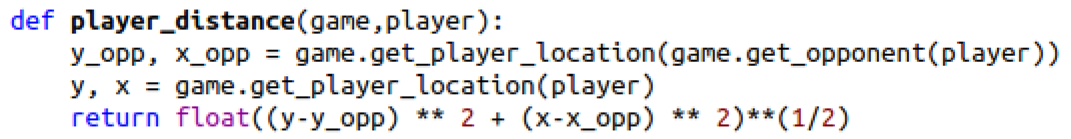
Heuristic Analysis
The basic evaluation function used in all the heuristics was the #my_moves - #opponent_moves. A couple of things were tried to see if the winning rate improved such as:

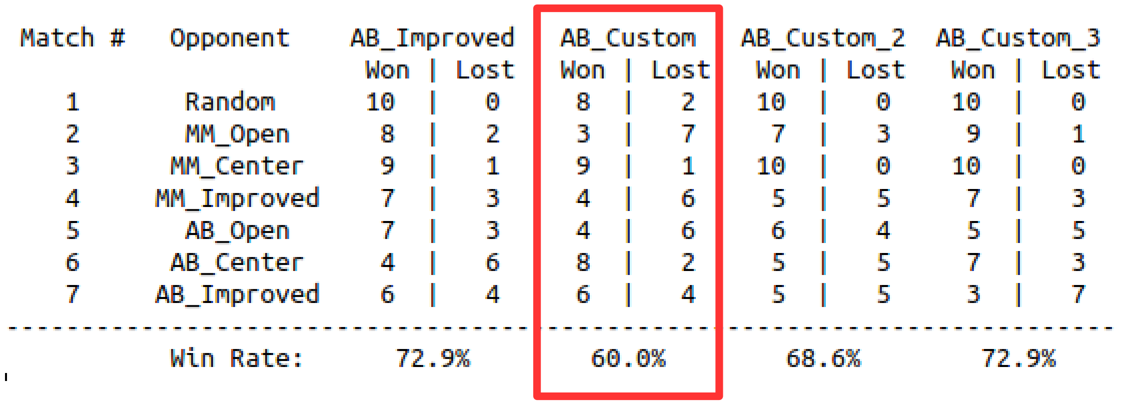
- Avoiding corners
- Following the opponent's moves
- Getting close to the center of the board
- Playing with weights in #my_moves and #opponent_moves

using the tournament.py script, I was able to test different configurations of these ideas and get an improved evaluation function.

custom_score()
Player distance: the idea is to penalize the heuristic function if the player is getting away from the opponent. In other words, we want the player to follow the opponent's moves.

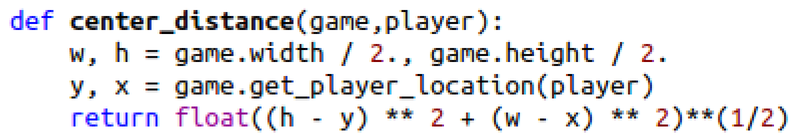


```
def player_distance(game,player):
    y_opp, x_opp = game.get_player_location(game.get_opponent(player))
    y, x = game.get_player_location(player)
    return float((y-y_opp) ** 2 + (x-x_opp) ** 2)**(1/2)
```

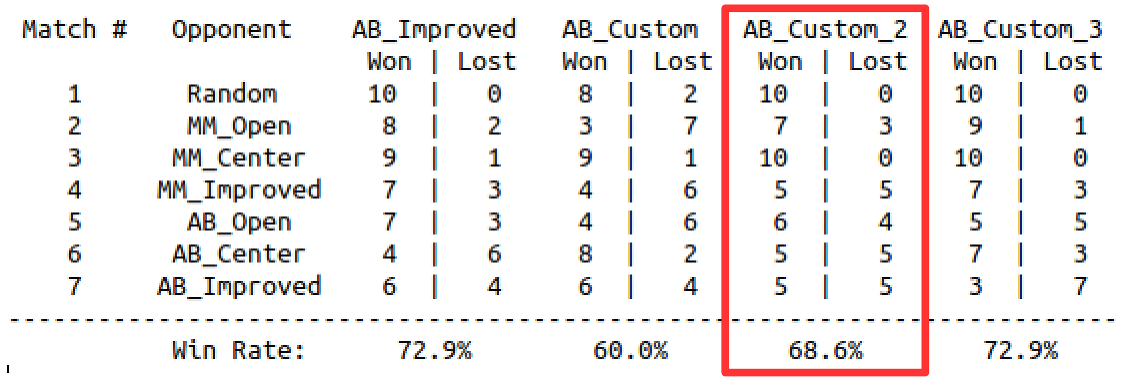


```
 Match #    Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                         Won | Lost   Won | Lost  Won | Lost   Won | Lost
    1        Random      10  |  0      8  |  2    10  |  0     10  |  0
    2        MM_Open      8  |  2      3  |  7     7  |  3      9  |  1
    3       MM_Center     9  |  1      9  |  1    10  |  0     10  |  0
    4      MM_Improved    7  |  3      4  |  6     5  |  5      7  |  3
    5        AB_Open      7  |  3      4  |  6     6  |  4      5  |  5
    6       AB_Center     4  |  6      8  |  2     5  |  5      7  |  3
    7      AB_Improved    6  |  4      6  |  4     5  |  5      3  |  7
--------------------------------------------------------------------------
          Win Rate:       72.9%        60.0%        68.6%        72.9%
```

custom_score2()
Center distance: the idea is to penalize the heuristic function if the player is getting away from the center of the board. The closer it can be to the center, the better, hoping that in the center there are more legal moves in the future.

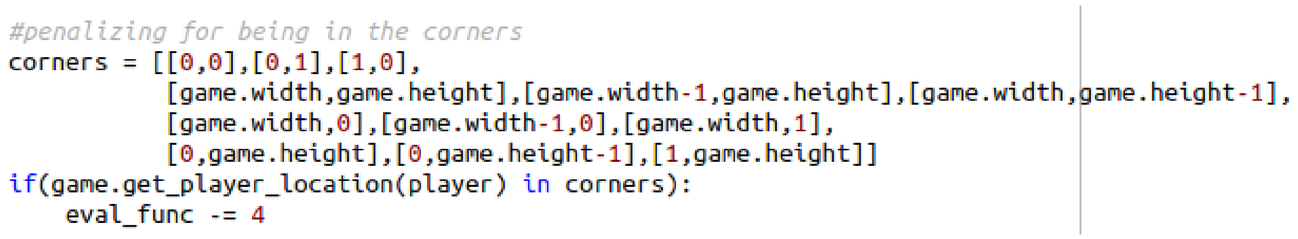


```
def center_distance(game,player):
    w, h = game.width / 2., game.height / 2.
    y, x = game.get_player_location(player)
    return float((h - y) ** 2 + (w - x) ** 2)**(1/2)
```

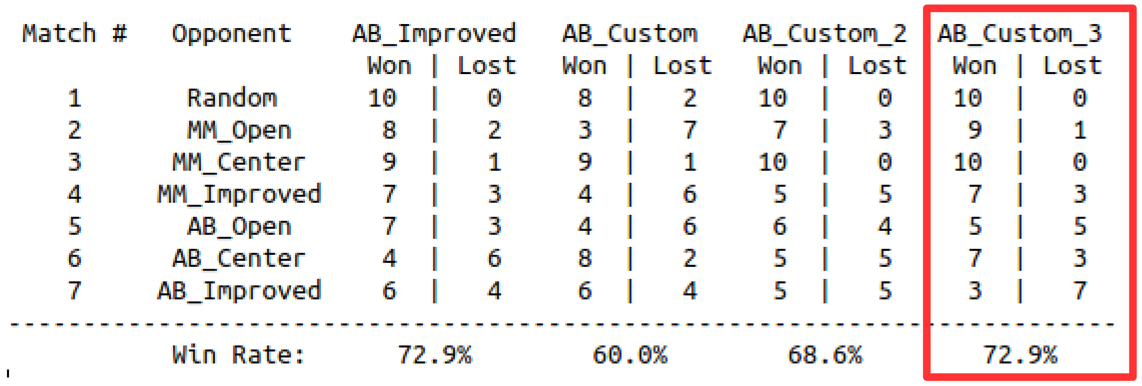


```
 Match #    Opponent     AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                         Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1        Random      10  |   0     8  |   2    10  |   0    10  |   0
    2        MM_Open      8  |   2     3  |   7     7  |   3     9  |   1
    3       MM_Center     9  |   1     9  |   1    10  |   0    10  |   0
    4      MM_Improved    7  |   3     4  |   6     5  |   5     7  |   3
    5        AB_Open      7  |   3     4  |   6     6  |   4     5  |   5
    6       AB_Center     4  |   6     8  |   2     5  |   5     7  |   3
    7      AB_Improved    6  |   4     6  |   4     5  |   5     3  |   7
--------------------------------------------------------------------------
            Win Rate:      72.9%        60.0%        68.6%        72.9%
```

custom_score3()
Avoiding the corners, and the places just horizontally and vertically next to the corners, seemed like a place where the player could get stuck, having fewer possibilities of moving. Thus, we penalize if the location of the player is in these places.



```
#penalizing for being in the corners
corners = [[0,0],[0,1],[1,0],
           [game.width,game.height],[game.width-1,game.height],[game.width,game.height-1],
           [game.width,0],[game.width-1,0],[game.width,1],
           [0,game.height],[0,game.height-1],[1,game.height]]
if(game.get_player_location(player) in corners):
    eval_func -= 4
```



```
 Match #    Opponent     AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                         Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1        Random      10  |   0     8  |   2    10  |   0    10  |   0
    2        MM_Open      8  |   2     3  |   7     7  |   3     9  |   1
    3       MM_Center     9  |   1     9  |   1    10  |   0    10  |   0
    4      MM_Improved    7  |   3     4  |   6     5  |   5     7  |   3
    5        AB_Open      7  |   3     4  |   6     6  |   4     5  |   5
    6       AB_Center     4  |   6     8  |   2     5  |   5     7  |   3
    7      AB_Improved    6  |   4     6  |   4     5  |   5     3  |   7
--------------------------------------------------------------------------
            Win Rate:      72.9%        60.0%        68.6%        72.9%
```

Conclusion
It seems that the best heuristic so far were the corners and center distance. The player distance heuristic did not yield better results than the benchmark AB_Improved. A combination of custom_score2() and custom_score3() could also be used. However, for the sakes of this report, we recommend the custom_score3() heuristic for checking corners for the following reasons:
-Ease of implementation: it is very simple to check if the current location is in the corners. Additional corners could even be added if wanted.
-Complexity: checking corner locations does not add any time or space complexity as there are no loops, just a quick check of the current location of the player.
-Performance: as games are random, the win rate results change. However, using the heuristics from custom3 should increase the overall win rate and showed the best overall performance, compared to AB_Improved with 72.9% win rate.