Following are the results that were obtained on running the tournament.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       9  |  1      9  |  1     10  |  0      9  |  1
    2       MM_Open      7  |  3      7  |  3      5  |  5      8  |  2
    3      MM_Center     8  |  2      8  |  2      9  |  1      9  |  1
    4     MM_Improved    8  |  2      7  |  3      8  |  2      7  |  3
    5       AB_Open      3  |  7      6  |  4      7  |  3      6  |  4
    6      AB_Center     5  |  5      5  |  5      6  |  4      4  |  6
    7     AB_Improved    3  |  7      5  |  5      6  |  4      4  |  6
--------------------------------------------------------------------------
           Win Rate:      61.4%        67.1%        72.9%        67.1%
```

Even after running the tests for multitple times the results fall in the same range .

For evaluating the state of the board it is necesscessary to give each place a value based on the options that they open up . Hence centre of the board should have the highest value and they should keep on decreasing as we move towards the corner . Because moving towards the corners makes the player more prone to isolation . A dictionary with key as the positions (x, y) and value as the place value was created .

Following is the representaiton of the place values corrosponding to the places on the board .

	0	1	2	3	4	5	6
0	1	2	3	4	3	2	1
1	2	4	6	8	6	4	2
2	3	6	9	12	9	6	3
3	4	8	12	16	12	8	4
4	3	6	9	12	9	6	3
5	2	4	6	8	6	4	2
6	1	2	3	4	3	2	1

eg.
position_value[(x,y)] = c
position_value[(3,3)] = 16

custom_score 1 :

This heuristic custom score 1 adds the place values of all the legal moves . The place values are taken from the above representation and returns them as a score . Certainly players with more legal moves towards the centre of the board will get a higher rating and players with more moves in the corners will get a lower score . This would facilitate the player to stick to the centre as long as possible , that would keep more options open in the preliminary stage of the game .

Custom_score 2 :

Based on the above theory of place values . This heuristic calculates the opponents score as well same as the player score and returns a difference between the two . Various experiments were attempted , altering the return value own_score – player_score by multiplying the opponent_score by 2 and 3 so that the player should play more aggressively . However i did not find any change in the results by doing so .

Custom_score 3 :

The board is divided into four parts so as to better understand the board state and rate each part which can be used to calculate the player score .

	0	1	2	3	4	5	6
0	1	2	3	4	3	2	1
1	2	4	6	8	6	4	2
2	3	6	9	12	9	6	3
3	4	8	12	16	12	8	4
4	3	6	9	12	9	6	3
5	2	4	6	8	6	4	2
6	1	2	3	4	3	2	1

Partition 1 :

	0	1	2	3
0	1	2	3	4
1	2	4	6	8
2	3	6	9	12
3	4	8	12	16

Number of elements in group = 16
Group ranges from
Top : (0,0) to (0,3)
Bottom : (3,0) to (3,3)

Partition 2 :

	4	5	6
0	3	2	1
1	6	4	2
2	9	6	3
3	12	8	4

Number of elements in group = 12

Group ranges from

Top : (0,4) to (0,6)

Bottom : (3,4) to (3,6)

Partition 3

	0	1	2	3
4	3	6	9	12
5	2	4	6	8
6	1	2	3	4

Number of elements in group = 12

Group ranges from

Top : (4,0) to (4,3)

Bottom : (6,0) to (6,3)

Partition 4 :

	4	5	6
4	9	6	3
5	6	4	2
6	3	2	1

Number of elements in group = 9

Group ranges from

Top : (4,4) to (4,6)

Bottom : (6,4) to (6,6)

The partitions were created to get a better understanding of the empty places , so that if one of the legal moves falls into the respective partition it could be multiplied with a factor that helps it to gain the understanding the empty places in that partition .

Eg . Consider partition 1

Partition 1 :

	0	1	2	3
0	0	1	0	1
1	1	0	1	1
2	1	1	1	0
3	1	1	0	1

1 – place occupied
0 – empty places .

Partition 1 place values :

	0	1	2	3
0	1	2	3	4
1	2	4	6	8
2	3	6	9	12
3	4	8	12	16

Hence to give the partition a rating we add the place values of the empty places :
That is for the above example .
Substuting the place values from the above representaiton for 0‘s .
Empty_places_score = 1 + 3 + 4 + 12 +12 = 32
multiplication_factor = empty_places_score / total_number of elements
= 32 / 16
= 2

The above procedure is done for all the four partitions . While the place value of legal move is multiplied by this multiplication factor .
own_score += position_value[(x,y)] * multiplication_factor

Similar equation is used to calculate the opponent score

opp_score += position_value[(x,y)] * multiplication_factor

The difference of the two was returned .

Analysis:

The custom score 1 simply returns the players position value and does not keep a track of the opponent . It is necessary to keep a track of the opponents position . The heuristic custom score 2 over comes this drawback by keeping a track of the opponents score and returns the difference of the two . While there is no way to know that the if one of the legal move is played what kind of options would it open up , that is if there were empty places in the region of the legal move . This drawback was overcome by the custom score 3 . However , this is computationally expensive . Hence , custom score 2 outperforms the other two .