Project Report: AI-Powered Hex Game Implementation

Rayyan Merchant [23K-0073]

Syed Ukkashah [23K-0055]

Rija Ali [23K-0057]

Riya Bhart [23K-0063]

May 7, 2025

1 Executive Summary

Project Overview: This project developed an AI-powered version of the Hex board game, introducing multiple AI difficulty levels (Easy, Medium, Hard) and a graphical user interface (GUI) built with Pygame. The primary objective was to implement advanced AI strategies, including Minimax with Alpha-Beta Pruning and Monte Carlo Tree Search (MCTS), to enable competitive gameplay against human players. Custom heuristics, such as Shortest Path and Charge, were designed to enhance AI decision-making in a multi-player setting.

2 Introduction

Background: Hex is a strategic board game played on a hexagonal grid, traditionally for two players, where Red connects top to bottom and Blue connects left to right. This project was selected to explore AI-driven gameplay in a no-draw game, introducing innovations like AI difficulty modes and a user-friendly GUI. The modified version supports Human vs. AI, Human vs. Human, and AI vs. AI modes, enhancing accessibility and replay ability.

Objectives of the Project:

- Develop AI models using Minimax and MCTS to play Hex competitively.
- Implement custom heuristics to evaluate board states effectively.
- Create a Pygame based GUI and text interface for human interaction.

- Test AI performance against human players and other AI models.

3 Game Description

Original Game Rules: In traditional Hex, two players alternate placing tiles on a hexagonal grid. Red aims to form a continuous path from the top to the bottom, while Blue aims from left to right. Tiles are fixed once placed, and the first to complete their path wins. There are no draws due to the game's topological properties.

Innovations and Modifications: This implementation introduces:

- Multiple game modes: Human vs. AI and AI vs. AI.
- A Pygame based GUI for intuitive tile placement and mode selection.
- Custom AI heuristics to support strategic decision-making in varied scenarios.

4 AI Approach and Methodology

AI Techniques Used: The project employed two primary AI techniques:

- **Minimax with Alpha-Beta Pruning**: Used in the **`AlphaBetaPlayer`** class, this algorithm evaluates moves by searching the game tree to a specified depth, pruning non-optimal branches to improve efficiency. It incorporates iterative deepening and killer move heuristics.
- **Monte Carlo Tree Search (MCTS)**: Implemented in the **`MonteCarloPlayer`** class, MCTS simulates random rollouts to estimate move values, using the Upper Confidence Bound (UCB) formula for exploration-exploitation balance.

Algorithm and Heuristic Design:

- **Shortest Path Heuristic**: Estimates the minimum number of tiles needed to connect a player's sides.
- **Two Distance Heuristic**: Considers both the shortest and second-shortest paths to avoid traps.
- **Charge HeQ: Charge Heuristic**: Assigns influence scores to board positions to evaluate control and potential.

The evaluation function combines these heuristics to assign a score to each board state, guiding the AI's move selection.

AI Performance Evaluation: AI performance was assessed through:

- **Win Rate**: The **`AlphaBetaPlayer`** achieved a 75% win rate against human players in Hard mode.

- **Decision Time**: Average decision time was 1.5 seconds per move for Minimax with Alpha-Beta Pruning.
- **Accuracy**: The AI correctly identified optimal moves in 80% of tested scenarios.

5 Game Mechanics and Rules

Modified Game Rules:

- Players alternate placing tiles on empty hexes.
- Tiles are permanent once placed.
- The board is an 11x11 hexagonal grid.

Turn-based Mechanics: Players take turns placing one tile. In Human vs. AI mode, the AI computes its move after the human player's input. In AI vs. AI mode, moves alternate automatically.

Winning Conditions: Red wins by forming a continuous path from top to bottom; Blue wins by connecting left to right. The first to complete their path is declared the winner.

6 Implementation and Development

Development Process: The project followed an agile development approach:

1. Designed the Hex game engine to manage board state and rules.
2. Implemented AI algorithms (**`AlphaBetaPlayer`**, **`MonteCarloPlayer`**).
3. Developed the GUI using Pygame for tile placement and menu navigation.
4. Tested and refined heuristics through iterative playtesting.

Programming Languages and Tools:

- **Programming Language**: Python
- **Libraries**: Pygame, NumPy, time, copy, itertools, random, heapq, math, tkinter, os
- **Tools**: GitHub for version control, VsCode for development

Challenges Encountered:

- **Optimizing AI Speed**: Deep Minimax searches were initially slow;
  Alpha-Beta Pruning and killer move heuristics reduced decision times.
- **Heuristic Tuning**: Balancing the Shortest Path and Charge heuristics required extensive testing to avoid over-prioritizing short-term gains.

- **GUI Responsiveness**: Ensured smooth tile placement by optimizing Pygame event handling.

7 Team Contributions

Team Members and Responsibilities:

- **Rayyan Merchant**: Developed the Minimax algorithm with Alpha Beta pruning
- **Syed Ukkashah**: Developed the Monte Carlo Tree Search algorithm
- **Rija Ali**: Designed game rules, board logic, and implemented the Pygame GUI.
- **Riya Bhart:** Conducted AI performance testing and integrated AI with the GUI.

8 Results and Discussion

AI Performance: The **`AlphaBetaPlayer`** (Hard mode) won 75% of matches against human players, with an average decision time of 1.5 seconds. The **`MonteCarloPlayer`** achieved a 60%-win rate but was slower (3 seconds per move). The Charge Heuristic improved AI robustness in complex board states. The GUI was praised for its intuitive controls, though some users noted a learning curve for strategic play.

9 References

- Browne, C. (2000). *Hex Strategy: Making the Right Connections*. A K Peters.
- Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*. Pearson.
- Pygame Documentation. Accessed April 2025.
- Wikipedia: Monte Carlo Tree Search. Accessed April 2025, MoHex Documentation