CS6380 - Artificial Intelligence: Search Methods for Problem Solving

Assignment 2: Othello

Vishnu Kiran[1] and CS18B067[1]

[1]Indian Institute of Technology Madras

November 27, 2021

1 Problem Statement

In this assignment, you will code a bot to play and win the game of Othello. 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 maximum number of coins is the winner.

2 Search Algorithm

Game Tree is a tree with different possible game states as nodes. **Mini-Max Algorithm** is a popularly used algorithm for game playing algorithms. **Alpha-beta pruning** is an optimization of the mini-max algorithm which fastens the search of the best move significantly.

Initially, I took the approach of trying to generate the entire game-tree up to a given depth(k-ply search). But, after realising that this approach goes against the goal of alpha-beta pruning (in terms of exploring only the "good" game states), I quickly moved away from this and implemented the alpha-beta algorithm as described in the slides. The final bot that I have submitted also uses the alpha-beta algorithm as the search algorithm to find the best move.

2.1 Depth

The deeper the game tree is, the more powerful the bot is. But, we have a limitation of 2 seconds for the bot to output a move. The look-ahead depth was fixed to be 6 after trying out different depths to figure out the largest depth whose computation completes within the given time limit. In other words, the bot uses a 6-ply search.

3 Evaluation Function

Evaluation function consists of five different components and a weighted aggregate of the components was used as the metric to evaluate a board position:

1. **Disc Count:** This basically provides the number of discs belonging to that particular player in the board. Even though, the end goal of the game is to maximise the player's disc count, this component is given less weightage as it can vary significantly from one move to next and is not stable(The opponent can gain a significant number of discs in just a single move). A bot which used disc count as a metric did not perform well.

2. **Number of legal moves:** This component of the metric function measures the number of legal moves that can be made by the player. It is a measure of the mobility of the player.

3. **Corner pieces:** The squares in the corners of the board are strategic positions that result in great advantage when occupied; furthermore, a disc in a corner square can never be flipped. Thus, the number of corner squares occupied by the pieces of a particular player is a significant component and is provided high weightage.

4. **Corner Closeness:** This component is heavily related to the previous part of the evaluation function. The value of corner pieces reflects that occupying pieces next to a corner is disadvantageous as this would enable the opponent to take control of the corner square.

5. **Positional value:** Different squares of the board are assigned different values based on the analysis of past - corners are good and the squares next to corners are bad. Disregarding symmetries, there are 10 different positions on a board, and each of these is given a value for each of the three possibilities: black disk, white disk and empty. The positional value used in this program were obtained from the following paper: Application of reinforcement learning to the game of Othello

	a	b	c	d	e	f	g	h
1	100	-20	10	5	5	10	-20	100
2	-20	-50	-2	-2	-2	-2	-50	-20
3	10	-2	-1	-1	-1	-1	-2	10
4	5	-2	-1	-1	-1	-1	-2	5
5	5	-2	-1	-1	-1	-1	-2	5
6	10	-2	-1	-1	-1	-1	-2	10
7	-20	-50	-2	-2	-2	-2	-50	-20
8	100	-20	10	5	5	10	-20	100

Figure 9. Othello position values.

Figure 1: The Positional Value Matrix

The value of a board is computed as a weighted average of the different components. The coefficient for each component was fixed after doing a grid search of different values. The final bot uses the following coefficients:

4 Result

The performance of the bot was measured by 20 games played between My Bot and the Random Bot where My Bot played BLACK for 10 of the games while Random Bot played BLACK for the rest 10. The results of the games conducted are as follows:

	BLACK SCORE	**RED SCORE**	**BLACK - RED SCORE**
Game 1	52	12	40
Game 2	48	16	32
Game 3	52	12	40
Game 4	49	15	34
Game 5	49	15	34
Game 6	47	17	30
Game 7	50	14	36
Game 8	52	12	40
Game 9	53	11	42
Game 10	52	12	40
Total	504	136	368

Table 1: My Bot playing as BLACK

	BLACK SCORE	RED SCORE	BLACK - RED SCORE
Game 1	21	43	-22
Game 2	24	40	-16
Game 3	40	24	16
Game 4	31	33	-2
Game 5	26	38	-12
Game 6	4	60	-56
Game 7	33	31	2
Game 8	46	18	28
Game 9	23	41	-18
Game 10	22	42	-20
Total	270	370	-100

Table 2: My Bot playing as RED

MyBot	874
RandomBot	406
Dfiference	468
Average score per game	23.4

Table 3: Aggregate score of each bot

5 Scope of improvement

The bot can be improved in the following aspects:

1. **Evaluation functions:** Even though the evaluation function consists of different components, it can be expanded to increase much more components such as patterns in the board. The grid search conducted for choosing the coefficients could also have been expanded to be swept over a larger grid. These actions can help improve performance of the bot.

2. **Playing RED:** As is clear from the results, the bot seems to be weak when playing as RED. It loses multiple games to the random bot and the behaviour of the bot while playing RED should be analyzed in depth to further improve the bot.