Heuristic Analysis

Summary

This notebook is a brief report that summarises three evaluation heuristics that were explored for the Isolation board game of the Udacity AIND course.

For determining the heuristics, we play a few games and we can observe that the following available information may be of use:

- The number of moves available to each player
- The position of each player, in particular the 'centrality' of the player as being close to the end of the edge of the board increases likelihood of being isolated
- The number of blank spaces around each player. Even when isolated, being surrouned by blank spaces still potentially allows a continuous run of 7 moves in this instance of Isolation.

The 4 Custom Score functions that we explore are:

1. Number of opponent moves
2. Linear combination of moves vs opponent.
3. Opponent points
4. Combination of moves with centrality minimisation

Custom Score 1: Number of opponent moves

This has to be one of the simplest heuristics. Essentially, we win when there is no more moves for the opponent. The simplicity and relative low cost of calculating this enables deeper traversal of our search tree.

In [ ]:

```
n_moves_opp = len(game.get_legal_moves(game.get_opponent(player)))
score = float(- n_moves_opp)
```

Custom Score 2: Linear combination of moves vs opponent.

Similar to Custom Score 1 but in cases where the opponent has the same number of moves, we can give preference to moves which maximises our available number of moves.

In [ ]:

```
n_moves = len(game.get_legal_moves(player))
n_moves_opp = len(game.get_legal_moves(game.get_opponent(player)))
score = float(0.5 * n_moves - n_moves_opp)
```

Custom Score 3: Opponent points

Similar to Custom Score 1 but we give each legal move for an opponent 10 points. When points are less then

20 (i.e only two legal moves left) we also evaluate how many legal subsequent moves there are.

In [ ]:

```
opp_moves = game.get_legal_moves(game.get_opponent(player))
opp_score = len(opp_moves) * 10
if opp_score <= 20:
    opp_score += sum([len(game._Board__get_moves((x, y))) for x, y in opp_moves])
score = float(- opp_score)
```

Custom Score 4: Combination of moves with centrality minimisation

Similar to Custom Score 2 in the sense that we look at the number of moves available to each player but we also penalise based on the centrality of the move. This assumes that moves close to central are good whilst moves near the edges of the board are bad and should be penalised.

In [ ]:

```
n_moves = sum([1 - abs(x / game.height - 0.5) - abs(y / game.width - 0.5)
               for x, y in game.get_legal_moves(player)])
n_moves_opp = sum([1 - abs(x / game.height - 0.5) - abs(y / game.width - 0.5)
                   for x, y in game.get_legal_moves(game.get_opponent(player))])
score = n_moves - n_moves_opp
```

Custom Score Performance Comparison

The following table shows the performance of the above custom score functions.

The number of opponent moves seems to more consistently outperform AB_Improved compared to the others. The most likely explanation is the low cost of evaluating the heuristic allows us to more deeply search the game tree. The concept of using centrality to determine the potential futures moves is interesting but comes at the cost which limits the number of times we can iteratively deepen our search. Another potential option would be to aggressively follow or run from opponent but again it appears that simply searching deeper into our tree is more effective.

In [ ]:

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

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       8  |   2    10  |   0     9  |   1     9  |   1
    2       MM_Open      6  |   4     8  |   2     6  |   4     6  |   4
    3      MM_Center     9  |   1     8  |   2     6  |   4     6  |   4
    4     MM_Improved    6  |   4     5  |   5     7  |   3     7  |   3
    5       AB_Open      5  |   5     9  |   1     4  |   6     6  |   4
    6      AB_Center     6  |   4     7  |   3     6  |   4     6  |   4
    7     AB_Improved    5  |   5     3  |   7     8  |   2     6  |   4
--------------------------------------------------------------------------
           Win Rate:      64.3%        71.4%        65.7%        65.7%

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

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       8  |   2     9  |   1    10  |   0    10  |   0
    2       MM_Open      8  |   2     9  |   1     9  |   1     7  |   3
    3      MM_Center     9  |   1    10  |   0     8  |   2    10  |   0
    4     MM_Improved    8  |   2     5  |   5     6  |   4     8  |   2
    5       AB_Open      3  |   7     6  |   4     4  |   6     3  |   7
    6      AB_Center     8  |   2     9  |   1     8  |   2     7  |   3
    7     AB_Improved    5  |   5     6  |   4     3  |   7     4  |   6
--------------------------------------------------------------------------
           Win Rate:      70.0%        77.1%        68.6%        70.0%

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

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3  AB_Custor
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost   Won | Lc
    1       Random      10  |   0    10  |   0    10  |   0     9  |   1     8  |
    2       MM_Open      8  |   2     9  |   1     8  |   2     8  |   2     7  |
    3      MM_Center     8  |   2    10  |   0     9  |   1     8  |   2     8  |
    4     MM_Improved    6  |   4     8  |   2     7  |   3     6  |   4     8  |
    5       AB_Open      7  |   3     6  |   4     7  |   3     6  |   4     4  |
    6      AB_Center     5  |   5     5  |   5     6  |   4     6  |   4     7  |
    7     AB_Improved    6  |   4     3  |   7     6  |   4     6  |   4     7  |
--------------------------------------------------------------------------
           Win Rate:      71.4%        72.9%        75.7%        70.0%        70.0%


"""
```

Custom Score Recommendation

Based on the above the simplest custom function of reporting the number of opponent moves would be the recommended heuristic.

The main reason is because it empiraclly has the best performance. This is most likely due to the relatively low cost of evaluating the heuristic, allowing for deeper game tree evaluation, when using iterative deepening.

The simplicity of this heuristic also makes it easier to explain and understand.

Finally, such a heuristic is very general and would easily extend to other variants of this game where we work with other rules, such as using a pawn or king instead of a knight.