Heuristic Analysis

Here is the output of tournament.py with the 3 custom evaluation functions.

```
(python3) LM-SJC-11002359:AIND-Isolation ssatpati$ python tournament.py

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       17  |   3    20  |   0    20  |   0    20  |   0
    2       MM_Open      16  |   4    15  |   5    13  |   7    17  |   3
    3      MM_Center     17  |   3    16  |   4    18  |   2    19  |   1
    4     MM_Improved    13  |   7    17  |   3    12  |   8    14  |   6
    5       AB_Open      11  |   9    11  |   9     6  |  14     8  |  12
    6      AB_Center     10  |  10    12  |   8    12  |   8    11  |   9
    7     AB_Improved    10  |  10    14  |   6     9  |  11     7  |  13
--------------------------------------------------------------------------
            Win Rate:      67.1%        75.0%        64.3%        68.6%
(python3) LM-SJC-11002359:AIND-Isolation ssatpati$
```

The 3 custom evaluation functions are as follows:

1. AB_Custom: # of legal moves of player – 2 X (# of legal moves of opponent)
2. AB_Custom_2: 2 X
   (# of legal moves of player) – 3 X (# of legal moves of opponent)
3. AB_Custom_3: # of legal moves of player – 3 X (# of legal moves of opponent)

As evident from above table, AB_Custom performed the best out of all three that were tested.

Also, as alpha beta pruning *enables deeper search by limiting to more-promising subtrees*, the AB_* agents performed better than the MM_* agents, since AB_* were able to search much deeper compared to MM_* given the same timeout was applied to both.

Recommendations: Evaluation Function

Out of all three, the best evaluation function is #1 from above list for the reasons below:

1. Supported by data (above). It consistently performed the best out of the three custom evaluation functions that were tested
2. It takes into account the number of options that a player has, and penalizes for the options that the opponent has, which is both logical and intuitive

3. Performance: easy to compute at any point or state of the game