Heuristic Analysis

There are 3 different heuristics were developed to find best scoring function. Each function check if current player win and returns +infinity score and check if current player lose and returns - infinity in this case. Otherwise all three heuristics are very different. Because of nondeterministic nature of game (Board returns shuffled available moves and because you don't know your opponent strategy) it's hard to compare different approaches. So, I run tournament.py 10 times and average results are very close to this round:

```
                    ************************
                         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     9  |   1     7  |   3     10 |   0
   2       MM_Open      5  |   5     6  |   4     7  |   3     9  |   1
   3      MM_Center     9  |   1     7  |   3     7  |   3     9  |   1
   4     MM_Improved    7  |   3     5  |   5     6  |   4     8  |   2
   5       AB_Open      5  |   5     7  |   3     4  |   6     5  |   5
   6      AB_Center     6  |   4     7  |   3     6  |   4     5  |   5
   7     AB_Improved    3  |   7     4  |   6     4  |   6     4  |   6
--------------------------------------------------------------------------
          Win Rate:      64.3%        64.3%        58.6%        71.4%


Your ID search forfeited 6.0 games while there were still legal moves availabl
o play.
```

AB_Improved: average score 63.7 % of win. Min 57.1, Max 71.4.

AB_Custom: average score 63 % of win. Min 57.1, Max 68.6. First strategy is based on players locations and amount of blank spaces. There is an assumption that if there is a lot of space available, then it's better to chase opponent trying to decrease number of available moves for the opponent. After there is not much free spaces available it's better to find new place far away from opponent to invade as much available space as possible to survive. This heuristic easily won against Random, MM_Open and MM_center, but sometimes fail to MM_Improved. It has not much advantages against any heuristic with AlphaBetaPlayer. On average it has performance similar to **AB_Improved** despite of different nature of scoring.

AB_Custom_2: average score 60.44 % of win. Min 52.9, Max 68.6. Second strategy is to use more "aggressive" behavior, it tries to invade moves that are also available for the opponent. So if player move to place, that is also available for its opponent (current player is active), then it's considered as the best possible move. If current player is not active and current opponent move collide with possible players moves, then it's considered as the worst possible move. For all other moves score is the amount of available moves for the current player. This strategy shows as not very effective one and have an advantage only against MinMaxPlayer. So, the most success

of this heuristic is because of using more advanced player. Average score is worse than **AB_Improved** and fail in most rounds against it.

AB_Custom_3: average score 73.57 % of win. Min 68.6, Max 78.6. Third strategy is mixed strategy. It's based on strategy when score is difference between available moves for current player and available moves for opponent. One improvement is that if the difference is the same, move with biggest amount of moves is prefered: `score += 0.2 * own_moves`. The second improvement is that current player should try to invade main diagonal and cross to make player stay away from "dangerous" corners. This heuristic has the best average score. It win easily against Random, MM_Open, MM_Center, MM_Improved and usually win against AB_Open and AB_Center. But randomly win or lose to **AB_Improved** so, optimization make sense. This improved version should be used because It has better average score, than **AB_Improved.** But according to the data it sometimes almost fail to very simple AB_Open and AB_Center. That shows it's not perfect and needs to be improved, maybe by combining with any of this heuristics. This heuristic give an advantage to positions in center, but it's not very efficient against AB_Improved. This shows that starting position in center is not so important as it seems.