Analysis of heuristic functions for the game of isolation

The score of this analysis is to evaluate the heuristic functions that I have implemented for the Game agent. The purpose of these heuristics is to beat the baseline implementation, which is Alpha Beta with improved score.
I have three strategies that I will discuss:

1. Be defensive and try to minimize the opponents options for every move
2. Start out defensively and then switch to offense as the number of moves increases
3. Start out offensively and then switch to defense as the number of moves increases

The results of the heuristics is shown below and will be discussed

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

Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                       Won | Lost   Won | Lost   Won | Lost   Won | Lost
   1       Random      176 |  24    188 |  12    191 |   9    189 |  11
   2       MM_Open     147 |  53    153 |  47    161 |  39    147 |  53
   3      MM_Center    180 |  20    174 |  26    176 |  24    166 |  34
   4     MM_Improved   140 |  60    149 |  51    149 |  51    140 |  60
   5       AB_Open     99  |  101   99  |  101   107 |  93    101 |  99
   6      AB_Center    109 |  91    114 |  86    108 |  92    110 |  90
   7     AB_Improved   101 |  99    103 |  97    105 |  95    93  |  107
--------------------------------------------------------------------------
          Win Rate:      68.0%        70.0%        71.2%        67.6%
```

The results shows it might be most beneficial to be defensive in the beginning. The first two algorithms, AB_Custom and AB_Custom_2, uses this principle. As can be seen they both beat the AB_Improved score.

The results were achieved by setting the number of matches to 100 in the tournament.py script. 100 was chosen to achieve a broader foundation to better judge the algorithms.

AB_Custom: block the opponent

In this function the player p will try to block the opponent o from the beginning. By setting ε to 2, we assure strictly blocking behaviour from the start till the end.

$$h(g,p) = m(p) - (\varepsilon * m(o))$$

The strategy is to just have more legal moves than the opponent and not if this move will search for the best result offensively.

AB_Custom_2: start out defensively

This is the opposite of heuristic 1. In this function the player will start out blocking the opponent and then gradually become more offensive. Again, this is based on the number of legal moves for the player $m(p)$ minus the number of legal moves for the opponent $m(o)$.
We have a start out blocking weight of 2 ε and an empirically tested decreasing value of 0.5 γ which is multiplied with the number of moves in the game gm so far.

$$h(g,p) = m(p) - (\varepsilon * m(o) - (\gamma * gm))$$

In this function we subtract the calculated increasing factor that is the result of the increasing moves in the game from the defensive value.

This heuristic seems to perform equally to

AB_Custom_3: start out offensively

This heuristic is formed based on the number of legal moves for the player minus the number of legal moves for the opponent. It then adds a move_count factor which consists of a weight, which is empirically set to 0.5 and then multiplied with the number of moves in the game.
In this case we have the heuristic function $h(g,p)$ where g is the game and p is the player. In the function we get the legal moves for the player p $m(p)$ and the legal moves for the opponent o $m(o)$.
We also see the weight γ that multiplies with the number of moves in the game gm so far.
We have this function:

$$h(g,p) = m(p) - (m(o) + (\gamma * gm))$$

This will start out with a strategy that seeks a move that gives more options than the opponent, but besides this does not care too much about trying to block the opponent. This will change as the game progresses, so eventually the player will try to block more and more as the game progresses.

Conclusion

The algorithm that has achieved the best result is AB_Custom_2. This is the algorithm that starts out defensively and then switch to offense as the game progresses.
This algorithm beats the AB_Improved with a score of 71.2% winning, whereas the AB_Improved has score of 68%. It means that AB_Custom_2 wins with a margin of 3.2 percentage points.
However the first algorithm, AB_Custom, also beats the AB_Improved score with 2 percentage points.

The recommendation is to use algorithm AB_Custom_2 as the preferred algorithm.