Performance Analysis of Reinforcement Learning Algorithms on mitigation of Forest Fires

Sagar Abhyankar
Department of Computer Engineering
Pune Institute of Computer Technolgy
Savitribai Phule Pune University
Pune, India

Omkar Bhosale
Department of Computer Engineering
Pune Institute of Computer Technolgy
Savitribai Phule Pune University
Pune, India

Aryan Dande
Department of Computer Engineering
Pune Institute of Computer Technolgy
Savitribai Phule Pune University
Pune, India

Sarang Joshi
Department of Computer Engineering
Pune Institute of Computer Technolgy
Savitribai Phule Pune University
Pune, India

***Abstract*—This paper intends to use and compare the Reinforcement Learning Algorithms' performance regarding stopping and mitigating forest fires. The Agents are trained using reinforcement learning allowing them to interact dynamically with the environment generating previously undiscovered but efficient fire-fighting strategies. The Agents are independent of each other and can coordinate their actions by communicating amongst themselves. The forest fire simulator uses an Open AI Gym based on Drossel and Schwabl's model of 2D Automata for spread of forest fires. This common benchmark will allow us to compare and analyze different Reinforcement Learning Algorithms like: Deep Q learning, Soft Actor Critic, Proximal Policy Optimization which can solve this real-world problem of forest fire mitigation. The Paper also puts forth the feasibility of Reinforcement Learning Models to solve Real World Environment Challenges.**

Keywords-Reinforcement Learning, Deep Q learning, Soft Actor Critic, Forest fire control

I. Introduction

Forest Fires have been a recurring problem that humanity has faced for several millennia. Often these forest fires have claimed lives and destroyed livelihoods and property. Large Fires can easily rake damages up to a billion dollars. Between 2021-2022, there have been damages of $11.2 billion caused by forest fires in the United States of America alone. Due to climate change and global warming, this problem will continue to increase. According to UNEP, there will be a 14% increase in intense wildfires by 2030 and a 30% increase by 2050. Thus, a problem of this severity must be looked at through a new perspective of Reinforcement Learning to search for suitable and optimal solutions. Research in this area will also allow us to check for the efficiency of current fire-fighting strategies and their feasibility.

Forest fires depend on the following three things: fuel, heat, and oxygen. Depriving the fire any one of them will mitigate and put out the forest fire. Methods for controlling the fire can be divided into airborne agents and ground agents. Mainly the methods involved in stopping forest fires can be divided into two parts: Airborne agents and Land Based Agents. Airborne agents like airplanes and helicopters deprive the forest fire by cutting it off from heat or oxygen (main components of fire) by enveloping the affected area with water or chemical retardants. Ground-based agents such as trucks or land rovers also use water from their water tanks to directly bring forest fires under control. Other ground agents are bulldozers, tractors, or people equipped with e.g., chainsaws. All of them stop the spread of fire by cutting it from its fuel source, trees. The choice of methods and equipment which are used depends on the country and the kind of environment. When the outbreak of fire is detected, a team of fire-fighters spring to action, and their fire manager draws out initial plans to curb the threat after evaluating the situation. This plan may be susceptible to biases and the pressure of the situation may lead to errors. While the Reinforcement Learning algorithms should not take complete control away from humans with regard to managing fire, the strategies developed by these models will help fire managers to make informed decisions and be invaluable assets in high-pressure situations.

This Paper aims to create a model for coordinating these independent agents and their course of action. Various Reinforcement Learning Algorithms like Deep Q learning, TD3, Soft Actor-Critic, and Proximal Policy Optimization have been considered. Additionally, these algorithms can be further calibrated according to various financial and geographic needs.

[VI] of the paper aims to compare and analyse the performances of these algorithms belonging to various types of sub-divisions of Reinforcement Learning to find out the best performing one in this specific use case of mitigation of forest fire in a 2d environment. The performance is based on the number of trees saved from burning and the amount of computing required to train the model.

II. Reinforcement Algorithms

A major disadvantage of Machine learning algorithms is that it requires a lot of data. The data can contain null values or can be not up to the mark and hence a lot of data pre-processing needs to be done. Reinforcement learning is a

branch of Machine learning where it does not need a predefined data set. It is used where writing the actual code is very difficult. Here the steps to be taken are decided on its own based on the reward the system gets for every right action. Some of the important terms include Agent, which performs actions in an environment. The environment provides a reward for certain actions. State is the present status of the Agent. The agent takes decisions based on the Policy. Every state is mapped to a value using the value function. A Markov Decision Process (MDP) is defined here [3].

There are two main approaches:

In model-based the decision for the action to be taken is taken before the action is made and in model-free it carries out the action to gain experience from it.

Examples of model based include chess, robotic arms in factories. While Model-free systems are better in self-driving cars. Our Approach: As our goal is to find and observe the optimal path, we will be using a model –free approach.

Target Policy is the policy that the agent finally desires to learn while the behavior policy is the policy is the policy that the used for exploration used in the environment.

On-Policy learning: In On-Policy learning the same policy is modified repeatedly to get a more efficient policy. Basically, Target Policy is the same as Behavior Policy. E.g. Policy Iteration, Value Iteration, Monte Carlo for On-Policy etc.

Off-Policy Learning: In Off-Policy learning algorithms a new more efficient algorithm is selected after evaluating the current policy. In short, Target Policy is not equal to Behavior Policy. E.g., Expected SARSA, Q learning

A. Policy Gradient

Instead of approximating a value function and using that to compute a deterministic policy(value-based),a stochastic policy is approximated in Policy gradient .θ is the vector of policy parameters and ρ the performance of the corresponding policy .Then the policy parameters are updated proportional to the gradient.[6] Policy gradient methods can handle continuous action spaces.

B. Actor Critic

Actor critic algorithms are a combination of On-Policy and Off-Policy algorithms by taking the best of the two. One part of the algorithm finds the action to be taken from the current state and the other part calculates Q values for that state. The input for the actor is state and output is action. The actor that is the first part is policy based while the critic is value based. Both participate in the working of the algorithm. Finally, the result produced is more efficient than the algorithms individually.[17] A3C is an asynchronous variant of A2C

C. Proximal Policy Optimization

PPO is like the gradient methods, as it calculates output probabilities in forward pass. The gradients are calculated to improve those decisions or probabilities in the backward pass. In PPO the old policy and updated policy are kept in proximity to each other (denoted by ε). Large updates are not allowed. PPO works well in continuous action spaces [7]. It is stable, high performance and scalable. One of the disadvantages is it is difficult to reproduce, and it converges to local maxims due to sensitivity of hyperparameters [8].

D. Trust Region Policy Optimization

Trust Region Policy Optimization, or TRPO, is a policy gradient method in reinforcement learning that takes large steps but also maintains a certain difference between old and new policy known as KL-Divergence.[18]

E. Deep Deterministic Policy Gradient

In DDPG the actor maps states to actions. It does not output the probability distribution across a discrete action space, it uses continuous action spaces. This algorithm improves the stability in learning. [9]

F. Twin Delay DDPG

TD3 is the successor of DDPG. DDPG can be unstable and thus have the problem of overfitting the Q values of the critic (value) network. TD3 solves this issue of overfitting by adding extra random noise to the target policy.TD3 trains a deterministic policy in an off-policy way.[15]

G. Soft Actor Critic

According to [10] SAC is defined for continuous action spaces. It uses a modified RL objective function. SAC is based on Entropy maximization. Entropy is the randomness of a variable here.

H. Q-learning

The s and a values are mapped into a memory table Q [s ,a]. The Q-value function returns an action and state pair. The agent needs to maximize this Q-Value function. γ denotes a discount factor. As a result, the immediate rewards get more value while rewards lose the value over time.[20]

I. Deep Q Network

It is an improvement in the Q-learning function as Q-learning was highly computational and hence Deep neural networks were used for this.

J. Categorical Deep Q Network

C51 is based on DQN. But the output layer instead of its mean $Q\pi(s,a)$, predicts the distribution of the returns for each action a in state s.

K. Quantile Regression -DQN

QR-DQN is improved version of C51 are DQN. Hindsight Experience Replay allows sample-efficient learning from rewards. It can be combined with an arbitrary off-policy RL algorithm and may be seen as a form of implicit curriculum.[13]

L. State Action Reward State Action

SARSA, State Action Reward State Action is an on-policy temporal difference learning method. In this we calculate Q π (s, a) for the current policy π and all pairs of (s-a). In SARSA, policy is updated based on action Q-value is updated based on error.[14]

SARSA uses Q (s, a, r, s', a'). s is the original state, a is the action, r is reward, s' is the new state and a' is the new action.

III. Literature Survey

[12] In this work, the spread of forest fire was modelled as 2d cellular fire automata which followed the rules:

- Each cell on the 2d grid is either vacant or covered by the grid.
- On each step, an `ignition' occurs with probability p, where p is a very small probability.
- When ignition occurs, the nearby cells have a probability q to ignite, and the entire cluster of trees connected to the ignited site burns down and becomes vacant.

The forest fire model is critical over a wide range of parameter values, and is robust with respect to slight modifications of the model rules. The fire in the model is initiated with a lightning strike.

[21] In this work using the foundations of Trust Region Policy Optimization, a less complex and easy-to-change policy is created that also does not allow the Policy to keep running Gradient Descent until it is indistinguishable from the old policy. PPO too, uses multiple epochs of stochastic gradient ascent to perform each policy update. This is done using a modified Objective Function that is simpler to implement than Trust Region Policy Optimization. The paper then conducts a benchmark test Proximal Policy Optimization showing that it outperformed other online policy gradient methods like A2C, CEM, TRPO and Vanilla. The paper also displays graphically the comparison of PPO and A2C on all 49 Atari games included in Open AI Gym. The paper then concludes on the note that PPO, whilst maintaining TRPO like reliability and stability, is much easier to implement requiring fewer lines of code but achieves overall better performance.

[22] This paper has used satellite images to capture real world data and analyze forest fires. The approach includes a fire spreading model. It was concluded that A3C is better at predicting the logic of spreading while MCTS is better for predicting where the fire will spread in the future. Over a period Supervised Gaussian algorithms prove better than Reinforcement algorithms.

[23] In this paper a team of distributed deep reinforcement learning (RL) aerial agents in the form of UAVs to control the forest fire is proposed. The forest fire dynamics are derived from the stochastic discrete-time model [8]. The model of the UAV agent considers the motility of the agent, its capability of communication with other agents, and its sensing power. The evaluation criteria used for the MADQN (multi-agent deep Q Network), and Sample heuristics is the fraction of remaining healthy trees to total trees. The fractions are then classified into three broad categories {losses, limited, wins} based on the value of the fraction. MADQN's results get better with an increase in the number of agents for the same number of initial forest fires as compared to heuristics as seen in Fig. 7 in [4]

[1] In this work a novel approach to control forest fires in a simulated environment using connectionist reinforcement learning algorithms has been described. A forest fire simulator was introduced to benchmark several popular model-free RL algorithms. The forest fire dynamics operate on four basic rules

- Heat Potential
- Ignition-Threshold
- Temperature
- Amount of Fuel

The approach used to put out the fire is to dig the ground around the active region of forest fire to contain it. The forest fire simulation was done on a grid of square either of the dimensions 10 x 10 or 14 x 14. Total four different Algorithms were tested in the paper, consisting of Q-Learning, SARSA, Dueling Q Networks and Dueling SARSA. The benchmark results for each algorithm were logged in a tabular format. A comparative graph was plotted for comparison of these algorithms based on the rewards generated and number of training episodes required. The results show that the demonstration data are necessary to learn very good policies for controlling the forest fires in the simulator and that the novel Dueling-SARSA algorithm performs best.

[4] This paper presents a model for coordinating teams of computational agents for the cause of performing simulated forest firefighting in real world environments. The simulation was performed on Pyrosim Platform. Pyrosim provides complex real-world entities like forest, grassland and variety of topological environments and therefore it acts as a testbed for team co-ordination models as well as an environment for

simulating fire tactics. The overall team coordination was carried out by giving every agent local autonomy but assigning a team leader for the team coordination to be centralized at higher levels. The Leader is aware of global situation, reasoning about it, and giving specific tasks to agents to carry out a given plan. The paper shows that centralizing team coordination provides many tactical advantages, yet has its own drawbacks which are to be dealt with like creating bottleneck on the leader agent and a few others which are inherent in almost all centralized systems. Figures [5]-[9] show the wet-line tactic in action.
The paper also presents comparative graphical results for various tactics implemented in the counter action. The tactic of surrounding the forest fire with a "Wet-Line" (ST6) is found out to be the most promising and stable amongst all the tested tactics. The paper concludes by declaring that the use of machine learning for this application can result in selection of better tactics dynamically to cater to most of the forest fire scenarios.

IV. Environment

The environment is primarily based on Open AI's gym tool [11]. The simulation follows the forest-fire model stated in [12] which depicts the forest fire as a 2D cellular grid. The cell has a set of three possible states {Empty, occupied by tree, Burning Tree}. The model runs on the following rules.

1. After a predefined amount of time, the tree in the burning cell is destroyed causing it to be empty/blank cell.
2. The spread of fire caused by neighboring trees is given by a predefined probability. If a particular threshold is met the cell catches fire.
3. The model also considers spontaneous fire initiation with a probability close to real world environment.
4. After some duration, an empty cell can be converted into a cell containing tree with a predefined probability.

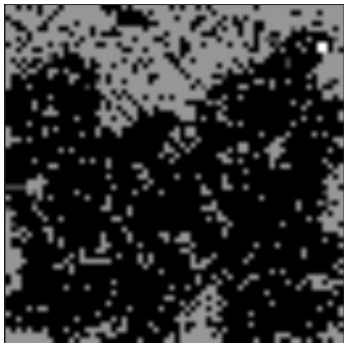


Fig. 1: The forest grid built with gym library. The red cells represent burning trees, black represents empty space and green (grey) represents tree cover. The agent single agent coverage area (shown in white) is also seen in the figure.

The agent tries to extinguish the fire and save the trees. Without the agent being actively involved, only init_tree x p_tree (Probability of a tree growing in an empty space.) remain at end of each episode which is considered as the baseline for the environment.

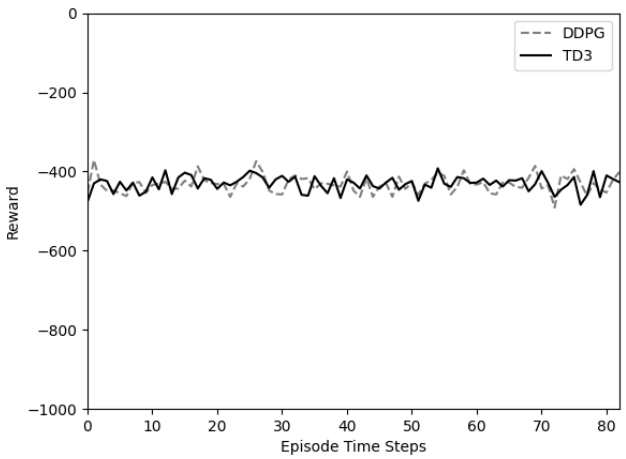
V. Rewards And Control

The environment as given in [12] can be modeled as a Markov Decision Process with the following considerations

1. The state represented is a 2D top orthographic view showing the forest with trees, fire and empty cells. The environment is 128x128 Numpy array with a value of 0 for empty cells, 1 for trees and 10 for the cells with active forest fire.
2. The fire firefighting team can have a dynamic location in the 2d environment given by [x, y], and the team puts off any fire in a predefined radius of the team's center coordinates.
3. In each stepping phase of the episode if the fire team successfully puts out fire from a burning cell, the agent gets a reward of 1 point, but if there is a fire which causes the tree to get destroyed, the team gets a reward of -1.
4. At the end of every episode, the total tree population saved or remaining is compared to the initial tree population. If it is more than half of the initial tree population, the fire team gets a reward of 300, if not then they get a reward of -300.
5. The transition probability depends on the variables of the environment model given in [12]

VI. Results

We now report the results which were obtained with the experiments in our implementation we executed the comparatively newer deep reinforcement algorithms such as DDPG (Deep Deterministic Policy Gradient) and its successor TD3, the results of these algorithms were then compared with the outcomes from other papers to generate a common performance comparison across all the reinforcement algorithms. The following plot shows the comparison results for TD3 vs DDPG, due to the limited computation power and memory storage at our disposal, we ran a total of seventy episodes, each having a three hundred training steps. Although the rewards are not in positive scale, it shows a good comparison between TD3 and DDPG in early training stages.



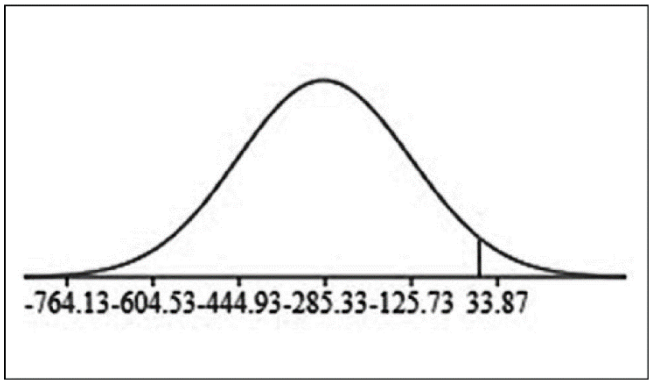
In [1] Fig.4 we find similar graphs plotted for a larger number of episodes for algorithms like Q-Network, SARSA, Dueling Q-Net, Dueling SARSA. The reward tends to slowly rise above positive X axis with increase in memories. All the algorithms start with a negative score as seen in the graph; this is due to agent still exploring all possibilities with minimum overhead. To achieve an overall comparison, we can make use of TABLE V from [1] for comparing the average rewards with zero episodes of demonstration data (loaded) as used in our case.

TABLE I

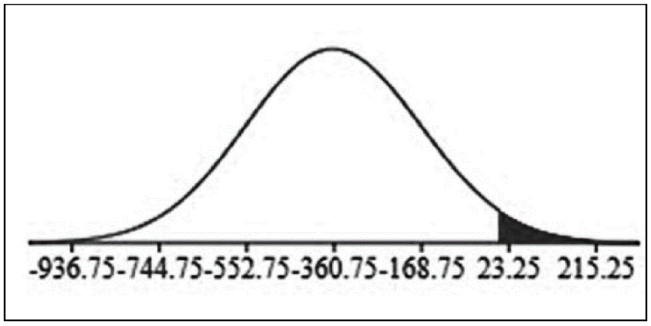
Algorithm	**Average Reward**	**Std. Error**	**Best Reward**
Baseline	**1152**	125	**1513**
Q-Network	-550	144	-139
SARSA	-398	116	-92
Dueling *Q*-Network	-40	335	349
Dueling SARSA	-455	134	-44
Deep Deterministic Policy Gradient	-427	126	-88
Twin Delayed DDPG	-395	147	-65

The following are the standard deviation graphs for the above tabulated results on average rewards (without considering the baseline).

GRAPH A



GRAPH B



Graph A shows the standard deviation including Twin Delayed DDPG and DDPG, Graph B shows the standard deviation without including them. Furthermore, from [15] evaluation graphs show that TD3 becomes largely more performant compared to other reinforcement algorithms over a training period of 0.2-million-time steps, carried out in seven different MuJoCo environments.

VI. CONCLUSION

In this paper, we have presented the comparative performance analysis of different reinforcement algorithms and the types of it. Special emphasis was given on testing latest developments Actor Critic algorithms for our use case which includes TD3 and DDPG. With in-depth study of our experiments and previous papers we conclude that both Dueling SARSA and Twin Delayed DDPG (TD3) are extremely performant when it comes to managing complex real-world environments like forest fires. However, as pointed out by our experiments, all the models perform similarly when not given enough demonstration data or enough training duration. We failed to find computationally lighter yet performant algorithm to match various use cases including ours.

In future work, we would like to use more complex forest fire simulation platforms and train the model on a greater number of episodes for better performance analysis, with better equipped computing infrastructure.

REFERENCES

[1] Hammond et al., "Forest Fire Control with Learning from Demonstration and Reinforcement Learning," *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1-8, doi: 10.1109/IJCNN48605.2020.9207548.

[2] Peter Dayana and Yael Niv — Reinforcement learning: The Good, The Bad and The Ugly, 2008)

[3] Modi et al., (2017). Markov Decision Processes with Continuous Side Information. *arXiv*. https://doi.org/10.48550/arXiv.1711.05726

[4] Moura et al., (2007). Fighting fire with agents: an agent coordination model for simulated firefighting.. 71-78. 10.1145/1404680.1404691.

[5] Wang et al., (2015). Dueling Network Architectures for Deep Reinforcement Learning.

[6] Sutton et al.. (2000). Policy Gradient Methods for Reinforcement Learning with Function Approximation. Adv. Neural Inf. Process. Syst. 12.

[7] Zhang et al., Proximal policy optimization via enhanced exploration efficiency,Information Sciences,Volume 609,2022,Pages 750-765,ISSN 0020-0255,https://doi.org/10.1016/j.ins.2022.07.111.

[8] Somanath et al., "Controlling stochastic growth processes on lattices: Wildfire management with robotic fire extinguishers," in 53rd IEEE Conference on Decision and Control, Dec 2014, pp. 1432–1437

[9] Lillicrap et al.,(2015). Continuous control with deep reinforcement learning. *arXiv*. https://doi.org/10.48550/arXiv.1509.02971

[10] J. -T. Chien and S. -H. Yang, "Model-Based Soft Actor-Critic," 2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2021, pp. 2028-2035.

[11] Brockman et al., (2016). OpenAI Gym. *arXiv*. https://doi.org/10.48550/arXiv.1606.01540

[12] B. Drossel, F. Schwabl, "Self-organized criticality in a forest-fire model",Physica A: Statistical Mechanics and its Applications, Volume 191, Issues 1â€“4,1992,Pages 47-50,ISSN 0378-4371, https://doi.org/10.1016/0378-4371(92)90504-J.

[13] Li et al., (2019). Curiosity-Driven Exploration for Off-Policy Reinforcement Learning Methods *. 1109-1114. 10.1109/ROBIO49542.2019.8961529.

[14] Reinforcement Learning: An Introduction Richard S. Sutton and Andrew G. Barto (chapter 6.4)

[15] Fujimoto et al.,, D. (2018). Addressing Function Approximation Error in Actor-Critic Methods. *ArXiv, abs/1802.09477*.

[16] Bellemare et al., (2017). A Distributional Perspective on Reinforcement Learning. *arXiv*. https://doi.org/10.48550/arXiv.1707.06887

[17] Mnih et al.,(2016). Asynchronous Methods for Deep Reinforcement Learning. *arXiv*. https://doi.org/10.48550/arXiv.1602.01783

[18] Schulman et al., (2015). Trust Region Policy Optimization. *arXiv*. https://doi.org/10.48550/arXiv.1502.05477

[19] Li, Y. (2017). Deep Reinforcement Learning: An Overview. *arXiv*. https://doi.org/10.48550/arXiv.1701.07274

[20] Jang et al., (2019). Q-Learning Algorithms: A Comprehensive Classification and Applications. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2941229.

[21] Schulman et al., Proximal Policy Optimization Algorithms. *arXiv*. https://doi.org/10.48550/arXiv.1707.06347

[22] Subramanian et al., (2018). Using Spatial Reinforcement Learning to Build Forest Wildfire Dynamics Models from Satellite Images. Frontiers in ICT. 5. 6. 10.3389/FICT.2018.00006.

[23] R. N. Haksar et al., "Distributed Deep Reinforcement Learning for Fighting Forest Fires with a Network of Aerial Robots," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 1067-1074, doi: 10.1109/IROS.2018.8593539.