Phase 2: Mid-Project Status Report - Comparing AI Agents for Connect 4

Team Matrix: Greeshma Chanduri, Pravalika Kamineni, Kiranmayi Modugu,
GitHub Repository: https://github.com/Greeshma-DS/Team-Matrix.git

- **Research Question**

We hypothesis which AI algorithm Minimax with Alpha-Beta Pruning or Monte Carlo Tree Search, performs better in Connect 4 in terms of win rate, decision time, and strategic decision-making?

- **Introduction**

Connect 4 is a two-player strategy game where players alternate dropping colored discs into a 6x7 grid, aiming to connect four discs vertically, horizontally, or diagonally. This project develops and compares two AI agents for Connect 4: Minimax with Alpha-Beta Pruning and Monte Carlo Tree Search (MCTS). The comparison focuses on win rate, decision time, and strategic behavior, providing insights into the strengths and weaknesses of deterministic and probabilistic AI approaches. This work is significant for understanding how AI techniques perform in zero-sum games, with applications in game theory, decision-making systems, and AI education.

- **Related Literature**

Allis, L. V. (1988): Used Minimax with hand-crafted heuristics for Connect 4. We differ by comparing the custom Minimax heuristic with MCTS.

Browne, C., et al. (2012): Surveyed MCTS for complex games like Go. We apply MCTS to Connect 4, comparing it with Minimax using custom UCT implementation.

Silver, D., et al. (2016): Combined MCTS with neural networks for Go. We use pure MCTS for Connect 4, compared against Minimax.

Our work uniquely compares Minimax and MCTS in Connect 4 using heuristic and Pygame GUI.

- **Approach and Implementation**

We implemented two algorithms, Minimax with Alpha-Beta Pruning and Monte Carlo Tree Search to compete against each other in Connect 4 using a Pygame GUI.

Minimax (Alpha-Beta Pruning): A depth-limited (default depth = 3, adjustable) algorithm using a custom heuristic that prioritizes center control, blocks threats, and rewards three-in-a-row patterns. Alpha-Beta Pruning improves efficiency. The game state is managed using a 6x7 NumPy board.

Monte Carlo Tree Search (MCTS): Runs 50 simulations per move (adjustable) using the UCT formula. Simulations aim to block opponent wins and build connections. Shares the same 6x7 board structure as Minimax.

Pygame GUI: Red (Minimax) and Yellow (MCTS) tokens animate real-time moves. The interface shows the move times and the winner. Upcoming features include difficulty sliders and board size options (5x6, 6x7, 7x8).

Game Flow: The game alternates between the two AIs, checks for a winner after each move, and logs move times. At the end, it displays the total and average decision time for each AI and announces the winner.

- **Achievements**

Fully implemented Minimax with Alpha-Beta Pruning and MCTS algorithms.

Developed a functional Pygame GUI showing Minimax vs. MCTS gameplay.

- **Challenges and Workarounds**

Challenge: MCTS was slow (initially 6+ seconds/move).
Solution: Reduced simulations to 50 per move, improving decision time to ~3.34 seconds. Plan to experiment with simulation optimization techniques (e.g., prioritizing high-value moves).

Challenge: Initial Minimax implementation without Alpha-Beta Pruning was computationally expensive.
Solution: Added Alpha-Beta Pruning, reducing average move time to 0.27 seconds. Further tuned the heuristic to prioritize central columns, improving strategic play.

Challenge: GUI development was delayed due to unfamiliarity with Pygame.
Solution: Studied Pygame documentation and implemented a basic GUI for real-time visualization. Currently refining it to include difficulty slides and board size options.

Challenge: MCTS win rate was lower than expected (1 win in 10 games).
Solution: Adjusted UCT constant to balance exploration and exploitation. Plan to increase simulations and refine simulation logic for better performance.

- **Preliminary Results**

Decision Time: Minimax (0.27 sec/move) is significantly faster than MCTS (3.34 sec/move), supporting our hypothesis.

Win Rate: MCTS won 1/10 games, suggesting Minimax is currently stronger, but further tuning of MCTS is needed.

Pruning Efficiency: Minimax prunes ~60% of branches, optimizing performance.

Simulation Efficiency: MCTS performance improves with more simulations but increases decision time, indicating a trade-off.

- **Next Steps**

Complete GUI with board size options (e.g., 5x6 or 7x8 boards).

Conduct games to robustly compare win rates and decision times.

Finalize the project report and demo video for Phase 3.