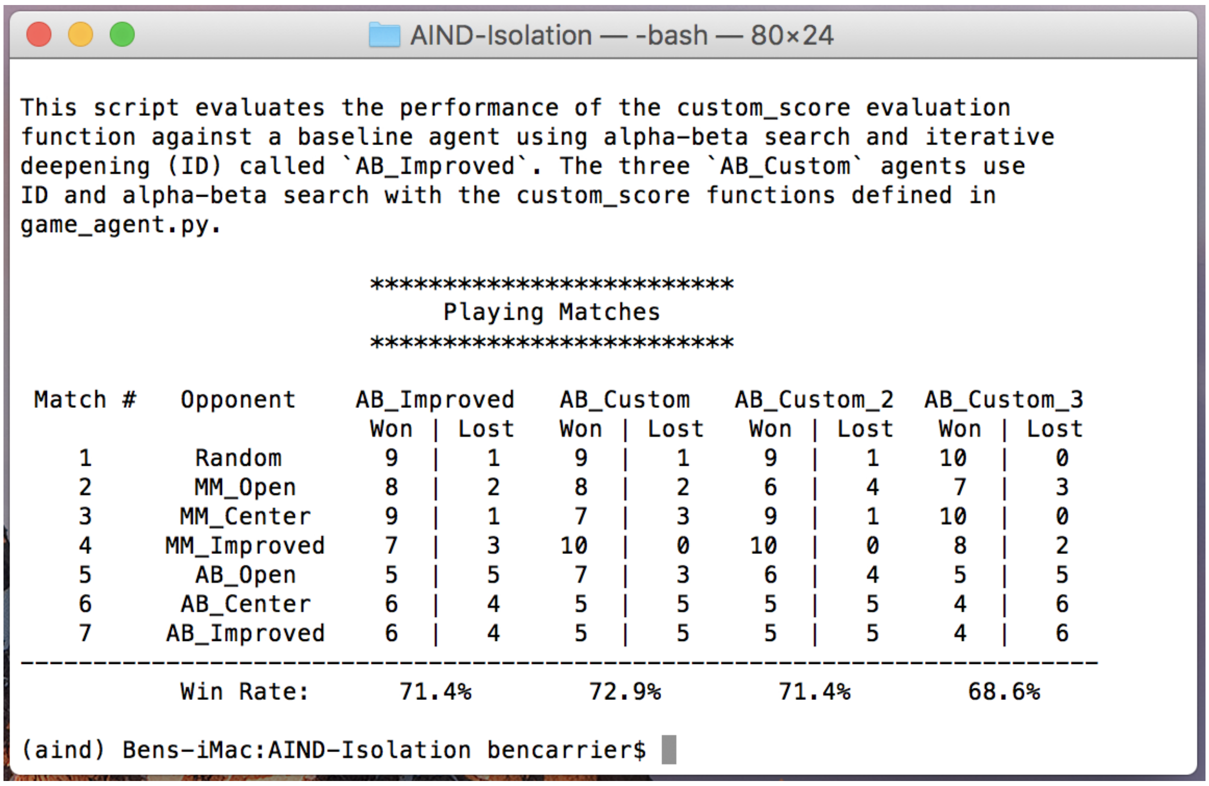
Heuristic Analysis



```
This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
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       9  |   1     9  |   1     9  |   1    10  |   0
    2       MM_Open      8  |   2     8  |   2     6  |   4     7  |   3
    3      MM_Center     9  |   1     7  |   3     9  |   1    10  |   0
    4     MM_Improved    7  |   3    10  |   0    10  |   0     8  |   2
    5       AB_Open      5  |   5     7  |   3     6  |   4     5  |   5
    6      AB_Center     6  |   4     5  |   5     5  |   5     4  |   6
    7     AB_Improved    6  |   4     5  |   5     5  |   5     4  |   6
--------------------------------------------------------------------------
           Win Rate:      71.4%        72.9%        71.4%        68.6%

(aind) Bens-iMac:AIND-Isolation bencarrier$
```

Custom Score 1

own_moves – (move_count/blank_spaces)*opp_moves))

This heuristic starts off by treating own_moves and opp_moves fairly equally, but as the game progresses, the heuristic becomes more and more aggressive by targeting boards where the opponent has less moves. This heuristic came out with the best win ratio of 72.9%. Although most of its wins came against the mini-max algorithm, it came out with a 56.6% win ratio over the Alpha-Beta version and never lost.

Custom Score 2

own_moves – (2*opp_moves)

This heuristic favours boards giving the opponent the least possible moves. It had a 71.4% win ratio, which is the same as the AB_Improved algorithm. It performed better against the minimax algorithm than the alpha-beta version.

Custom Score 3

(2*own_moves) – opp_moves

This heuristic favours boards where the agent has the most available moves over the opponent. Whereas it performed slightly better than the AB-Improved algorithm against mini-max, it performed worse against the alpha-beta version.

Conclusion

My chosen algorithm would be Custom Score 1, as it performs consistently well against all variations of algorithm and heuristics. Its worst score was a 5 – 5 draw, meaning it never lost over the 7 opponents and came out with highest win-ratio of all heuristics. It appears that a heuristic favouring space at the start of the game and restricting opponents moves at the end of the game performs better than a heuristic favouring its own moves or the opponents moves throughout the entire game. It is also easy to implement due to its low complexity as all you need to know is the number of own_moves, opp_moves, total_moves and blank_spaces on the game board, and perform a simple mathematical equation. A low complexity heuristic also means it is fast, allowing it to search deeper into the game tree. A more complex heuristic may give a better estimate, but will have to trade off with a shallower search due to the time it takes for the heuristic to calculate. Searching deeper in the game tree can be more beneficial, than a better estimate at a shallower level in the tree.