Strategic Grid-Based Treasure Hunt Game

Vanshika Kumawat
Roll No.: M22MA207

Abstract

This project implements a competitive two-player grid-based game where one player employs the Minimax algorithm and the other utilizes a Deep Q-Network (DQN). The game introduces uncertainty with randomized treasures and traps, creating a balance between strategy and adaptability. The project explores reinforcement learning and adversarial search strategies in uncertain environments.

1 Introduction

This game is designed as a 2-player competition where players navigate a 7x7 grid. Treasures (+10 points) and traps (-5 points) are placed randomly, introducing uncertainty. Player strategies differ: one uses Minimax for strategic adversarial search, while the other leverages DQN to adapt and improve over time. The project highlights how AI techniques perform in non-deterministic, competitive environments.

2 Game Description

2.1 Grid Setup

The grid is initialized with treasures and traps randomly placed. Each cell has one of three states:

- **Treasure (+10):** Boosts the player's score.
- **Trap (-5):** Penalizes the player's score.
- **Neutral (0):** No effect.

2.2 Players and Strategy

- **Minimax Player:** Uses a depth-limited Minimax algorithm with alpha-beta pruning to evaluate the best possible moves.
- **DQN Player:** Implements Q-learning with an epsilon-greedy policy for balancing exploration and exploitation.

2.3 Game Loop

Players alternate turns, moving within the grid and updating their scores based on cell values. The game ends after a fixed number of turns or when all treasures are collected.

3 Implementation

3.1 Algorithms Used

Minimax Algorithm: Evaluates states and performs depth-2 search using alpha-beta pruning. Scores are assigned based on treasures and traps.
DQN: A Q-table approximates the reward for each action at any state. The Bellman equation is used for updates:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$

3.2 Graphics and Visualization

The grid and player paths are visualized using Python's `turtle` module. Treasures and traps are colored differently, and the winner is displayed on the grid upon game completion.

3.3 Outputs

Figure 1 shows an example of the game's graphical output. Player paths are highlighted, and scores are updated dynamically.

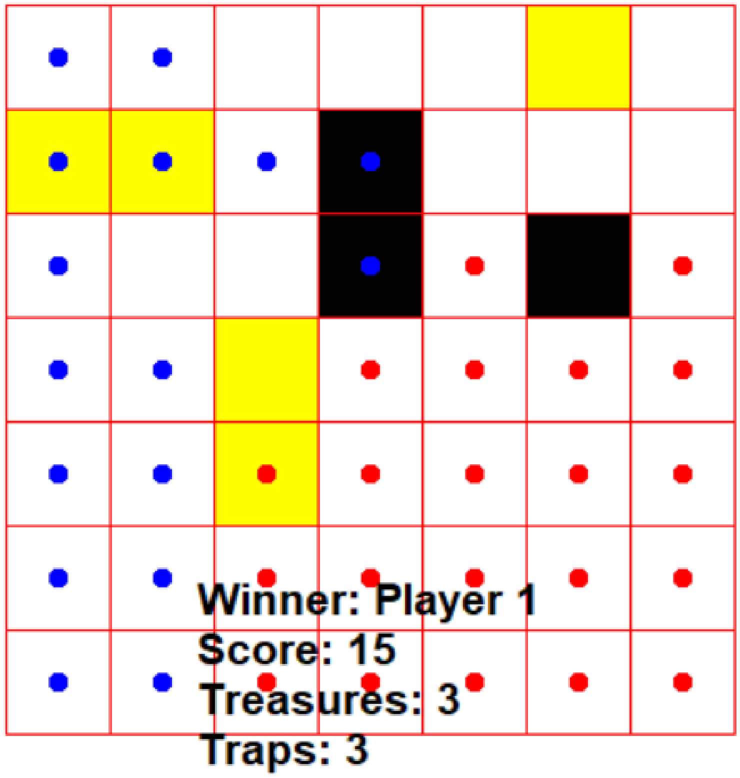

Figure 1: Sample Output of Treasure Hunt Game.

4 Results

4.1 Winner Display

The game ends with the winner and their details (score, treasures found, traps crossed) displayed on both the console and the graphical interface.

4.2 Insights

- The Minimax player performs well in predictable environments.
- The DQN player adapts to randomness, balancing exploration and exploitation.

5 Conclusion

This project demonstrates the integration of reinforcement learning and adversarial search in a competitive scenario. The randomized grid setup ensures variability, testing the strengths and weaknesses of both AI approaches.

References

1. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction.*
2. Russell, S. J., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach.*