CS 312: Artificial Intelligence Laboratory

Lab 5 report

— B Siddharth Prabhu (200010003)

— Sourabh Bhosale (200010004)

- **Introduction:**

The objective of this task is to code a Bot to play the game of Othello in an optimal way, in order to win the game. With a given a board configuration and a turn, your 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.

- **Brief description of the algorithms:**

A) Minimax Algorithm:

Minimax is a decision rule used in decision theory, game theory, statistics, and philosophy for minimizing the possible loss for a worst case (maximum loss) scenario. This algorithm makes a tree of positions as it explores all possible moves by the opponent. When it reaches the required depth, it assesses the position using a heuristic/evaluation function. It evaluates the best move such that it minimizes the best move of the opponent in the course of game.

B) Alpha Beta Pruning:

Alpha-Beta Pruning seeks to minimize the number of positions explored in the search tree by the Minimax algorithm. It stops evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move. So, such moves need not be evaluated further and hence it prunes the search tree such that the outcome of the algorithm remains unchanged in the algorithm.

- **Heuristic functions:**

A) Coin Parity:

This heuristic returns the lead of the player with respect to their opponent (in terms of number of coins).

```
def coinParity(position)
    if position.turn == BLACK then
        return board.getBlackCount() - board.getRedCount()
    else
        return board.getBlackCount() - board.getRedCount()
```

B) Mobility:

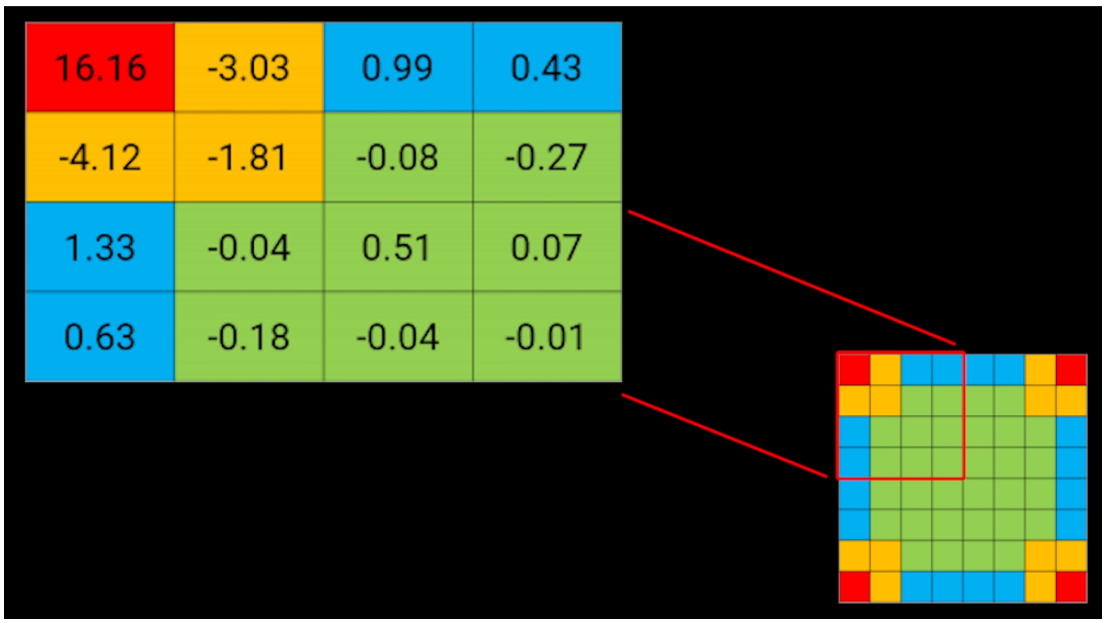
Increase in number of choices, is likely to mean that we have an advantage over the opponent. This heuristic tries to improve our freedom to make a variety of moves with respect to our opponent.

```
def mobility(position):
        myMoves = position.getValidMoves(position.turn).size()
        oppMoves = position.getValidMoves(position.turn.opponent).size()
        return myMoves - oppMoves
```

C) Corner Capture:

The number of corners occupied improves the chances of winning manifold. This heuristic returns the difference between opponents with respect to edge and corner occupation. The reference weights for the heuristics were taken from following figure. The main idea of algorithms remains same.

```
def cornersCaptured(position):
    myCorners, oppCorners = 0, 0
    for corner in position.corners() do
        if corner == position.turn then
            myCorners++
        if corner == position.turn then
            oppCorners++
    return (myCorners - oppCorners)
```



Corners & Neighbours Captured Heuristics

Reference:

https://www.youtube.com/watch?v=y7AKtWGOPAE

- **Trees moves for a given the board configuration:**

All the output files have been given in the *all_trees* directory.

- **Comparison Between Minimax and Alpha-Beta Pruning:**

Both algorithms should perform ideally well given that Alpha-Beta Pruning is an optimized version of the Minimax Algorithm in an ideal case.

A) Space and Time Complexity:

The Alpha-Beta Pruning Algorithm has lesser time and space complexity as it moves that are guaranteed to be worse than previously examined moves, are not further explored. Thus by eliminating worse states that need not be explored, the space complexity is reduced. Since, lesser states are explored, in comparison to Minimax, the time complexity is also lesser in case of Alpha-Beta Pruning Algorithm

B) Winning Criteria:

Alpha Beta bot has the advantage of exploring greater depths compared to the Minimax Bot, due to the time constraint per move of 2 seconds. When this is relaxed, both the bots are ideally expected to play equally well.

Multiple trials show that the two bots play nearly equally well and that there is a general trend of the winning bot being the one that starts the game first. This is the effect of heuristics over the winning chances of the bot.