Heuristic Analysis - Game Playing Agent

In the game of Isolation, we can abstract possible evaluation strategies into 2 techniques:

1. Following the opponent
    - This technique can be used to block the opponents next best move
2. Following maximum freedom
    - This technique values a higher number of possible moves and will generally avoid states which lead to smaller number of moves, for example: being closer to walls

To create evaluation heuristics for the techniques mentioned above, we write a few functions where given the game state and the current player, these functions return a value to estimate the how good the current state for the player. More evaluation heuristics are shown in the results table below.

Here are the top 3 evaluation methods tested:

1. **Moves Delta** *(see CUSTOM_SCORE_1 in results table below)*
   Using the number of player moves and the number of opponent moves. If the current player has higher moves for the given game state, the heuristic returns a higher value.

   ```
   value = my_moves_count - opponent_moves_count
   ```

2. **Unique Moves** *(see CUSTOM_SCORE_2 in results table below)*
   This heuristic values having a higher number of unique moves which entails being able to explore parts of the game board which the opponent cannot explore. This technique could be useful when "running away" from the opponent.

   ```
   value = my_unique_moves_count
   ```

3. **Weighted Moves Delta** (see CUSTOM_SCORE_3 in results table *below*)
   Similar to the first scoring function however, this technique values the opponents moves count with a weight. Using a weighted number of moves for the opponent allows for a more aggressive strategy.

   ```
   weight = 1.5
   value = my_moves_count - (opponent_moves_count * weight)
   ```

   After running the tournament script, this heuristic seems to perform best and consistently. It also outperforms the `AB_Improved` heuristic provided in the project.

Results:

Match #	**Opponent**	**AB_Improved**	**CUSTOM_1**	**CUSTOM_2**	**CUSTOM_3**	**DELTA_WALLS**	**DELTA_UNIQ_EXP**
		Won | Lost	Won | Lost	Won | Lost	Won | Lost	Won | Lost	Won | Lost
1	Random	10 | 0	9 | 1	9 | 1	10 | 0	10 | 0	9 | 1
2	MM_Open	9 | 1	7 | 3	7 | 3	6 | 4	8 | 2	7 | 3
3	MM_Center	9 | 1	9 | 1	9 | 1	10 | 0	9 | 1	10 | 0
4	MM_Improved	7 | 3	8 | 2	7 | 3	8 | 2	6 | 4	10 | 0
5	AB_Open	4 | 6	5 | 5	5 | 5	7 | 3	4 | 6	6 | 4
6	AB_Center	4 | 6	7 | 3	4 | 6	9 | 1	5 | 5	7 | 3
7	AB_Improved	4 | 6	6 | 4	4 | 6	5 | 5	4 | 6	4 | 6
Win Rate:		67.1%	72.9%	64.3%	78.6%	65.7%	75.7%

- *DELTA_WALLS*: a heuristic which calculates the difference in the number of moves available to each player while penalizing moves at walls towards the end of the game. This strategy might not be consistent because it checks for a very specific case which is the walls.

- DELTA_UNIQ_EXP: a heuristic which calculates the difference the in the number of unique moves available to each player, while also applying an exponential growth factor which allows the value function to become more aggressive as the reaches towards the end.

Other Notes:

- One interesting approach to take would be to introduce a changing “weight” value in weighted heuristics such as `CUSTOM_3` above. This allows the game playing agent to get more/less aggressive as the game moves on (i.e. as there are less blank moves).
  - This changing value can either be exponential growth or exponential decay.
  - An example of this is the `DELTA_UNIQ_EXP` heuristic shown in the results table above.

- The tournament results could vary which means to get really consistent results, the tournament script must be run multiple times and an average win rate can be taken across all tournament instances for each evaluation heuristic.