Checkers Report

Name: Muhammad Sayed Mahdy Mahmoud
Sec: 2
B.N.: 14
Problem number: 4

Files Structure

1. ***Checkers.py*** contains all the functionalities of the checkers game such as move generators, minimax algorithm and evaluation functions. The functions are well documented.
2. ***Game.py*** contains the functions of the GUI
3. ***MinimaxVsMinimax.py*** contains code to run a minimax agent against another minimax agent with different or same parameters. It's just for comparing between different evaluation functions and hyperparameters.
4. ***MinimaxVsRandom.py*** contains code to run minimax agent against random playing agent.It's for the same purpose as ***MinimaxVsMinimax.py***

How to Run

You must have python 3 installed.
just type in the terminal ***python Game.py*** to run the game.
it's cross-platform.
You can also run the other files ***MinimaxVsMinimax.py*** and ***MinimaxVsRandom.py*** and tweak the hyperparameters.

Hyperparametrs

1. Max depth of the alpha-beta minimax algorithm. The more the max depth is, the harder the agent and the more time it takes to move.
2. Evaluation function (more in this below).
3. Size of the board (you can change it by passing the size to the ***Checkers*** constructor.
4. You can uncomment the code of state counter to allow for the rule of (if the exact same state repeated 3 times, then it will be draw) this code can be found in ***MinimaxVsMinimax.py*** and ***MinimaxVsRandom.py***

Experiments and Results

Evaluation Function 1

$$Evaluation\ function\ 1 = (2 * \#maximizer\ kings + maximizer\ men - (2 * opponent\ kings + opponent\ men)) * 5$$

This was the first evaluation function I wrote, and it's very intuitive, you need to maximize the number of your pieces and minimize the number of the opponent pieces, giving the kings more weight

Evaluation function 2

I implemented the evaluation function suggested in this paper.
It depends on several parameters.

1. Number of men
2. Number of kings
3. Number of pieces in the middle box (the middle 2 rows and middle 4 columns). This is an advantageous position.
4. Number of pieces in the middle row but not in the middle box
5. Number of vulnerable pieces, that is in an attacked position
6. Number of protected pieces, that is protected by other pieces or by a wall.

I used the weights suggested by the paper.

Sum of Distances

When there are few pieces left on the board, the previous functions tend to draw if the maximum depth wasn't large, they try to minimize the number of lost pieces. So a good evaluation function in this situation is the sum of distances between every piece of the maximizer and all pieces of the opponent. If the maximizer pieces are more than the opponent pieces, then he should come closer to him and attack while keeping his pieces alive, otherwise if the number of maximizer pieces is less than or equal to the opponent pieces, it's good to run away from him to avoid being attacked.
Keeping maximizer pieces alive is achieved by the same idea as in *Evaluation function 1.*
I activate this function when the number of moves without capturing exceeds 25. (the draw is after 100 moves without capturing).

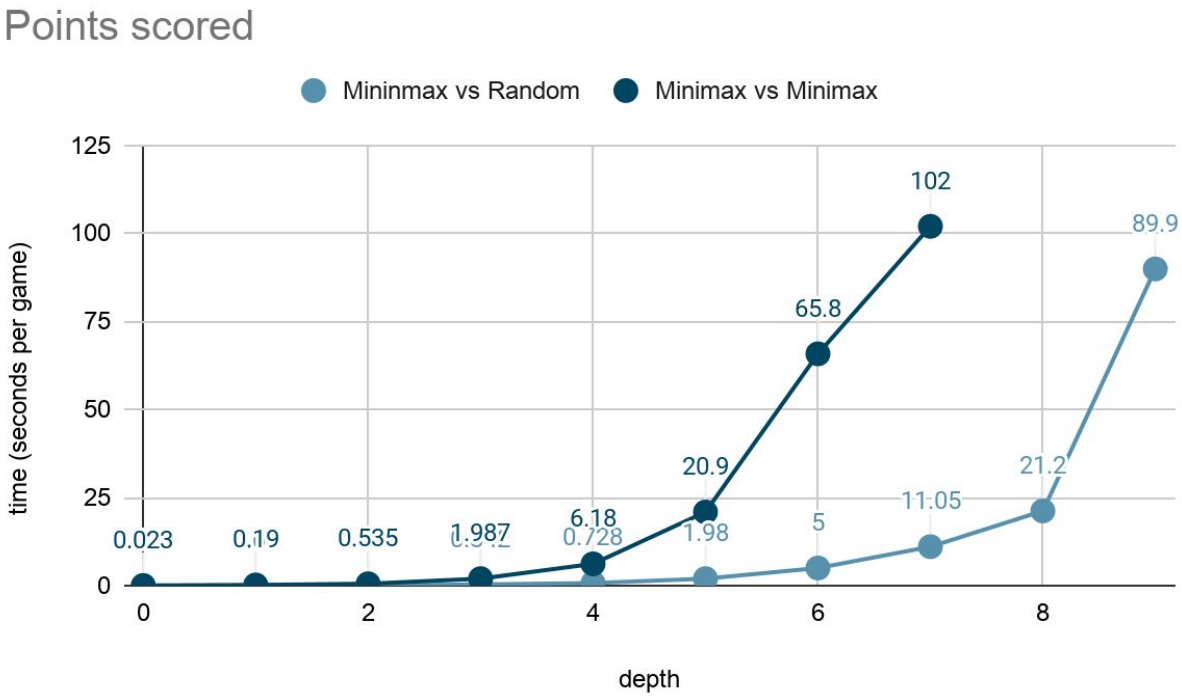
Changing Maximum Depth

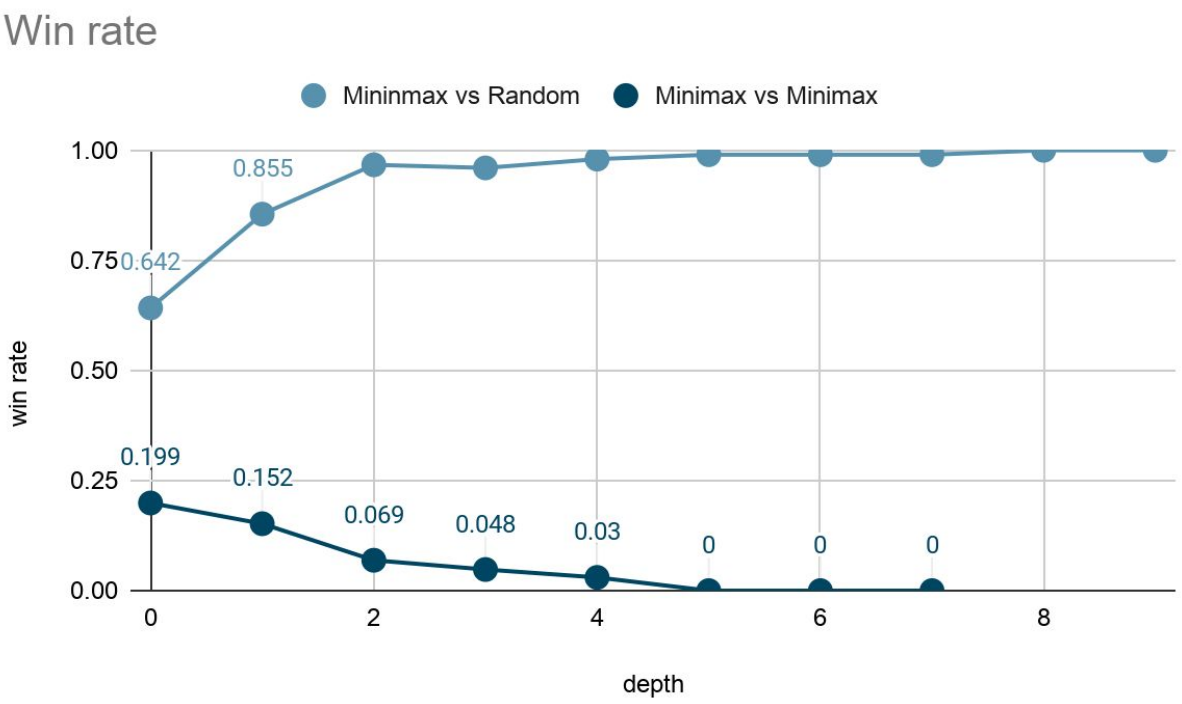
Increasing maximum depth of the alpha-beta minimax algorithm improved the results, but the time increases exponentially. I run the agents two times by this settings

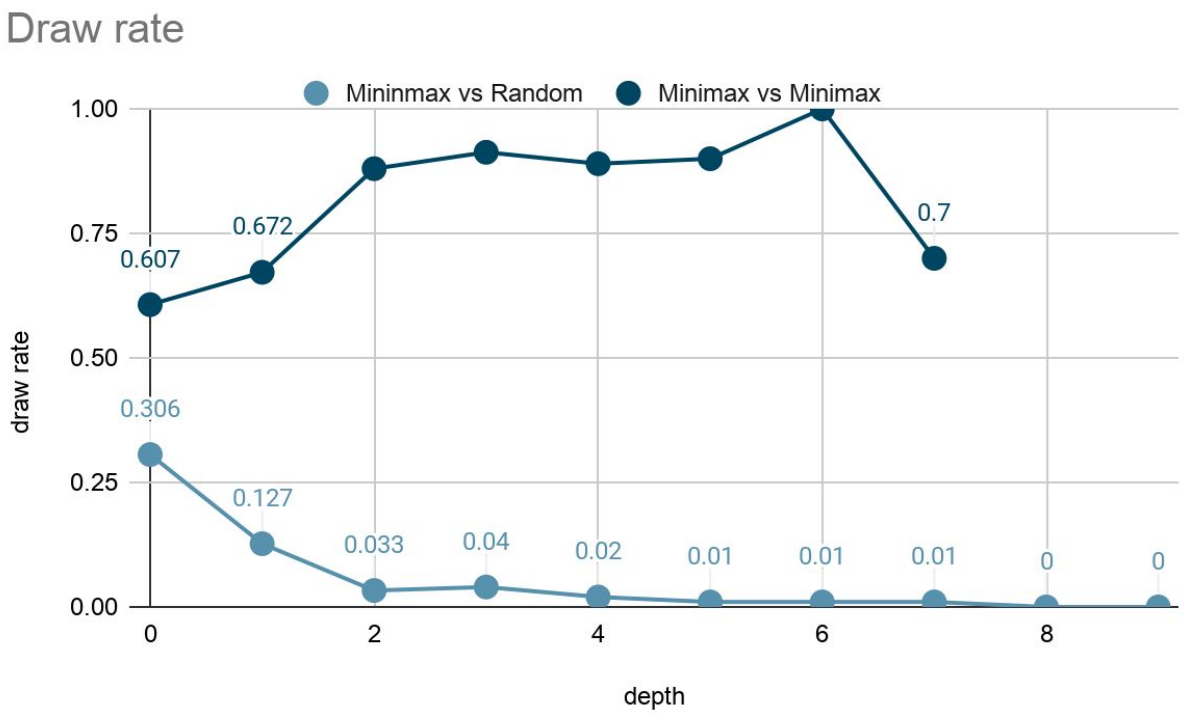
1. minimax (evaluation function 1) vs random.
2. minimax (evaluation function 1) vs minimax (evaluation function 1).

It produced the following results.

for the complete set of results refer to this [sheet]







Comparison between evaluation functions

Evaluation function 2 vs Evaluation function 1				
Depth	Win rate	Draw rate	Lose rate	time/game (sec)
0	0.269	0.574	0.157	0.037
1	0.27	0.659	0.071	0.273
2	0.008	0.854	0.138	1.02
3	0.01	0.83	0.16	3.95
4	0	0.82	0.18	16.28
5	0	0.9	0.1	75.4

After experimenting, evaluation function 2 is better than evaluation function 1 but at lower values of maximum depth. When increasing the depth, evaluation function 1 performed better.

Adding the sum of distances evaluation improved the results of evaluation function 2.