Heuristic Evaluations in Competitive Tournaments

Q: For each of your **three** custom heuristic functions, ***evaluate** the performance of the heuristic* using the included **tournament.py** script.

A: As seen in the table below, the **AB_Custom, AB_Custom_2**, and **AB_Custom_3** heuristics performed at a level of **60%**, **60%** and **75.1%** win rate respectively when engaged in the tournament as opposed to the **AB_Improved** Agent which performed at a **65.7%** win rate.

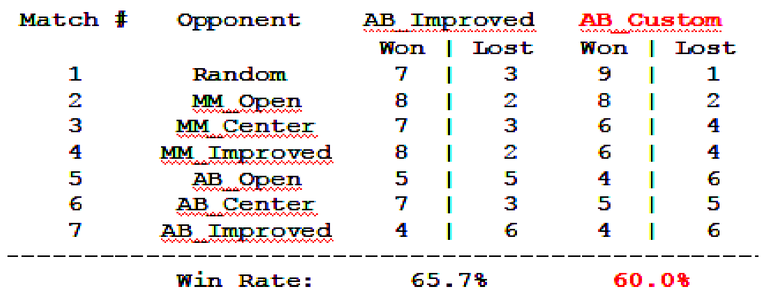
```
#                            *************************
#                                 Playing Matches
#                            *************************
#
#  Match #    Opponent    AB_Improved   AB_Custom    AB_Custom_2  AB_Custom_3
#                          Won | Lost   Won | Lost   Won | Lost   Won | Lost
#    1         Random      7   |   3    9   |   1    9   |   1    8   |   2
#    2         MM_Open     8   |   2    8   |   2    7   |   3    6   |   4
#    3        MM_Center    7   |   3    6   |   4    6   |   4    6   |   4
#    4       MM_Improved   8   |   2    6   |   4    4   |   6    6   |   4
#    5         AB_Open     5   |   5    4   |   6    5   |   5    5   |   5
#    6        AB_Center    7   |   3    5   |   5    7   |   3    3   |   7
#    7       AB_Improved   4   |   6    4   |   6    4   |   6    6   |   4
# --------------------------------------------------------------------------
#             Win Rate:       65.7%        60.0%        60.0%        57.1%
```

Q: **Then write up a brief summary of your results**, describing the performance of the agent using the different heuristic functions verbally and using appropriate visualizations.

Custom Score Agent

- **Name: heuristic_square_difference_available_moves**
- **Description:** Outputs a score equal to the square of the difference in the number of moves available to the two players.
- **Commentary:** The concept was to experiment with derivations of the difference between the number of moves left between players. The "Improved" evaluation function did well by only taking the simple difference, so the thought was to use something related but different (like the square) and see if any improvements could be made.
- **Evaluation:** This heuristic actually did OK (60% win rate) but not as well as the heuristic in which it was modeled after.
- **Code:**

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



```
Match #    Opponent     AB_Improved   AB_Custom
                         Won | Lost   Won | Lost
   1        Random        7  |  3      9  |  1
   2        MM_Open       8  |  2      8  |  2
   3       MM_Center      7  |  3      6  |  4
   4      MM_Improved     8  |  2      6  |  4
   5        AB_Open       5  |  5      4  |  6
   6       AB_Center      7  |  3      5  |  5
   7      AB_Improved     4  |  6      4  |  6
--------------------------------------------------
           Win Rate:       65.7%        60.0%
```

Custom Score 2 Agent

- **Name: heuristic_player_moves_vs_player_distance_to_center**
- **Description:** Outputs a score equal to the difference in the number of moves available to the two players multiplied by the active players distance to the center of the board.
- **Commentary:** Remembering that the lessons argued that the first choice being in the center of the board was an excellent one, and the **center_score** heuristic in **sample_players.py** calculated it's heuristic relative to the center of the board - the idea was to relate the moves remaining to their distance to the center of the board.
- Evaluation: This heuristic did as well as the first heuristic (60% win rate) - which is interesting.
- **Code:**

```
# Get the num of moves left for both players and take the difference
own_moves = len(game.get_legal_moves(player))
opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
moves_diff = own_moves - opp_moves

# Get the location of the player relative to the center of the board
w, h = game.width / 2., game.height / 2.
y, x = game.get_player_location(player)
dist_to_center = ((h - y)**2 + (w - x)**2)

return float(moves_diff * dist_to_center)
```

```
 Match #    Opponent     AB_Improved   AB_Custom_2
                          Won | Lost   Won | Lost
    1        Random        7  |  3      9  |  1
    2        MM_Open       8  |  2      7  |  3
    3       MM_Center      7  |  3      6  |  4
    4      MM_Improved     8  |  2      4  |  6
    5        AB_Open       5  |  5      5  |  5
    6       AB_Center      7  |  3      7  |  3
    7      AB_Improved     4  |  6      4  |  6
--------------------------------------------------------
           Win Rate:        65.7%         60.0%
```

Custom Score 3 Agent

- **Name: heuristic_player_opponent_vs_center_distance**
- **Description:** Outputs a score equal to square of the distance from the center of the board relative to the difference of the position of the player vs opponent.
- **Commentary:** This heuristic was an experiment to see if there was any correlation between the location on the board where the players were operating and the center of the board. The heuristic performed alright, better than average at 57.1% win rate, but not as well as the others.
- Code:

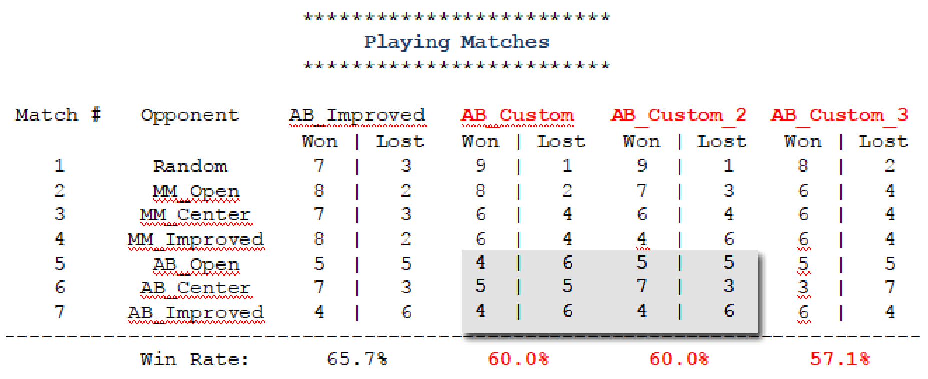
```
# Get the dimmensions of the Board
w, h = game.width / 2., game.height / 2.

# Get Player Location
py, px = game.get_player_location(player)

# Get Opponents location
oy, ox = game.get_player_location(game.get_opponent(player))

# Return the heuristic calculation
return float( (h - abs(py -oy))**2 + (w - abs(px - ox))**2 )
```

```
 Match #    Opponent     AB_Improved  AB_Custom_3
                          Won | Lost  Won | Lost
    1        Random        7  |  3     8  |  2
    2        MM_Open       8  |  2     6  |  4
    3       MM_Center      7  |  3     6  |  4
    4      MM_Improved     8  |  2     6  |  4
    5        AB_Open       5  |  5     5  |  5
    6       AB_Center      7  |  3     3  |  7
    7      AB_Improved     4  |  6     6  |  4
--------------------------------------------------------
           Win Rate:        65.7%        57.1%
```

Evaluation Function Recommendation

The evaluation function recommendation for usage is **AB_Custom_2** for the following reasons:

1. In the table below, the opponents get smarter down the list (1 being the easiest opponent, who plays a random game), and the heuristic AB_Custom_2 wins slightly more games with the more difficult opponents - see highlighted scores below.
2. The heuristic AB_Custom_2 performs at the same level **overall** as the AB_Custom heuristic
3. The heuristic AB_Custome_2 performs at the same level as AB_Custom with the **easier** opponents



```
                           *************************
                                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    8  |   2
    2         MM_Open      8  |   2     8  |   2    7  |   3    6  |   4
    3        MM_Center     7  |   3     6  |   4    6  |   4    6  |   4
    4       MM_Improved    8  |   2     6  |   4    4  |   6    6  |   4
    5         AB_Open      5  |   5     4  |   6    5  |   5    5  |   5
    6        AB_Center     7  |   3     5  |   5    7  |   3    3  |   7
    7       AB_Improved    4  |   6     4  |   6    4  |   6    6  |   4
--------------------------------------------------------------------------
           Win Rate:      65.7%        60.0%        60.0%        57.1%
```