Tic-Tac-Toe and Wild Tic-Tac-Toe using Minimax

Sirish Prabakar, Dane Jew

Tic-Tac-Toe and Wild Tic-Tac-Toe using Minimax

Abstract

Adversarial algorithms are essential for computer games. The simplest of these is Minimax, which can be used for deterministic games such as Tic-Tac-Toe. The goal of this project is to implement Minimax for both Tic-Tac-Toe and Wild Tic-Tac-Toe to demonstrate the use of adversarial search.

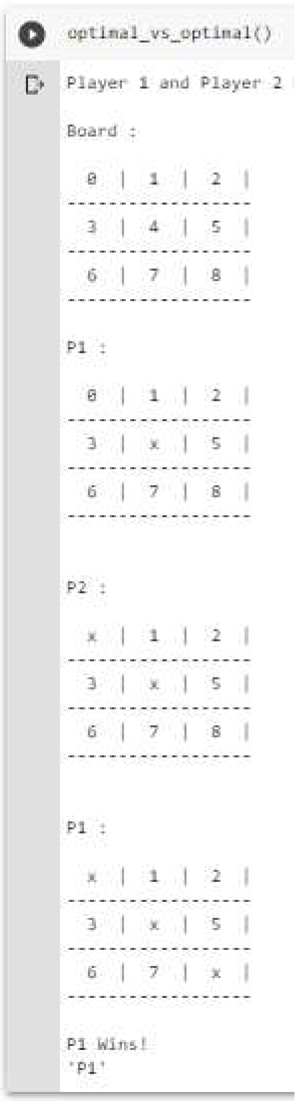
Problem Statement

The goal of this project is to implement Minimax functions on both Tic-Tac-Toe and Wild Tic-Tac-Toe. This is to be verified by the characteristics of each algorithm.

Methodology

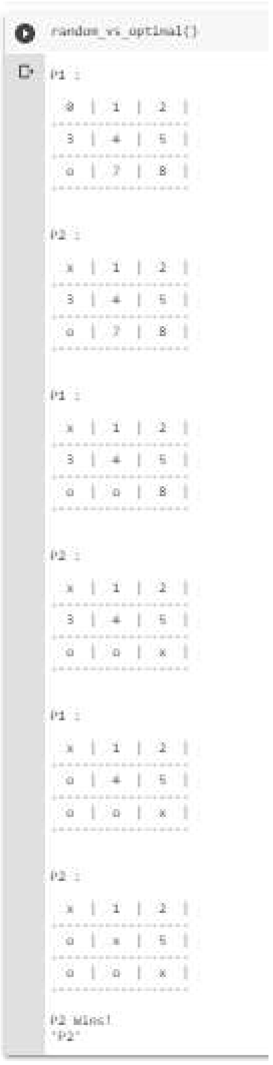
The Tic-Tac-Toe and Wild Tic-Tac-Toe projects as with others was implemented in Python Jupyter Notebook. The only part that required work was simply implement the Minimax functions for both search algorithms.

Figure 1
Minimax for Wild Tic-Tac-Toe
Optimal vs. Optimal



```
optimal_vs_optimal()

Player 1 and Player 2

Board :

 0  |  1  |  2  |
-----------------
 3  |  4  |  5  |
-----------------
 6  |  7  |  8  |
-----------------

P1 :

 0  |  1  |  2  |
-----------------
 3  |  x  |  5  |
-----------------
 6  |  7  |  8  |
-----------------


P2 :

 x  |  1  |  2  |
-----------------
 3  |  x  |  5  |
-----------------
 6  |  7  |  8  |
-----------------


P1 :

 x  |  1  |  2  |
-----------------
 3  |  x  |  5  |
-----------------
 6  |  7  |  x  |
-----------------

P1 Wins!
'P1'
```

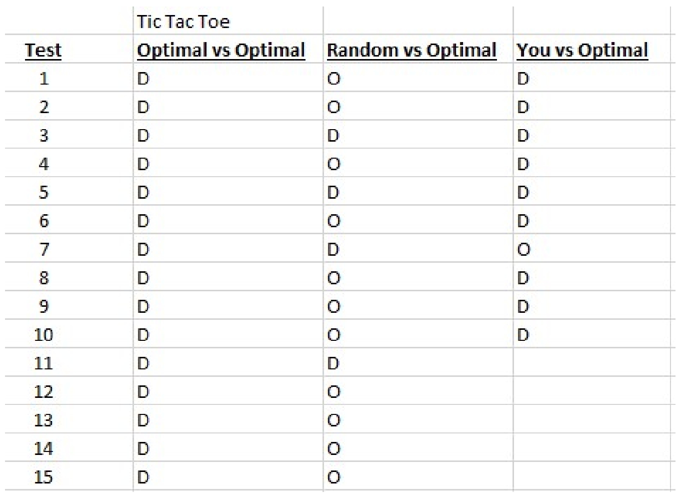
Figure 1
Minimax for Wild Tic-Tac-Toe



```
random_vs_optimal()
```

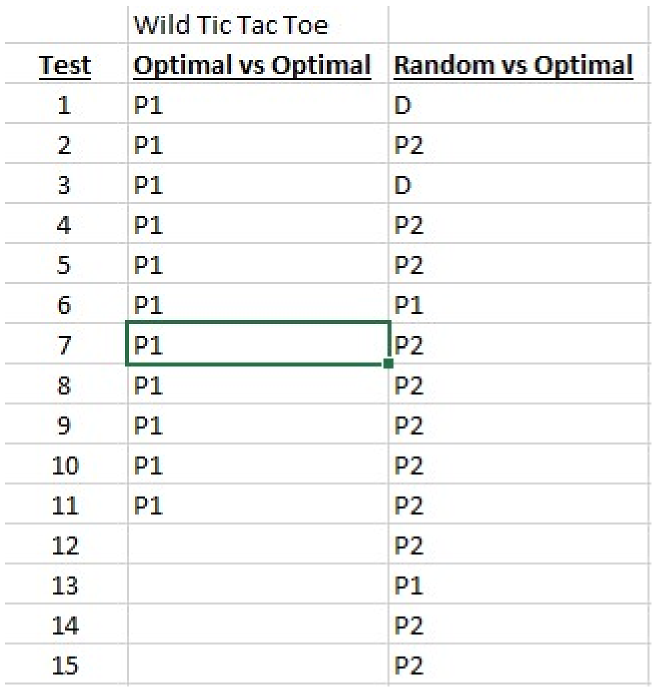
```
P1 :

 0 | 1 | 2 |
-------------
 3 | 4 | 5 |
-------------
 o | 7 | 8 |
-------------

P2 :

 x | 1 | 2 |
-------------
 3 | 4 | 5 |
-------------
 o | 7 | 8 |
-------------

P1 :

 x | 1 | 2 |
-------------
 3 | 4 | 5 |
-------------
 o | o | 8 |
-------------

P2 :

 x | 1 | 2 |
-------------
 3 | 4 | 5 |
-------------
 o | o | x |
-------------

P1 :

 x | 1 | 2 |
-------------
 o | 4 | 5 |
-------------
 o | o | x |
-------------

P2 :

 x | 1 | 2 |
-------------
 o | x | 5 |
-------------
 o | o | x |
-------------

P2 Wins!
'P2'
```

Figure 2
Tic-Tac-Toe



	Tic Tac Toe		
Test	**Optimal vs Optimal**	**Random vs Optimal**	**You vs Optimal**
1	D	O	D
2	D	O	D
3	D	D	D
4	D	O	D
5	D	D	D
6	D	O	D
7	D	D	O
8	D	O	D
9	D	O	D
10	D	O	D
11	D	D	
12	D	O	
13	D	O	
14	D	O	
15	D	O	

Figure 3
Wild Tic-Tac-Toe



	Wild Tic Tac Toe	
Test	**Optimal vs Optimal**	**Random vs Optimal**
1	P1	D
2	P1	P2
3	P1	D
4	P1	P2
5	P1	P2
6	P1	P1
7	P1	P2
8	P1	P2
9	P1	P2
10	P1	P2
11	P1	P2
12		P2
13		P1
14		P2
15		P2

Experimental Results and Analysis

After the Tic-Tac-Toe algorithms was implemented, both the Optimum_vs_Optimum and Random_vs_Optimum was executed 100 times each. As a results it can be confirmed that when both players are optimal it always ends in a draw. See Figure 1. When player one, which is "x" is random, the other player "o", who is optimal, most of the times wins.

Figure 1
Tic-Tac-Toe
Outcome*

	Optimal vs. Optimal	Random vs. Optimal
X Wins	0.00	0.00
O Wins	0.00	0.79
Draws	1.00	0.21

* Results are from 100 games.

The Wild Tic-Tac-Toe algorithms, on the other hand, has very poor performance due to complexity.

Figure 2
Wild Tic-Tac-Toe
Outcomes*

	Optimal vs. Optimal	Random vs. Optimal
X Wins	1.00	0.10
O Wins	0.00	0.82
Draws	0.00	0.08

* Results for Optimal vs. Optimal is out of 10. Random vs. Optimal is out of 50.

Task Division and Project Reflection

The work breakdown is as follow in the table below:

Coding		
	Work Pct	
Dane Jew	50%	
Sirish Prabakar	50%	

Documentation		
	Work Pct	
Dane Jew	50%	
Sirish Prabakar	50%	

Challenges

The challenges come from implementation of the Minimax functions for both games. The Tic-Tac-Toe implementation was the more simple of the two, although implementing a recursive function is inherently complex. The other issue is that python does not allow pass by value for parameter in function declarations.

The challenge with Wild Tic-Tac-Toe is that running of the recursive function requires a much larger search space or tree. This causes execution to run extremely slow and requires the use of Google Colab to run the simulations. This reduced the number of test that can be run for the Optimum vs. Optimum.

Conclusion

In the end, both the Tic-Tac-Toe and Wild Tic-Tac-Toe algorithms ran very well in terms of implementation. The both are able to win matches at a high rate.