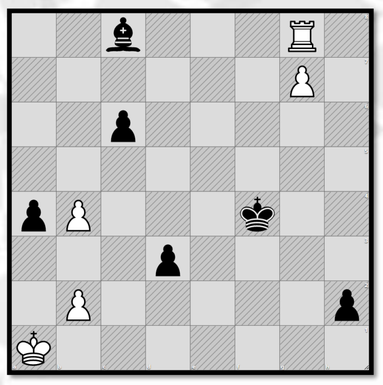
Chess Algorithm Comparison

Background



- **Chess Endgame**

Classically the stage of the game where there are a few pieces left. There is no clear distinction between middlegame and endgame. The characteristics of an endgame are usually when players are attempting to promote their pawns and utilizing their king to attack.

- **Heuristic algorithm**

Essentially is an approximation of a solution that is obtained efficiently. Often used when an exact solution cannot be found in a reasonable time through classical means.

- **Minimax**

Originally formulated for two player zero-sum games. Utilizes an evaluation function to determine which player has the advantage, high value favoring the first player, vice versa. Explores the search space for k depth while maximizing the value on player one's turn and minimizing on player two's turn. This is done by assuming the opposing player moves optimally, selecting the lowest value on every turn. The move that leads to the highest/best outcome at k depth is selected.

- **Monte-Carlo Tree Search**

Applicable for incomplete information games. It utilizes an expected outcome model instead of an evaluation function. MCTS obtains the most promising move by expanding the search tree based on random sampling of the search space. Each turn, k number of games are played with random moves until they end and the move with the highest win probability is selected.

Motivation

Blunders in the endgame are what causes an advantaged player to draw or lose instead of winning and vice versa. Because the search space is smaller in the endgame, I am curious to how heuristics perform when there is little room for error. Studying performance in endgames can improve decision making and analysis skills.

Aim & Objectives

Analysis and comparison of heuristic algorithms applied in chess endgames.

1. Research data structures and programming language required to implement the algorithms.
2. Research and implement the heuristic algorithms to a chess board program.
3. Compare and justify the best heuristic algorithm by time and space complexity performance**.**

Methodology

1. Research and create pseudo-code of traditional algorithms for chess (Minimax) and alternative algorithms (Monte-Carlo Tree Search)
2. Fork a basic C++ chess program and adapt the game to allow for algorithms to play
3. Import a set of endgame scenarios of similar difficulty
4. Implement algorithms and time and resources monitoring
5. Analyze the time and space complexity and conclude results
   a. Time per move vs move number
   b. Memory per move vs move number
   c. Expected number of moves to solution vs obtained
   d. Average win rate of playing both sides for each endgame

Future Development

Creating a clear criteria for the opening, middlegame, and endgame will allow a better comparison of performance. Then playing the game from the start will allow analysis between different stages. Adding understanding of the strengths and weaknesses of the algorithms and what the optimal playstyle is at each stage. Comparison with a retrograde analysis solution can show the efficiency versus accuracy trade off.

Nicholas Ng 170470865 – Supervisor Sergiy Bogomolov