Meta Tic-Tac-Toe with AI Agent

Amey Rangari, Divyanshu Borana, Naman Varshney,
Vedant Barde, Vineet Kumar Tatu

Aim

The aim of this project is to develop a Meta Tic-Tac-Toe game with an AI agent capable of playing optimally. The AI agent progresses from a simple random move strategy to more sophisticated algorithms, including Minimax, Alpha-Beta Pruning, and Monte Carlo Search, to provide an engaging and challenging experience for players.

Code Explanation

The game is implemented using Python and the Tkinter library for the graphical user interface. The Meta Tic-Tac-Toe game board is 4-dimensional, consisting of 9 smaller 3x3 grids, making it more complex than the traditional Tic-Tac-Toe. Each small grid represents a smaller game of Tic-Tac-Toe, and the objective is to win the larger grid by winning the smaller grids.

Class Definition: The core of the game is the `MetaTicTacToe` class, which encapsulates the game logic and the user interface. It initializes the game board as a 4D array and handles user input through button presses. The state of each grid (whether it is won or still open) is tracked using the `grid_status` and `large_grid_status` arrays.

Move Logic: The game ensures that players must play in specific grids based on the last move made. The user is assigned 'X' and the AI is 'O'. After each move, the game checks for a winner in the small grid and then in the large grid. If a player wins, a message box announces the winner and the game is reset.

Game Flow: The game alternates between the user and the AI. The AI initially selects moves randomly, but the implementation of the Minimax and Alpha-Beta Pruning algorithms later allows the AI to make more strategic decisions. The AI can either play in the grid specified by the last user move or choose any open grid if the target grid is closed.

Constraints of Meta Tic-Tac-Toe and CSP Implementation

Meta Tic-Tac-Toe introduces several complexities compared to the standard Tic-Tac-Toe:

- **4D Board:** The game board consists of a 3x3 grid of 3x3 sub-grids, creating a 4D structure that increases the search space.
- **Grid Constraints:** A move in one of the smaller grids dictates the grid in which the opponent must play. This constraint adds a layer of strategic depth, making it more challenging to develop an AI that plays optimally.

- **Large Grid Winner:** The overall winner is determined by winning the majority of the smaller grids within the larger grid, which requires additional checks for winning conditions in both the smaller and larger grids.

To handle these constraints and ensure the game rules are respected, the concept of a *Constraint Satisfaction Problem (CSP)* is applied. CSP involves finding values for variables that satisfy a number of constraints. In the case of Meta Tic-Tac-Toe, the variables are the open cells on the board, and the constraints are:

- A player can only move in an open cell.
- A player must move in a grid specified by the previous move.
- A grid is closed once it is won by a player.

The game logic checks for valid moves by considering these constraints.

AI Agent

The AI agent was implemented in stages to provide a progressively more intelligent gameplay experience:

- **Random Move:** The simplest AI strategy, where the agent selects any valid move at random. This approach ensures basic functionality but lacks any strategic depth.
- **Minimax Algorithm:** This algorithm explores all possible game states to determine the optimal move by minimizing potential losses and maximizing gains. However, due to the 4D nature of Meta Tic-Tac-Toe, the computational complexity makes this method impractical for real-time gameplay.
- **Alpha-Beta Pruning:** By pruning branches of the game tree that do not need exploration, this optimization improves the efficiency of the Minimax algorithm. While it significantly reduces computation time, it still struggles with the game's large search space and may not always make optimal moves due to the inherent complexity of the game.
- **Monte Carlo Tree Search Algorithm:** It is an advanced method that can be implemented as further scope of improvement. This approach uses random simulations to predict the best move by analyzing the likelihood of winning from different positions. It is more adaptable to complex games like Meta Tic-Tac-Toe and offers a balance between strategic depth and computational efficiency.

Conclusion

The Meta Tic-Tac-Toe game with an AI agent is an innovative extension of the classic Tic-Tac-Toe. The game introduces strategic depth through its multi-level grid structure and constrained move logic. The AI agent progresses from a basic random move strategy to more sophisticated approaches, including Minimax, and Alpha-Beta Pruning, to provide an engaging and challenging experience for players. Future improvements could involve Monte Carlo Tree Search and other optimizations in the AI's decision-making processes and incorporating machine learning techniques to allow the AI to adapt and learn from player behavior.