# Heuristic Analysis for Isolation Game

Annie Flippo

The Isolation Game is two player game where each player, in turn, moves to a square on a grid thereby removing that square from possible future moves on the board. The object is isolate the opponent so she/he cannot move.  Here, in this game, each player can move in an L-shape pattern similar to a knight in chess.

I see this game as one that is maximizing your own moves while minimizing your opponent's moves and I have designed my three heuristic scoring function accordingly.

## Heuristic 1

This measures the manhattan distance of the player versus the opponent's from the center square.  The idea is that I want to maximize the difference between the opponent's distance and mine.

```
my_score = manhattan_distance_score(game, player)
opp_score = manhattan_distance_score(game, game.get_opponent(player))

Maximize: opp_score - my_score
```

## Heuristic 2

This one builds on Heuristic 1 where I measure a player's number of possible moves minus its position from the center (as measured by Manhattan distance).  It was the hope that one player's score conveys 2 data points.  This is then compared with the opponent's equivalent score.

```
my_score = len(game.get_legal_moves(player)) - \
   manhattan_distance_score(game, player)
opp_score = len(game.get_legal_moves(game.get_opponent(player))) - \
   manhattan_distance_score(game, game.get_opponent(player))

Maximize: myscore - opp_score
```

## Heuristic 3

There are different number of possible moves at each position on the board.  If you assign a score for each square equals to the possible moves for the square, in essence, you are capturing how advantageous that square is. See the score for the board using this heuristic.  Here are the possible moves for a 7x7 board.

```
2 | 3 | 4 | 4 | 4 | 3 | 2
3 | 4 | 6 | 6 | 6 | 4 | 3
4 | 6 | 8 | 8 | 8 | 6 | 4
4 | 6 | 8 | 8 | 8 | 6 | 4
4 | 6 | 8 | 8 | 8 | 6 | 4
3 | 4 | 6 | 6 | 6 | 4 | 3
2 | 3 | 4 | 4 | 4 | 3 | 2
```

In this iteration, I'm evaluating how many moves each of ***my possible next moves*** will result.  I want to maximize the number of my over ***my opponent's possible next moves***.  The function score_centrality( ) which is generalized to any board larger than 5x5 returns the sum the score for all possible next moves for a player.

```
my_score = score_centrality(my_legal_moves)
opp_score = score_centrality(opp_legal_moves)

Maximize: my_score - opp_score
```

## Conclusion

With a combination of the maximizing the difference on how advantageous the player is over the opponent's, it results in a high proportion of wins.  Sample tournament.py output.  I ran the heuristic scoring function 20 times.

| Win Rate | AB_Improved | AB_Custom | AB_Custom_2 | AB_Custom_3 |
|---:|---:|---:|---:|---:|
| 1 | 68.60% | 65.70% | 68.60% | 70.00% |
| 2 | 70.00% | 77.10% | 75.70% | 68.60% |
| 3 | 78.60% | 64.30% | 65.70% | 68.60% |
| 4 | 70.00% | 64.30% | 75.70% | 71.40% |
| 5 | 65.70% | 62.90% | 68.60% | 72.90% |
| 6 | 74.30% | 58.60% | 71.40% | 72.90% |
| 7 | 70.00% | 68.60% | 70.00% | 70.00% |
| 8 | 77.10% | 60.00% | 74.30% | 64.30% |

| | | | | |
|---|---|---|---|---|
| 9 | 71.40% | 65.70% | 71.40% | 71.40% |
| 10 | 64.30% | 64.30% | 71.40% | 77.10% |
| 11 | 77.10% | 68.60% | 65.70% | 77.10% |
| 12 | 72.90% | 62.90% | 70.00% | 70.00% |
| 13 | 67.10% | 72.90% | 72.90% | 62.90% |
| 14 | 71.40% | 65.70% | 70.00% | 71.40% |
| 15 | 68.60% | 72.90% | 70.00% | 68.60% |
| 16 | 72.90% | 67.10% | 75.70% | 72.90% |
| 17 | 64.30% | 67.10% | 65.70% | 67.10% |
| 18 | 72.90% | 67.10% | 65.70% | 65.70% |
| 19 | 75.70% | 65.70% | 62.90% | 72.90% |
| 20 | 71.40% | 64.30% | 64.30% | 75.70% |
| **Average** | **71.22%** | **66.29%** | **69.79%** | **70.58%** |
| **Standard Dev** | 4.12% | 4.33% | 3.91% | 3.85% |

It is surprising that neither heuristic 1, 2 or 3 are consistently better than AB_Improved. I would think that introducing more information such as manhattan distance from the center for player and opponent would be better than just calculating the difference of number of possible moves.

Although each tournament gives very different results as to which scoring performs the best, it seems heuristics 3 outperforms heuristics 2 & 1 but about the same as AB_improved. Heuristics 3 which is the sum of available moves for the **current possible moves for the next round** gives the similar performance as the difference in number of **available moves this round**. Heuristics 3 which accounts for all the possible moves for the next round gives more information than just the available moves for a given player. Say, you know you have 2 possible moves this round but in heuristics 3 you will know that your next 2 moves are moving towards the center. This is much richer information than just you have 2 available moves. I think that's why it is better than the rest of the scoring function which gives less information.