Othello Assignment

CS6380:AI-SMPS
November 28,
2021

Problem

Othello is a two player game that is classified as a zero sum game i.e if the two players play perfectly then the match would result in a draw.Also at any point of time the players have the entire knowledge of the board and also the valid moves they can make,which makes the game a deterministic one.Artificial intelligence has always been interested in developing a bot which could play Othello effectively and could probably excel at the game.In this report I have tried to present a way in which an AI bot was developed and improved upon various versions to play in the given “Desdemona” framework.

1 Othello AI Bot

1.1 Game Tree

All the valid moves possible by a player at any level is represented in a hierarchical structure called the othello game tree and these game tree is searched to find an optimal move at any given point.Since to generate the entire game tree takes exponential time, the tree at any given point is generated up to a certain depth only and is searched to find a good move with the help of a heuristic value.

1.2 Search Algorithm

The search algorithm searches through every possible move upto a certain depth from the current level and returns the best possible move.Two algorithms while developing the bot was used which have been discussed below:

- **Minimax Algorithm:**

  Minimax algorithm operates on the concept of two players - max and min player

  - the max player tries to maximize its score by choosing a move that maximizes its value hence to win the game while the min player tries to minimize the max player’s score.It searches every node till it reaches a leaf node which represents the end state nodes where with the help of heuristic value it decides the move to choose.

- **Alpha-beta Pruning :**

  Minimax algorithm searches all the nodes of the game tree irrespective of the max or min value.In order to avoid computations of certain paths of the game tree which does not contribute to the final move alpha-beta pruning is used.It searches for almost 30efficient than it

Othello Assignment

1.3 Heuristic Function

The heuristic function is one of the major aspects of the othello AI bot.It determines the state of the current board based on the position of the discs.Its value is used to determine if the move which leads to the position of the board is good or not.The heuristic function used tries to follow the key points as mentioned by Ted Landau in his book “Othello: Brief and Basic” to choose between all the potential moves.Some of its points are mentioned below:

- Get More Stable Discs, Not Just More Discs
- Not All Squares Were Created Equal
- Control of the Game: Mobility Optimization and Dynamic Square Evaluation
- Good Moves and Bad Moves: Gaining Control
- Blocking Techniques: Access and Poisoned Moves
- Edge Play: Developing and Resolving Edges

In order to follow these key points, these features were taken into account.

- **Parity:**

  Parity refers to the piece difference between the two players at a certain position of the board.The game is essentially won or lost by which side has a greater number of pieces, and therefore it must factor into the heuristic. It is numerically calculated as the percent of total pieces that are of the maximizing color (black) or the minimizing color(Red).

- **Stability:**

  Stability is one of the most important feature in the heuristic function since discs at certain positions of the board are more valuable i.e stable than the disc at other positions.For example the disc at the corner position is the most valuable as it controls the all the horizontal,vertical as well as diagonal lines.Therefore each position in the board is associated with some weights to it based upon its stability.Static weights have been used for calculating the the stability feature.

- **Mobility:**

  Mobility is an important factor in Othello strategy because if mobility is restricted at a given point, the opponent can force the other player into taking unfavorable moves or passing the turn. Mobility is measured by the number of available moves in the given board.

- **Corner Occupancy:**

  Corners are the most powerful positions in the game as once taken, they cannot be flipped.It is almost impossible to lose if a player captures all the corner positions.

- **Closeness:**

  The positions next to empty corner spaces are considered the worst positions in the game. Taking a position next to an empty corner potentially gives your opponent the ability to take the corner immediately, if not later on in the game.

FINAL SUM is calculated by multiplying a certain factor to each of these features.Clearly giving a high weight to parity feature would not result in a good heuristic function if the entire game tree is not built.These is because having disc at important positions are more important than the number of discs.So the parity feature is given a small positive weight.Mobility and corner occupancy is very important as it gives a major control to the player and so a high positive value is associated with these two features.Corner closeness are one of the worst positions to take so a high negative weight is given to it.Furthermore a good ratio of all the weights are chosen by playing with the random bot and by itself with different weights to best suit for the required time constraint.

2 Development

2.1 Initil phase

Most of the time was spent in understanding the “Desdemona” framework and knowing all the functionalities and methods given by the framework.A simple minimax algorithm was used with 4 levels and only statibility feature was taken into consideration for calculating the heuristic function.Two different table with different weights for each position in the board was considered.The first table was considered till first 20 moves and then the second table for further more moves. The bot was presented for the trail run which performed very badly and hence a newer and better version of the bot was required.

2.2 Final phase

The minimax algorithm required a lot of unnecessary computations and so alpha-beta prun- ing was implemented to save a lot of computational time.The 4-ply search was increased to 6-ply search with necessary condition added to avoid a time limit of 2 secs for each move.Also the heuristic function was improved by taking into account all the important features men- tioned above along with their appropriate weights. The weights were revised based on its performance by playing itself with different weights.