Heuristic Analysis

Of

Isolation Game-Playing Agent

****

Playing Matches

****

Match #	Opponent	AB_Improved	AB_Custom	AB_Custom_2	AB_Custom_3
		Won | Lost	Won | Lost	Won | Lost	Won | Lost
1	Random	7 | 3	9 | 1	9 | 1	6 | 4
2	MM_Open	7 | 3	7 | 3	7 | 3	8 | 2
3	MM_Center	7 | 3	8 | 2	8 | 2	7 | 3
4	MM_Improved	6 | 4	9 | 1	6 | 4	6 | 4
5	AB_Open	6 | 4	5 | 5	5 | 5	5 | 5
6	AB_Center	5 | 5	4 | 6	6 | 4	5 | 5
7	AB_Improved	6 | 4	5 | 5	4 | 6	4 | 6
	Win Rate:	62.9%	67.1%	64.3%	58.6%

#1 - custom_score()

This heuristic is an improvisation of improved_score() function where in this case the number of available legal moves for the opponent is weighted. So penalizing the opponent by weighting their moves will make the computer player more cautious to any changes in their position.

Also, this is the preferred heuristic since it has the highest percentage chance of winning the matches compared to others as shown in the above table.

Implementation:

```
def custom_score(game, player,w=1.5):
    if game.is_loser(player):
        return float("-inf")


    if game.is_winner(player):
        return float("inf")


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

#2 - custom_score_2()

This heuristic outputs a score equal to square of the distance from the center of the board to the position of the player. Also, it is the next best heuristic in winning the games.

Implementation:

```
def custom_score_2(game, player):
    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")

    w, h = game.width / 2., game.height / 2.
    y, x = game.get_player_location(player)
    return float((h - y)**2 + (w - x)**2)
```

#3 - custom_score_3()

It was kind of an experimental heuristic which weighs the previous two heuristics and outputs the sum as the heuristic score value. Depending on which one performs better in the previous heuristics the weights can be modified for a better accuracy.

Implementation:

```
def custom_score_3(game, player, w1=0.4, w2=0.6):
    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")

    return ( w1*custom_score(game, player) + w2*custom_score_2(game, player) )
```