Heuristics Analysis for AIND-Isolation

Function 1 (Best function) :

Custom function 1 uses a linear combination of difference between the agent's moves and opponent's moves, and the distance away from the center of the board.

This stems from the following ideas:

- If a player is has significantly more moves than his opponent, the odds are in his favor
- Being away from the center of the board prevents one from being cut-off

Both scores are given equivalent weighted and summed together.

Function 2:

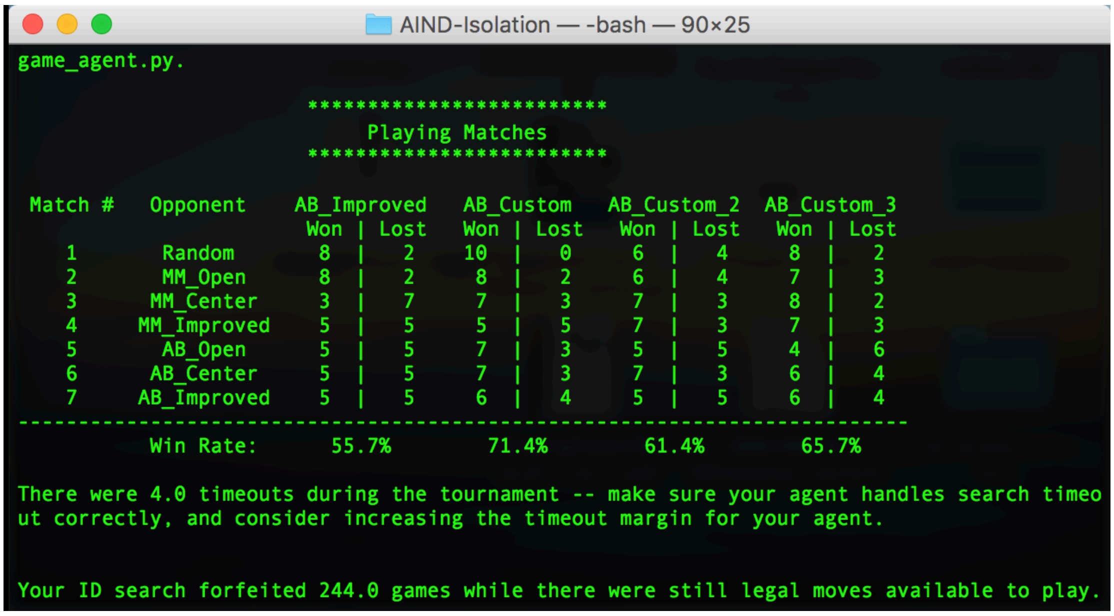
Custom function 2 stems from the notion of being cornered on the board.

This is done by first finding the closest corner to the agent. Next, we find the distance between opponent and the aforementioned closest corner. Finally, the score is given by an exponential of the difference between the agent's distance and the opponent's distance to the corner. A higher score means that the agent is further away and hence has a higher chance to corner his opponent.

Function 3:

Custom function 3 is an aggressive implementation of the difference between the agent's moves and the opponent's moves, scaled by the number of empty spaces left on the board. This stems from the idea that having significantly more moves than your opponent when there are less spaces left presents a higher opportunity to win the game.

Final Simulation Outcome



```
game_agent.py.

                         *************************
                              Playing Matches
                         *************************

 Match #     Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                          Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1         Random       8  |   2    10  |   0     6  |   4     8  |   2
    2         MM_Open      8  |   2     8  |   2     6  |   4     7  |   3
    3        MM_Center     3  |   7     7  |   3     7  |   3     8  |   2
    4       MM_Improved    5  |   5     5  |   5     7  |   3     7  |   3
    5         AB_Open      5  |   5     7  |   3     5  |   5     4  |   6
    6        AB_Center     5  |   5     7  |   3     7  |   3     6  |   4
    7       AB_Improved    5  |   5     6  |   4     5  |   5     6  |   4
--------------------------------------------------------------------------
           Win Rate:      55.7%        71.4%        61.4%        65.7%

There were 4.0 timeouts during the tournament -- make sure your agent handles search timeo
ut correctly, and consider increasing the timeout margin for your agent.


Your ID search forfeited 244.0 games while there were still legal moves available to play.
```

Above is a snapshot of the final simulation outcome running the game playing agent with the given heuristics. From this result, we can see that having an algorithm such as Minimax already improves our win rate to be above 50%. Having additional heuristics or human input as to how to evaluate board states can further improve the win rates.

Finally, the custom_score (first function) is chosen to be the optimal choice for this exercise due to its high win-rate, low complexity and strategy diversity. Also, the run-time using such a function was not significantly slower compared to other strategies.

However, whilst I am convinced that having heuristics and a minimax/AB-pruning strategy improves our chances of winning the game, I noticed that the performance of the agents were not always that consistent. Some heuristics performed better than others in other simulations. However, they maintined a more than 50% win rate most of the time.

Additionally, I believe that whilst heuristics and Minimax/AB-pruning strategies have improved our win rates, the inconsistency could be due to the horizon effect when searching for the best possible move. Since we are unable to search to the end of the game tree, the best possible moves might not be discovered, resulting in unstable win rates. The agent can be further improved by reducing the search space to enable searching to the end of the game tree.