Lab Report: Implementing an AI Algorithm for Tic-Tac-Toe

Introduction:

Tic-Tac-Toe is a classic two-player game that involves placing Xs and Os on a 3x3 grid, with the objective of forming a winning pattern. This project aims to implement an AI algorithm to play Tic-Tac-Toe intelligently against a human opponent. The importance of this project lies in showcasing the application of artificial intelligence in simple board games, demonstrating how AI can make strategic decisions, and providing an enjoyable gaming experience.

Algorithm Description:

The AI algorithm used in this project is the Minimax algorithm. Minimax is a decision-making algorithm that systematically explores the game tree, considering all possible moves and their consequences. It evaluates the board state to find the best move, maximizing its chances of winning and minimizing the opponent's chances.

Strengths of Minimax:

- Guarantees optimal play
- Suitable for games with a finite and manageable number of states.
- Adaptable to various game environments.

Weaknesses of Minimax:

- Computationally expensive for larger game boards.
- Cannot handle games with incomplete information.
- Cannot be used in more complex and longer games since it will not accurately predict the best future.

In comparison to other algorithms, such as Monte Carlo Tree Search (MCTS), the Minimax algorithm with alpha-beta pruning is well-suited for the game of Tic-Tac-Toe due to its deterministic nature and relatively small state space. While Minimax explores the entire game tree, MCTS leverages a stochastic approach by simulating random playouts, making it more applicable to games with larger state spaces and more complex decision-making.

Example Scenario:

Let's walk through a scenario where the AI algorithm (Player_2) makes a move:

```
- X O
- X -
O - -
```

The AI evaluates all possible moves and their consequences to select the best one. The minimax function is used, starting with the AI's ('O') turn. It explores the game tree and assigns a score to each leaf node.

The AI (Player_2) simulates making a move at (0, 0), and the board becomes:

```
O X O
- X -
O - -
```

The AI then runs the minimax function recursively, and the score for this move is calculated based on the worst-case scenario.

The AI simulates making a move at (1, 2), and the board becomes:

```
- X O
- X O
```

O - -

Again, the minimax function is used, and a score is assigned. The AI continues this process for all available moves.

The AI selects the move with the highest score (in this case, at (0, 0)) and updates the game board:

O X O

- X -

O - -

This strategy allows the AI to make informed decisions and strive for victory.

Conclusion:

In this project, we successfully implemented an AI algorithm for playing Tic-Tac-Toe using the Minimax algorithm. We discussed the strengths and weaknesses of the algorithm and compared it to other potential approaches. The Minimax algorithm demonstrates the AI's ability to make strategic decisions in a simple game. Future improvements could include implementing more sophisticated AI algorithms for more complex games, enhancing the user interface, and optimizing the algorithm's performance, such as the way we added alpha-beta pruning into the algorithm.