Environment specifications:

- Computer: 2016 MacBook Pro
  - Processor: 2.4 GHz Intel Core i7
  - Memory: 16 GB
- IDE: PyCharm
- tournament.py: NUM_MATCHES = 10

HEURISTIC ANALYSIS

For this project, I implemented the following heuristics:

1. **Weighted improved_score** – This heuristic modifies the improved_score heuristic defined in sample_players.py to add a weight to the #_opp_moves and demonstrate a more aggressive playing style.
2. **The distance of agent from opponent** – This heuristic makes the agent create a space between it and its opponent, maximizing the value of a move the further away it is from its opponent.
3. **The number of the agent's moves plus the number of blank spaces**

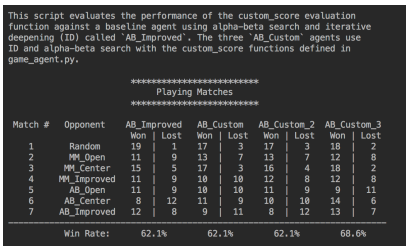
CHOOSING THE WEIGHT FOR WEIGHTED IMPROVED_SCORE HEURISTIC

In order to maximize win rates against the testing agent, I ran through various weights ranging from 1.3 to 2.

The custom_scores were first modified to take in the following values: 1.3, 1.6, 2.0. This was to test whether a more aggressive playing style would maximize wins for the gaming agent. The results for this are as follows:

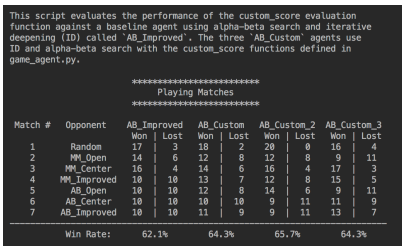
Weights	**Run #1 Win Rate (%)**	**Run #2 Win Rate (%)**	**Run #3 Win Rate (%)**	**Average Win Rate (%)**
ID_improved (control) = 1	62.1	60.0	62.1	61.4
custom_score = 2.0	62.1	62.1	64.3	62.8
custom_score_2 = 1.6	62.1	62.9	65.7	63.6
custom_score_3 = 1.3	68.6	59.3	64.3	64.1

Run 1



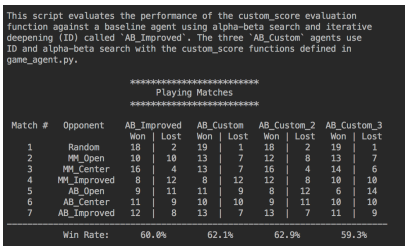
```
This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

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

Match #   Opponent      AB_Improved     AB_Custom     AB_Custom_2     AB_Custom_3
                         Won | Lost    Won | Lost    Won | Lost    Won | Lost
    1       Random        19  |  1      17  |  3      17  |  3      18  |  2
    2      MM_Open        11  |  9      13  |  7      13  |  7      12  |  8
    3    MM_Center        15  |  5      17  |  3      16  |  4      18  |  2
    4   MM_Improved       11  |  9      10  |  10     12  |  8      12  |  8
    5     AB_Open         11  |  9      10  |  10     11  |  9      9   |  11
    6    AB_Center         8  |  12     11  |  9      10  |  10     14  |  6
    7   AB_Improved       12  |  8      9   |  11     8   |  12     13  |  7
--------------------------------------------------------------------------------
          Win Rate:      62.1%          62.1%          62.1%          68.6%
```

Run 2



```
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        17  |  3      18  |  2      20  |  0      16  |  4
    2      MM_Open        14  |  6      12  |  8      12  |  8      9   |  11
    3    MM_Center        16  |  4      14  |  6      16  |  4      17  |  3
    4   MM_Improved       10  |  10     13  |  7      12  |  8      15  |  5
    5     AB_Open         10  |  10     12  |  8      14  |  6      9   |  11
    6    AB_Center        10  |  10     10  |  10     9   |  11     11  |  9
    7   AB_Improved       10  |  10     11  |  9      9   |  11     13  |  7
--------------------------------------------------------------------------------
          Win Rate:      62.1%          64.3%          65.7%          64.3%
```

Run 3



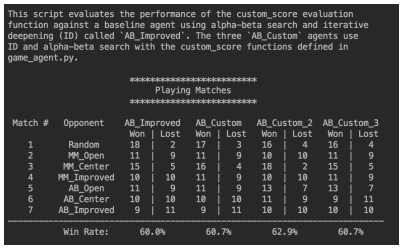
```
This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

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

Match #   Opponent      AB_Improved     AB_Custom     AB_Custom_2     AB_Custom_3
                         Won | Lost    Won | Lost    Won | Lost    Won | Lost
    1       Random        18  |  2      19  |  1      18  |  2      19  |  1
    2      MM_Open        10  |  10     13  |  7      12  |  8      13  |  7
    3    MM_Center        16  |  4      13  |  7      16  |  4      14  |  6
    4   MM_Improved       8   |  12     8   |  12     12  |  8      10  |  10
    5     AB_Open         9   |  11     11  |  9      8   |  12     6   |  14
    6    AB_Center        11  |  9      10  |  10     9   |  11     10  |  10
    7   AB_Improved       12  |  8      13  |  7      13  |  7      11  |  9
--------------------------------------------------------------------------------
          Win Rate:      60.0%          62.1%          62.9%          59.3%
```

Based on the results, we see that as the weight gets closer to 1, we tend to win more games rather than going the opposite direction. However, weight = 1 does the poorest performance than the rest. The weight of 1.6 was chosen due to having more consistent win rates than 1.3 although 1.3 had a better average win rate.

Another test was ran to check if the winning rate could be improved by deviating +/- 0.1 from 1.6 and the results are as follows:

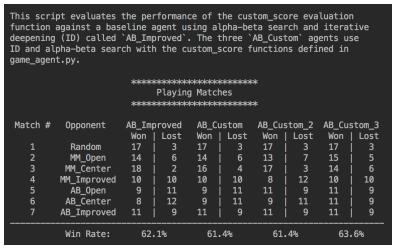
Weights	**Run #1 Win Rate (%)**	**Run #2 Win Rate (%)**	**Run #3 Win Rate (%)**	**Average Win Rate (%)**
ID_improved (control) = 1	60.0	62.1	65.0	62.4
custom_score = 1.7	60.7	61.4	60.7	60.9
custom_score_2 = 1.6	62.9	61.4	65.7	63.3
custom_score_3 = 1.5	60.7	63.6	62.1	62.1

Run 1



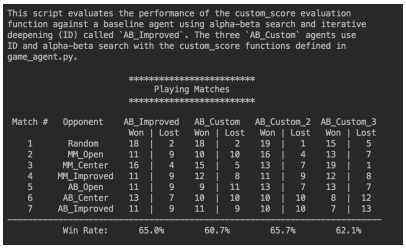
```
This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

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

Match #   Opponent      AB_Improved     AB_Custom     AB_Custom_2     AB_Custom_3
                         Won | Lost    Won | Lost    Won | Lost    Won | Lost
    1       Random        18  |  2      17  |  3      16  |  4      16  |  4
    2      MM_Open        11  |  9      11  |  9      10  |  10     11  |  9
    3    MM_Center        15  |  5      16  |  4      18  |  2      15  |  5
    4   MM_Improved       10  |  10     11  |  9      10  |  10     11  |  9
    5     AB_Open         11  |  9      11  |  9      13  |  7      13  |  7
    6    AB_Center        10  |  10     10  |  10     11  |  9      9   |  11
    7   AB_Improved       9   |  11     9   |  11     10  |  10     10  |  10
--------------------------------------------------------------------------------
          Win Rate:      60.0%          60.7%          62.9%          60.7%
```

Run 2



```
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        17  |  3      17  |  3      17  |  3      17  |  3
    2      MM_Open        14  |  6      14  |  6      13  |  7      15  |  5
    3    MM_Center        18  |  2      16  |  4      17  |  3      14  |  6
    4   MM_Improved       10  |  10     10  |  10     8   |  12     10  |  10
    5     AB_Open         9   |  11     9   |  11     11  |  9      11  |  9
    6    AB_Center        8   |  12     9   |  11     9   |  11     11  |  9
    7   AB_Improved       11  |  9      11  |  9      11  |  9      11  |  9
--------------------------------------------------------------------------------
          Win Rate:      62.1%          61.4%          61.4%          63.6%
```

Run 3



```
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        18  |  2      18  |  2      19  |  1      15  |  5
    2      MM_Open        11  |  9      10  |  10     16  |  4      13  |  7
    3    MM_Center        16  |  4      15  |  5      13  |  7      19  |  1
    4   MM_Improved       11  |  9      12  |  8      11  |  9      12  |  8
    5     AB_Open         11  |  9      9   |  11     13  |  7      13  |  7
    6    AB_Center        13  |  7      10  |  10     10  |  10     8   |  12
    7   AB_Improved       11  |  9      11  |  9      10  |  10     7   |  13
--------------------------------------------------------------------------------
          Win Rate:      65.0%          60.7%          65.7%          62.1%
```

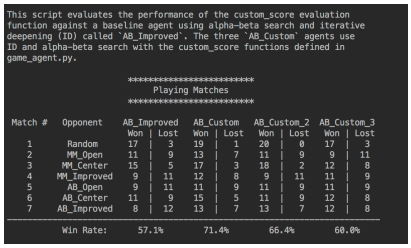
The results from this test solidified the decision to choose 1.6 as the weight for this implementation because of its consistence and higher average win rate.

CUSTOM HEURISTIC PERFORMANCE

After running the custom heuristics against ID_improved, the results showed the custom_score that I would recommend scored the highest against all other heuristics and consistently outperforming the testing agent. The figures below show each run.

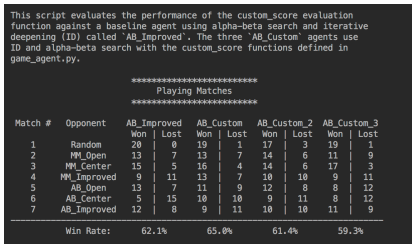
Heuristics	Run #1 Win Rate (%)	Run #2 Win Rate (%)	Run #3 Win Rate (%)	Average Win Rate (%)
AB_improved	57.1	62.1	57.9	59.0
AB_Custom (weighted improved_score)	71.4	65.0	62.9	66.4
AB_Custom_2 (agent's distance from opponent)	66.4	61.4	57.9	61.9
AB_Custom_3(#_my_moves + #_blank_spaces)	60.0	59.3	62.1	60.5

Run 1



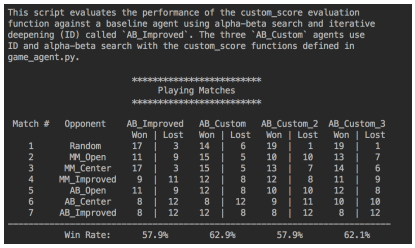
```
This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

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

Match #   Opponent      AB_Improved     AB_Custom     AB_Custom_2   AB_Custom_3
                        Won | Lost      Won | Lost      Won | Lost    Won | Lost
    1     Random        17  |  3        19  |  1        20  |  0      17  |  3
    2     MM_Open       11  |  9        13  |  7        11  |  9       9  | 11
    3     MM_Center     15  |  5        17  |  3        18  |  2      12  |  8
    4     MM_Improved    9  | 11        12  |  8         9  | 11      11  |  9
    5     AB_Open        9  | 11        11  |  9        11  |  9      11  |  9
    6     AB_Center     11  |  9        15  |  5        11  |  9      12  |  8
    7     AB_Improved    8  | 12        13  |  7        13  |  7      12  |  8
--------------------------------------------------------------------------------
          Win Rate:      57.1%        71.4%        66.4%        60.0%
```

Run 2



```
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        20  |  0        19  |  1        17  |  3      19  |  1
    2     MM_Open       13  |  7        13  |  7        14  |  6      11  |  9
    3     MM_Center     15  |  5        16  |  4        14  |  6      17  |  3
    4     MM_Improved    9  | 11        13  |  7        10  | 10       9  | 11
    5     AB_Open       13  |  7        11  |  9        12  |  8       8  | 12
    6     AB_Center      5  | 15        10  | 10         9  | 11       8  | 12
    7     AB_Improved   12  |  8         9  | 11        10  | 10      11  |  9
--------------------------------------------------------------------------------
          Win Rate:      62.1%        65.0%        61.4%        59.3%
```

Run 3



```
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        17  |  3        14  |  6        19  |  1      19  |  1
    2     MM_Open       11  |  9        15  |  5        10  | 10      13  |  7
    3     MM_Center     17  |  3        15  |  5        13  |  7      14  |  6
    4     MM_Improved    9  | 11        12  |  8        12  |  8      11  |  9
    5     AB_Open       11  |  9        12  |  8        10  | 10      12  |  8
    6     AB_Center      8  | 12         8  | 12         9  | 11      10  | 10
    7     AB_Improved    8  | 12        12  |  8         8  | 12       8  | 12
--------------------------------------------------------------------------------
          Win Rate:      57.9%        62.9%        57.9%        62.1%
```

HEURISTIC RECOMMENDATION

As the data shows, the weighted improved_score heuristic performed the best, thus this heuristic is my recommendation. It consistently outperformed the testing agent as well as the other custom_scores, having an average win rate of 66.4%. This heuristic scored the highest win rate out of all tests performed and it performs less calculations than my second-best heuristic (distance from opponent) producing a result in less time.