Checkers Report

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.

Configuration

1. You can change the size of the checkers instead of the default 8*8 to any even number greater than 3.
2. You can change the mode of the game, it can be either ***Mode.MULTIPLE_PLAYER*** or ***Mode.SINGLE_PLAYER***
3. You can change the starting player in ***Game.py*** it can be either **Checkers.BLACK** or **Checkers.WHITE**
4. You can change the algorithm used to play a computer move, it can be either ***Algorithm.MINIMAX*** or ***Algorithm.RANDOM*** (minimax is much harder).

Minimax configuration

1. You can change the ***MAX_DEPTH***, the higher the max depth the harder the level of play and the more time it takes to compute the play.
2. You can change the evaluation function, it can be either ***Checkers.evaluate1***, ***Checkers.evaluate2*** or ***Checkers.sumDistances***

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 modified 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

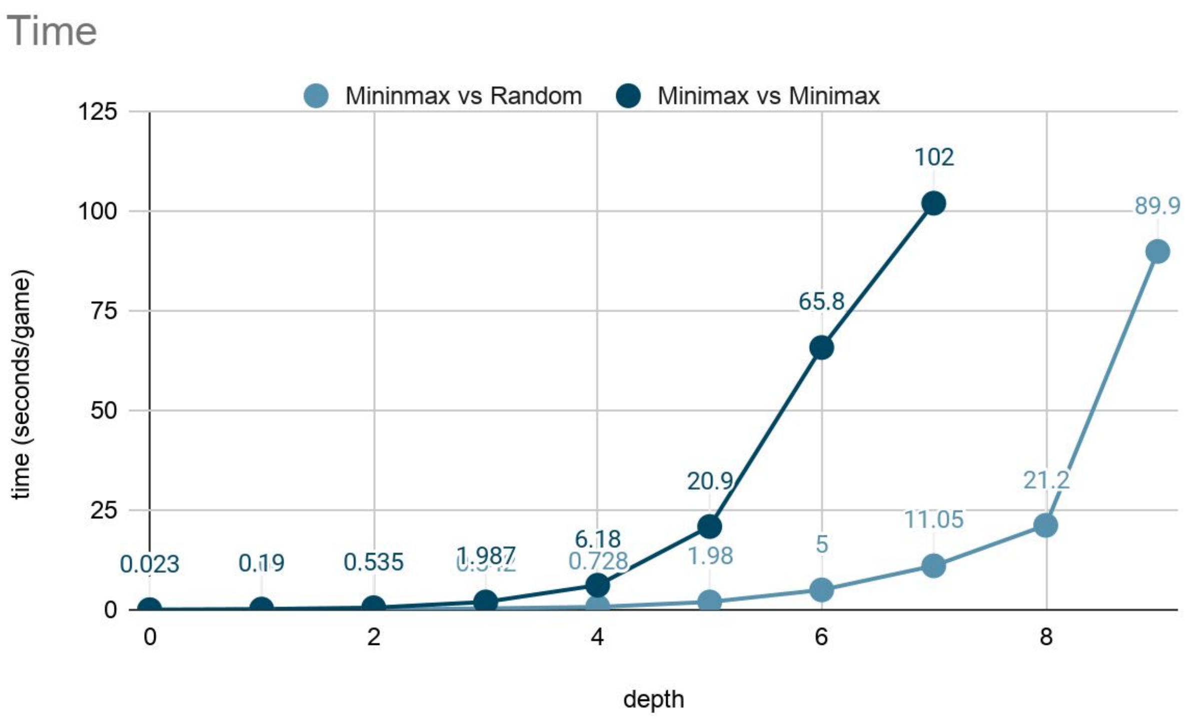
Increasing maximum depth of the alpha-beta minimax algorithm improved the results, but the time increases exponentially. I run the agents two times by this settings

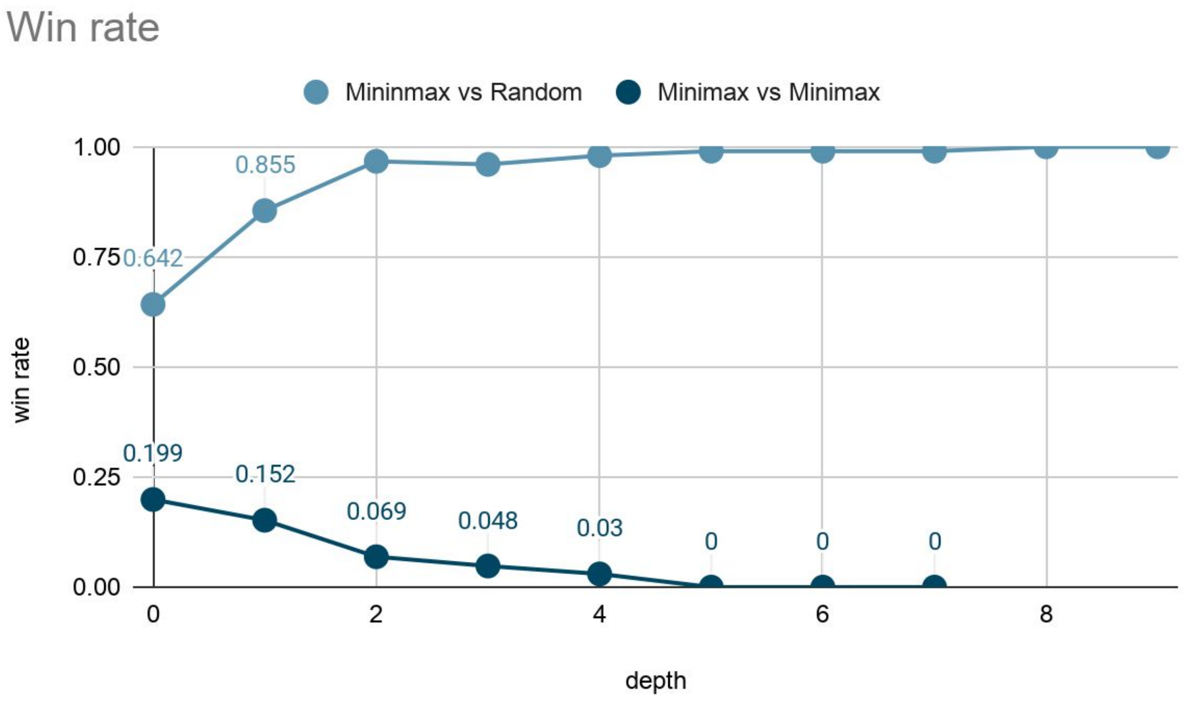
1. minimax (evaluation function 1) vs random.

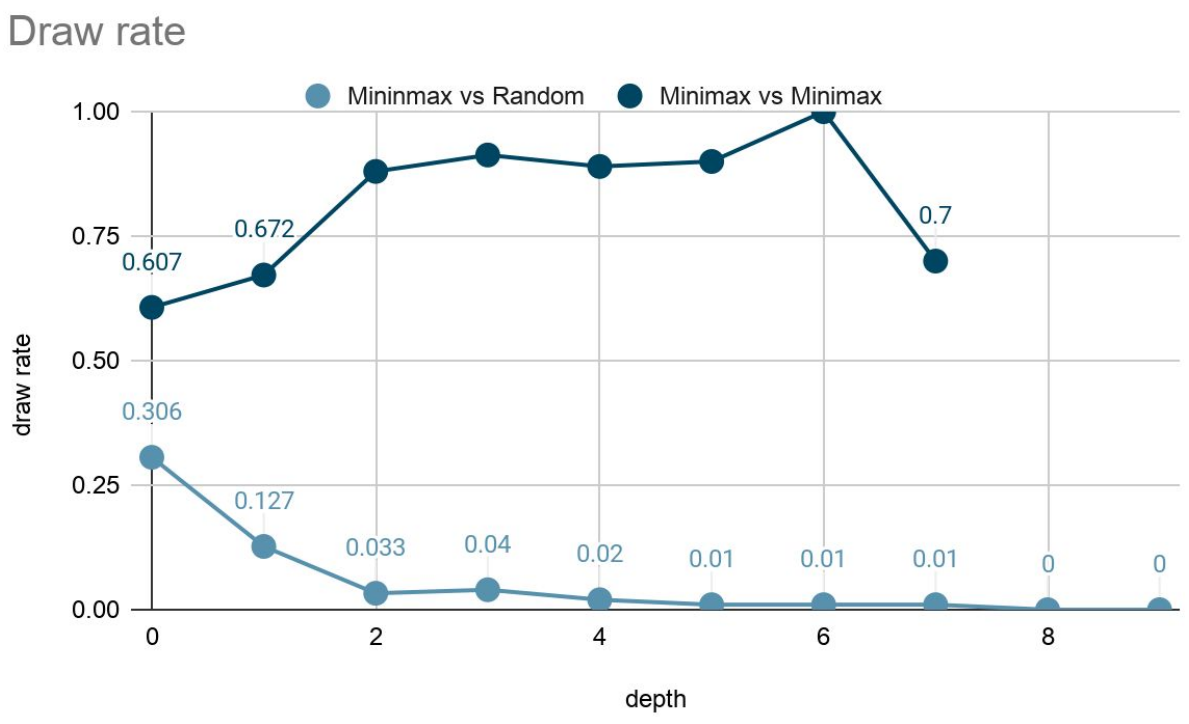
2. minimax (evaluation function 1) vs minimax (evaluation function 1).

It produced the following results.

for the complete set of results refer to this sheet







Conclusions

- ***MinimaxVsMinimax*** took more time than ***MinimaxVsRandom***.
- As the depth increases, the win rate of ***MinimaxVsRandom*** increases and the draw rate decreases.
- As the depth increases, the win rate of ***MinimaxVsMinimax*** decreases and the draw rate increases.

Comparison between evaluation functions

Evaluation function 2 vs Evaluation function 1				
depth	win rate	draw rate	losing rate	time (seconds/game)
0	0.247	0.578	0.175	0.037
1	0.204	0.658	0.138	0.328
2	0.072	0.77	0.158	1.06
3	0.045	0.885	0.07	5.28
4	0.18	0.7	0.12	21.12

(Evaluation function 2 + sum distances) vs (Evaluation function 1 + sum distances)				
depth	win rate	draw rate	losing rate	time (seconds/game)
0	0.416	0.377	0.207	0.039
1	0.29	0.508	0.202	0.324
2	0.25	0.498	0.252	0.94
3	0.23	0.66	0.11	4.5
4	0.28	0.6	0.12	15.56
5	0.3	0.65	0.05	57.2

Conclusions

- Evaluation function 2 is better than evaluation function 1.
- Adding the sum of distances evaluation decreased the draw rate and increased both the win rate and the losing rate.