e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
Volume:02/Issue:12/December -2020 Impact Factor- 5.354 www.irjmets.com

COMPARISON OF HEURISTIC METHODS IN THE OTHELLO GAME

Pramita Kastha [*1]

[*1]Student, Department of Electronics and Communication Engineering, Indian Institute of Technology, Kharagpur, Kharagpur, West Bengal, India.

ABSTRACT

Heuristics have become a widely accepted and used method for usability evaluation in software development. In this paper, we implement the simple board game of Othello in python, propose four strategies and compare them against each other by simulating a number of games and noting the percentage of wins, losses and ties.

Keywords: Heuristics, games, board games, evaluation, comparison, artificial intelligence.

I. INTRODUCTION

Othello is a very popular board game. The basic idea of the game is the maximize the number of the chips of your color, and at the same time attempt to reduce those of the opponents. To win over the opponent, a good strategy is paramount. However, what may initially appear to be a bad strategy might prove fruitful in some cases and vice-versa. Thus, statistical analysis on the performance of common strategies is needed.

II. HOW TO PLAY OTHELLO

Othello has a 8x8 board and tiles that are black on one side and white on the other. It is played by two people using one board. The aim of this game is to get one's opponent's chips flipped, as many as possible. The player who has more chips of his own type remaining on the board at the end of the game is declared winner. End of the game is a situation where either the board is full, or there is no chip which can make a jump.

The way to play this game is quite easy. For example, say the first player uses the black chip, and the second player uses the white chip. Black, which was initially at E4, jumps through the white chip at D4 into C4. Now, C4 has black, E4 has black, and D4, which was initially white, is flipped over to black as it has been jumped through. Figure 1 and Figure 2, respectively, show the initial and final state of the board after this move.

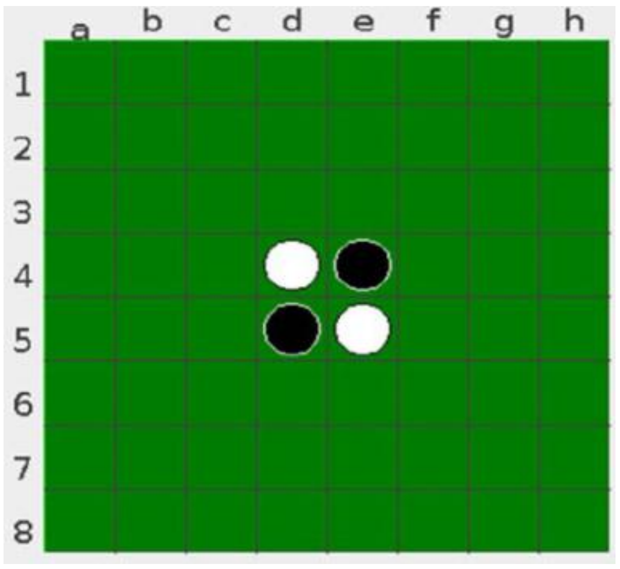


Fig.-1: Initial state of the board

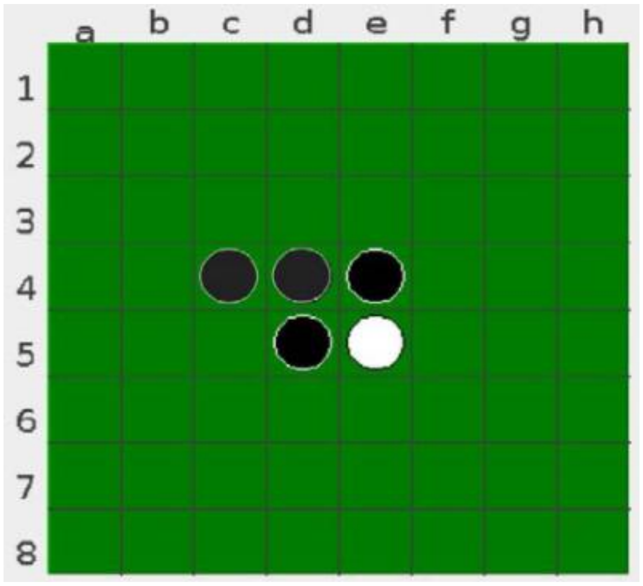


Fig.-2: Final state of the board

III. IMPLEMENTATION

We start by coding the python implementation of the Othello game where in one player is the user who supplies the move and the other player is the computer. Following are the steps for the same:

Drawing the Game Board Data Structure

The game board is a represented by a matrix. In python, that is a list of lists. We initialize all values as empty string (except the initial coordinates). Then, based on the player or the computer's move, they are initialized to either 'O'(for white) and 'X'(for black).

Checking Whether a Move is Valid

In order for a move to be valid, it needs to flip at least one of the opponent's tiles by sandwiching the current player's new tile with one of the player's old tiles. That means that the new tile must be next to one of the opponent's tiles. Next, we check whether there are adjacent tiles that can be flipped over. Finally, we check whether the resulting coordinates are a valid one, i.e the resulting coordinate lies within the matrix.

Getting the Score, Getting the Player Choice, Determining the First Move and Placing a Move

For getting the score, we simply use a nested loop to traverse the matrix, check for all 64 positions and store the count. of 'X's and 'O's.

For getting the Player choice, we prompt the user to enter either 'X' or 'O'.

The player who moves first is determined randomly.

Placing a move is simply modifying the game board matrix in accordance to the rules.

Getting the Player's Move

The user has to enter the coordinates of his choice of move. If it is valid, the move is executed, else it is prompted to enter a valid one.

Getting the Computer Move

Here, we propose the following strategy:

1. Generate a list of all possible moves and shuffle them randomly.
2. From the shuffled list, return the first corner move. This is because once a tile has been placed on the corner, it can never be flipped over.
3. If there is no corner move, loop through the list, calculate the score after each of the possible moves and return the one with the highest score.

At this stage, we are done with creating a simple game where a user can play against a computer. Next, we code four functions, each with four different strategies, simulate them against one another and study the results.

Corner-Best AI

This is the same strategy that we used initially, i.e:

1. Generate a list of all possible moves and shuffle them randomly.
2. From the shuffled list, return the first corner move. This is because once a tile has been placed on the corner, it can never be flipped over.
3. If there is no corner move, loop through the list, calculate the score after each of the possible moves and return the one with the highest score.

Worst-Move AI

1. Generate a list of all possible moves and shuffle them randomly.
2. Loop through the list, calculate the score after each of the possible moves and return the one with the lowest score.

Random-Move AI

1. Generate a list of all possible moves.
2. Return a random move.

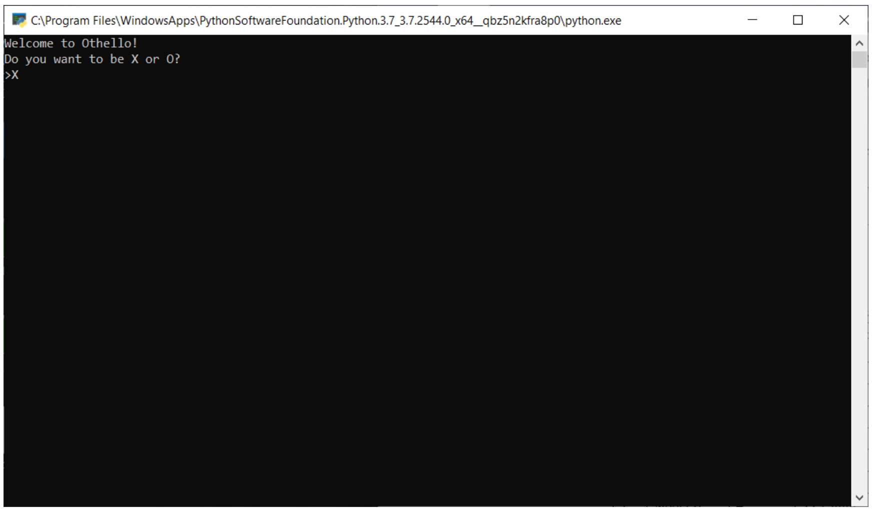
Corner-Side-Best AI

1. Generate a list of all possible moves and shuffle them randomly.
2. From the shuffled list, return the first corner move. This is because once a tile has been placed on the corner, it can never be flipped over.
3. If there is no corner move, make a side move, as they have a lesser chance of being flipped over as compared to those on the inside.
4. If there is no side move, loop through the list, calculate the score after each of the possible moves and return the one with the highest score.

Finally, we modify our game code to simulate a certain no. of games and record the count of wins losses and ties, along with their percentages.

IV. RESULTS AND DISCUSSION

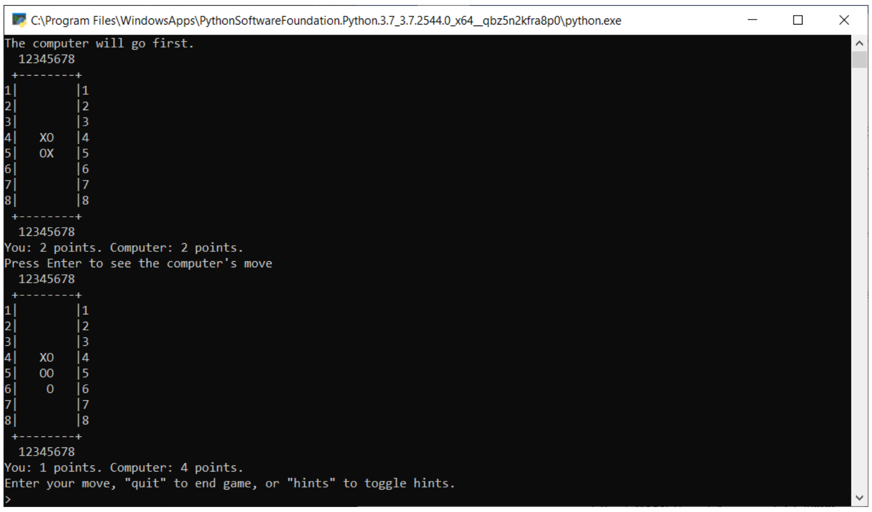
Before coding the simulation, we give a sample run of our game code where the user plays with the computer, which is following the Corner-Best AI strategy.


C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.7_3.7.2544.0_x64__qbz5n2kfra8p0\python.exe

```
Welcome to Othello!
Do you want to be X or O?
>X
```

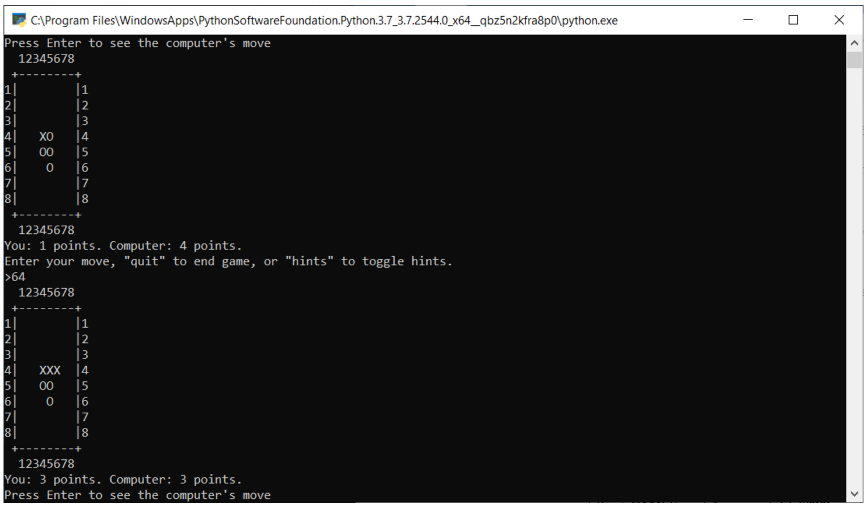
Fig.-3: Start screen of game window

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
Volume:02/Issue:12/December -2020 Impact Factor- 5.354 www.irjmets.com



```
C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.7_3.7.2544.0_x64__qbz5n2kfra8p0\python.exe
The computer will go first.
  12345678
 +--------+
1|        |1
2|        |2
3|        |3
4|   XO   |4
5|   OX   |5
6|        |6
7|        |7
8|        |8
 +--------+
  12345678
You: 2 points. Computer: 2 points.
Press Enter to see the computer's move
  12345678
 +--------+
1|        |1
2|        |2
3|        |3
4|   XO   |4
5|   OO   |5
6|    O   |6
7|        |7
8|        |8
 +--------+
  12345678
You: 1 points. Computer: 4 points.
Enter your move, "quit" to end game, or "hints" to toggle hints.
>
```

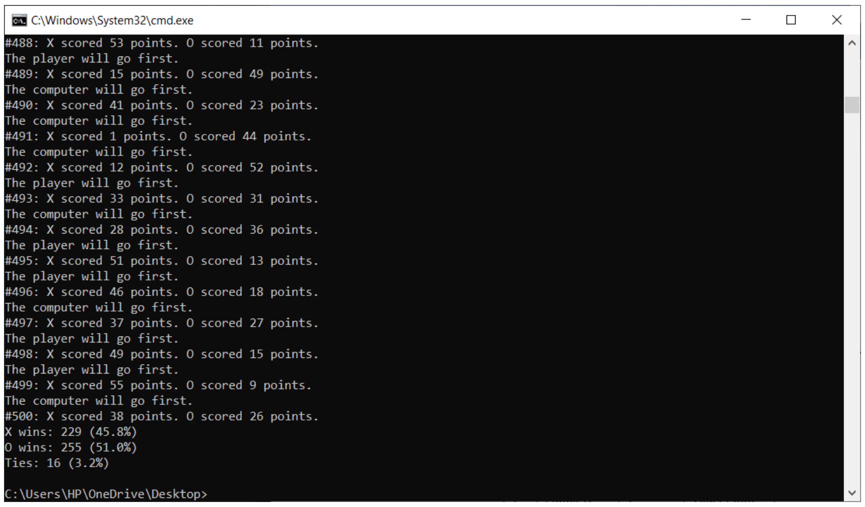
Fig.-4: The computer makes the first move



```
C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.7_3.7.2544.0_x64__qbz5n2kfra8p0\python.exe
Press Enter to see the computer's move
  12345678
 +--------+
1|        |1
2|        |2
3|        |3
4|   XO   |4
5|   OO   |5
6|    O   |6
7|        |7
8|        |8
 +--------+
  12345678
You: 1 points. Computer: 4 points.
Enter your move, "quit" to end game, or "hints" to toggle hints.
>64
  12345678
 +--------+
1|        |1
2|        |2
3|        |3
4|   XXX  |4
5|   OO   |5
6|    O   |6
7|        |7
8|        |8
 +--------+
  12345678
You: 3 points. Computer: 3 points.
Press Enter to see the computer's move
```

Fig.-5: The player makes the move after the computer

Now, we run the simulation taking two of the strategies at a time. Following is the output of Corner-Best AI against Corner-Side-Best AI.



```
C:\Windows\System32\cmd.exe
#488: X scored 53 points. O scored 11 points.
The player will go first.
#489: X scored 15 points. O scored 49 points.
The computer will go first.
#490: X scored 41 points. O scored 23 points.
The computer will go first.
#491: X scored 1 points. O scored 44 points.
The computer will go first.
#492: X scored 12 points. O scored 52 points.
The player will go first.
#493: X scored 33 points. O scored 31 points.
The computer will go first.
#494: X scored 28 points. O scored 36 points.
The player will go first.
#495: X scored 51 points. O scored 13 points.
The player will go first.
#496: X scored 46 points. O scored 18 points.
The computer will go first.
#497: X scored 37 points. O scored 27 points.
The player will go first.
#498: X scored 49 points. O scored 15 points.
The player will go first.
#499: X scored 55 points. O scored 9 points.
The computer will go first.
#500: X scored 38 points. O scored 26 points.
X wins: 229 (45.8%)
O wins: 255 (51.0%)
Ties: 16 (3.2%)

C:\Users\HP\OneDrive\Desktop>
```

Fig.-6: Simulation of 500 games of Corner-Best AI against Corner-Side-Best AI

The below table presents shows the results of 500 games, taking two strategies at a time, for all 6 cases.

Table-1: Comparison of performance of different strategies (Total no. of games=500)

SN.	Player 1	Player 2	Percentage of Wins by Player 1	Percentage of Wins by Player 2	Percentage of Ties
1	Corner-Best AI	Corner-Side-Best AI	51.0	45.8	3.2
2	Corner-Best AI	Worst-Move AI	80.8	16.0	3.2
3	Corner-Best AI	Random-Move AI	78.2	19.6	2.2
4	Corner-Side-Best AI	Worst-Move AI	83.0	15.0	2
5	Corner-Side-Best AI	Random-Move AI	72.4	25.8	1.8
6	Random-Move AI	Worst-Move AI	57.4	37.8	4.8

V. CONCLUSION

The experiment reveals the following results:

1. Making the immediate highest scoring move after making a corner move is more effective than making a side move.
2. Even the worst move strategy perform pretty well against highly scoring strategies like Corner-Best AI and Corner-Side-Best AI.
3. The random move strategy is only slightly better than the worst move strategy.

VI. REFERENCES

[1] Novaryan, Jeremy & Ch, T.. (2014). IMPACT ANALYSIS OF THE CAPABILITY ON STRATEGY VIA OTHELLO GAME. SISFORMA. 1. 5. 10.24167/sisforma.v1i1.86..

[2] Desurvire, Heather & Heather, & Caplan, & Martin, & Toth, & A, Jozsef. (2004). Using heuristics to evaluate the playability of games. 10.1145/985921.986102.

[3] Şişeci, Melike & Pence, Ihsan & Cetisli, Bayram. (2016). Comparison of search algorithms and heuristic methods in the solution of chess endgame problems. 1509-1512. 10.1109/SIU.2016.7496038..

[4] Szubert, Marcin & Jaśkowski, Wojciech & Krawiec, Krzysztof. (2011). Learning Board Evaluation Function for Othello by Hybridizing Coevolution with Temporal Difference Learning. Control and Cybernetics. 40. 805-831.

[5] http://inventwithpython.com/invent4thed/