Othello AI

Yilun Jiang

**General Information about the program**

The Othello AI program is written in C++ and everything is inside one source file. The program is compiled and tested to be working correctly under a Cygwin terminal. However, the program only provides a user friendly ASCII text interface (protect against bad inputs, has different color for the board and pieces, automatic prompt for all possible legal moves during the player’s turn), a GUI interface might be a possible future improvement. Regarding the algorithm used in the program, it combines iterative deepening and an alpha-beta search (i.e., a minimax search with alpha-beta pruning), along with a hard-coded heuristic function of my own construction.

**Instructions on using the program**

1. Compiling in Cygwin

g++ -o Othello\_AI Othello\_AI\_Yilun\_Jiang.cpp

1. Running the program

./ Othello\_AI

1. Playing with the program

Type in the correct form of inputs based on the prompt of the program.(Ex. Time limit for AI to use in each move, which side do you want to play…etc.) The program is protected against bad inputs so if you messed up, the program will not crash.

**Heuristic used in the program**

The heuristic function is a linear combination of two parts. One is mobility and the other is the board value. Due to the rule of Othello, mobility (number of legal moves you have) is more important than the number of pieces you have on the board. In evaluating the state, the program counts how many legal moves you will be able to choose and times this factor with a high coefficient. The other factor included in heuristic is the board value. Instead of simply counting how many pieces on the board, I assign different weights for different cells on the board. For example, the corner is assigned with a high positive value and places like X squares and C squares are assigned with relatively high negative value since they tend to give away the corners to the opponent. Other places are just plain vanilla positive values with sides and S squares being somewhat higher than the center.

Exception in using the heuristic function is that when the alpha-beta pruning has searched all the way towards the end game, the program simply counts how many pieces are on the board. Since when the game is at an ending position, what matters is only how many pieces are on the board.

