CS6380: Othello Game-Playing Bot

V.C.Ashwin
EE16B041
Muqeeth Mohammed
EE16B026

October 20, 2019

1 Introduction

Othello is a board game between 2 players say, *black* and *red*, played on an $8X8$ board. It starts with four discs being placed at the center of the board and typically *black* takes the first move. The goal of the game is to place discs such that you have more number of discs than that of the opponent when the game ends. It involves capturing the discs of the opponent and replacing them with your discs.

We have implemented a game-playing bot which emulates a player by performing moves with certain optimal strategies. For this purpose we have used the $Minimax$ Algorithm with $iterative - deepening$ and $\alpha - \beta$ pruning to develop a strategy for the best next move possible.

2 Alpha-Beta Algorithm

We find the best possible move by considering the best one out of all the valid moves of the MAX node. We calculate the heuristic value of the nodes at the final depth from the $evaluation - function$. These values are translated upwards by recursion based on the MAX node or MIN node. If at any instance the value of the node is $> \beta$ or $< \alpha$ then it is pruned and not explored further.

Figure 1: $\alpha - \beta$ Algorithm

```
AlphaBeta(j, α, β)
  1     /* To return the minimax value of a node j */
  2     /* Intially α = -LARGE, and β = +LARGE */
  3 if Terminal(j)
  4     then  return e(j)
  5     else  if j is a MAX node
  6                then for i ← 1 to b     /* j_i is the j^th child of j */
  7                       do α ← Max(α, AlphaBeta(j_i, α, β))
  8                          if α ≥ β   then return β
  9                          if i = b     then return α
 10                else   /*j is MIN */
 11                   for i ← 1 to b
 12                       do β ← Min(β, AlphaBeta(j_i, α, β))
 13                          if α ≥ β   then return α
 14                          if i = b      then return β
```

3 Evaluation Function

3.1 Corners

This evaluates the effect of the corners of the board,*i.e* $[0,0],[0,7],[7,0],[7,7]$ as they play a crucial role in deciding the winner. If a MAX node is placed at this location *high* value is assigned and vice-versa in the case of MIN node.

3.2 Difference of Discs

We calculate the disc difference on the current board state and assign a reasonable weight during mid-game and end-game. At the start of the it is not very significant since the positional value and the mobility are more dominant.

3.3 Parity

It measures the player who is expected to make the last move,i.e $parity = (EmptyLocations)\%2$. If it is 0 then the opponent has a higher chance and vice-versa.

3.4 Mobility

For a given state of the board mobility is defined as the difference of the number of valid moves of the first player and that of the second player. $Mobility = Black_{moves} - Red_{moves}$

3.5 Positional Value

We assign weights to each square on the board and calculate the value for a given board state. High values are assigned at the last rows and columns, negative values are assigned to the penultimate rows/columns. This is because logically a higher chance of placing discs on the last rows/columns indicates a greater chance of winning.

4 Summing Up...

To sum up the entire procedure, we calculate the $\alpha - values$ of each valid move through iterative deepening and return then best move. For a given depth the heuristic value is obtained from the evaluation function as follows:

```
For Start-Game i.e Total Discs on Board <= 24)
    eval = 20*(PositionalValue) + 5*(Mobility) + 1000*(CornerValue)

For Mid-Game i.e Total Discs <= 62
    eval = 10*(PositionalValue) + 2*(Mobility) + 10*(DiscDiff) +
           1000*(CornerValue) + 100*(Parity)

For End-Game i.e  62 < Total Discs <= 64
```