Charles Zhao
November 24, 2015

AI Othello Log

1. Winter Break 2015 I implemented minimax with iterative deepening and a board-value heuristic that puts extra weight on stable pieces. Initially, each non-stable piece had a weight of one, while stable-pieces had a weight of 10. I then improved the heuristic by giving stable pieces a weight of 20 at the beginning of the game, and then have this weight linearly decrease down to 5 when the board was full. I also gave the X and C squares (the squares surrounding the corners) negative the weight of stable pieces. Stable pieces were defined as pieces where, in at least four directions (up or down, left or right, upper-right or lower-left, and upper-left or lower-right), there were only the same-colored pieces. I.e., there were no spaces or any of the opponent's pieces. My program was able to go on average about four moves deep given approximately 1.5 seconds per move.

2. January 4, 2016 I implemented alpha-beta pruning by editing my minimax algorithm. This allowed my program to reach a depth of around six moves given approximately 1.5 seconds per move.

3. January 12, 2016 I improved my determination of stable moves, making the algorithm more efficient and accurate. I changed my determination of a stable piece to be all corners and all pieces horizontal or vertical of each corner owned by the player. These pieces had to be in a continuous line of the player's pieces to a corner. For example, in Figure 1 below, with the left-most space as space 0, spaces 0, 1, and 2 are stable pieces. An O breaks the continuous line, so space 4 is not stable.

X	X	X	O	X		X	

Figure 1: The first row of an Othello board.

4. January 14, 2016 I improved my heuristic by subtracting two times the number of possible moves the opponent would have from the board's value. This accounts for mobility.

5. January 15, 2016 I wrote a program that tested every combination of weights for the heuristic with maximum stable weights (at the beginning of the game) from 10 to 40, minimum stable weights (at the end of the game) from 1 to 10, and with mobility weights from -1 to -4. It was tournament-styled, so two algorithms would be randomly selected to play against each other and whichever won six games first moved on to the next round.

6. January 17, 2016 I used the same program as above, except this time I had a tournament among just the top twenty heuristics from the previous tournament. This time, I allowed whichever algorithm won ten games first move on to the next round.

From this, I determined that the best heuristic had a minimum stable weight of 11, a maximum stable weight of 20, and a mobility weight of -4.

7. January 23, 2016 Moves at the beginning of the game generally do not matter as much as those in the mid to late game. Therefore, I accounted for this by restricting my program to five moves deep while fewer than twenty pieces had been played, and to six moves deep while fewer than forty pieces had been played. This way, my program has more time to calculate moves later in the game.

8. January 27, 2016 I implemented principle variation search (PVS), often equated with negascout, which is an improvement upon alpha-beta pruning. For every move between moves 30-45, which I estimate are the most important moves, I store the predicted best move in a transposition table along with the depth used to determine that this was the best move. This move is the principle variation (PV), and starting with this node first in a search allows alpha-beta pruning to cut off more of the game tree. This is done through searching with a null window, i.e., assuming that the PV is the best move and therefore only searching through enough of the tree to ascertain this. I store the depth of the PVs so that if I am able to search deeper later on, I may update the transposition table with more accurate PVs. The transposition table is stored in an external file with JSON.

9. February 2, 2016 I improved my board value heuristic by accounting for the "openness" of each of a player's pieces. Essentially, having open sides creates weaknesses, so I subtract 0.5 from the board value for each open side of each of the player's pieces. This new strategy had about an 85% win rate against my previous version.