Assignment 2: Artificial intelligence.

Group 2392. Pair 1.

Kevin de la Coba and Juan Luis Sanz.

1. Minimax and alpha-beta pruning

a. Implementation

i. Which tests have been designed and applied to determine whether the implementation is correct?

In the demos of TicTacToe and Reversi, we created players which were using the MinimaxAlphaBetaStrategy and the minimaxStrategy. First, we did matches between the minimaxStrategy players, then we did the same matches with the MinimaxAlphaBetaStrategy. We came to the conclusion both strategies should choose the same movements, but the ones of alpha-beta should be faster.
So, we checked that the movements were always the same.

ii. Design.

We followed the already implemented minimaxStrategy class design and we used the assignment pseudocode for the MinimaxAlphaBetaStrategy. The differences between these 2 strategies pseudocodes are the pruning and “alpha” and “beta” values, so our design focused on implementing the pruning and naming variables correctly for the min_value and max_value functions.

iii. Implementation.

We tried to follow as strictly as we could the pseudocode given. We can see that the algorithm is the same until the pruning, so we copied the minimax implementation until that part and then we implemented the pruning. Small changes were made on that copy.
To be more specific, in the function next_move, we created the variables alpha and beta, and we initialized them to -∞, ∞ respectively. As we are the max, we have to update the alpha by checking the successor’s min.
In the min_value and max_value functions we have to update beta and alpha respectively and for the pruning we must compare if the biggest value taken is bigger than beta (max_value case) and if the smallest value taken is smaller than alpha (min_value case).

b. Efficiency of alpha-beta pruning.

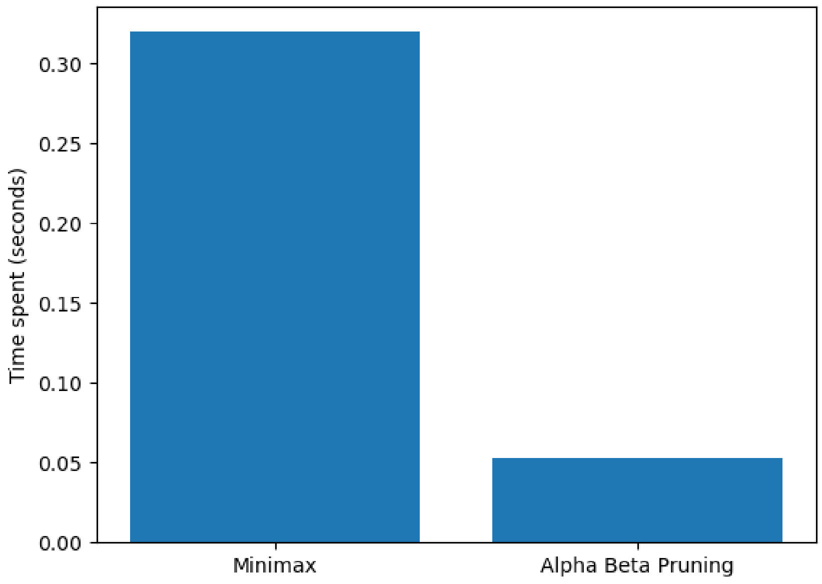
i. Complete description of the evaluation protocol.

We created the file tester.py, in this one we first define the heuristic used for testing and then we buil a game state from the TicTacToe game in which the board is empty. We chose this state (empty board state) because is the one in which can choose more movements. Once we built the game state, we started testing the strategies using the timeit library.
We test each strategy 10 times, from depths 1 to 4. After each depth, we plot a histogram with the time spent by minimaxStrategy and MinimaxAlphaBetaStrategy. After all the depths, we plot how many times is alpha-beta faster than minimax in each depth. This last test is the computer independent test.

ii. **Tables in which times with and without pruning are reported.**

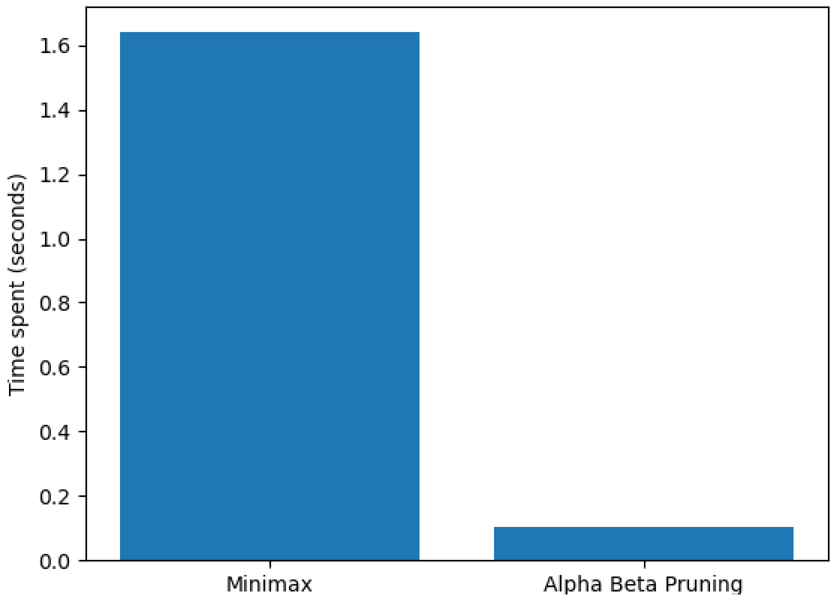
In the following histograms we can see both strategies and the time spent by each one in the Y axis.

Depth 3:



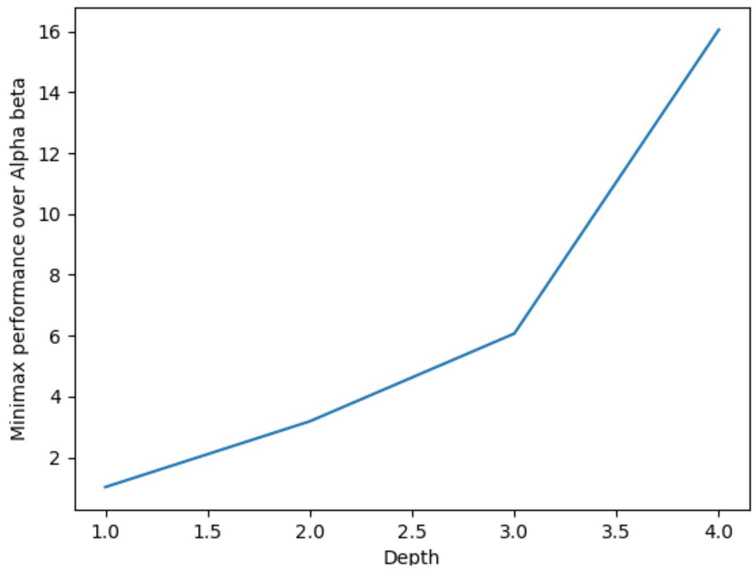
- Minimax time was 0.3202123seconds.
- Alpha-beta pruning time 0.052769269999999994 seconds.

Depth 4:

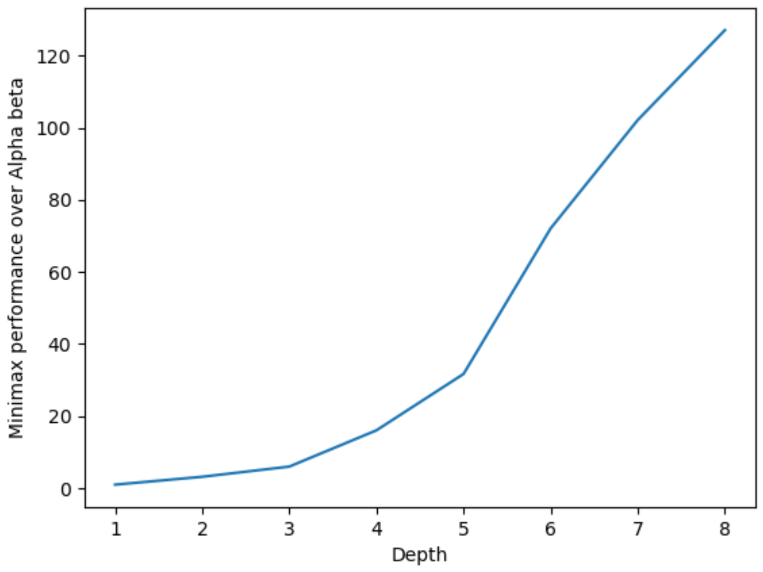


- Minimax time was 1.6395015999999998 seconds.
- Alpha-beta pruning time was 0.10207853999999941 seconds.

iii. **Computer independent measures of improvement.**
On the Y axis we can see how many times is alpha beta faster than minimax. On the X axis we can see the depths for each value.



From depths 1 to 4, the time spent of minimax divided by the time spent of alpha-beta.



From depths 1 to 8, the value of the division is getting bigger, this means that the gap between these 2 strategies is growing.

iv. **Correct, clear, and complete analysis of the results.**
For depths 3 and 4 we could see that the bigger the depth, the bigger the difference between the strategies. This is because the “tree” gets bigger with the depth and minimax has to compare more nodes than alpha-beta (thanks to the pruning).

v. **Other relevant information.**
We saw that just in the depth 1, alpha-beta pruning has a worst performance than minimax, this is because at that depth, the time spent to check if we have to prune is bigger than the overall time spent by minimax only comparing nodes.

2. Documentation of the design of the heuristic

a. Review of previous work on Reversi strategies, including references in APA format.

Basically, our strategies were designed by us by playing the Reversi game and finding possible ways to get an advantage over the other player. In the next section it is explained how we reached those designs and a brief explanation of each of them.

b. Description of the design process:

i. How was the design process planned and realized?

Just before starting with the design and implementation of the heuristics, we played one against the other on an online Reversi to really understand how it worked and to learn some strategies to win more easily. We realized that when having the control of the corners, it was impossible for the other player to "eat" that token and that gave a really important advantage. Also, we realized that having the control of the positions which were in the sides of the board also gave some kind of advantage over the other player.

We did not have a clear plan for the heuristics, but at first, we started designing simple heuristics that fulfill the requirements to have a first contact and then improving them. One of the first was the SimpleKJ in which we designed one simple heuristic with the previous idea we had, following the premise of wall cells and corners give advantage so they reduce the total remaining cost more than any other cell. This first approach was nice, but we needed a more complex design to improve it.

Then, we found on the internet a weighed board which specified more each cell than the previous design, giving a specific value depending on how likely was to get an advantage over the other player if the new token was placed there. We coded an approach to that weighted board in an heuristic called WeightedBoardKJ and we expected very good results in the ranking because it was way better than SimpleKJ and it beat the given test heuristics, but it was not as good as we thought and it did not have good results.

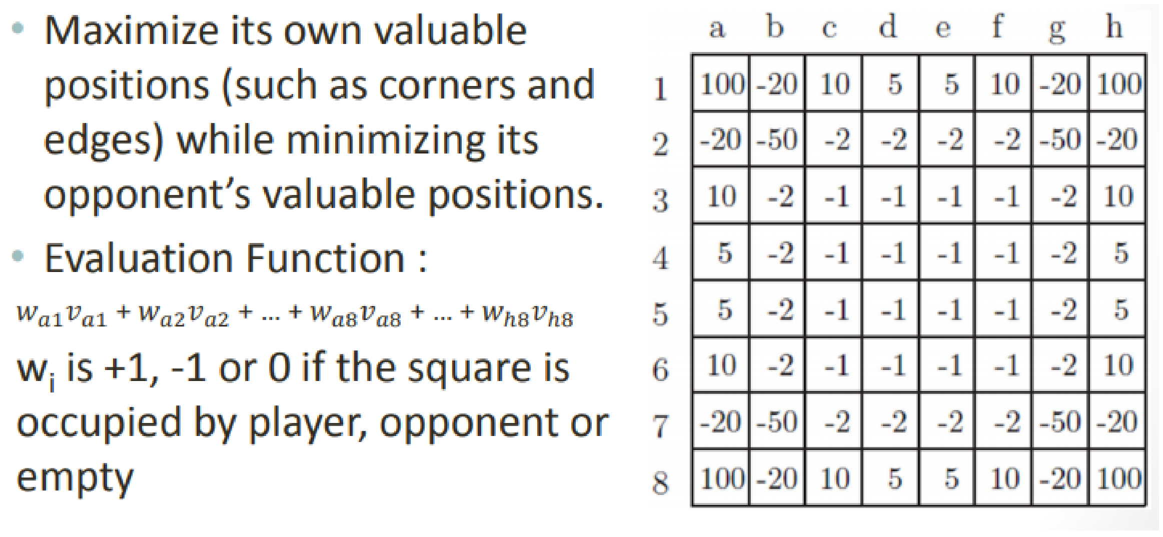
The last heuristic we designed (MaxCellsKJ) was a very simple but effective one, especially when the game state was advanced because it followed a score method in which the more cells the player had, the better. Although it was way simpler than the previous one, it had really good results in the ranking.

ii. Did we have a systematic procedure to evaluate the heuristics designed?

For evaluating the heuristics, we first used the demo_reversi.py and demo_tictactoe.py (the heuristics are designed for the reversi game but we tested the in demo_tictactoe to see if we didn't have any kind of syntax error) files adding a new player with our new heuristic and also with the tournament.py in which we could make all of the heuristics given and our heuristics compete against each other and see which of them was better.

iii. Did we take advantage of strategies developed by others? If these are publicly available, provide references in APA format; otherwise, include the name of the person who provided the information and give proper credit of the contribution as "private communication".

For the Reversi strategies we mainly focused on the idea of having a weighted board, searching on the internet we found a weighted board like this:



- Maximize its own valuable positions (such as corners and edges) while minimizing its opponent's valuable positions.
- Evaluation Function :

$w_{a1}v_{a1} + w_{a2}v_{a2} + \ldots + w_{a8}v_{a8} + \ldots + w_{h8}v_{h8}$

w_i is +1, -1 or 0 if the square is occupied by player, opponent or empty

	a	b	c	d	e	f	g	h
1	100	-20	10	5	5	10	-20	100
2	-20	-50	-2	-2	-2	-2	-50	-20
3	10	-2	-1	-1	-1	-1	-2	10
4	5	-2	-1	-1	-1	-1	-2	5
5	5	-2	-1	-1	-1	-1	-2	5
6	10	-2	-1	-1	-1	-1	-2	10
7	-20	-50	-2	-2	-2	-2	-50	-20
8	100	-20	10	5	5	10	-20	100

Parekh, P., Singh Bhatia, U., & Sukthankar, K. Othello/Reversi using Game Theory techniques (p. 6). PDF. Taken from http://play-othello.appspot.com/files/Othello.pdf

c. Description of the final heuristic submitted.

All of the three heuristics proposed on the 2b-i section have been uploaded, but if we had to choose one it would be the MaxCellsKJ strategy which improves its results as it advances the game state, and we can have a real advantage if in the tournament we start from an advanced board game state. This heuristic beats by far all the reference heuristics that were tested and exposed in the ranking so it is a good one to choose.

The operation of the MaxCellsKJ heuristic was previously explained, but to explain it in more detail, we will say that each cell cost is one at the beginning and in each step the board is refreshed, it is calculated the sum of all cells in the board minus the ones occupied by us. The less that number is, the better.