Heuristic Analysis

Patricio Beltran Molas

February 2018

I had never heard about the Isolation board game before, so thinking about great evaluation functions was rather hard at first. I had to play several games in order to grasp a basic knowledge of which strategies work. Three different evaluation functions were created.

1 Aggressive with game progression awareness

```
own_moves = len(game.get_legal_moves(player))
opp_moves = len(game.get_legal_moves(game.get_opponent(player)))

# euclidean distance to center of board
center = _distance_2_center(game, game.get_player_location(player))

# returns a decimal between 0 and 1: move_count / board_size
progress = _game_progression(game)

if progress > .61:
    return float((own_moves - 4 * opp_moves))
else:
    return float((own_moves - 4 * opp_moves) - (center))
```

The main idea here was to create an improved version of AB_Improved. The **progress** value determines how "full" the board is. As the game advances, the distance to center is not as important as plainly staying alive. The **.61** value was the result of trial and error. That was the sweet spot. An aggressive multiplier of 4 was added to the opponent's moves in order to force the agent to continuously harass the opponent.

2 Aggressive with no progression awareness

```
own_moves = len(game.get_legal_moves(player))
opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
```

```
return float(own_moves - 4 * opp_moves)
```

This one was the one the first step toward creating the recommended function. I thought of making the agent overly aggressive, even if that meant that leaving the opponent in a bad position was more important over taking care of itself.

3 Distance from center

```
return -1 * (_distance_2_center(game, game.get_player_location(player)))
```

Custom score 3 was a really bad idea, but I decided to leave it for comparison purposes. Initially I thought the game could be easily won by staying near the center squares (much like the center principle of chess). While playing some simple games on paper, it seemed like a good idea, but with a bigger board and somewhat sophisticated search, it just performed poorly.

4 Tournament results

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

Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                      Won | Lost   Won | Lost   Won | Lost   Won | Lost
   1       Random      9  |   1    10  |   0    7  |   3    7  |   3
   2       MM_Open     9  |   1    10  |   0    7  |   3    1  |   9
   3      MM_Center    9  |   1     8  |   2    9  |   1    1  |   9
   4     MM_Improved   7  |   3     8  |   2    7  |   3    0  |  10
   5       AB_Open     4  |   6     5  |   5    9  |   1    2  |   8
   6      AB_Center    7  |   3     5  |   5    4  |   6    0  |  10
   7     AB_Improved   6  |   4     6  |   4    2  |   8    1  |   9
-------------------------------------------------------------------------
          Win Rate:      72.9%        74.3%        64.3%        17.1%
```

As I said before, Custom 3 was the worst. Proof that only worrying about one's position in the board is not enough to survive the game. It waas really interesting to see how much better Custom was over Custom 2. The awareness of when to use distance to center and when to ignore it gave Custom, my recommended function, a relatively important competitive edge. Perhaps further improvements to the recommended function could be to calculate the density of only the center area, and avoid the distance to center score if it is already too packed.