GOLD MINE GAME DESIGN

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology

In

Computer Science and Engineering

School of Engineering and Sciences

Submitted by

Candidate Name

AP21110011357	**M. Jyothi**
AP21110011370	**I. D. Surya Teja**
AP21110011402	**Ch. Sanjeet**
AP21110011404	**S.Prasanth**

Under the Guidance of

Dr. Neeraj Kumar Sharma

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

Nov, 2023

Certificate

Date: 3-Dec-23

This is to certify that the work present in this Project entitled "**GOLD MINING**" has been carried out by **M. Jyothi, I. D. Surya Teja, Ch. Sanjeet, S. Prasanth** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

Supervisor

(Signature)

Dr. Neeraj Kumar Sharma

Designation,

Affiliation.

Co-supervisor

(Signature)

Dr. [Name]

Designation,

Affiliation.

Acknowledgements

We are grateful that we completed our Python project in the allotted time period given by our lecturer, Neeraj Kumar Sharma. Without the effort and cooperation of our group members, M. Jyothi, I. D. Surya Teja, Ch. Sanjeet, and S. Prasanth, this project cannot be finished. We also sincerely thank our lecturer for guiding us in this course and for helping us to complete this project with his guidance and encouragement. Finally, we would like to thank our friends and respondents for their help and willingness to spend some time to clarify the doubts.

Table of Contents

Certificate i
Acknowledgements iii
Table of Contents v
Abstract vii
Abbreviations ix
List of Figures xi
1. Introduction 1
1.1 PROBLEM STATEMENT: 1
1.2 PPO Algorithm 1
1.3 PPO in Miner's Gold Collection Simulation 1
2. Methodology 4
2.1 SOFTWARE REQUIREMENTS: 4
2.1.1 HARDWARE REQUIREMENTS: 4
2.1.2 ALGORITHM: 4
2.1.3 GENERATED OUTPUT: 6
3. Concluding Remarks 8
4. Future Work 10
References 11

Abstract

This report explores the implementation and visualization of a gold mining scenario utilizing the Proximal Policy Optimization (PPO) algorithm. The simulation involves a miner navigating a grid-based environment to collect gold while employing a reinforcement learning approach for decision-making.

The code simulation leverages Python with libraries such as NumPy for matrix operations and tkinter for graphical visualization. The miner's movements are guided by an algorithm resembling PPO, an efficient policy optimization method in the realm of reinforcement learning.

The algorithm iterates through the grid, choosing optimal directions to move based on available gold, and updates its policy through a mechanism resembling advantage estimation and surrogate objectives. The code incorporates visualization functions to animate the miner's actions and the gold distribution within the grid, aiding in understanding the learning process and decision-making of the miner.

This report provides an overview of the PPO algorithm, detailing its principles of policy optimization while ensuring stability through controlled policy updates. It explores the code structure, function roles, and the visualization aspect to illustrate the miner's actions in the gold mining environment.

Through this analysis, we aim to dissect the PPO-based approach in the context of a gold mining simulation, shedding light on its applicability, strengths, and potential areas for further enhancements in reinforcement learning algorithms.

Abbreviations

PPO	Proximal Policy Optimization

List of Figures

Figure 1. Output-1…………………………..………………6

Figure 2. Output-2…………………………..………………6

1. Introduction

1.1 PROBLEM STATEMENT:

Consider a n*m-dimensional gold mine. The amount of gold (in tonnes) is represented by a positive integer in each field of this mine. The miner starts out in the first column but could be in any row. The miner can only move in the following directions: diagonally up towards the right, diagonally down towards the right, or right ->, right up /, right down\) from a given cell. Find out the maximum amount of gold a miner can collect.

1.2 PPO Algorithm

In the realm of reinforcement learning, the Proximal Policy Optimization (PPO) algorithm stands as a pivotal technique for training agents to navigate complex environments and make sequential decisions. This report delves into the implementation and analysis of a gold mining simulation driven by PPO, showcasing its efficacy in guiding a miner through a grid-based landscape to maximize gold collection.

1.3 PPO in Miner's Gold Collection Simulation

The project's foundation lies in simulating a miner's quest to collect gold within a grid environment. Leveraging Python programming, specifically utilizing libraries like NumPy for efficient matrix operations and tkinter for visualization, the simulation integrates PPO as the guiding algorithm for the miner's decision-making process.

The core principle of PPO centers around optimizing policies for decision-making while ensuring stability through controlled updates. Within this simulation, PPO orchestrates the miner's movements, dynamically selecting paths to gather gold, and refining its strategies iteratively based on collected rewards.

This report unveils the intricate interplay between the PPO algorithm and the gold mining environment. It explores the nuances of PPO's policy optimization techniques and investigates their manifestation in guiding the miner's actions to maximize gold accumulation. Additionally, it scrutinizes the impact of algorithmic decisions on the efficiency of gold collection within the simulated grid.

Through this exploration, the report aims to shed light on the practical application of PPO in guiding decision-making processes within complex environments, using the gold mining simulation as an illustrative example. By dissecting the miner's actions guided by PPO, we aim to uncover insights into the algorithm's adaptability, effectiveness, and potential for optimizing decision-making strategies in similar scenarios.

2. Methodology

The gold mining simulation within a grid environment, tasking a miner to collect gold by navigating specific movements: **rightward, diagonally up-right, and diagonally down-right**. The grid represents gold tonnage in each cell. The code prompts user input for grid dimensions, initiating a visualization using tkinter. It generates a random grid with gold values. The implemented algorithm dictates the miner's moves, optimizing gold collection through Proximal Policy Optimization (PPO) concepts. This setup lays the groundwork for visualizing the miner's path and analyzing PPO's role in maximizing gold accumulation efficiency within the grid-based environment.

The library and tools used in the implementation should be detailed, including NumPy for matrix operations and tkinter for visualization. Parameter settings should be discussed, including the chosen or tuned hyperparameters and their impact on the simulation. The data generation process should detail the creation of the grid, gold, and initial conditions for the miner. The visualization approach should be described, allowing readers to understand the miner's actions and gold distribution.

2.1 SOFTWARE REQUIREMENTS:

Jupyter Notebook/ Python idle/ Visual studio

2.1.1 HARDWARE REQUIREMENTS:

8GB RAM

2.1.2 ALGORITHM:

```
1. visualize_mine_animation(gold_mine, miner_positions, canvas, rows, cols):
   a. Clear canvas
   b. Iterate through rows and columns:
      - Create rectangles to represent cells in the grid
      - Display text (gold amount) in each cell
```

```
    c. Display the miner's path as rectangles and show the collected gold in each cell
    d. Draw gridlines for the visualization
    e. Update canvas and sleep for a short duration

2. start_animation(gold_mine, canvas, total_reward_label, rows, cols):
    a. Initialize miner positions, policy, and maximum gold collected
    b. Loop through the columns of the grid:
        - Visualize the current state of the gold mine grid
        - Initialize variables for maximum gold and best move
        - Define possible movements in eight directions
        - Evaluate each movement's potential gold collection
        - Update the policy based on the best move and update miner position
        - Perform a simplified PPO-like policy update
        - Accumulate the maximum gold collected
    c. Final visualization and update total reward label

3. generate_random_integer_matrix(rows, cols, low, high):
    a. Create an empty matrix of size (rows x cols)
    b. Iterate through rows and columns, filling the matrix with random integers
within the specified range
    c. Return the generated random integer matrix
4. start():
    a. Read the number of rows and columns from user input
    b. Create the setup window for user input
    c. Set up the animation window for the gold miner simulation
    d. Generate a random gold mine grid based on user-provided rows and columns
    e. Start the animation of the gold mining scenario
```

5. GUI setup:

a. Create setup window labels and entries for rows and columns input

b. Create a start button to initiate the simulation

2.1.3 GENERATED OUTPUT:

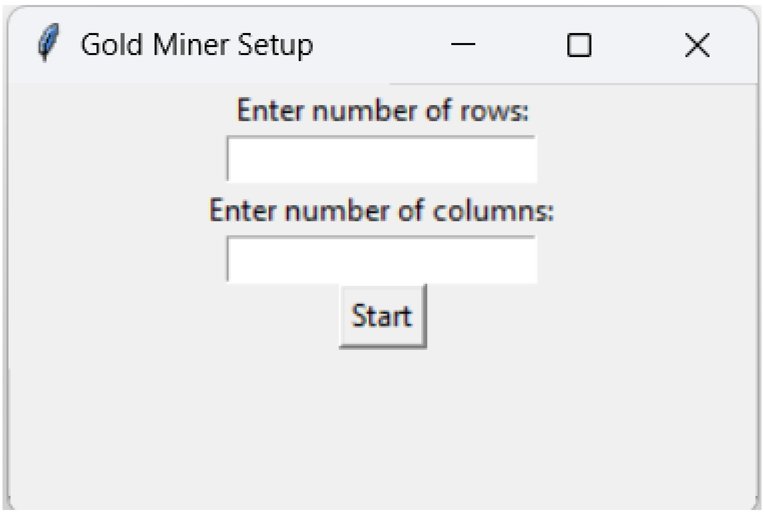
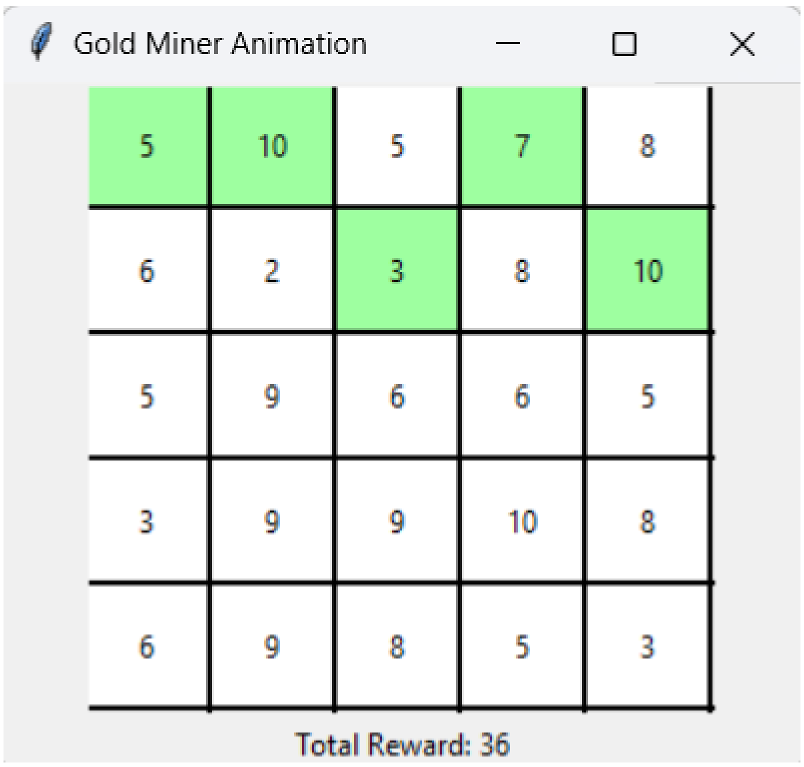


Figure 1. Output-1



🟩 5	🟩 10	5	🟩 7	8
6	2	🟩 3	8	🟩 10
5	9	6	6	5
3	9	9	10	8
6	9	8	5	3

Total Reward: 36

Figure 2. Output-2

3. Concluding Remarks

The project delved into applying Proximal Policy Optimization (PPO) within a gold mining simulation, unveiling its efficacy in reinforcement learning. Through Python and libraries like NumPy and tkinter, PPO orchestrated a miner's actions, showcasing its adaptability in maximizing gold accumulation within a grid-based environment.

By simulating the miner's quest for optimal gold collection, the project highlighted PPO's ability to iteratively refine strategies, dynamically choosing paths, and maximizing rewards. The visualization tools provided deeper insights into PPO's policy updates, aiding in understanding its impact on decision-making for gold accumulation.

This exploration not only exemplified PPO's adaptability but also set the stage for future research. It showcased the potential for advanced algorithms, improved visualizations, and expanded scenarios to enhance understanding in reinforcement learning. Overall, this project contributes to the practical applications of reinforcement learning, emphasizing PPO's effectiveness in guiding decision-making processes within complex environments and laying the foundation for continued advancements in this field.

4. Future Work

1. Enhanced Reinforcement Learning Algorithms:

- Implement more advanced reinforcement learning algorithms beyond the basic PPO approach, like DQN (Deep Q-Networks), A3C (Asynchronous Advantage Actor-Critic), or other policy gradient methods.

2. Visualizations and Analytics:

- Enhance visualizations to display metrics such as learning curves, reward distributions, or heatmaps representing the learned policies.

3. Parallelization and Performance:

- Optimize the code for performance, and explore parallelization techniques to speed up computations, especially when dealing with larger grid sizes or more complex algorithms.

4. Machine Learning Libraries:

- Implement the simulation using machine learning libraries like TensorFlow or PyTorch to leverage their capabilities for neural network-based reinforcement learning algorithms.

5. Multi-Agent Scenarios:

- Extend the simulation to accommodate multiple agents interacting within the same environment, exploring scenarios with cooperation or competition.

6. Hyperparameter Tuning and Optimization:

- Conduct systematic hyperparameter tuning or explore automated methods for optimizing algorithm parameters to enhance learning efficiency.

References

1. https://www.geeksforgeeks.org/a-brief-introduction-to-proximal-policy-optimization/
2. https://paperswithcode.com/paper/proximal-policy-optimization-algorithms
3. https://lightning.ai/pages/community/tutorial/how-to-train-reinforcement-learning-model-to-play-game-using-proximal-policy-optimization-ppo-algorithm/