There are three heuristics I developed.

1. The first one gives a single value and uses a two step approach. The first step is a simple one to know the difference between the number of moves of the opponent and the number of moves for the opponent. The issue with using this approach is, the next level can be a problem. So I added a different hueristic, if the opponent can be a neighbour to the new move, then give it a higher penalty. This will ensure that the opponent will not be able to pursue the player
2. I divided the game into 2 parts. One where the number of open spaces is more than 75% of the time. If so, I do a minimal check to see if there are any moves which block the move of the opponent. For such moves, I give higher score. If the opening is past and moved to middle portion, the aim is to go a level deeper and check if there are positions of blocking.
3. The third one is to check if the opponent has a chance to become neighbour. This will imply that the opponent has ability to pursue. So for such moves I give less score.

The second choice works better times because of the following reasons…

1. The first benefit is that it distinguishes between beginning ad middle portions of the game. The extra level is being used only after 50% is reached. So the search space is reduced. Because it has the ability to deal in this manner, the overall performance must be better.
2. The third option is useful with a more aggressive opponent, and is a defensive strategy. If the opponent can be a neighbour, he will have the ability to block in the next step.
3. THe first one is similar, but in the beginning of the game, it does not matter if the opponent is close. So it is necessary to distinguish between start and middle game.

The test results of the execution from tournament.py are given below…

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

 Match #   Opponent    AB_Improved  AB_Custom_2
                    Won | Lost   Won | Lost
   1       Random       0  |  10     9  |   1
   2       MM_Open      0  |  10     6  |   4
   3       MM_Center    0  |  10     7  |   3
   4       MM_Improved  0  |  10     2  |   8
   5       AB_Open      5  |   5    10  |   0
   6       AB_Center    5  |   5    10  |   0
   7       AB_Improved  5  |   5    10  |   0
--------------------------------------------------------------------------
         Win Rate:      21.4%        77.1%
```

AB_Improved only won 21.4% of the time while the custom_2 won 77% of the time. This is because of the way scoring is done in the evaluation function.