Heuristic Analysis of a Game Playing Agent

Ju Yang

1. Heuristic 1

Outputs a score equal to the difference in the number of moves available to the two players with a weight on the opponent moves. With a weight applied on the opponents, the heuristic function penalizes the computer player more for moves open to the opponent, essentially the computer player will try to block the opponent open moves more aggressively.

```
own_moves = len(game.get_legal_moves(player))
opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
return float(own_moves - 2 * opp_moves)
```

2. Heuristic 2

Outputs a score equal to the ratio of the number of moves available to the two players.

```
own_moves = len(game.get_legal_moves(player))
opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
if opp_moves == 0:
    return float("inf")
return 1.*own_moves / opp_moves
```

3. Heuristic 3

Outputs a score equal to difference of the square of the distance from the center of the board to the player and the opponent. This takes into consideration of the location of both players on the board, and try to stay as far as possible away from the opponent.

```
y1, x1 = game.get_player_location(player)
y2, x2 = game.get_player_location(game.get_opponent(player))
return float(y2 - y1)**2 + float(x2 - x1)**2
```

4. Comparison and Result

From the tournament.py result below, we can see that all 3 custom_score functions outperforms AB_improved when `NUM_MATCHES = 5`.

```
This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.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       9  |   1    10  |   0     9  |   1    10  |   0
    2        MM_Open     10  |   0     9  |   1     9  |   1     8  |   2
    3       MM_Center     9  |   1    10  |   0    10  |   0    10  |   0
    4     MM_Improved     9  |   1     8  |   2     9  |   1     9  |   1
    5        AB_Open      3  |   7     3  |   7     5  |   5     4  |   6
    6       AB_Center     5  |   5     6  |   4     5  |   5     5  |   5
    7     AB_Improved     5  |   5     7  |   3     4  |   6     6  |   4
--------------------------------------------------------------------------
           Win Rate:      71.4%        75.7%        72.9%        74.3%


Your ID search forfeited 15.0 games while there were still legal moves
```

I decide to use the best custom_score function 1 as my heuristic function because of the following 3 reasons 1) it has the highest winning rate 2) the final goal of the game is to block the opponents' legal moves, therefore, the number of moves is a straightforward evaluation parameter 3) blocking opponent's move more aggressively seems to be a good strategy in isolation, and it considers both players' moves to make a well-rounded evaluation.