Alexander Khoma, Jakwanul Safin
ak4109, js5142

11/22/2019

Project Proposal

Create an algorithm to decide on the best moves in a game of suicide chess. Sucide chess is a variant of chess where the player's goal is to lose all their pieces. The rules are the same as in normal chess except that if a player is able to take a piece they are required to do so (importantly however if there are multiple moves that take pieces any one of those moves can be chosen).

See https://en.wikipedia.org/wiki/Losing_chess for the precise ruleset.

Algorithm Idea:

The game can be analyzed similar to any 2 player zero-sum game. The possible moves at each step can be represented as a tree that can be searched using the minimax algorithm. Optimizations can be made along the way to improve performance (iterative deepening, alpha-beta pruning, etc.)
The minimax algorithm is sequential, so there isn't a clear way to parallelize searching different branches, but popular chess engines that use parallel search usually have a transposition table that is shared by the different threads. (A transposition table is a common chess engine feature, it's cache that stores a position, depth-searched-to, and evaluation, so that the same position, which can be reached through different sequences of moves, is never evaluated twice).

Challenges:

Just as in a standard chess game, the search tree increases exponentially so it is practically impossible to find an exact solution. However there are many methods available for finding approximate solutions as described above. Additionally due to the side-effect free design of Haskell random simulations of moves would be a challenge to implement. Standard heuristics used by humans in standard chess, such as board control and piece values, do not extend well to suicide chess, as the game is highly senstive to the state.