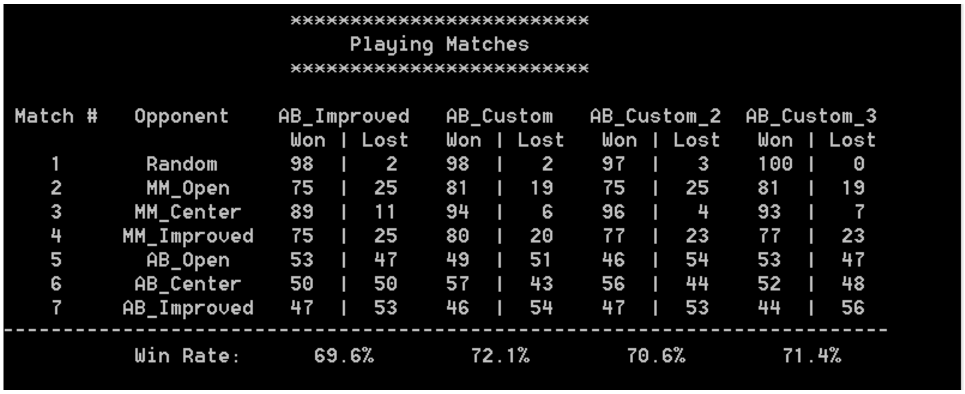
Saurabh Vyas
Udacity Artificial Intelligence Nanodegree
05-Feb-18

Evaluating Game Playing Heuristics

Project 2: Build a Game-Playing Agent

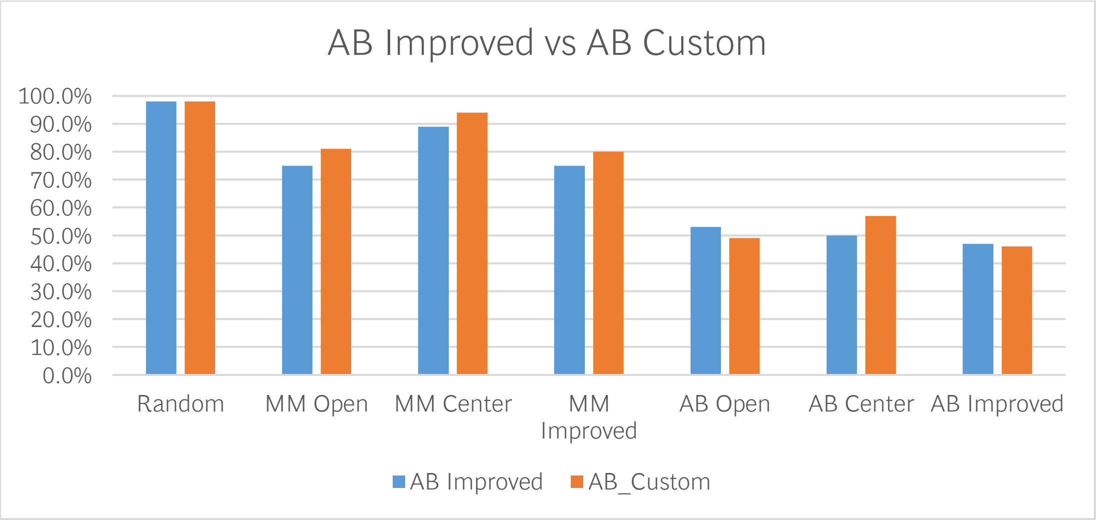
I evaluated three basic heuristics that can be used to score in Isolation game. All of these heuristics were used with alpha beta pruning. The following is the results of tournament.py.



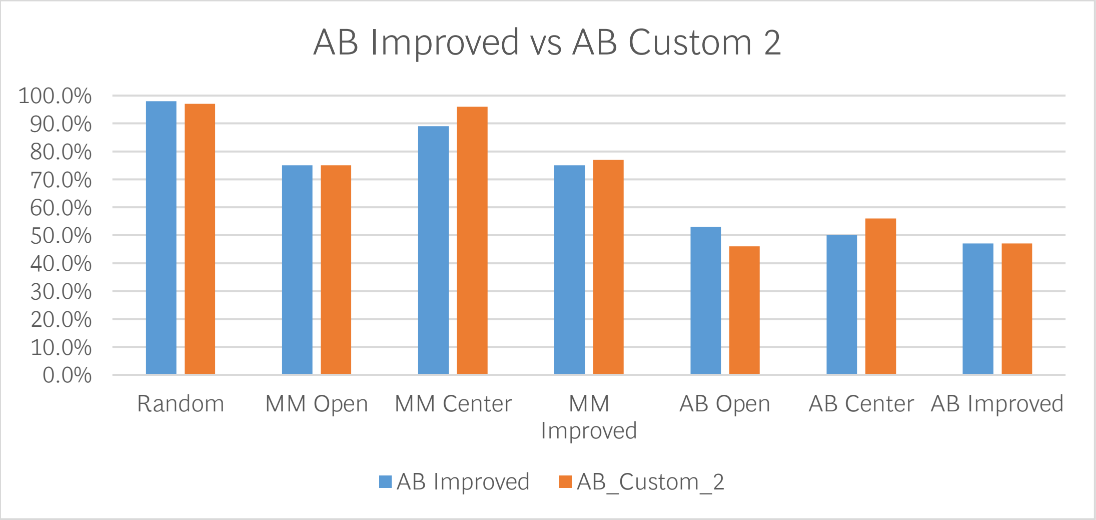
```
                        *************************
                             Playing Matches
                        *************************

 Match #    Opponent     AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                          Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1        Random       98  |   2    98  |   2    97  |   3   100  |   0
    2        MM_Open      75  |  25    81  |  19    75  |  25    81  |  19
    3       MM_Center     89  |  11    94  |   6    96  |   4    93  |   7
    4      MM_Improved    75  |  25    80  |  20    77  |  23    77  |  23
    5        AB_Open      53  |  47    49  |  51    46  |  54    53  |  47
    6       AB_Center     50  |  50    57  |  43    56  |  44    52  |  48
    7      AB_Improved    47  |  53    46  |  54    47  |  53    44  |  56
--------------------------------------------------------------------------
           Win Rate:      69.6%        72.1%        70.6%        71.4%
```

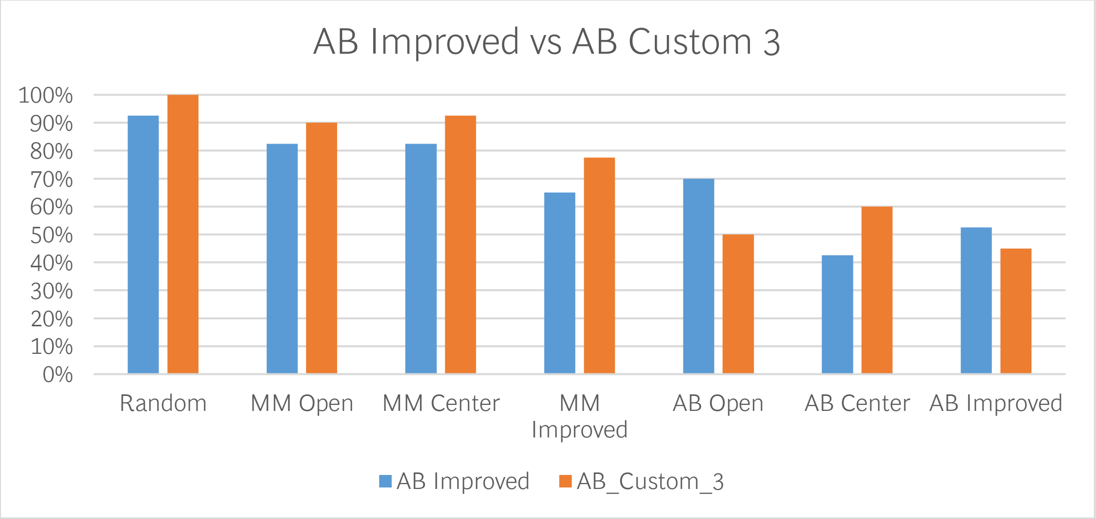
1. **AB_Custom (Own moves – Opp moves – distance from center):** This heuristic calculates the number of moves available to the player and then subtracts number of available moves to opponent and distance of the current move from center of the board. The heuristic works pretty well and has better performance as compared to AB_Improved (72.1% vs 69.6%). The reason why it works well is that it tries to maximize the number of moves available for the player and minimize opponent's moves and distance from the center. Distance from the center part tries to avoid the walls and opponent move part tries to make sure that opponent has fewer choices so that it eventually runs out of moves. In a nutshell, it penalizes bad moves severely.



2. **AB_Custom_2 (Own moves – sqrt(Opp moves)):** This heuristic focuses on maximizing difference between own moves and square root of opponent moves. Performance of this method is also better than AB Improved (70.6% vs 69.6%). However, it's worse than AB_Custom. In this method, I am trying to improve AB_Improved by reducing the penalty on Opp moves in a nonlinear way. This shows that in AB Improved, we are applying aggressive penalty, which sometimes rules out really good moves. By reducing the penalty, performance improves.



3. **AB_Custom_3 (Own moves – 0.5*(Opp moves)):** This option evaluates if applying a different linear penalty on opponent moves leads to a better function. For this heuristic, the different penalties were tried such as OwnM – 1.5*OppM and OwnM – 0.5*OppM. The latter option worked better that is consistent with the explanation given in AB_Custom_2. More aggressive penalty doesn't improve the performance. The performance of this heuristic was very good as compared to AB Improved (71.4% vs 69.6%) and slightly worse than AB_Custom (71.4% vs 72.1%).



Recommendation

The above results show that the first heuristic performs significantly better than the other two. This leads to the conclusion that an objective function on the below lines could be a very good choice:

F(x) = a*OwnMoves – b*OppMoves – c*(distance from center)

We can try to find the optimum values of ‘a’, ‘b’ and ‘c’ by simulation.

The reason why this function works well could be due to the following:

1. It tries to maximize the own moves
2. It also applies a penalty to those moves that give large number of options to the opponent and tries to avoid going towards walls or corners.
3. The penalties could be optimized further using simulations, which would lead to even better results.

Based on above it could be conclude that finding good heuristic function is a major part of any game strategy.