Program structures and algorithms
Project Milestone Report

Team Members:
Sai Lalith Pulluri (NUID: 002056829)
Chinnasurya Prasad Vulavala (NUID:002056815)
Github url: https://github.com/chinnasuryaprasad1612/INFO-6205_Project

Overview:

Tic-Tac-Toe is a well-known two-player strategy game played on a 3x3 grid. The goal is to align three of one's own symbols (X or O) in a row, column, or diagonal. The project implements an intelligent agent that plays Tic-Tac-Toe using Monte Carlo Tree Search (MCTS), a search algorithm known for balancing exploration and exploitation in large decision spaces.

Implementation Details

The implementation begins with a State representing the current game board and the player whose turn it is. A TicTacToeNode wraps this state for tree-based reasoning. The MCTS algorithm drives move selection for the AI player through the following four phases:

1. **Selection**: Starting from the root node, child nodes are recursively selected using the Upper Confidence Bound (UCB1) formula to balance the tradeoff between exploring new moves and exploiting promising ones.
2. **Expansion**: Once a leaf node is reached (a node without children), it is expanded by generating all valid moves for the current player and adding corresponding child states.
3. **Simulation (Playout)**: A random or heuristic-guided simulation is run from the expanded state until a terminal game outcome is reached (win, loss, or draw).
4. **Backpropagation**: The result of the simulation is propagated back up the tree, incrementing visit counts and win statistics along the path. We assign a score of 2 for a win and 1 for a draw.

Heuristic Enhancements

To improve performance, we integrated a simple **heuristic-based move selection** strategy that is used during simulations:

- **Win First**: If a move immediately wins the game, it is selected.
- **Block Opponent**: If the opponent can win in their next move, block it.
- **Prefer Center**: The center tile (1,1) is prioritized, as it is strategically strong.
- **Fallback Random**: If none of the above apply, a random move is chosen.

This hybrid approach helps the simulation phase converge toward better quality decisions and makes the MCTS more effective with fewer iterations.

Metrics Collected

We measured and recorded the following metrics across multiple test runs and iteration counts:

- **Total Games**: Number of games played for each iteration count.
- **X Wins / O Wins / Draws**: Number of outcomes for each player.
- **Win Rates**: Percentage of wins for X and O players.
- **Average Game Length**: Number of moves until game termination.
- **Total Simulations**: Total playouts executed across games.
- **Average Simulations per Game**: Simulations run per game.
- **Time (ms)**: Execution time for each configuration.

Observations & Analysis

1. **More Iterations = Better Strategy**: As we increase the number of MCTS iterations, the decision quality improves significantly. This is reflected in more balanced win rates between X and O and fewer random errors in gameplay.
2. **X Bias at Low Iterations**: When the number of iterations is low (e.g., 50–200), player X tends to dominate. This is expected because X plays first and random playouts don't allow O to counter effectively.
3. **Heuristics Improve Learning**: Introducing heuristics during simulations significantly improves win rates for the second player (O), reducing bias toward X. This shows that even basic domain knowledge helps guide MCTS to better results.
4. **Draws Increase with Iterations**: As simulations grow deeper, games tend to result in more draws, which is consistent with optimal Tic-Tac-Toe play where perfect strategies from both sides lead to a draw.
5. **Diminishing Returns**: After a certain iteration threshold (e.g., 3200+), improvement in win rates slows, while computation time increases sharply. This highlights a tradeoff between performance and compute.

Conclusion

Our MCTS implementation for Tic-Tac-Toe demonstrates how a combination of probabilistic search and simple heuristics can produce strong gameplay even in a small domain. It showcases the importance of balancing computation with strategy depth, and how guiding MCTS with even lightweight heuristics can enhance its decision-making under limited resources.