P(s' \mid s, a)$: Fire spreads from burning trees to adjacent healthy trees with probability
$$P_{\text{ignite}} = 1 - (1 - \alpha)^n,$$
where $n$ is the number of burning neighbors and $\alpha \in [0, 1]$ controls the spread rate. Burning trees become burnt with probability
$$P_{\text{burnt}} = 1 - \beta + \delta_\beta \cdot \mathbb{1}_{\text{agent\_above}},$$
where $\beta$ is the burn persistence and $\delta_\beta$ increases the extinguishing chance if a UAV is above. When a UAV moves above a burning tree, it increases the chance that the tree transitions to

a burnt state, effectively extinguishing it. Once a tree is burnt, it cannot ignite nearby trees. Thus, agent actions affect future states by controlling fire spread and limiting propagation through that cell.

Figure 1 shows examples of an initial state and a later state when fire has spread.



(a) Initial state of the environment. There are two agents at (3, 3) and (8, 8), and the fire starts with size $2 \times 2$ at (5, 5), (5, 6), (6, 5), and (6, 6).



(b) Later state of the environment. Agents extinguished the fire at (5, 5), (5, 6), and (6, 5) but fire spread from (6, 6) and has not yet been contained.

Figure 1: Example states of the environment.

We train and evaluate four different reinforcement algorithms: PPO, DQN, A2C, and RPPO.

The Proximal Policy Optimization (PPO) algorithm developed by Schulman et al. (2017) uses the advantage estimates to update the parameters of a policy's action distribution. However, the key aspect of PPO is that it clips the probability ratios between policy updates to ensure that parameters do not change too much. Specifically, the loss function is

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[ \min \left( r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ and $\hat{A}_t$ is the advantage estimate. The Recurrent Proximal Policy Optimization (RPPO) algorithm is an extension of PPO that includes a recurrent neural network (RNN). Sequences of observations are fed into the RNN, which allows the RNN to incorporate memory into training.

The Deep Q-Network (DQN) algorithm developed by Mnih et al. (2015) uses a neural network to estimate the Q-values for each state-action pair. It also uses experience replay, which improves sample efficiency, and updates parameters for the Q-network with the following loss function:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}} \left[ \left( r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \right)^2 \right]$$

Advantage Actor-Critic (A2C) algorithm developed by Mnih et al. (2016) has an actor that updates the parameters for the policy's action distribution using the gradient $\nabla_\theta \log \pi(a_t \mid s_t; \theta) \cdot A(s_t, a_t; \theta, \theta_v)$. A critic then estimates the value function and provides the advantage estimates $\sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k}; \theta_v) - V(s_t; \theta_v)$ that the actor uses to improve its actions by placing a greater weight on actions that perform better than expected.

## 4 Experimental Setup

We want fire suppression to scale with the size of the fire, so we used single-agent training across multiple agents. This allows new UAVs to be added easily, with each following its own policy. In other words, if a fire is spreading and we want to add more UAVs to suppress the fire, we can choose the single-agent policy that best matches the location of the UAV relative to the fire.

We evaluate performance based on the number of burnt trees and the number of trees on fire at the end of an episode, which is when there are no more trees on fire or after 100 timesteps. The total fire spread is the sum of the number of burnt trees and the number of trees on fire. Disaggregating the two allows us to determine whether the fire was fully suppressed, and if so, how much did it spread. When evaluating, we average performance across ten different episodes.

We first test seven different reward functions with PPO and DQN on a $12 \times 12$ grid with two agents at (3, 3) and (8, 8), and a fire initialized with size $2 \times 2$ at (5, 5), (5, 6), (6, 5), and (6, 6). We select $\alpha = 0.2$, $\beta = 0.99$, and $\delta_\beta = 0.9$ for all experiments. We select the top two reward functions and train these with PPO, DQN, A2C, and RPPO on the same small grid with fixed fire initialization. We select the top two performing algorithms and train these with each of the top two reward functions on the small grid with random fire initialization, large grid with fixed fire initialization, and large grid with random fire initialization.

We compare our models to two baseline policies. One is a random policy where agents randomly select across the five actions. The other is a heuristic where the agent selects the movement that brings it closer to the fire edge, i.e., the closest tree on fire that is adjacent to a healthy tree. The idea behind the heuristic is that trees on the fire edge are capable of spreading the fire and thus must be extinguished to mitigate fire propagation.

In total, we conduct $7 * 2 + (4 - 2) * 2 + 2 * 2 * 3 + 2 * 4 = 38$ different experiments.

## 4.1 Reward Engineering

Although we evaluate performance based on fire spread, it is possible that other reward objectives allow the agents to learn policies that enable them to better minimize wildfire spread. We first conduct reward engineering where we test different reward functions to determine what objectives drive agents to learn. We select the top two reward functions based on final evaluations trained on PPO and DQN on the fixed-initialization small grid with two agents. We have three base reward functions and four combo reward functions for a total of seven reward functions.

Our three base reward functions target different objectives that we believe incentivize agents to suppress wildfire.

1. **Minimize Fires:** The reward is $-1\times$ the number of trees on fire in the current state. This is our ultimate objective and evaluation metric because we want to suppress wildfires.
2. **Minimize Distance:** The reward is $-1\times$ the L1 distance to the closest tree that is on fire and is adjacent to a healthy tree. In other words, it is the distance to the fire edge. With our fire spread dynamics, we believe targeting the trees that could spread the fire—i.e., the fire edge—is a useful strategy.
3. **Maximize Extinguishing:** The reward is $+1$ for extinguishing a tree. This incentivizes agents to actually try to extinguish the fire, which is of course necessary to suppress the fire.

It is possible, and we believe likely, that the optimal reward function combines multiple objectives, so we create four combo reward functions.

1. **Proactive Firefighter:** reward $= 0.0R_{minfire} + 0.6R_{distance} + 1.0R_{extinguish} - 0.1R_{still}$.
2. **Efficient Suppressor:** reward $= 0.8R_{minfire} + 0.0R_{distance} + 1.2R_{extinguish} - 0.0R_{still}$.
3. **Dense Combo:** reward $= 0.3R_{minfire} + 0.5R_{distance} + 1.0R_{extinguish} - 0.2R_{still}$.
4. **Anti Idle:** reward $= 1.0R_{minfire} + 0.0R_{distance} + 0.0R_{extinguish} - 0.3R_{still}$.
   The stillness penalty provides built-in exploration and, in real-world application, ensures agents do not fixate on any one area.

## 4.2 Algorithm Selection

After selecting the top two reward functions, we then test four different reinforcement learning algorithms from stable-baselines3 developed by Raffin et al. (2019) on the fixed-initialization small grid with two agents.

1. **PPO:** this algorithm provides stable learning in stochastic environments like fire propagation due to clipped policy updates.
2. **DQN:** this algorithm is sample-efficient for discrete action spaces like this where agents select one of five action movements because it can reuse old trajectories.
3. **A2C:** this algorithm suits our dense reward environment because it can determine the advantage of a specific action.

4. **RPPO:** this algorithm incorporates memory which is useful for delayed feedback such as ours where the importance of burning a specific tree may not be known for some time as we wait to determine fire spread.

### 4.3 Generalization

We select the top two performing algorithms and top two performing reward functions on the fixed-initialization small grid with two agents. We then test these four reward+algorithm combinations on three grid setups to analyze generalizability of our models. The small grid with random fire initialiation, large grid with fixed-fire initialization, and large grid with random fire initialization. These three different grids enable us to understand if the reward function + algorithm models are robust to different environments, whether the robustness is focused to only grid size or fire starting points, or if the policies can only perform well in the small grid, fixed fire initialization setup.

1. **Small grid with random fire initialization:** a $12 \times 12$ grid with two agents initialized at (3, 3) and (8, 8) and a $2 \times 2$ fire with a random starting location.
2. **Large grid with fixed fire initialization:** a $17 \times 17$ grid with two agents initialized at (2, 2) and (14, 14) and a $2 \times 2$ fire initialized at (8, 8), (8, 9), (9, 8), and (9, 9).
3. **Large grid with random fire initialization:** a $17 \times 17$ grid with two agents initialized at (2, 2) and (14, 14) and a $2 \times 2$ fire with a random starting location.

## 5 Results

Through our quantitative and qualitative analyses, we found that PPO with Minimize Fires and Dense Combo consistently outperformed all other models on fixed initializations and outperformed the random baseline in all experiments. However, Dense Combo seems to generalize better to large and random initializations. The heuristic seems to perform optimally.

### 5.1 Quantitative Evaluation

We first compared our three base, single-objective reward functions across our random policy, heuristic policy, PPO, and DQN on the small grid with fixed fire initialization for two agents. Table 1 shows the results. It is clear that Minimize Fires and Distance to Fire can both incentivize great performance, as demonstrated by PPO fully extinguishing the fire very quickly with Minimize Fires and Distance to Fire almost always enabling the agents to extinguish the fire quickly with a mean number of trees on fire at the end of the episode of 0.60. Somewhat surprisingly, Extinguish Fires did not perform very well and performed similarly to the random policy.

| | Minimize Fires | | Distance to Fire | | Extinguish Fires | |
|---|---|---|---|---|---|---|
| Algorithm | Burnt | On Fire | Burnt | On Fire | Burnt | On Fire |
| RANDOM | 15.20 | 13.00 | 15.20 | 8.80 | 14.30 | 7.90 |
| HEURISTIC | 4.70 | 0.00 | 4.70 | 0.00 | 4.70 | 0.00 |
| DQN | 13.80 | 21.90 | 9.50 | 7.00 | 11.90 | 18.40 |
| PPO | 5.00 | 0.00 | 7.10 | 0.60 | 12.00 | 7.30 |

Table 1: Performance comparison of baselines, PPO, and DQN across single-objective base reward functions: Minimize Fires, Distance to Fire, and Extinguish Fires (2 agents, 12x12 grid, fixed initialization). Metrics are mean number of burnt trees and trees on fire at episode end across 10 episodes.

We then compared our four multi-objective reward functions across our random policy, heuristic policy, PPO, and DQN on the small grid with fixed fire initialization for two agents. Table 2 shows the results. It is clear that combining multiple objectives can allow for significant learning as Proactive Firefighter, Anti Idle, and Dense Combo all perform very well and successfully suppress the fire completely with PPO, and do so very quickly as there are 4.80, 4.70, 4.80 total burnt trees on average.

Our qualitative analysis suggests that Dense Combo provides the "smartest" policy, which is why we choose it.

| | Proactive Firefighter | | Efficient Suppressor | | Dense Combo | | Anti Idle | |
|---|---|---|---|---|---|---|---|---|
| Algorithm | Burnt | On Fire | Burnt | On Fire | Burnt | On Fire | Burnt | On Fire |
| RANDOM | 18.00 | 11.20 | 15.80 | 13.80 | 18.70 | 12.60 | 14.20 | 11.90 |
| HEURISTIC | 4.70 | 0.00 | 4.70 | 0.00 | 4.70 | 0.00 | 4.70 | 0.00 |
| DQN | 10.40 | 1.90 | 14.80 | 20.10 | 8.90 | 6.10 | 13.20 | 21.30 |
| PPO | 4.80 | 0.00 | 11.80 | 6.20 | 4.80 | 0.00 | 4.70 | 0.00 |

Table 2: Performance comparison of baselines, PPO, and DQN across multi-objective reward functions: Proactive Firefighter, Efficient Suppressor, Dense Combo, and Anti Idle (2 agents, 12x12 grid, fixed initialization). Metrics are mean number of burnt trees and trees on fire at episode end across 10 episodes.

Note that the random policy's performance is independent of the reward function, so the differences in evaluation metrics demonstrate the high stochasticity of both the policy and the environment. The heuristic's performance is also independent of the reward function, so its consistent metrics illustrate how it behaves the same way on the fixed initialization setup.

We select Minimize Fires because of its high performance with PPO and how it best represents our ultimate objective and Dense Combo because it again showed great quantitative performance and smart actions based on our qualitative analysis.

After selecting our two reward functions, Minimize Fires and Dense Combo, we experiment with A2C and RPPO. Table 3 shows the results. We see that the heuristic continues to outperform the other models, and that PPO and A2C significantly outperform DQN and RPPO across both reward functions. PPO always suppresses the wildfire completely and relatively quickly with the number of trees on average at the end being 0.0. A2C often suppresses the wildfire completely and has a mean number of trees on fire at the end of 0.50 with the Dense Combo reward function.

| | Minimize Fires | | Dense Combo | |
|---|---|---|---|---|
| Algorithm | Burnt | On Fire | Burnt | On Fire |
| RANDOM | 15.20 | 13.00 | 18.70 | 12.60 |
| HEURISTIC | 4.70 | 0.00 | 4.70 | 0.00 |
| PPO | 5.00 | 0.00 | 4.80 | 0.00 |
| DQN | 13.80 | 21.90 | 8.90 | 6.10 |
| A2C | 8.60 | 3.00 | 7.10 | 0.50 |
| RPPO | 13.90 | 24.00 | 11.60 | 9.80 |

Table 3: Performance comparison of top two reward functions across different algorithms: PPO, DQN, A2C, and RPPO (2 agents, 12x12 grid, fixed initialization). Metrics are mean number of burnt trees and trees on fire at episode end across 10 episodes.

After selecting our two algorithms (PPO and A2C) and two reward functions (Minimize Fires and Dense Combo), we tested the generalizability of our models. We experimented on our small grid with random fire initialization, on our large grid with fixed fire initialization, and on our large grid with random fire initialization. Table 4 shows the results. All the RL models significantly outperform the random policy, but the heuristic outperforms all the RL models. PPO outperforms A2C across all experiments. Performance actually improves for PPO and A2C when moving from small grid fixed initialization to small grid random initializaiton.

The Dense Combo reward function proves to be more generalizable when moving towards a random initialization on the large grid. Performance actually improves when moving from fixed to random on the large grid with Dense Combo as shown by how the number of trees on fire decreased from 4.60 to 1.50 for PPO and 13.30 to 8.70 for A2C. Our best performing model on the large grid was the PPO +

Dense Combo model with random initialization, with a mean number of burnt trees of 9.70 and mean number of trees on fire of 1.50.

| | **Fixed Fire Init** | | | | **Random Fire Init** | | | |
|---|---|---|---|---|---|---|---|---|
| | Small Grid | | Large Grid | | Small Grid | | Large Grid | |
| Algorithm | Burnt | On Fire | Burnt | On Fire | Burnt | On Fire | Burnt | On Fire |
| *Minimize Fires Reward* | | | | | | | | |
| RANDOM | 15.20 | 13.00 | 16.70 | 21.00 | 14.90 | 8.10 | 15.00 | 20.30 |
| HEURISTIC | 4.70 | 0.00 | 6.60 | 0.00 | 4.60 | 0.00 | 6.40 | 0.00 |
| A2C | 8.60 | 3.00 | 13.80 | 27.30 | 7.90 | 9.50 | 13.10 | 23.90 |
| PPO | 5.00 | 0.00 | 9.10 | 2.10 | 4.70 | 0.00 | 16.10 | 10.80 |
| *Dense Combo Reward* | | | | | | | | |
| RANDOM | 18.70 | 12.60 | 14.30 | 24.00 | 12.60 | 10.40 | 15.20 | 18.40 |
| HEURISTIC | 4.70 | 0.00 | 6.60 | 0.00 | 4.60 | 0.00 | 6.40 | 0.00 |
| A2C | 7.10 | 0.50 | 17.60 | 13.30 | 6.10 | 1.80 | 12.70 | 8.70 |
| PPO | 4.80 | 0.00 | 11.50 | 4.60 | 4.70 | 0.00 | 9.70 | 1.50 |

Table 4: Performance comparison of algorithms across fire initialization types (Fixed vs. Random), grid sizes (Small and Large), and reward functions (Minimize Fires and Dense Combo). Metrics are mean number of burnt trees and trees on fire at episode end. Best results are underlined.

## 5.2 Qualitative Analysis

On the small grid with fixed initialization, PPO with Minimize Fires appears to immediately move towards the fire and frequently extinguishes it immediately. However, if the fire was able to spread at all, the policy does not seem to understand how to respond. Anti Idle appears to act similarly, which is expected due to the almost identical reward functions.

Dense Combo also immediately moves towards the fire and frequently extinguishes it immediately. Though when the fire is able to spread, agents sometimes optimally move to the new trees on fire, showing better decision-making than with other reward functions. On the other hand, Proactive Firefighter also frequently extinguishes the fire immediately, though when the fire is able to spread, agents sometimes move in the opposite direction of where they should.

Surprisingly, PPO with Distance to Fire does not behave similarly to the heuristic. Agents generally move towards the fire at first but then seem to take random actions rather than specifically targeting fire edge trees. On the other hand, the heuristic performs exactly as intended, as each agent always moves directly towards the nearest fire edge.

Despite the slightly better quantitative performance of Proactive Firefighter, we select Dense Combo as our preferred reward function because of how agents better respond to trees that start on fire after initialization. We also select Minimize Fires because it performs similarly to other reward functions, but most matches our evaluation metric, which could allow for better understanding for how agents should behave.

On the large fixed initialization grids, PPO with Dense Combo seems to behave extremely intelligently. The agents move towards the fire and both continue to target new fire when it spreads. On the other hand, it seems like one agent learned much better than the other agent for A2C. One agent does not behave intelligently at all, while the other somewhat moves to extinguish the fire. PPO with Minimize Fires seems to move towards the fire then takes random actions.

On the large random grids, PPO with Dense Combo always moves directly towards the fire, but then seems to act almost randomly. On the other hand, A2C with Dense Combo only sometimes moves towards the fire, and when it does, it takes a meandering path! PPO with Minimize Fires seems to move towards the fire then takes random actions.

It seems like the improved performance on random initializations is due to fires being initialization closer to an agent rather than due to improved policies.

The most intelligent policy seems to be PPO with Dense Combo. The agents move directly towards the fire and continue to target new fire when it presents itself.

### 5.3 Analysis

We found that PPO significantly outperforms all other RL algorithms, and A2C is the second best-performing algorithm. Both DQN and RPPO perform very poorly, which is surprising. PPO determines the optimal policy based on maximizing the future reward and does not change parameters much between steps. As a result, it is not so sensitive to noise and random changes. This allows it to handle this stochastic environment well. Our reward functions create a dense reward environment because there is consistently a changing reward for each timestep. As a result, A2C-trained policies can measure the advantage of a specific action. However, the weaker performance could be attributed to the difficulty of incorporating delayed rewards, such as how if you burn a superspreader tree, you may not see the positive effect for a while. DQN maps state-action pairs to future rewards, so it struggles with stochastic transitions where the resulting state from a state-action pair is not always, or even consistently, the same. This may explain its extremely poor performance. Arguably the most surprising result is the extremely poor performance of RPPO. We believed RPPO would train good policies for this setting because of how it can use memory to capture delayed rewards and potentially then be able to identify which trees are more important to extinguish. However, RPPO requires a significant amount of training and can be sensitive to parameterization. It is possible that with more training and hyperparameterization, RPPO could still prove to be the best algorithm.

## 6 Discussion

The main takeaway is the clearly superior performance of our multi-objective Dense Combo reward function. The Dense Combo reward function is reward $= 0.3R_{minfire} + 0.5R_{distance} + 1.0R_{extinguish} - 0.2R_{still}$. Early on, when the agent is far from the fire, the distance to fire dominates the reward, forcing the agent to move towards the fire. This creates a dense reward environment, and ensures useful feedback due to the built in exploration. Rather than randomly exploring and rarely reaching the fire, this ensures agents quickly reach the fire, allowing for useful feedback and aiding learning. Once the agent reaches the fire, the reward function is based on the number of trees on fire, and extinguishing the fire. The stillness penalty ensures agents do not get "stuck" and is also a built in exploration strategy. The reward for extinguishing fire encourages agents to put out the fire, but the main part of the reward is still based on the overall objective of minimizing the number of trees on fire. These results suggest that learning may be better incentivized by specific goals that incorporate how the environment may change over time. Having a reward function that only targets one thing may not be able to manage incentivizing specific actions that are required in specific states, such as actions when far away from the fire. A single-objective reward function may also not successfully balance exploration and exploitation.

Another result is the clear superior performance of the heuristic. It is not clear whether there is a more efficient way to suppress the fire in these initializations. One possible takeaway is that RL is not as well suited for this task as this heuristic. What is more likely, we argue, is that this heuristic is optimal for the specific dynamics of this environment, but will not generalize as well to the real-world as RL would. This environment has a fixed probability of fire propagation across the entire discrete grid and over time. As a result, our heuristic specifically targets the way fire spreads in this environment. However, our environment inherently assumes uniform fuel for fire across geography and uniform dynamics over time. On a more complex environment, it is likely that RL agents would better be able to target the unexpected stochasticity. Optimal behavior will likely still target trees on the fire edge, but they would be able to identify "superspreaders". In other words, there may be trees that due to the high fire fuel around them, are susceptible to quickly spreading the fire, and it is worth the extra timesteps to extinguish these super spreaders than it is to extinguish the closest fire edge tree.

We also demonstrate the usefulness of multiple policies. The pairs of agents exhibited non-identical behavior on the same grid, indicating that the policies incorporated relational and geographical information. In some cases, such as A2C on large fixed-initialization grids, one agent outperformed the other. This suggests the benefit of having multiple single-agent policies that are able to learn individualized geographical information.

Several constraints limit the generalizability of our findings. First, the wildfire-environment simulation is a more simple model than real-world wildfires. It lacks important factors like wind, heterogeneous terrain, and fuel variability. These abstractions may cause us to overestimate the effectiveness of our tested strategies. Second, although we adopt a single-agent policy approach for scalability, this does not allow for coordination like multi-agent policies and could lead to worse global behavior in larger or more complex deployments. Third, compute and time constraints restricted the extent of our hyperparameter tuning and the total number of training steps, which likely hindered the performance of sensitive algorithms like RPPO. Lastly, our evaluation averaged results across only ten episodes per setting, which may not fully capture variability in outcomes caused by the stochastic nature of fire propagation.

The broader impact of this project is the potential of RL to support real-world disaster mitigation. Developing effective wildfire suppression strategies is an area of increasing importance under climate change. If deployed effectively, RL systems could reduce risks to human firefighters, allow rapid fire containment, and better conserve resources like water and fire retardant. However, deploying such systems in the real world would demand more robust safety testing and careful integration and coordination with existing human firefighting operations.

One of the biggest challenges we encountered was finding a suitable simulation environment. We initially aimed to use SimHarness but faced setup and compatibility issues. After experimenting with PyroRL and Cell2Fire, we transitioned to wildfire-environment v0.1.9, which required significant adaptation to support our reward functions and evaluation protocols. Training RL agents in this environment was challenging due to the sparsity of meaningful rewards in larger grids, where agents often failed to reach the fire early in training. We had to tune the grid size during testing. If it was too small, the task became trivial. If it was too large, agents struggled to learn due to infrequent feedback.

## 7 Conclusion

Our results suggest that RL is a promising approach to UAV training for wildfire suppression, and this application should continue to be explored. We showed that a reward function with multiple objectives better enables RL agents to learn strategies that balance exploration and exploitation across different states and is more generalizable to different environment initializations. Our best model is trained with PPO on the dense combination of rewards, though it is still outperformed by a heuristic where agents move to the closest tree on fire that borders a healthy tree.

Our results show that reward design is an important component of reinforcement learning, especially in sparse environments. By combining multiple objectives, we transformed a sparse reward landscape into a denser one, allowing agents to receive more consistent feedback and learn behaviors that generalize across different environments. At the same time, a simple heuristic that targets the fire edge consistently matched or outperformed all RL models in our setting. This shows the value of evaluating learned policies against strong, interpretable baselines, especially in structured domains where heuristics can be highly effective.

### 7.1 Future Work

Future work should train more agents and on a more complex and realistic environment that better captures wildfire dynamics. Future work can also explore training beyond a simulation environment and incorporate imitation learning where human-controlled drones collect initial data. More training and hyperparameterization may be required for RL training in this complex application.

## 8 Team Contributions

- **Amy Guan:** Env. setup, training/eval pipeline, reward functions, experiments, and report
- **Ryne Reger:** Env. adaptation, heuristic, reward functions, experiments, and report
- **Emily Tianshi:** Env. setup, literature review, reward functions, experiments, and report

**Changes from Proposal** We originally intended to build on the SimFire and SimHarness environments developed by Tapley et al. (2023) to train RL agents representing firefighters for wildfire mitigation strategy optimization by extending their single-agent DQN baseline, experimenting with modified reward functions, and introducing real-world constraints. After running into difficulties implementing SimHarness, we also explored PyroRL (Guman et al. (2024)), an RL environment simulating wildfire evacuation, and Cell2Fire (Pais et al. (2019)), a cell-based forest fire growth model. Ultimately, we landed on wildfire-environment v0.1.9 (2024) , which similarly to SimHarness, explores the potential of reinforcement learning in wildfire suppression, although specifically with UAVs rather than with people.

## References

Abdulelah Altamimi, Constantino Lagoa, José G. Borges, Marc E. McDill, C. P. Andriotis, and K. G. Papakonstantinou. 2022. Large-Scale Wildfire Mitigation Through Deep Reinforcement Learning. *Frontiers in Forests and Global Change* Volume 5 - 2022 (2022). https://doi.org/10.3389/ffgc.2022.734330

Maggie Guman, Brittany Fiore-Gartland, Vivian Lin, Shruti Mishra, Erin Sparks, Nisarg Shah, Rishabh Ahuja, Julia Wilkening, Jon Wilkening, Serge Levine, Pieter Abbeel, Joseph E. Gonzalez, and Steve Yadlowsky. 2024. PyroRL: A Reinforcement Learning Environment for Wildfire Evacuation. *Journal of Open Source Software* 9, 101 (2024), 6739. https://doi.org/10.21105/joss.06739

Ravi N. Haksar and Mac Schwager. 2018. 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)*. 1067–1074. https://doi.org/10.1109/IROS.2018.8593539

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. arXiv:1602.01783 [cs.LG] https://arxiv.org/abs/1602.01783

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (Feb. 2015), 529–533.

Lucas Murray, Tatiana Castillo, Jaime Carrasco, Andrés Weintraub, Richard Weber, Isaac Martín de Diego, José Ramón González, and Jordi García-Gonzalo. 2024. Advancing Forest Fire Prevention: Deep Reinforcement Learning for Effective Firebreak Placement. arXiv:2404.08523 [cs.LG] https://arxiv.org/abs/2404.08523

Cristobal Pais, Jaime Carrasco, David L. Martell, Andres Weintraub, and David L. Woodruff. 2019. Cell2Fire: A Cell Based Forest Fire Growth Model. arXiv:1905.09317 [stat.CO] https://arxiv.org/abs/1905.09317

Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. 2019. Stable Baselines3. https://github.com/DLR-RM/stable-baselines3.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG] https://arxiv.org/abs/1707.06347

Reza Shami Tanha, Mohsen Hooshmand, and Mohsen Afsharchi. 2023. UAV-based Firefighting by Multi-agent Reinforcement Learning. 117–123. https://doi.org/10.1109/ICCKE60553.2023.10326250

Alexander Tapley, Marissa Dotter, Michael Doyle, Aidan Fennelly, Dhanuj Gandikota, Savanna Smith, Michael Threet, and Tim Welsh. 2023. Reinforcement learning for wildfire mitigation in simulated disaster environments. *arXiv preprint arXiv:2311.15925* (2023).

wildfire-environment v0.1.9. 2024. wildfire-environment v0.1.9. https://pypi.org/project/wildfire-environment/0.1.9/. Python library.