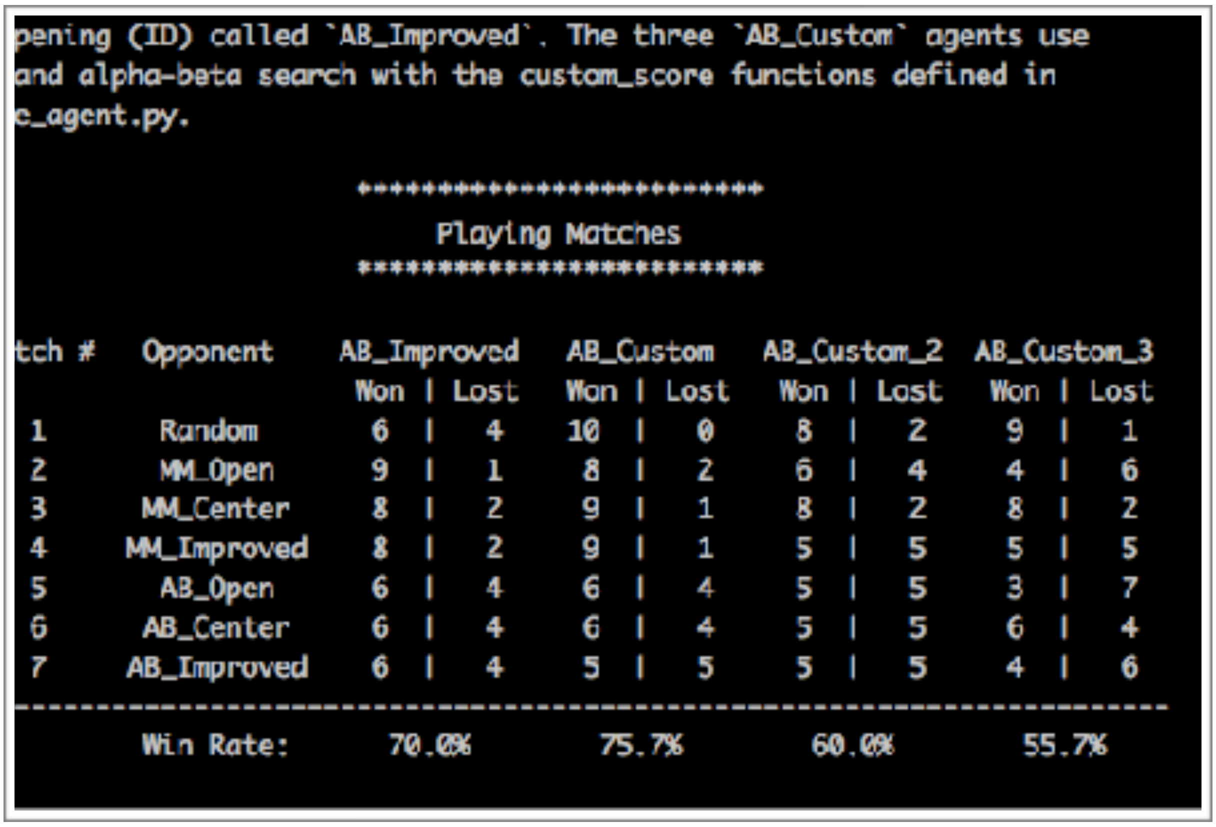
Isolation Heuristic Analysis



```
pening (ID) called `AB_Improved`. The three `AB_Custom` agents use
and alpha-beta search with the custom_score functions defined in
c_agent.py.

                          *************************
                               Playing Matches
                          *************************

tch #    Opponent      AB_Improved    AB_Custom     AB_Custom_2   AB_Custom_3
                        Won | Lost   Won | Lost    Won | Lost    Won | Lost
 1        Random         6  |  4     10  |  0       8  |  2       9  |  1
 2        MM_Open        9  |  1      8  |  2       6  |  4       4  |  6
 3       MM_Center       8  |  2      9  |  1       8  |  2       8  |  2
 4      MM_Improved      8  |  2      9  |  1       5  |  5       5  |  5
 5        AB_Open        6  |  4      6  |  4       5  |  5       3  |  7
 6       AB_Center       6  |  4      6  |  4       5  |  5       6  |  4
 7      AB_Improved      6  |  4      5  |  5       5  |  5       4  |  6
--------------------------------------------------------------------------
         Win Rate:      70.0%        75.7%         60.0%         55.7%
```

Artificial Intelligence Nanodegree

May Cohort

Isolation Heuristic Analysis

Heuristic 1

The first heuristic is concerned with the number of remaining moves. It takes into account the number of remaining moves of the opponent as well as own and uses a simple formulae, which internally just computes the relative importance with respect to other, to calculate the score. The performance which i usually got was very good, around 75% in total.

The formulae which i used to compute this is,

```
if own_moves > opp_moves
        100 * (own_moves/own_moves+opp_moves)
elif own_moves < opp_moves
        - 100 * (opp_moves/own_moves+opp_moves)
else
        0
```

The above pseudo code defines this heuristic. It is easy to compute just we need to calculate the total possible remaining moves, and we are pretty much done. But it takes into account the opponent moves also. It is a linear computation in total, as finding possible moves internally takes linear time. It is very good, the chances of winning are high, as it tries to know your current position with respect to the other player.

Heuristic 2

The second heuristic attempts to capture the relative difference between the number of possible moves for ourselves and the opponent players, with the intent of restricting the opponent's mobility and increasing one's own mobility. The performance was between 60-65% in total (against all the competitors) on average.

The pseudo code for this heuristic is,

```
if own_moves + opp_moves != 0
        100 * (own_moves - opp_moves)/(own_moves + opp_moves)
else
        0
```

Similar to heuristic 1, it also takes just the constant time to compute the value, once you know the number of possible moves for yourself and your opponent at the present scenario of the game (which internally i suppose takes linear time).

Heuristic 3

The third heuristic just tries to calculate the manhattan distance of the piece from the centre block of the board. Its performance is pretty average being around 58% (on average).

It is very straightforward and gives a quick score. It is a constant time computation. Though not very efficient, but still have better chances to win than normal on average. It doesn't takes into account the opponent's or the board's situation in the game at the present moment.

There are pretty more heuristics we can apply like corners captured and stability, and we can even combine multiple heuristics to get the best performance.

On the basis of the above given heuristics, i would definitely go with heuristic 1.

Its efficiency is good. It increases the chances of winning to almost 75% in total if playing against wealthy heuristics (strictly on basis of tournament score). In general if played against random player it mostly gives perfect result, (i.e. almost wins). Secondly its calculation is also not very restrictive or hardware dependent, its just a linear computation. Thirdly it compares our player with the opponent and the remaining space in the board, which is the core of this isolation game. Though it doesn't goes into much depth to identify or use any extravagant search, its sole purpose is to give a score on the basis of present scenario of itself and its opponent.