Checkmatr Written Report

Andrew Mascillaro, Nabih Estefan

Overview

Chess has simple rules, but the game space is extremely large. With the strength of our current computers, we aren't even close to solving the game of chess by brute force. As a fallback, engines use a heuristic to try to find the best moves for a given player, which basically traverses a "game tree" (tree of possible moves for both players) and quantitatively evaluates the resulting position. This project primarily focuses on how to efficiently traverse such a game tree and create an effective heuristic to maximize the strength of a chess engine. To test the engine's strength, we have scripts that allow the engine to play agains humans or Stockfish (a top open source chess engine) at different rating levels. Our engine outperformed our expectations, and has won games against the 2200 rated Stockfish engine, which basically means this engine is strong enough to be a titled player in the chess world.

Background Information

Game Tree Traversal

Minimax

The most basic tree traversal is a minimax algorithm. This algorithm assumes that white and black will play their optimal moves and tries to find the most favorable move *assuming that both players play perfectly.* More information can be found **here**. While the space complexity of this is negligible, the time complexity is m^p, where each player has roughly m moves per ply p (a ply is half of a move or just one turn for one player). The proof for this is explained in the linked research.

Alpha Beta Pruning

A standard minimax traversal can be very slow, so shortcuts are very useful.

ProbCut

Click **the link**

Positional and Material Heuristic

Endgame Tablebases as a Heuristic

Our Implementation