Heuristic Analysis

Factors of Heuristic Function

At first, I designed several factor based on `improved_score()` in `sample_players.py`, such as a coefficient multiplied to `my_moves` or `oppnent_moves`, `distance_from_center` or `distance_between_players`. But I didn't know which number of coefficient and signs(positive or negtive) of these factors I should choose. So I refactored `tornament.py` in order to compare different coefficients and signs(positive or negtive) of all these factors.

Here are some heuristic functions I designed for testing in `my_tornament.py`:

```
if game.is_loser(player):
    return float("-inf")
if game.is_winner(player):
    return float("inf")
my_moves = len(game.get_legal_moves(player))
opponent_moves = len(game.get_legal_moves(game.get_opponent(player)))
w, h = game.width / 2., game.height / 2.
y_m, x_m = game.get_player_location(player)
y_o, x_o = game.get_player_location(game.get_opponent(player))
center_distant = math.sqrt((h - y_m) ** 2 + (w - x_m) ** 2)
player_distant = math.sqrt((y_m - y_o) ** 2 + (x_m - y_o) ** 2)
```

- opp_x2() - `opponent_moves` is multiplied by 2

```
score = float(my_moves - 2 * opponent_moves)
```

- my_x2() - `my_moves` is multiplied by 2

```
score = float(2 * my_moves - opponent_moves)
```

- opp_x3() - `opponent_moves` is multiplied by 3

```
score = float(my_moves - 3 * opponent_moves)
```

- my_x3() - `my_moves` is multiplied by 3

```
score = float(3 * my_moves - opponent_moves)
```

- pos_center_dis() - base score plus distance from the center of game board

```
score = float(my_moves - opponent_moves + center_distant)
```

- neg_center_dis() - base score minus distance from the center of game board

```
score = float(my_moves - opponent_moves - center_distant)
```

- pos_player_dis() - base score plus distance between players

```
score = float(my_moves - opponent_moves + player_distant)
```

- neg_player_dis() - base score minus distance between players

```
1 score = float(my_moves - opponent_moves - player_distant)
```

In order to evaluate these heuristic functions better, I set `NUM_MATCHES` to `10` and `TIME_LIMIT` to `500`. Then I got the performances table:

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

Match #    Opponent    AB_Improved     opp_x2        my_x2        opp_x3        my_x3      pos_ct_dis    neg_ct_dis    pos_p_dis    neg_p_dis
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost   Won | Lost   Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      13  |   7    13  |   7    16  |   4    14  |   6    16  |   4    13  |   7    16  |   4    14  |   6    16  |   4
    2       MM_Open     13  |   7    16  |   4    13  |   7    13  |   7    12  |   8    13  |   7    14  |   6    15  |   5    13  |   7
    3      MM_Center    14  |   6    13  |   7    13  |   7    14  |   6    15  |   5    13  |   7    14  |   6    15  |   5    15  |   5
    4     MM_Improved   13  |   7    17  |   3    13  |   7    11  |   9    10  |  10    13  |   7    12  |   8    15  |   5    14  |   6
    5       AB_Open     10  |  10    11  |   9     8  |  12    12  |   8     9  |  11     9  |  11    10  |  10    10  |  10    11  |   9
    6      AB_Center    11  |   9    13  |   7    12  |   8    13  |   7    10  |  10    12  |   8    12  |   8    10  |  10     9  |  11
    7     AB_Improved   12  |   8    10  |  10     9  |  11     7  |  13    10  |  10    10  |  10     9  |  11    12  |   8    10  |  10
-----------------------------------------------------------------------
          Win Rate:      61.4%        66.4%        60.0%        60.0%        58.6%        59.3%        62.1%        65.0%        62.9%
```

Combination of Factors

From the performance table above, I decided the sign of each factor:

- Choice: opp_x2

  > opp_x2 > opp_x3 = my_x2 > my_x3

- Choice: neg_center_dis

  > neg_center_dis > pos_center_dis

- Choice: pos_player_dis

  > pos_player_dis > neg_player_dis

Then I combined every two of factors as well as all of them to get candidates of the finally heuristic function:

- ox2_ppd() - opp_x2() & pos_player_dis()

```
1 score = float(my_moves - opponent_moves * 2 + player_distant)
```

- ox2_ncd() - opp_x2() & neg_center_dis()

```
1 score = float(my_moves - opponent_moves * 2 - center_distant)
```

- ncd_ppd() - neg_center_diss() & pos_player_dis()

```
1 score = float(my_moves - opponent_moves - center_distant + player_distant)
```

- ox2_ncd_ppd() - opp_x2() & neg_center_diss() & pos_player_dis()

```
1 score = float(my_moves - opponent_moves * 2 - center_distant + player_distant)
```

Performances Table:

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

Match #   Opponent    AB_Improved     ox2_ppd      ox2_ncd      ncd_ppd    ox2_ncd_ppd
                       Won | Lost   Won | Lost   Won | Lost   Won | Lost   Won | Lost
   1       Random       14  |   6    15  |   5    17  |   3    16  |   4    14  |   6
   2       MM_Open      12  |   8    15  |   5    11  |   9    15  |   5    13  |   7
   3      MM_Center     13  |   7    16  |   4    14  |   6    16  |   4    15  |   5
   4     MM_Improved    13  |   7    16  |   4    15  |   5    13  |   7    14  |   6
   5       AB_Open       9  |  11    12  |   8    11  |   9     9  |  11     7  |  13
   6      AB_Center     11  |   9     8  |  12     9  |  11    11  |   9    11  |   9
   7     AB_Improved    11  |   9    11  |   9     9  |  11    11  |   9    12  |   8
-----------------------------------------------------------------------------------
          Win Rate:      59.3%        66.4%        61.4%        65.0%        61.4%
```

Final Heuristic Function

Finally, from the performances table of conbination factors above, I decided the oder of final heuristic functions in `game_agent.py`:

- custom_score():

  > score = float(my_moves - opponent_moves * 2 + player_distant)
  > Win Rate: 66.4%

- custom_score_2():

  > score = float(my_moves - opponent_moves - center_distant + player_distant)
  > Win Rate: 65.0%

- custom_score_3():

  > score = float(my_moves - opponent_moves * 2 - center_distant + player_distant)
  > Win Rate: 61.4%

Analysis:

It seems that `custom_score()` is the best heuristic function of all functions. In my opinion, the reson is as following:

- `my_moves` maximize legal moves of active player so that player will get more opportunties in future decisions.
- `-opponent_moves` minimize legal moves of opponent. This action help player 'isolate' the opponent. Also, in this heuristic function, `opponent_moves` is multiplied by 2, and this measure forces the player's strategy to be more radical, which means to be more aggressive.
- `player_distant` maximize the distance between two players. Maybe the agent think it's better to stay away from the opponent and get more opportunties in future decisions.