# Assignment-5, Report

R.Chidaksh (200010046), R.Sai Krupakar (200010047)

Feb 28, 2022

## 1 Problem Statement :

In this assignment, we have to code a bot to win the game of Othello. Given a board configuration and a turn, the bot will return a valid move. The game ends when neither of the players can make a valid move. The player with the maximum number of coins is the winner.

## 2 Algorithms :

### 2.1 Minimax Algorithm

The minimax algorithm is a backtracking algorithm used in decision making, game theory and artificial intelligence (AI). It is used to find the optimal move for a player, assuming that the opponent is also playing optimally. Mini-Max algorithm uses recursion to search through the game tree. When the algorithm reaches the required depth it calculates the heuristic value of the node.

### 2.2 Alpha Beta Pruning

Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm. The algorithm stops evaluating a move when it founds a worse move compared to the previous move examined. This is a

technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called pruning. This involves two threshold parameter Alpha and beta for future expansion, so it is called alpha-beta pruning.

## 3 Heuristic Functions :

The value returned by this heuristic is the number of coins that the player leads with respect to his opponent.

### Heuristic 1 :

```
procedure Mobility(position)
  int Player_2_moves = board.getValidMoves(other(this->turn)).size();
  int Player_1_moves = board.getValidMoves(this->turn).size();
  int Moves_differ = Player_1_moves - Player_2_moves;
  return Moves_differ
```

### Heuristic 2 :

```
procedure LeadCoin(position)
    if position.turn == Black then
        return board.getBlackCount() – board.getRedCount()
    else
        return board.getRedCount() - board.getBlackCount()
```

This heuristic tries to improve our possibilities to make a best move.

- **cornerOccupancy** - Calculates the number of corners captured by the player. Corners are a major part of the game as they cannot be outflanked in any way.
  This grants immense stability to the player who captures this position.

- **positionalScore** - A score is attached to every square on the board. Corners have a high score whereas the tiles adjacent to corners have extremely high scores. A weighted sum of all coins is calculated with this score. This helps the player filter better moves more efficiently.

## 4 Comparison between Minimax and Alpha-Beta pruning :

### 4.1 Winning Criteria

As mentioned in the question each bot can take atmost 2 seconds to return a valid move, this time constraint does not allow the Minimax algorithm to explore greater depths compared to Alpha Beta Pruning. Comparison between the two bots is made using the same heuristic. There

is a tendency of winning the game for a bot if it starts the game first. If there is no time constraint given then both the bots are expected to play equally well.

### 4.2 Space and Time Complexity

The time complexity for Alpha Beta pruning is less compared to Minimax as it eliminates some of the states while exploring. The space complexity is also less for Alpha Beta pruning compared to Minimax algorithm. The reason for this is in minimax algorithm we check for all the nodes possible, but in alpha beta pruning we assume that the opponent always chooses the minimum from the nodes available and we choose the maximum from the minimums, while exploring this we no need to explore other minimum nodes as what we want is maximum.