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    18  |   2     8  |  12
    2       MM_Open     18  |   2    18  |   2    15  |   5     0  |  20
    3      MM_Center    19  |   1    19  |   1    16  |   4     0  |  20
    4     MM_Improved   18  |   2    14  |   6    10  |  10     1  |  19
    5       AB_Open     16  |   4     9  |  11     8  |  12     3  |  17
    6      AB_Center     9  |  11     7  |  13     9  |  11     0  |  20
    7     AB_Improved   12  |   8     8  |  12     8  |  12     1  |  19
--------------------------------------------------------------------------
            Win Rate:      77.9%        67.9%        60.0%        9.3%
```

The 3 custom evaluation functions are as follows:

1. # of legal moves of player – 2 X (# of legal moves of opponent)
2. # of legal moves of player / # of blank spaces left
3. # of legal moves of player + 2 X (square root of distance from the center)

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. This was tried with different variations of weights, but the one above performed the best.

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