REPORT : Othello Game Bot Assignment

K Jothir Sashank EE19B137

November 28, 2021

1 Problem statement

The aim of this assignment is to build a game playing bot for the board game Othello. The bot is to be implemented in C++ using the Desdemona framework. A time limit of two seconds is placed on each move.

2 Solution Outline

Two important components of any game bot are as follows.

2.1 Search method:

A search method which explores as few leaf nodes as possible to find the best move.

2.2 Evaluation function:

An evaluation function which provides a good quantitative estimation of the position on the board. The following sections will expand more on these two aspects.

3 Search method : Alpha-beta pruning

In this assignment, the search method employed was alpha-beta pruning. This was used due to its ease of implementation. The specifics of the implementation are as follows.

- The search ply is set at 6 - This was obtained by repeatedly running the algorithm for multiple depths with an increment of 1 everytime. At 6 the whole search space is searched, at 7 the time limit exceeds 2 seconds.
- The LARGE value (limit on the heuristic values) is set at INF .
- The computation time is kept track of using the ctime library.
- Due to the time limit for each move, when the computation time exceeds 1.95 seconds, value returned is +LARGE or LARGE (for max and min nodes respectively), to force the pruning of game tree.

4 Evaluation function

The components for evaluation function used in this assignment is largely based on the ideas discussed in [2] . The heuristic function uses two components as is from the paper, viz., coin parity (coin count) and actual mobility heuristic values. Calculating the other two, corner and stability, is computationally expensive and hence due to the time limit per move, a crude approximation is used in their place. In the place of stability, a static heuristic is used to quantify a position. In the place of corner heuristic value, corner occupancy and corner closeness (discouraging occupancy of the dangerous C and X squares) are utilised based on the ideas discussed in the videos at [1]. A weighted average of all these

six components is calculated to obtain the heuristic value. The static matrix used was obtained by the trail-and-error method, starting from the static weights matrix given in [2], and observing the games played by the bot. The weights for averaging were also calculated in a similar manner, referred to github. The final static weights matrix and the weights are given in Tables 1 and 2 respectively.

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8
20	-3	11	8	8	11	-3	20
-3	-7	-4	1	1	-4	-7	-3
11	-4	2	2	2	2	-4	11
8	1	2	-3	-3	2	1	8
8	1	2	-3	-3	2	1	8
11	-4	2	2	2	2	-4	11
-3	-7	-4	1	1	-4	-7	-3
20	-3	11	8	8	11	-3	20

Table 1 - Static Weights

Heuristic	Weight
Coin parity	1190.0
Actual mobility	8692.2
Static weight	10
Corner occupancy	21268.1
Corner closeness	3820.26
Vulnerability	7839.6

Table 2 - Weights for computing average

5 Performance

This bot was made to play with the random bot provided, the static bot, which utilised the static weights heuristic as mentioned in [2] and the trial run submission bot, which also utilised the same matrix as the static bot with higher weights for corners. The results are tabulated in Table 3. The score for random bot is for a single run. Even though the result varied, the final bot always won.

Opponent Bot	Bot colour - Black	Bot colour - Red
Random bot	62-2	11-53
Slow Bot	52-12	13-51

Table 3 - Results of games: (Black score - Red score)

References

1 Othello lessons. url: Youtube Video

2 Vaishnavi Sannidhanam and Muthukaruppan Annamalai. An Analysis of Heuristics in Othello. url: Research Paper

3 kartikkukreja's Github page related to othello evaluation function url : Github