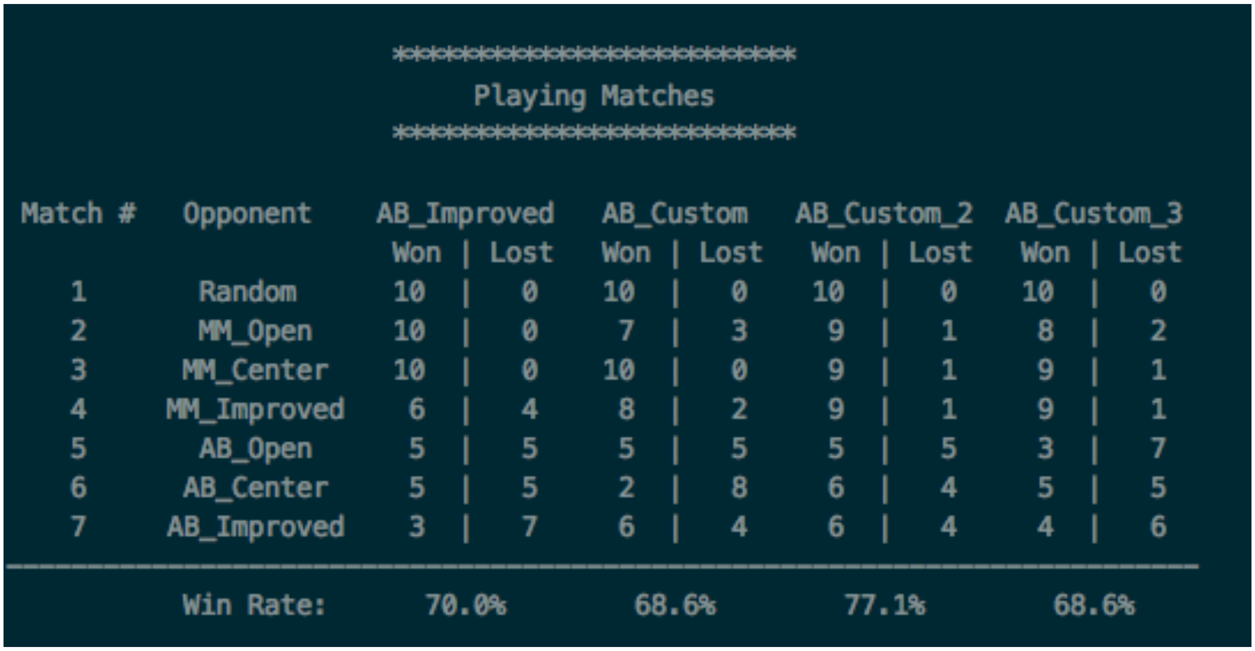
Heuristic Analysis for Isolation Game Playing Agent

-Radhika Pasari

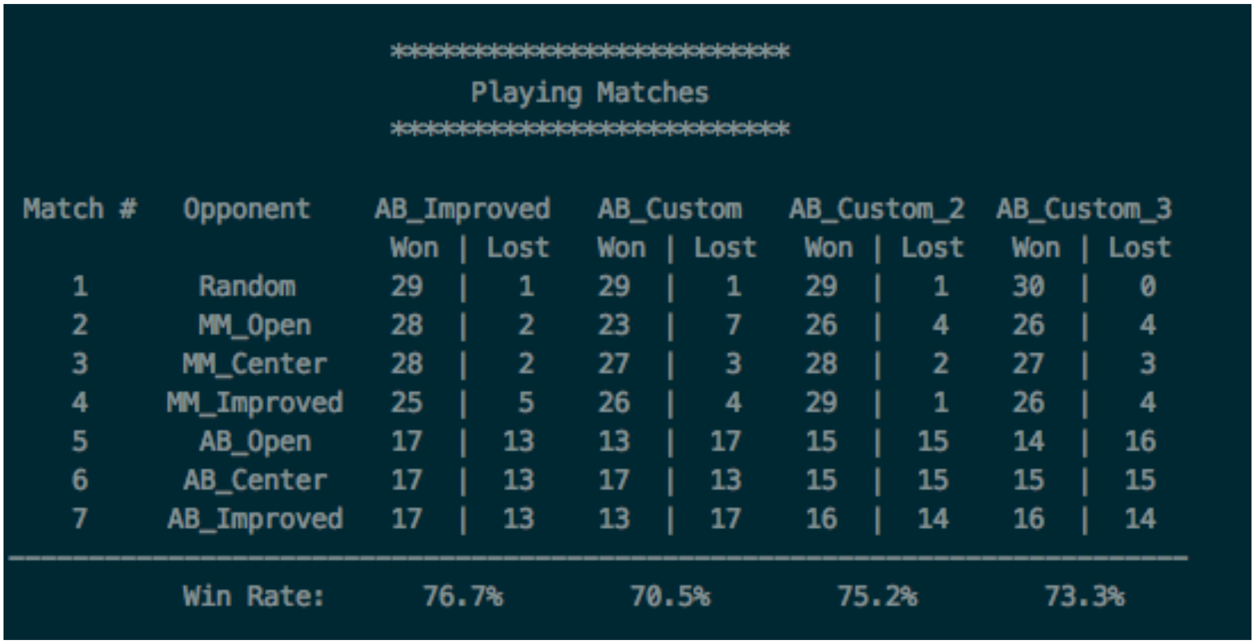
Results for 10 game runs –



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

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      10  |   0    10  |   0    10  |   0    10  |   0
    2       MM_Open     10  |   0     7  |   3     9  |   1     8  |   2
    3      MM_Center    10  |   0    10  |   0     9  |   1     9  |   1
    4     MM_Improved    6  |   4     8  |   2     9  |   1     9  |   1
    5       AB_Open      5  |   5     5  |   5     5  |   5     3  |   7
    6      AB_Center     5  |   5     2  |   8     6  |   4     5  |   5
    7     AB_Improved    3  |   7     6  |   4     6  |   4     4  |   6
--------------------------------------------------------------------------
           Win Rate:      70.0%        68.6%        77.1%        68.6%
```

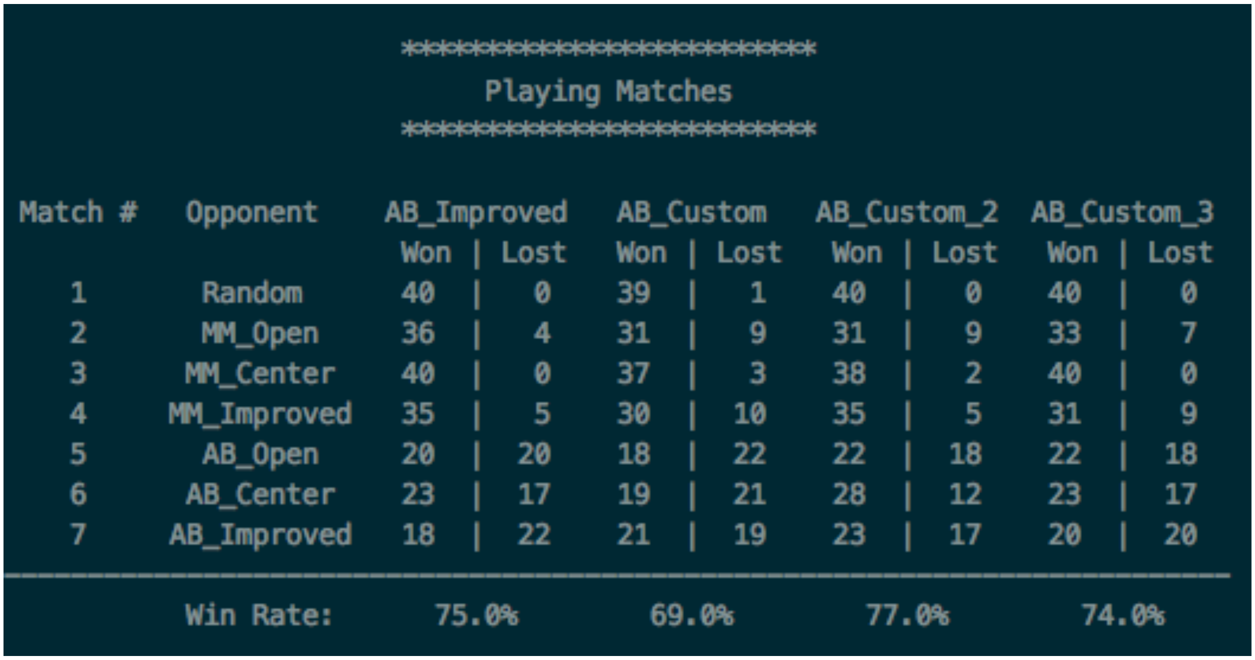
Results for 30 game runs –



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

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      29  |   1    29  |   1    29  |   1    30  |   0
    2       MM_Open     28  |   2    23  |   7    26  |   4    26  |   4
    3      MM_Center    28  |   2    27  |   3    28  |   2    27  |   3
    4     MM_Improved   25  |   5    26  |   4    29  |   1    26  |   4
    5       AB_Open     17  |  13    13  |  17    15  |  15    14  |  16
    6      AB_Center    17  |  13    17  |  13    15  |  15    15  |  15
    7     AB_Improved   17  |  13    13  |  17    16  |  14    16  |  14
--------------------------------------------------------------------------
           Win Rate:      76.7%        70.5%        75.2%        73.3%
```

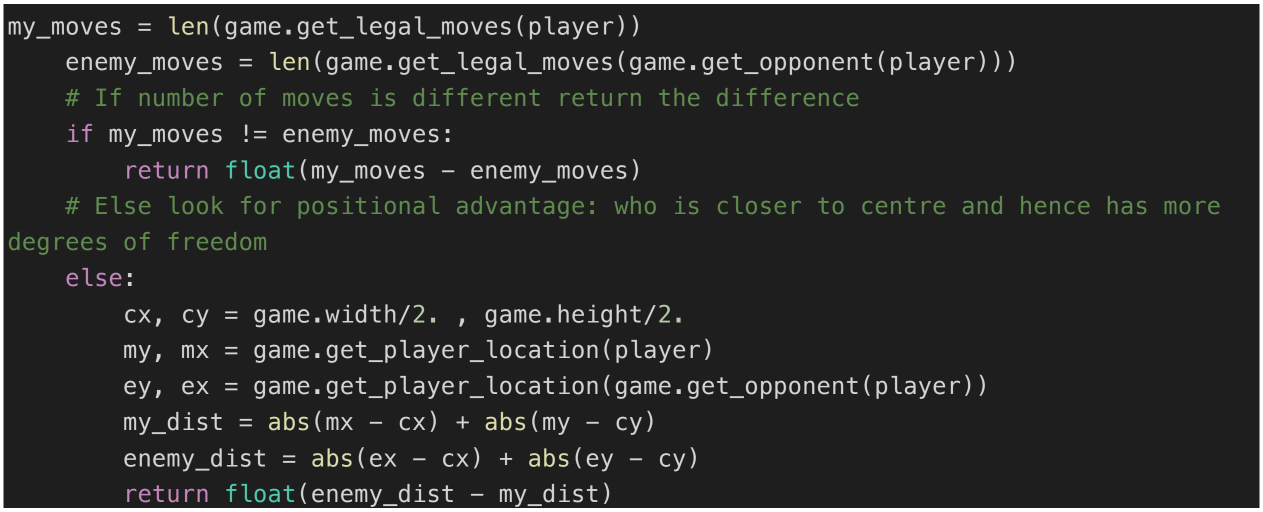
Results for 40 game runs –



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

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      40  |   0    39  |   1    40  |   0    40  |   0
    2       MM_Open     36  |   4    31  |   9    31  |   9    33  |   7
    3      MM_Center    40  |   0    37  |   3    38  |   2    40  |   0
    4     MM_Improved   35  |   5    30  |  10    35  |   5    31  |   9
    5       AB_Open     20  |  20    18  |  22    22  |  18    22  |  18
    6      AB_Center    23  |  17    19  |  21    28  |  12    23  |  17
    7     AB_Improved   18  |  22    21  |  19    23  |  17    20  |  20
--------------------------------------------------------------------------
           Win Rate:      75.0%        69.0%        77.0%        74.0%
```

1. **Heuristic 1:**
   If (My moves! = Opponents moves) then (My moves - Opponents moves) Else (Opponents distance to center – My distance to center) [Here distance is Manhattan distance]



```
my_moves = len(game.get_legal_moves(player))
    enemy_moves = len(game.get_legal_moves(game.get_opponent(player)))
    # If number of moves is different return the difference
    if my_moves != enemy_moves:
        return float(my_moves - enemy_moves)
    # Else look for positional advantage: who is closer to centre and hence has more
degrees of freedom
    else:
        cx, cy = game.width/2. , game.height/2.
        my, mx = game.get_player_location(player)
        ey, ex = game.get_player_location(game.get_opponent(player))
        my_dist = abs(mx - cx) + abs(my - cy)
        enemy_dist = abs(ex - cx) + abs(ey - cy)
        return float(enemy_dist - my_dist)
```

In this heuristic, we use the Manhattan distance instead of Euclidean since it makes more sense in a discrete board game structure. Also, it takes into consideration the distance to center advantage only when there is a difference in number of moves left

2. **Heuristic 2:**
   (My moves – Opponents moves) + (Opponent distance to center – My distance to center) / (2 * Total number of game moves) [Here distance is Euclidean distance]

```
my_moves = len(game.get_legal_moves(player))
    enemy_moves = len(game.get_legal_moves(game.get_opponent(player)))
    cx, cy = game.width/2. , game.height/2.
    my, mx = game.get_player_location(player)
    ey, ex = game.get_player_location(game.get_opponent(player))
    my_dist_to_center = float((cy - my)**2 + (cx - mx)**2)
    enemy_dist_to_center = float((cy - ey)**2 + (cx - ex)**2)
    return float((my_moves - enemy_moves) + (enemy_dist_to_center -
my_dist_to_center)/(2*game.move_count))
```

   In this function, I try to capture distance to center because the further to the corners that we are we eliminate our degrees of freedom. But I divide that distance factor to dwindles as we go further in the game since the importance of staying in the center reduces and the importance of maximizing your own moves left increases.

3. **Heuristic 3:**

2 * My moves – Opponents moves – 0.5*Jumps to center

```
my_moves = len(game.get_legal_moves(player))
    enemy_moves = len(game.get_legal_moves(game.get_opponent(player)))
    my, mx = game.get_player_location(player)
    cx, cy = game.width/2. , game.height/2.
    jumps_to_center = min(abs(mx - cx) , abs(my - cy))
    return (2*my_moves - enemy_moves - 0.5*jumps_to_center)
```

   In order to improve on the second heuristic, I did two things – instead of distance to center I used an approximate jump to center function and used a weighted function. I gave more weight to my moves left and varied them based on multiple runs.

Recommendation-

I would recommend my second function since it performed the best. I think its performance can be attributed to the following reasons-

1. Along with number of moves left it also takes into account board game position advantage by measuring distance to center.
2. Towards the end of the game when distance to center becomes less important and your own moves left become more important it scales the weights of those two factors accordingly.
3. Lastly, it is simple to compute and hence very fast.