Department of Engineering and Exact Sciences

Systems Engineering Career

OTHELLO GAME (HUMAN VS COMPUTER)

Intelligent Systems - Assignment #3

Alfred Brandon Garcia Arias

Cochabamba – Bolivia

October 20, 2023

Index

1. Problem description

The problem to resolve consists of programming the game named "Othello" and the "Min-Max (with depth)" algorithm.

The objective of programming these programs is to create an AI which plays Othello against another player. Both players playing the game must have the option to be a user or to be the AI programmed with the "Min-Max" algorithm.

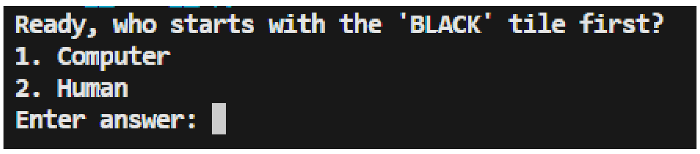
1.1 Goals

After Othello is working correctly and the AI working correctly and is able to play Othello as a player, we have to do the following experiments and modifications:

- Add the "Pruning" functionality to the Min-Max algorithm.
- Calculate the average time the AI takes to make its move.
- Make sure the AI is making a move in less than 15 seconds.
- Get the total number of expanded states by the AI.

That is basically the description of the problem and objectives.

<u>We start the game, the first to start is the black piece:</u>



```
Ready, who starts with the 'BLACK' tile first?
1. Computer
2. Human
Enter answer:
```

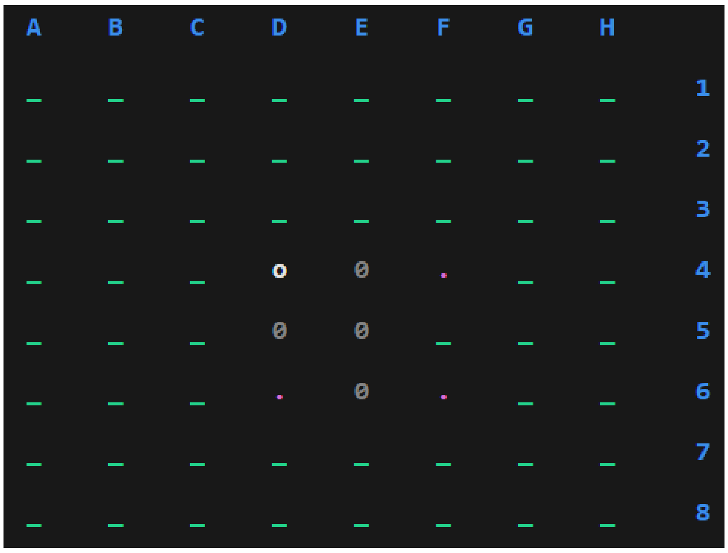
1.2 Requirements

The specific requirements are the following:

- The program reports the computer(AI) response time after each turn.
- The program reports the average response time of the computer at the end of the game.
- The program reports the number of black and white tiles that exist on the board at the end of the game, determining a winner.
- The program has an initial menu where you can choose to be black or white.
- The program shows the current state of the board after each turn.

- The MinMaxWithDepthest algorithm has been successfully implemented.
- The α β pruning algorithm is properly integrated into the MinMaxWithDepth algorithm as seen in classes.
- Othello is working correctly and according to the official game rules.

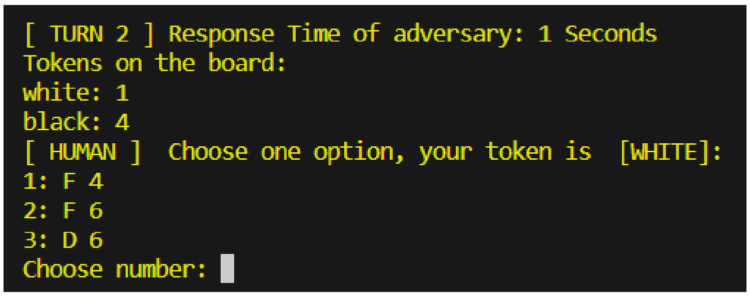
Starting Board:



```
A   B   C   D   E   F   G   H

_   _   _   _   _   _   _   _   1
_   _   _   _   _   _   _   _   2
_   _   _   _   _   _   _   _   3
_   _   _   o   0   .   _   _   4
_   _   _   0   0   _   _   _   5
_   _   _   .   0   .   _   _   6
_   _   _   _   _   _   _   _   7
_   _   _   _   _   _   _   _   8
```

Game Data:



```
[ TURN 2 ] Response Time of adversary: 1 Seconds
Tokens on the board:
white: 1
black: 4
[ HUMAN ]  Choose one option, your token is  [WHITE]:
1: F 4
2: F 6
3: D 6
Choose number:
```

2. Experiments

In order for the program to make an "intelligent" decision about where to put its token next, it must do the following:

2.1 Implement the MinMax, MinMax+α-β running and MinMax Heuristic algorithms. Compare them and define which one is better.

The MinMax algorithm, MinMax with alpha-beta pruning and Minimax with heuristics are techniques used in decision making in games and search problems:

- ☑ MinMax is the basic approach, and has no alpha-beta pruning or heuristics. It is the simplest but is not efficient in complex games.
- ☑ MinMax with alpha-beta pruning improves efficiency by avoiding exploring all branches of the search tree. It is faster than MinMax, but it does not guarantee that the best move will always be found.
- ☑ MinMax with Heuristics incorporates a heuristic evaluation function that allows you to estimate the value of a state without exploring all possibilities. It can be even faster, but the quality of the heuristics is essential for the accuracy of the decisions.

The choice of the best algorithm depends on the characteristics of the game and the needs of the problem or objective, BUT taking into account the results that will be shown below:

I. If we have sufficient computing capabilities and the game is not too complex, MinMax provides accurate results.
II. If the game is more complex or we want a faster response, MinMax with alpha-beta pruning is a solid choice.
III. If the game is extremely complex and we need quick answers, MinMax with heuristics may be the best option... of course as long as we have a good heuristic evaluation function.

2.2 Implement two heuristics for the MinMax Heuristic and define which one is better.

We defined a variety of heuristic functions to be used in our Min-Max algorithm for the "Eval()" function. The heuristic functions we defined are :

2.2.1 Highest score heuristic

This strategy consists in returning the number of tokens on board to determine who has the upperhand.

2.2.2 Best flank heuristic

This best flank heuristic consists of making a move which flanks more enemy tokens. For Example:

If we reach a "Cut off" state and we have a move #1 which flanks 2 enemy tokens and a move #2 which flanks 4 enemy tokens, the next move will be the move #2 because it has a bigger number of flanks done. (We didn't implement this heuristic)

2.3.3 Mobility strategy heuristic

Consists in returning the number of possible moves to determine the upper hand. The player who has more moves at the start of the game, around 20 turns, has a higher chance to force the opponent to make bad moves in the late game.

When we are in the late game the heuristic changes to the highest score heuristic. The heuristics can be changed in the Settings class. (setting the heuristic methods defined in the heuristicFunctionCollection class).

3. Results

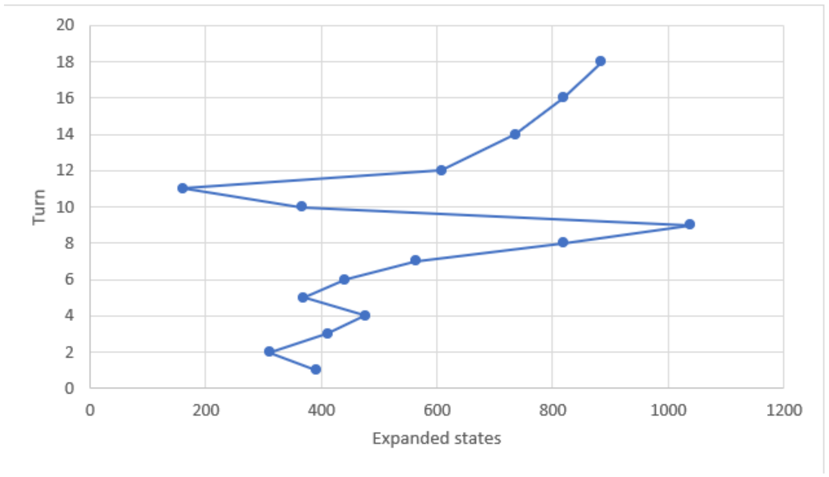
3.1 Turn VS Expanded States

with ‘depth’ = 3

Move	Shift number	Expanded states
1	2	215
1	4	535
1	6	427
1	8	1035
1	10	272
1	12	533
1	14	435
1	16	1016
1	18	1270
2	2	265
2	4	420
2	6	492
2	8	724
2	10	371
3	1	393
3	3	412
3	5	370
3	7	564
3	9	1039
3	11	161
4	2	451
4	4	476
4	6	407
4	8	696
4	10	457
4	12	684
4	14	1038
4	16	621
4	18	499

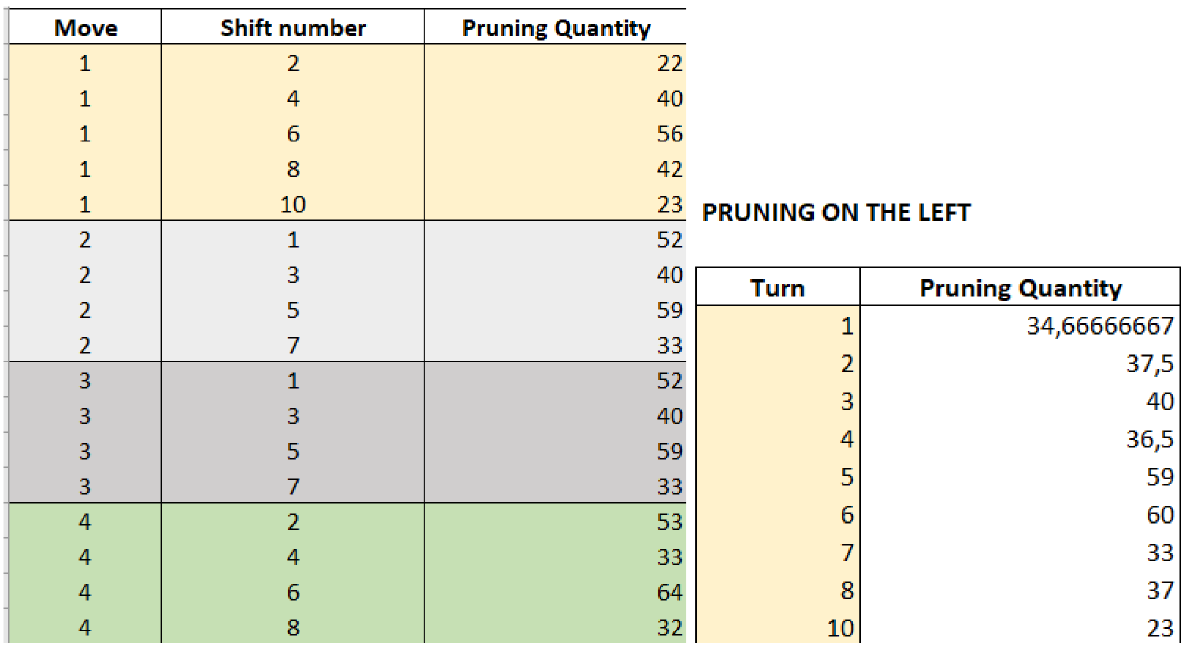
Turn	Expanded states
1	393
2	310,3333333
3	412
4	477
5	370
6	442
7	564
8	818,3333333
9	1039
10	366,6666667
11	161
12	608,5
14	736,5
16	818,5
18	884,5

<u>Display:</u>



3.2 Turn VS Number of Prunings

<u>On the left</u>

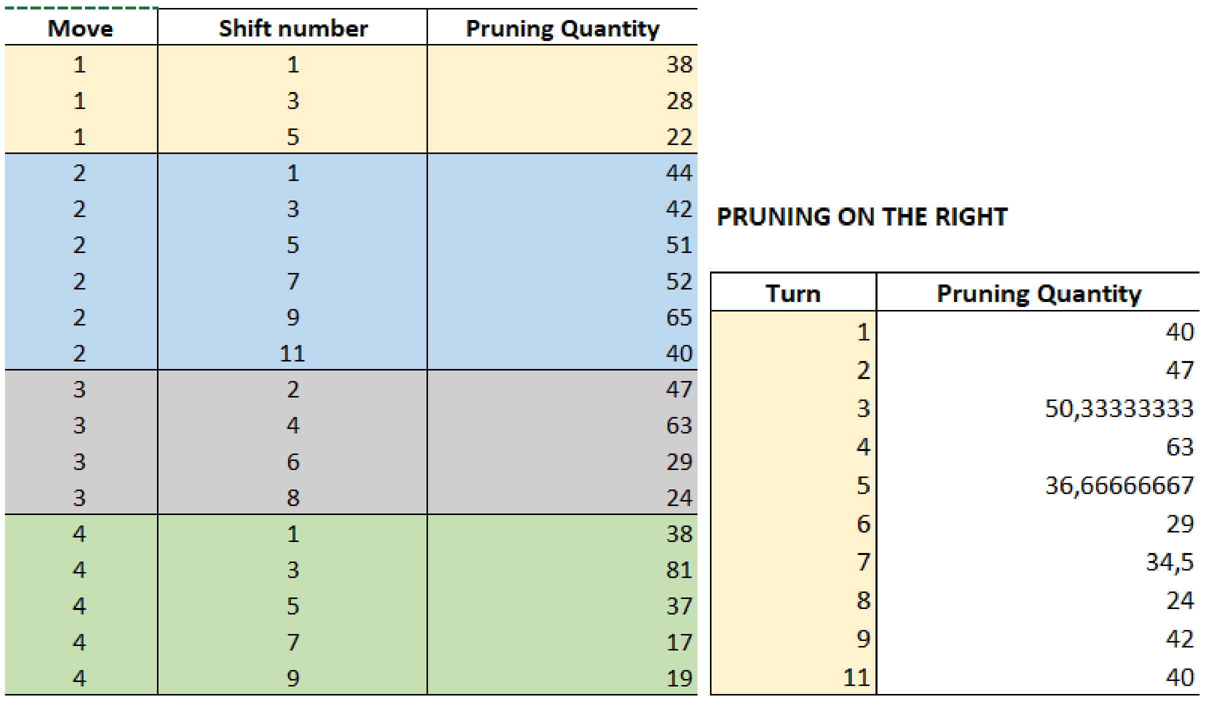


Move	Shift number	Pruning Quantity
1	2	22
1	4	40
1	6	56
1	8	42
1	10	23
2	1	52
2	3	40
2	5	59
2	7	33
3	1	52
3	3	40
3	5	59
3	7	33
4	2	53
4	4	33
4	6	64
4	8	32

PRUNING ON THE LEFT

Turn	Pruning Quantity
1	34,66666667
2	37,5
3	40
4	36,5
5	59
6	60
7	33
8	37
10	23

On the right

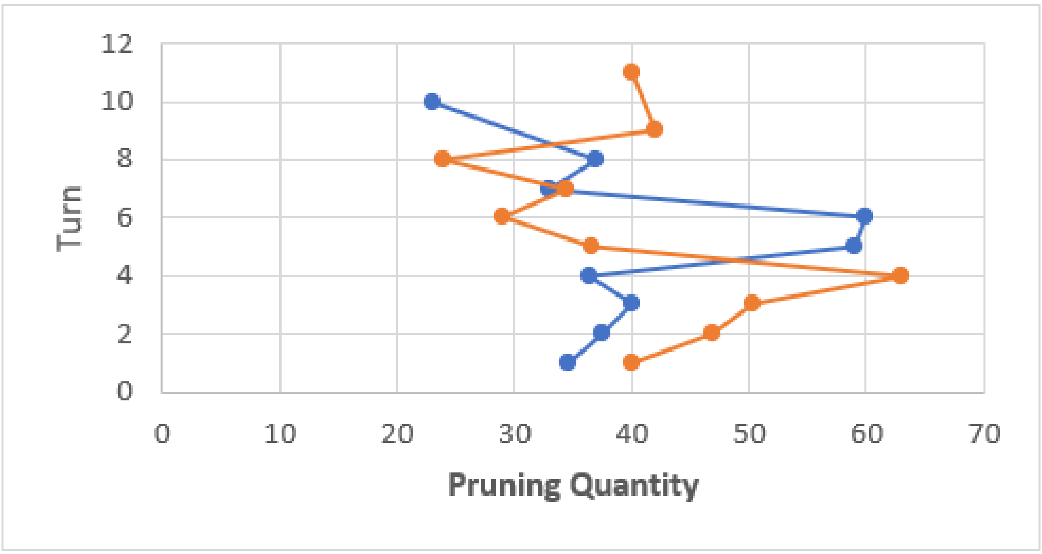


Move	Shift number	Pruning Quantity
1	1	38
1	3	28
1	5	22
2	1	44
2	3	42
2	5	51
2	7	52
2	9	65
2	11	40
3	2	47
3	4	63
3	6	29
3	8	24
4	1	38
4	3	81
4	5	37
4	7	17
4	9	19

PRUNING ON THE RIGHT

Turn	Pruning Quantity
1	40
2	47
3	50,33333333
4	63
5	36,66666667
6	29
7	34,5
8	24
9	42
11	40

Display:



3.3 Time

Average decision making time from the AI.

The average time the AI takes to make a move is: between 6 and 11 seconds.

4. Conclusions

I have observed that heuristics are very important.

We observed that the highest score strategy is more consistent in terms of the average time the computer takes to respond.

We observed that the mobility strategy is sometimes better, but there are cases where it takes up to 40 seconds to respond. But at the end it returns a very low average response time around 6 seconds.