Parallelized Min-Max Chess Engine (PM)*

Feitong Qiao (flq2101), Yuanyuting Wang (yw3241)

November 2021

1 Problem Formulation

In this project, we propose a parallelized implementation of a simple chess engine. The classical approach to implementing a chess engine is by using a min-max algorithm that explores the tree of all possible moves up to a certain depth. For each move, we assign a "strength" index to the move that's determined by the chess pieces remaining on the board by the end of the move. We can then select the "strongest" move for each player given each discrete state of the board by traversing through the tree.

It is obvious that this basic strategy has some serious flaws in terms of time complexity. Consequently, the alpha-beta pruning algorithm is commonly used to minimize the number of nodes evaluated in the min-max strategy.[1] The essential idea is that, as we traverse through the tree, we keep track of the maximal and minimal "strength"s (α and β) of the moves so far; and if a node proves to produce sub-optimal values compared to α and β, we stop evaluating its descendant nodes but move to the next peer node instead. This significantly reduces the number of nodes we need to traverse, especially in the case where the first few nodes are heuristically superior compared to the rest.

However, the basic alpha-beta pruning algorithm is sequential in nature, as the strategy constantly depends on the α and β values generated thus far. The challenge then becomes bringing in an aspect of parallelism to further optimize this program, while retaining and accommodating the basic concepts of min-max and alpha-beta pruning algorithms.

2 Implementation Plan

To parallelize the process of exploring the entire tree, the simplest approach would be to evaluate each sub-tree (or move) on a spark. In other words, this

*If it doesn't work so well, this name refers to Parallel Min-Max; otherwise, please call it Parallel Master (inspired by Grandmaster (GM) in the international chess ranking system)

[1] https://www.freecodecamp.org/news/simple-chess-ai-step-by-step-1d55a9266977/

would be a parallel version of the original minimax algorithm. However, one of the main problem with this approach is that this does not do any pruning.

To incorporate the ideas of alpha-beta pruning, we are proposing to do a sequential evaluation on the left-most branch in order to acquire a baseline alpha, α'. With α', we can then explore the rest of the moves all in parallel and prune using this baseline alpha.

This approach is inspired by the Principle Variation Splitting, an advance and more chess-specific parallel algorithm.[2] Our hope is that this baseline alpha can be on average good enough to be used in pruning. The actual performance will need to be evaluated empirically.

3 Deliverables

1. A chess engine program that is interactive (we can play against this engine) and parallel (effectively utilises the available cores);

2. Comparisons with traditional techniques like minimax (no pruning) and minimax (with alpha-beta pruning) using performance benchmarks.

[2] http://worldcomp-proceedings.com/proc/p2011/SER3956.pdf