AI Final Project Report Winter 2022

Ling Fei Zhang, 260985358
Brandon Ma, 260983550

April 5, 2022

1 Motivation

1.1 Approach

March 25th 2022 Today, we played a few games against each other. We believe that in order to design a good AI agent, we must first understand the fundamentals of the game as well as to develop a few basic strategies. Our first instinct is to avoid corners as we can easily get boxed in. This means that generally speaking, we would like to move towards the center.

2 Theoretical Approach

2.1 Basic Strategies

First, we need to implement some basic strategies. We implemented a Depth First Search in order to enumerate all the possible moves given a specific position, a chess board and a maximum number of steps. Below we have some other basic strategies:

1. At every turn, our agent checks if it can win in 1 move.
2. Trying to box in the opponent as much as possible.
3. Use our agent as a wall to limit the opponent's movement.
4. Make an heursitic function that is balanced between centralization, not getting trapped and cornerning the oppponent.

2.2 More Complexe Functions

We also implemented some other functions that are more complexe, such as minimax and alpha-beta pruning. The goal is to use a heuristic function to give a score to each move. This implies that the agent can decide which move has better winning chances relative to others. We also have functions to count each player's potential territory. This is particularly useful in endgames, where 1 or 2 walls in the world ends the game.

3 Advantages and Disadvantages

3.1 Advantages

In our algorithm, on top of minimax, we have implemented a mating search. This means that up to a certain depth, our algorithm can find a forced win by choosing a sequence of moves (very much like Chess). This function is used in conjuction with the minimax and alpha-beta pruning.

3.2 Disadvantages

One of the disadvantages of our approach using minimax is that this algorithm assumes that the opponent plays the best possible move when it's their turn. This implies that if the opponent is not optimized (i.e. versus a random agent or event another student's agent), and doesn't play the best move, then we could've potentially obtained a better score by choosing another move.

3.3 Expected Failure

Some expected failures.

3.4 Weaknesses

Some weaknesses.

4 Other Approaches

We've tried other approaches at the beginning such as using a Breadth First Search in order to generate the tree of possible moves. We realized that at each iteration of a Breadth First Search, we would also have to keep track of all the nodes up to the depth that we are searching for. On the other hand,

an Iterative Deepening Search will not have to keep all of them in memory since either some nodes are already visited or they are nodes that we will visit later. Not to mention, Iterative Deepening Search also does better in our situation since we have a time constraint. In the event that we reach the time limit, the algorithm will still have some sense of what a good score is even without a full traversal of the desired depth.

5 Further Improvements

We believe that one improvement to our agent can be improving its heuristic with some added features to consider such as:

1. some extra feature