Fighting Forest Fire Project

Course code: DS251

Group no: 3

***Abstract****: Our goal is to create a simulator for forest fires and use Multiagent Reinforcement Learning to simulate teamwork in battling fires. AI agents, together with equipment, will cooperate to put out the fire. To accommodate different eventualities, we want to integrate several cost functions. The goal of this project is to build a dynamic, learning-based system that allows agents to optimize their tactics for efficient fire management in addition to simulating the spread of fire in a realistic manner.*

***Team members****:*

1. *Mohd Adil - 12141080*
2. *Ratnakar Gautam - 12141360*
3. *Roushan Kumar - 12141420*
4. *Shishir Kushwah - 12141510*

**INTRODUCTION**:

Forest fires represent a grave threat, causing substantial economic, environmental, and property damage globally. With each incident incurring costs surpassing billions of dollars for suppression, rehabilitation, and public support, tackling this menace is imperative. Given the challenges posed by terrains, weather dynamics, and risk to human life, aerial robotics emerges as a promising solution for monitoring and controlling forest fires. Unmanned Aerial Vehicles (UAVs) offer agility, scalability, and the potential for large-scale deployment in a distributed manner, obviating the need for a centralized decision-making entity. Integrating multi-agent systems, like a fleet of UAVs, could provide crucial information to firefighters, aiding their planning, monitoring, and firefighting endeavors.

In our report, we begin by introducing a simplified stochastic model based on lattices, capturing the fundamental behavior of forest fires. Additionally, an agent model encompassing motion and operational limitations is presented. We demonstrate the infeasibility of precise and approximate control methods reliant on exhaustive state-space enumeration. To address the complexity, we explore task decomposition strategies and detail a manually crafted heuristic to drive control inputs for agents.

Our approach extends Q-learning for multi-agent reinforcement learning (RL), focusing on persistent forest fire monitoring.While model-based RL proves suitable, existing methods falter due to domain complexity or necessitate frequent recomputation with altering parameters. To tackle this, we leverage task abstraction to enhance long-term planning efficacy. Our proposed solution is deep Q-learning to ensure scalability.

**MODELING**:

1. ***Problem Modeling***

The forest fire model described here represents a stochastic discrete-time system where the forest is depicted as a collection of trees on a two-dimensional lattice. Each tree can be in one of three states: healthy, on fire, or burnt denoted by H, F, and B respectively. Actions in this model involve applying fire retardant on a tree, represented by binary variables that are available only if a UAV is positioned over the tree.

***Assumptions***:

The state transitions of a tree are influenced by its current state and neighboring trees. The neighborhood is defined as trees located one unit away in Manhattan distance. The transition probabilities for a single tree depend on its current state and the states of its neighbors. These transitions are governed by specific probabilities:

1. Forests contain only one kind of tree and with equal distance of unit 1.
2. Healthy trees will transition to being on fire probabilistically if at least one neighboring tree is on fire.
3. Trees on fire either remain on fire or transition to a burnt state based on a probability that is influenced by the applied action.
4. Burnt trees stay burnt indefinitely.

The parameters controlling these transitions include:

- α, which defines the likelihood of fire spreading from an on-fire tree to a healthy tree.

- β, describing the probability of an on-fire tree remaining on fire.

- ∆β, indicating the impact of actions on the persistence of fire.

The aggregate state of the entire forest is determined by a joint probability distribution considering the states of all individual trees and their transitions based on the states of their neighbors and applied actions.

This model accounts for the dynamics of forest fire propagation, considering the interplay between individual tree states, neighboring influences, and the effects of actions taken to control the fire. The parameters α, β, and ∆β can be extended to accommodate additional factors like wind and terrain variations, although in this particular context, they are considered constant.

1. ***Agent Modeling***

The UAV agent model presented here outlines the characteristics and capabilities of the agents involved in managing the forest fire scenario. Key components of this model include motion constraints, communication abilities, and sensor functionalities:

1. Motion:

1. Each agent moves on a two-dimensional lattice similar to the forest layout. Agents have the freedom to move infinitely in any direction.
2. There are nine possible actions available to agents that allow movement within the lattice.
3. Start of the simulation all the agents will reach the mean position of the fire.

2. Sensors:

1. Agents are equipped with two types of sensors:
2. A downward-facing infrared (IR) camera that captures the states of a grid of trees. The captured image encompasses a specified grid size, padded with healthy trees if the agent is at the forest edge.
3. A radio sensor that provides information about the initial average position of trees on fire across the entire forest.

3. Communication:

1. . Agents communicate with the nearest other agent, sharing their positions and unique labels.
2. Only the nearest agent exchanges information with another, transmitting its position and label to the designated agent.
3. Communication occurs without exchanging additional data, maintaining a relatively low-bandwidth communication model.

4. Control Action:

1. Fire retardant is applied to a tree when an agent's position coincides with a tree on fire within a specific time interval.
2. Multiple agents reaching the same tree within this interval do not compound the effect of the retardant. Agents must collaborate to prevent wastage of control efforts.
3. Agents have a specified retardant capacity, limiting the number of trees they can treat. Upon depletion, agents return to a designated base station for refilling before redeployment into the forest.

5. Memory:

1. Agents maintain memory of whether their position has coincided with a tree that was on fire or burnt. This information is derived from the image data captured by their sensors.

Implementation considerations:

1. The assumed capabilities of the agents align with certain UAV (Unmanned Aerial Vehicle) capabilities described in literature, particularly in trajectory control for quadrotor vehicles.
2. These capabilities include maintaining constant altitude and lateral movement to achieve discrete motion patterns similar to those described in the agent motion model.
3. Sensors like infrared cameras and high-frequency radios are common in many UAV platforms and are relatively compact and cost-effective.
4. Agents are assumed to be distributed throughout the forest before a fire ignition event, with the location communicated to them via broadcasting.
5. Communication between agents is kept low-bandwidth and relatively simple, minimizing error susceptibility.
6. The model assumes a practical storage capacity for fire retardant on agents, allowing treatment of a specified number of trees before requiring a refill at a base station.

Overall, these agent and forest fire models are highly abstracted but maintain essential attributes necessary for devising an effective fire-fighting strategy employing a team of UAVs.

1. ***Problem Complexity***

The problem complexity associated with modeling the forest fire scenario using Markov Decision Processes (MDPs) and Partially Observable Markov Decision Processes (POMDPs) is considerable due to the sheer number of possible configurations and constraints imposed by agent capabilities:

1. Forest Representation:

1. Each tree in the forest is described as an infinite horizon, discounted MDP considering its state and reward function.
2. The whole forest consists of locally-interacting MDPs, where each tree's region of interaction is defined by its neighbor set.
3. The number of possible configurations for even a moderately sized forest is astronomically high, making it challenging to control this networked MDP effectively.
4. Considering only reachable states based on non-zero probability transitions can reduce complexity, yet the number of possible configurations remains enormous even for small fire scenarios.

2. Agent Constraints:

1. Agents are limited in their observations due to restricted field-of-view sensors (e.g., cameras) and motion constraints within the lattice.
2. Consequently, the problem falls under Partially Observable MDP (POMDP) framework due to both motion and partial observability constraints.
3. POMDP solutions are computationally challenging and can be intractable for exact solutions, especially for infinite horizon problems.

3. Factored MDPs (FMDPs):

1. FMDPs leverage problem structure to create a more compact representation compared to traditional MDPs by storing individual state space, action space, transition function, and reward function for each constituent MDP.
2. Despite the compact representation, exact solutions for FMDPs still require enumeration of the aggregate state space.

4. Solution Complexity:

1. While FMDPs offer a more structured representation, exact solutions necessitate exploring the aggregate state space, posing computational challenges.
2. Approximate methods exist for both FMDPs and factored representations of POMDPs, but they might struggle with the added complexity of partial observability constraints.

5. Proposed Solution:

1. The goal is to find a solution method that doesn't require complete recomputation when the number of trees in the forest changes.
2. Deep Reinforcement Learning (RL) methods are suggested as potential scalable solutions to tackle overwhelming complexity and constraints posed by agent capabilities and partial observability.

In summary, the forest fire model's complexity arises from the enormous number of possible configurations, agent constraints, and the challenges in finding computationally feasible solutions for effective control and decision-making in such scenarios. Techniques like Factored MDPs and Approximate methods offer some relief but might still struggle with the complexities introduced by partial observability and the overall scale of the problem. Deep RL methods are anticipated to provide scalable solutions to address these challenges.

**APPROACH:**

The approach adopted here involves a hand-tuned method that aims to generate actions for agents independent of the forest size or the number of agents by decomposing the problem into individual agent-level tasks. This decomposition simplifies the complexity associated with agent motion, sensing, and communication constraints. The specific tasks assigned to each agent are outlined as follows:

***Task Decomposition:***

Each agent is assigned two primary objectives:

1. Approach the Initial Forest Fire Location:
2. Agents are tasked with reaching the location in the forest where the fire initially started.
3. Their objective is to find the shortest path from their current position to the fire's origin.
4. Agents aim to minimize their distance from the initial fire location without considering communication or collaboration with other agents.
5. Move to Suppress the Forest Fire:
6. After reaching close proximity to the initial fire location, agents transition to the second task.
7. The criterion for this transition is based on an agent's memory `ck`, which is triggered when the agent encounters a tree that is not in a healthy state.
8. This detection indicates that the agent is in the vicinity of the fire, prompting them to switch focus to suppressing the fire.

These tasks aim to simplify the decision-making process for individual agents by breaking down the complex problem into sequential objectives: initially navigating towards the fire's origin and subsequently taking actions to combat and control the fire upon reaching close proximity to its location.

***Deep Q-Learning Approach:***

This approach involves utilizing a deep reinforcement learning (RL) method termed Multi-Agent Deep Q Network (MADQN) to generate decentralized solutions that outperform heuristic approaches while being scalable independently of forest size or the number of agents.

***Agent Reward Algorithm:***

This algorithm calculates the reward for an agent based on its actions and the state it encounters during the forest fire scenario. Here's a summary of the steps:

1. Input: Agent's feature set and action.
2. Output: Agent reward).
3. The initial reward is set to zero.
4. If the agent moves to a boundary tree on fire, the reward is increased by 1. If not, the reward is decreased by 2.
5. If the agent moves to a healthy tree with at least one neighboring tree on fire or burnt, the reward is increased by 0.5; otherwise, it's decreased by 1.
6. Depending on specific conditions (e.g., distance to an obstacle), the reward is adjusted accordingly.
7. The reward is updated based on agent movement direction (e.g., clockwise).

***Neural Network Architecture:***

1. Input: Agent's feature set (represented by a black rectangle).
2. Hidden Layers: Three fully-connected layers with Rectified Linear Unit (ReLU) activations (represented by blue rectangles).
3. Output: The output is a vector containing values for each possible action the agent can take (represented by a red rectangle).

Training Algorithm (Algorithm 3):

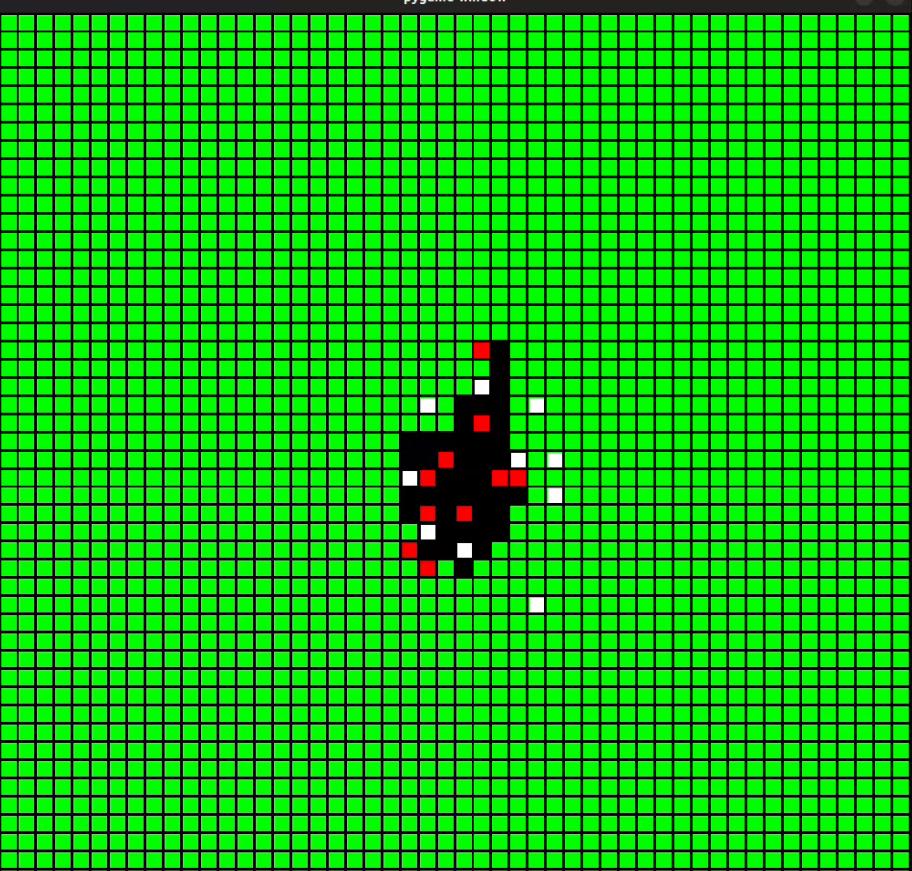
1. The training of the MADQN involves episodes where an initial configuration of trees on fire and agent locations is specified (depicted in Figure 6a).
2. Each agent contributes experiences…

**EXPERIMENT AND RESULT:**

The experiments in the study utilized specific parameters and methodologies to simulate and evaluate forest fire control methods.

1. Model Parameters: The study utilized various parameters derived from previous research to simulate the spread and persistence of forest fires.

1. MADQN Architecture: The method employed a specific neural network architecture with three hidden layers for determining the best actions in controlling the fire spread.
2. Training Details: Training of the MADQN involved specific techniques and parameters to train the network for effective decision-making.
3. Simulation Experiments: The experiments were divided into distinct scenarios to test the efficiency of the control methods under different constraints and initial conditions.
4. Evaluation Metric: The evaluation of both control methods was based on the fraction of remaining healthy trees at the end of each simulation.
5. Simulation Results: The results were presented in a table comparing the performance of the heuristic method and MADQN across different scenarios involving varying numbers of initial fires and agents.
6. Performance Comparison: The comparison indicated the scalability and adaptability of MADQN compared to the heuristic method in various scenarios, emphasizing its effectiveness in preserving healthy trees.
7. Action Comparison: A comparison was made between the actions suggested by the heuristic method and those predicted by MADQN to illustrate their differences in approaching fire control strategies.



*Here is the sample image of the simulation.*

*Color coding: Agent (white), Healthy Tree (green), Burnt Tree (black) and Burning Tree (red)*

**CONCLUSION**:

This work introduces a decentralized forest fire control strategy using deep RL, accommodating agent constraints. Validated via Monte Carlo simulations, it aims to address complexities in real-time fire management. Future enhancements involve refining task decomposition based on existing literature, adapting the approach for real-world factors (e.g., wind, terrain), and conducting extensive experiments with quadrotors in simulated fire scenarios. These efforts aim to bolster the proposed deep RL method, ensuring scalability and practicality for effective forest fire control in diverse, real-world environments.

REFERENCES:

1. <https://ieeexplore.ieee.org/document/8593539>
2. <https://www.youtube.com/playlist?list=PLZbbT5o_s2xoWNVdDudn51XM8lOuZ_Njv>
3. <https://www.geeksforgeeks.org/introduction-deep-learning/>
4. <https://www.ibm.com/topics/deep-learning>