Heuristic Review

For this experiment, I'm using a 13' macbook pro to run this task, the heuristic I use is as followed. Note that due to the computation ability of a laptop is limited, the AB_Improved column can be treated as a benchmark of how well the different heuristics applied.

Heuristic1:

score = (my_moves - 2 * my_opp)

This is the heuristic given in the lecture, it is the traditional game scheme that to find as much blank spaces as possible while chasing your opponent to the corner, which makes him isolated.

Heuristics2:

score = distance to the center

The second heuristics is to let the player stay at the center of the board as much as possible. Since the center of the board has more symmetry and more places to move. If the player stays at the center, it indicates that he gains more chances and choice to go than the one at the corner of the board.

Heuristics3:

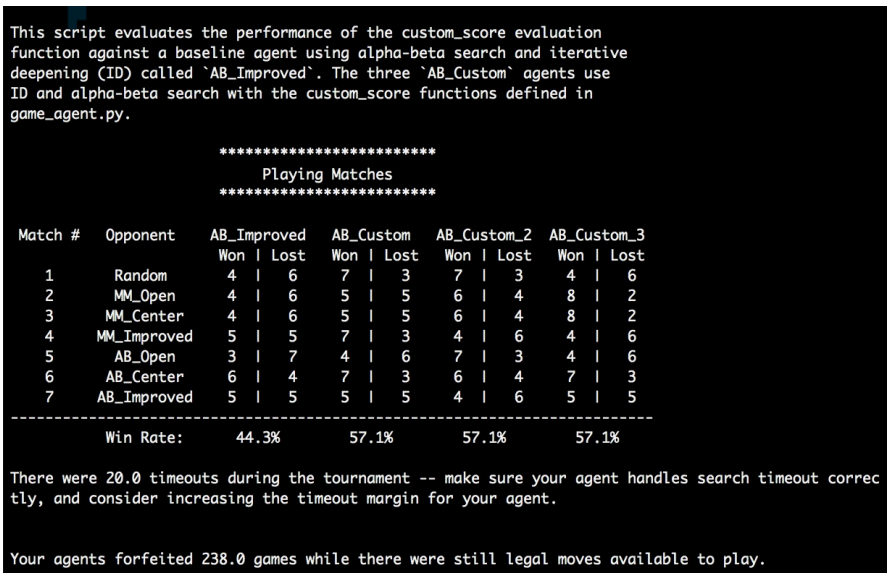
score = add the previous two

This heuristic combined the beneficial of both previous heuristics. Since the score is additive, simply add them together generate a score with equal weighted of both scheme. However, sometimes the unequal weigh may make the heuristics performs better.

The difference between following result is time-limit is set to 150ms, 300ms and 600ms.

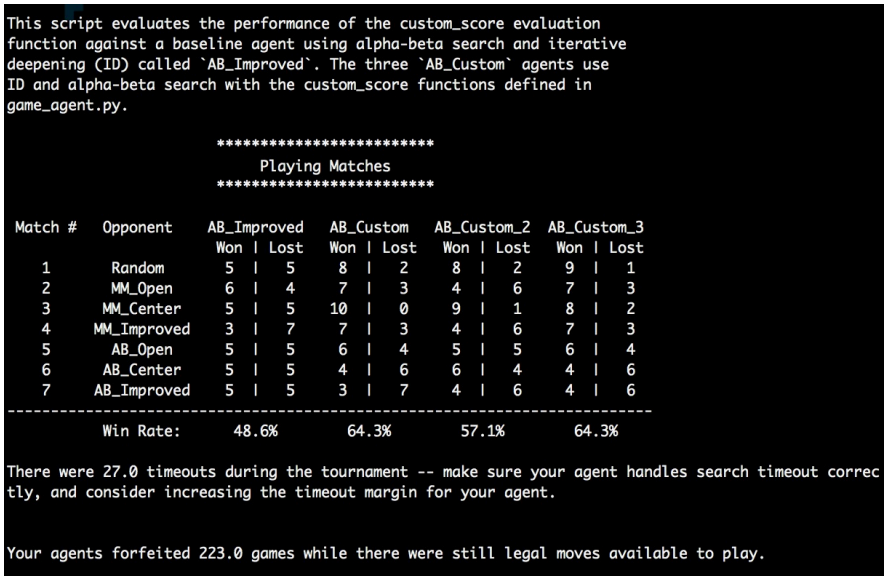
```
This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

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

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       7  |   3     7  |   3     7  |   3     5  |   5
    2       MM_Open      4  |   6     7  |   3     5  |   5     4  |   6
    3      MM_Center     6  |   4     6  |   4     5  |   5     6  |   4
    4     MM_Improved    3  |   7     6  |   4     6  |   4     4  |   6
    5       AB_Open      4  |   6     5  |   5     3  |   7     5  |   5
    6      AB_Center     4  |   6     5  |   5     4  |   6     4  |   6
    7     AB_Improved    5  |   5     4  |   6     5  |   5     2  |   8
--------------------------------------------------------------------------
          Win Rate:      47.1%        57.1%        50.0%        42.9%

There were 44.0 timeouts during the tournament -- make sure your agent handles search timeout correc
tly, and consider increasing the timeout margin for your agent.


Your agents forfeited 210.0 games while there were still legal moves available to play.
```

Fig1. 150ms time limit



```
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       4  |   6     7  |   3     7  |   3     4  |   6
    2       MM_Open      4  |   6     5  |   5     6  |   4     8  |   2
    3      MM_Center     4  |   6     5  |   5     6  |   4     8  |   2
    4     MM_Improved    5  |   5     7  |   3     4  |   6     4  |   6
    5       AB_Open      3  |   7     4  |   6     7  |   3     4  |   6
    6      AB_Center     6  |   4     7  |   3     6  |   4     7  |   3
    7     AB_Improved    5  |   5     5  |   5     4  |   6     5  |   5
--------------------------------------------------------------------------
          Win Rate:      44.3%        57.1%        57.1%        57.1%

There were 20.0 timeouts during the tournament -- make sure your agent handles search timeout correc
tly, and consider increasing the timeout margin for your agent.


Your agents forfeited 238.0 games while there were still legal moves available to play.
```

Fig2. 300ms time limit



```
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       5  |   5     8  |   2     8  |   2     9  |   1
    2       MM_Open      6  |   4     7  |   3     4  |   6     7  |   3
    3      MM_Center     5  |   5    10  |   0     9  |   1     8  |   2
    4     MM_Improved    3  |   7     7  |   3     4  |   6     7  |   3
    5       AB_Open      5  |   5     6  |   4     5  |   5     6  |   4
    6      AB_Center     5  |   5     4  |   6     6  |   4     4  |   6
    7     AB_Improved    5  |   5     3  |   7     4  |   6     4  |   6
--------------------------------------------------------------------------
          Win Rate:      48.6%        64.3%        57.1%        64.3%

There were 27.0 timeouts during the tournament -- make sure your agent handles search timeout correc
tly, and consider increasing the timeout margin for your agent.


Your agents forfeited 223.0 games while there were still legal moves available to play.
```

Fig3. 600ms time limit

There still some timeout happened due to the performance of my laptop, although the search tree is pruned, the agent still can't go deep enough to find the good result. By doubling or even quadruple the time-limit, the average performance of AB_Improved does not change very much. For heuristic1, double the time-limit doesn't change anything while quadruple it makes the winning rate higher. For heuristic2, doubling time-limit do change the winning rate while double it again does not improve anything. And for the heuristic3, although it performs not good with very short time limit. Its accuracy does become better and better after giving more search time. And it really combined the benefit of the previous two heuristic.

So, in this case, I should choose the heuristic3 as the best solution to this experiment.