Institute of Information Technology
University of Dhaka

Topic: Minichess

Artificial Intelligence (CSE-604)

Submitted by:

Nusrat Jahan Lia: 1306

Labib Muntasir: 1319

Tasnim Mahfuz Nafis: 1327

Submitted to:

Dr. Ahmedul Kabir
Associate Professor
Institute of Information Technology
University of Dhaka

Introduction

In this project, we implemented a chess AI capable of making optimal moves using the Minimax algorithm enhanced with Alpha-Beta pruning. The AI evaluates the game tree to determine the best move by searching through potential game states. The program combines several computational strategies, including search algorithms, evaluation heuristics, and pruning techniques, to efficiently handle the complexity of chess decision-making.

Game Tree Search

The Minimax algorithm is used to explore the game tree systematically. At each level, the AI alternates between maximizing and minimizing players, representing White and Black, respectively. The depth of search is limited to a predefined `DEPTH` value (e.g., 4), allowing the AI to consider multiple moves ahead while avoiding excessive computational overhead.

Alpha-Beta pruning is integrated to eliminate irrelevant branches during the search, improving efficiency by discarding moves that cannot influence the final decision. The algorithm evaluates potential moves and stops further exploration when a move's value is determined to be worse than the previously established best move.

Implementation of Minimax Algorithm

The Minimax algorithm determines the best possible move by recursively evaluating game states. The AI:

- **Maximizes** the score for White (maximizing player).
- **Minimizes** the score for Black (minimizing player). The algorithm uses a helper function, `scoreBoard`, to assess the value of board states and propagates these values back up the tree. Alpha-Beta pruning ensures the algorithm evaluates only critical branches.

Evaluation Function

The evaluation function (`scoreBoard`) determines the goodness of a board state by:

1. **Material Value**: The value of each piece is based on its importance (e.g., Queen = 10, Knight = 7).
2. **Positional Value**: Each piece is assigned a score based on its location, leveraging predefined tables that reflect strategic positions.

3. **Game Outcomes**: Checkmate is assigned a maximum score (`CHECKMATE`), while stalemate is neutral (`STALEMATE`).

This combination ensures the AI prioritizes both material and positional advantages, enhancing decision-making quality.

Early Stopping Mechanism

The algorithm features an early stopping mechanism by terminating the search when the maximum depth is reached or when the remaining branches are guaranteed to be irrelevant due to Alpha-Beta pruning. This allows the program to return the best known move without completing an exhaustive search of the game tree.

References:

[1] engines - What is an accurate way to evaluate chess positions? - Chess Stack Exchange