## Overall perfomance

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

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      137 |  63    136 |  64    131 |  69    130 |  70
    2       MM_Open     128 |  72    138 |  62    124 |  76    120 |  80
    3      MM_Center    150 |  50    139 |  61    151 |  49    149 |  51
    4     MM_Improved   123 |  77    126 |  74    130 |  70    125 |  75
    5       AB_Open     107 |  93    102 |  98    113 |  87    91  |  109
    6      AB_Center    107 |  93    117 |  83    101 |  99    110 |  90
    7     AB_Improved   100 |  100   103 |  97    108 |  92    117 |  83
--------------------------------------------------------------------------
           Win Rate:      60.9%        61.5%        61.3%        60.1%
```

As we can see from above, the three score functions' result is just above the `AB_improved` winning rate, which is the benchmark of the test. I set game number between each set players to be `200`, for minizing the variation on winning rate.

### Custom 1

the score function I used in `custom 1` is

```
player_moves - opponent_moves + 0.1 * player_distance_to_center
```

This function combines both `AB_improved` and `AB_open`. We put less weight on the distance since the perfomance of `AB_improved` is better than `AB_open`, also the scale of the distance is from 0~5, and we don't want it to be the dominating part.

It turns out is slightly better than `AB_improved`, but not far ahead.

### Custom 2

The score function in `custom 2` is

```
players_moves - 2 * opponent_moves
```

This function is similar to `AB_improved`, but we put more weight on minimizing the opponent's available moves. Its perfomance is slight worse than `custom 1` and better than `AB_improved`.

### Custom 3

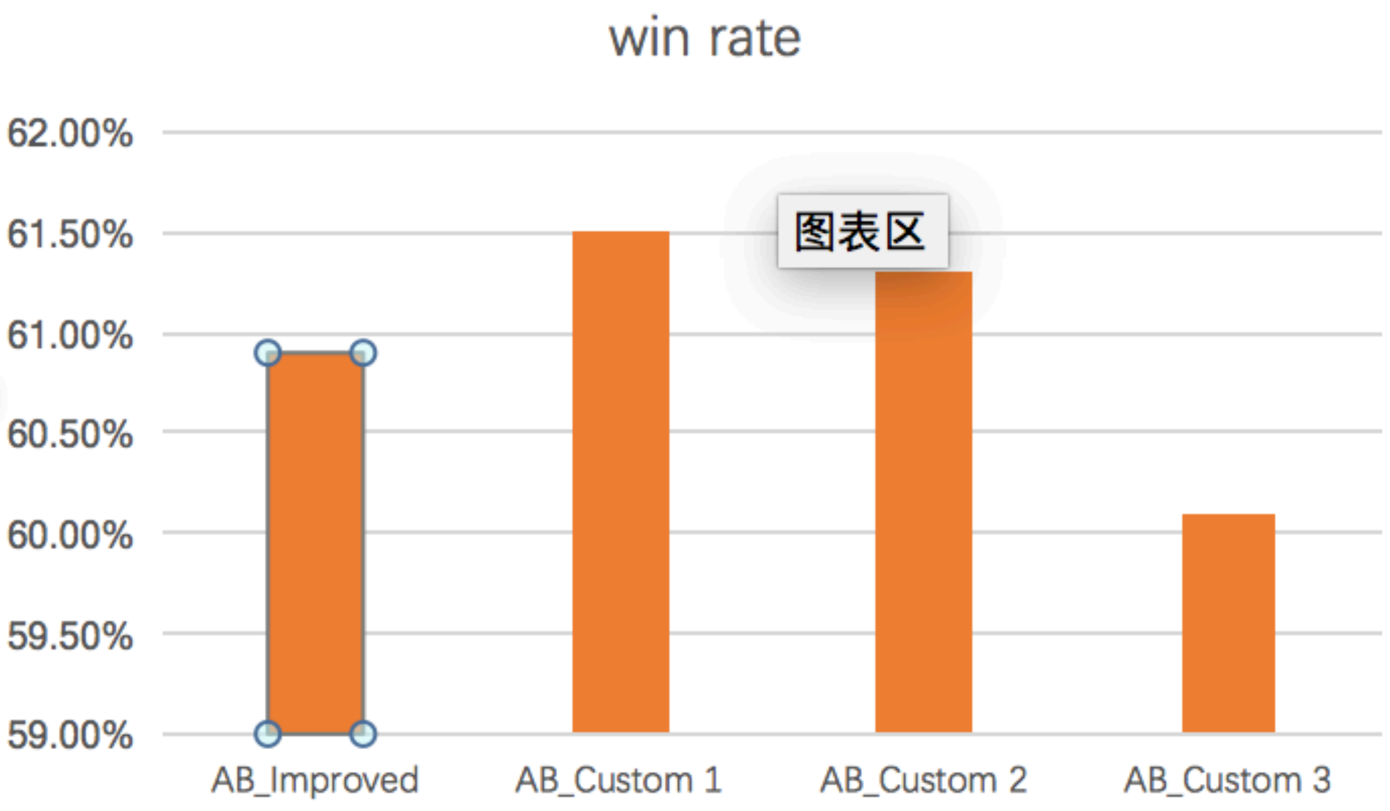
The score funtion I used in `custom 3` is

```
min(player_moves - opponent_moves, player_distance_to_center)
```

This function is another comination of `AB_improved` and `AB_open`. It turns out is slightly worse than `AB improved`.

## Conclusion

For computation complex, `AB_custom 2` clearly is the easiest funtion to compute, and then the `AB_custom 3`. The other two `AB_improved` and `AB_custom 1` are the more difficulty funtions to compute as they involve square and square roots. But on the other hand, they all just functions envlove only constants, which means they are the same when considering level of computational complex. As the result, they will go into the similar depth of the game tree.



For the winning rate, we can see from above, our `AB_custom 1` has the best performance.

To conclude, I would recommond `AB_custom 1` here. As it has the best performance result, with the similar complex level as the other heuristic funcitons.