Project Report: Development of 4-in-a-Line Game AI Using Alpha-Beta Pruning

Introduction:

This report outlines the development of an AI for the game "4-in-a-Line," an 8x8 board game where players alternate placing pieces, aiming to form a line of four consecutive pieces. The project leverages the alpha-beta pruning algorithm in Python, with a focus on achieving efficient and intelligent game play.

Implementation Overview:

The core of the AI is based on the minimax algorithm with alpha-beta pruning. The game board is represented as an 8x8 NumPy array, with different integer values indicating the presence of a human player's piece, the computer's piece, or an empty cell. The game loop allows for alternating turns between the human player and the AI.

Key Functions and Algorithm:

1. **Board Evaluation**:
    - **Winning Condition**: Checks for a sequence of four identical, non-zero elements in rows or columns.
    - **Dynamic Scoring**: Adjusts the scoring strategy based on the game's progress. In the endgame, the AI prioritizes immediate wins or blocks, while in the midgame, it focuses on strategic positioning.
    - **Positional Value**: Assigns higher scores to pieces placed in central and strategically advantageous positions.
2. **Minimax Algorithm with Alpha-Beta Pruning**:
    - **Depth-Limited Search**: The algorithm searches up to a certain depth, balancing between search depth and computation time.
    - **Alpha-Beta Pruning**: Optimizes the minimax search, reducing the number of nodes evaluated.

Challenges and Solutions:

1. **Detecting Winning Conditions**:

- **Initial Challenge**: The AI initially struggled to detect winning conditions, leading to suboptimal moves.
- **Solution**: Enhanced the winning condition checks to accurately evaluate rows, columns, and update the evaluation function accordingly.

2. **Dynamic Scoring System**:
    - **Initial Challenge**: The AI's static scoring system failed to adapt to different game stages.
    - **Solution**: Implemented a dynamic scoring system that changes strategies based on the number of remaining moves, giving higher priority to blocking or winning moves as the game progresses.
3. **Performance Optimization**:
    - **Initial Challenge**: The algorithm's performance was initially subpar, particularly in deeper searches.
    - **Solution**: Fine-tuned the depth of the search and implemented more efficient board evaluation methods. Additionally, incorporated alpha-beta pruning to significantly reduce the search space.

Conclusions and Future Work:

The developed AI demonstrates competent gameplay in "4-in-a-Line," effectively balancing between strategic depth and computational efficiency. Future improvements could include machine learning techniques for adaptive gameplay, deeper search algorithms, and more complex evaluation functions that consider potential future states.

The project provided valuable insights into game AI development, particularly in optimizing search algorithms and dynamically adjusting strategies based on the game state. These learnings can be applied to more complex games and AI applications.