Heuristic analysis

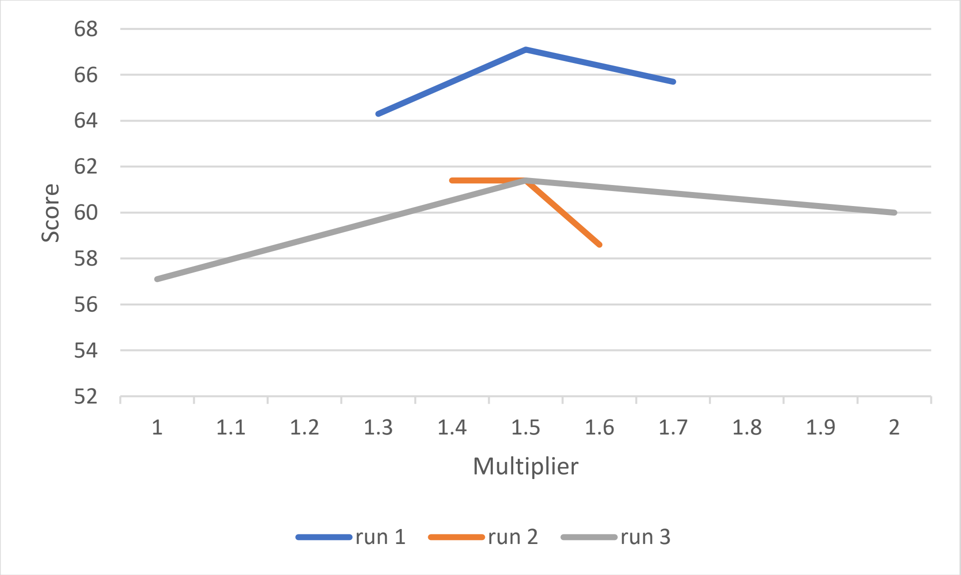
```
                               *************************
                                    Playing Matches
                               *************************

 Match #     Opponent      AB_Improved    AB_Custom    AB_Custom_2  AB_Custom_3
                            Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1         Random         7  |   3    10  |   0     7  |   3     8  |   2
    2         MM_Open        5  |   5     7  |   3     9  |   1     8  |   2
    3        MM_Center       8  |   2     8  |   2     8  |   2     6  |   4
    4       MM_Improved      7  |   3     7  |   3     6  |   4     6  |   4
    5         AB_Open        5  |   5     7  |   3     5  |   5     5  |   5
    6        AB_Center       7  |   3     7  |   3     7  |   3     6  |   4
    7       AB_Improved      3  |   7     7  |   3     4  |   6     4  |   6
--------------------------------------------------------------------------------
           Win Rate:       60.0%         75.7%        65.7%        61.4%
```

Custom score 3

I started with a simple number_of_players_moves - 1.5 * number_of_opponents_moves like the one the lecture talked about, but I wondered if 1.5 was the optimal multiplier.

So a started testing 7 different multipliers between 1 and 2, the result varied between the different runs, so I included all of them:



But no matter how much the varied, 1.5 was always the best. There is no difference between 1.4 and 1.5, but I think it is because there aren't enough examples.

But I settled with 1.5 as the multiplier for my next score functions.

The overall score for custom score 1 is only a little better then AB_Improved, and started to lack behind the more advanced the opponent got. But is a start for my more advanced score functions.

Custom score 2

In this version, reduced the score if the player if it got to close to the edge. players_moves - player_edge_bonus * 0.5 - (1.5 * opponents_moves). I did it because as closer you get to the edge, the fewer fields the agent is able to take, and will automatically go to the middle, but I wanted the agent to use all the board. I improved the overall score from 61.4% to 65.7%. The agent got worse if I weighted the edge score too much.

This score function was significant better against the more advanced score functions compared to custom score 3. And is better or the same then AB_Improved (only 1 was worse).

Custom score 1

I build on top of score 2 by adding the edge score for both agent, as well as adding the length between the agents to the score. I put up a grid search for finding the best multipliers for "player_edge_bonus", "opponents_edge_bonus" and "space_between_players". After searching through 221 combinations, I found the multipliers 0.4, 0.4 and 0.6 worked best. players_moves + player_edge_bonus * 0.4 - (1.5 * opponents_moves) + opponents_edge_bonus * 0.4 + max(5, space_between_players)*.6. I did not search for a new multiplier for opponents_moves because the number of combinations exploded. I got the agent overall score up to 75.7% with those multipliers.

This score function is better or the same then AB_Improved and the other score function in every single way and manage to win every single time over the random agent.

I recommend this function for following reasons:

- It has the best overall score of my custom score functions, 15% and 10% better compared to custom score 2 and custom score 1.
- It is simple and fast, there is no loops or complex math. If a score function gets to complicated and slow, then it won't give the iterative deepening enough time to search deep enough.
- It has more insight to the game because it got more variable to make up a score.