AI Agent for the 4-in-a-Row Game

Anish Karnik, Ayush Modi, Kaushal Kothiya
Foundations of AI: Multiagent Systems, IIT Gandhinagar

November 20, 2024

Abstract

This project was undertaken as part of the course Foundations of AI: Multiagent Systems to demonstrate the Minimax algorithm's utility in game-playing AI. We developed an AI agent for the 2-player game 4-in-a-Row and implemented a web application. The webapp allows users to play against the AI at three difficulty levels: easy, medium, and hard. The AI's decision-making is based on the Minimax algorithm with alpha-beta pruning, varying depths, and heuristics. This report outlines the methodology, essential theory, and key techniques of the project.

1 Introduction

4-in-a-Row is a popular 2-player game played on a 7x6 grid where players alternate dropping discs into columns, aiming to form a horizontal, vertical, or diagonal line of four discs. This project focuses on creating an AI agent for the game, using the Minimax algorithm to simulate competitive decision-making. The primary goal is to develop an interactive platform to test AI capabilities and strategies while allowing users to challenge the agent at different difficulty levels.

2 Methodology

The AI agent's decision-making relies on the Minimax algorithm, a classic adversarial search technique enhanced with alpha-beta pruning to optimize its performance. The web application supports three difficulty levels:

Easy Difficulty

Moves are determined with a 50% probability of selecting a random move and a 50% probability of using a Minimax tree of depth 2. This level introduces variability and less optimal play, making it suitable for beginners.

Medium Difficulty

The AI agent uses the Minimax algorithm with a tree depth of 3. This provides a balanced performance, thus making the gameplay moderately challenging.

Hard Difficulty

The AI agent employs the Minimax algorithm with a tree depth of 5, augmented with the following heuristics:

- **Central Column Preference**: Prioritizes moves near the center to maximize board control.

- **Prevent Opponent's Winning Moves**: Detects when the user is close to forming a line of three and blocks it.
- **Maximize Winning Opportunities**: Identifies and pursues situations where the AI can complete a line of three.

The webapp interface allows the user to start the game, select a difficulty level, and play against the AI agent, which alternates moves with the user. The implementation uses Flask for the backend, Python for the AI logic, and HTML/CSS for the front-end design.

3 Deep Dive: Alpha-Beta Pruning in Minimax Algorithm

Alpha-beta pruning is a critical optimization applied to the Minimax algorithm to reduce the number of nodes evaluated in the search tree. It works by maintaining two bounds **Alpha**: The best score the maximizer is assured of, and **Beta**: The best score the minimizer is assured of.

During the search, branches that cannot influence the final decision are pruned, significantly reducing computation time without affecting the algorithm's correctness. For instance, in the Hard mode with a depth of 5, alpha-beta pruning allows the agent to evaluate deeper strategies more efficiently, ensuring robust decision-making under computational constraints.

This optimization was instrumental in achieving playable speeds for the AI in real-time, particularly in the Hard mode, where the search tree grows exponentially.

4 Results and Discussion

The AI agent demonstrates consistent and competitive performance across all difficulty levels:

- In the **Easy** mode, the AI often makes suboptimal moves due to the random component, leading to a high win rate for users.
- In the **Medium** mode, the AI provides a balanced challenge, making strategic moves while occasionally overlooking complex threats.
- In the **Hard** mode, the AI showcases strong gameplay by efficiently blocking the user's advances and prioritizing winning moves, making it significantly difficult to beat.

The heuristic enhancements in Hard mode significantly improve the AI's ability to simulate human-like strategic reasoning. However, the computational complexity increases with tree depth, leading to longer decision times in Hard mode, especially in the late game.

5 Conclusion

This project successfully demonstrates the application of the Minimax algorithm with alpha-beta pruning to develop an AI agent for 4-in-a-Row. The implementation highlights the importance of heuristic design and computational trade-offs in AI systems. The project provides a foundational understanding of adversarial search techniques and their practical applications in multi-agent systems.

6 References

1. http://blog.gamesolver.org/solving-connect-four/08-iterative-deepening/
2. https://www.youtube.com/watch?v=IpTFe0H52JM