Juliana Yurie de Heer
Project: Advanced Game Playing
September 10, 2017

Part 1: Coding and Analysis

For my custom heuristics, I first implemented one that gives a weight to each possible move based on its location. I had this idea after observing that, on an empty board, from any of the middle positions (i.e., positions within a margin of at least 2 squares from the edge on every side), there are 8 possible moves, and, this number decreases as we get closer to the corners. It’s easy to prove by mathematical induction that the middle positions are always a better choice on an empty board for any board with width and height greater than or equal to 4 (see images below for square boards of sizes 2x2 to 8x8. They describe the number of available moves from each position on an empty board.)

<table>
<tr><th>2x2</th><th>3x3</th><th>4x4</th><th>5x5</th></tr>
<tr>
<td>0 1
0 | 0| 0|
1 | 0| 0|</td>
<td>0 1 2
0 | 2| 2| 2|
1 | 2| 0| 2|
2 | 2| 2| 2|</td>
<td>0 1 2 3
0 | 2| 3| 3| 2|
1 | 3| 4| 4| 3|
2 | 3| 4| 4| 3|
3 | 2| 3| 3| 2|</td>
<td>0 1 2 3 4
0 | 2| 3| 4| 3| 2|
1 | 3| 4| 6| 4| 3|
2 | 4| 6| 8| 6| 4|
3 | 3| 4| 6| 4| 3|
4 | 2| 3| 4| 3| 2|</td>
</tr>
</table>

<table>
<tr><th>6x6</th><th>7x7</th><th>8x8</th></tr>
<tr>
<td>0 1 2 3 4 5
0 | 2| 3| 4| 4| 3| 2|
1 | 3| 4| 6| 6| 4| 3|
2 | 4| 6| 8| 8| 6| 4|
3 | 4| 6| 8| 8| 6| 4|
4 | 3| 4| 6| 6| 4| 3|
5 | 2| 3| 4| 4| 3| 2|</td>
<td>0 1 2 3 4 5 6 7
0 | 2| 3| 4| 4| 4| 4| 3| 2|
1 | 3| 4| 6| 6| 6| 6| 4| 3|
2 | 4| 6| 8| 8| 8| 8| 6| 4|
3 | 4| 6| 8| 8| 8| 8| 6| 4|
4 | 4| 6| 8| 8| 8| 8| 6| 4|
5 | 4| 6| 8| 8| 8| 8| 6| 4|
6 | 3| 4| 6| 6| 6| 6| 4| 3|
7 | 2| 3| 4| 4| 4| 4| 3| 2|</td>
<td>0 1 2 3 4 5 6
0 | 2| 3| 4| 4| 4| 3| 2|
1 | 3| 4| 6| 6| 6| 4| 3|
2 | 4| 6| 8| 8| 8| 6| 4|
3 | 4| 6| 8| 8| 8| 6| 4|
4 | 4| 6| 8| 8| 8| 6| 4|
5 | 3| 4| 6| 6| 6| 4| 3|
6 | 2| 3| 4| 4| 4| 3| 2|</td>
</tr>
</table>

With this in mind, I designed a heuristic that weighted each possible move based on these numbers (which works on any board greater than a 4x4 since their distribution is symmetrical). My thought was that, in the case when we have, say, 2 positions with the same number of available moves, I wanted to favor the one for which the next moves available are located closer to the center.

Although choosing a center position is a fine heuristic for determining the first move, in reality this proved to be the worst performing heuristic in almost all of my tests. Thus this was named `custom_score_3`.

As for the other two heuristics, I wanted to use a modified version of `improved_score` to test whether giving more weight to `own_moves` versus `opp_moves` would make a difference. As it turns out, the giving more weight to the latter consistently increased the win rate, while the first consistently decreased it, as compared to `improved_score`. Thus I named the first heuristic `custom_score_2`, which is defined as `2*own_moves - opp_moves`. My winning heuristic, `custom_score`, was defined as `own_moves - 2*opp_moves`. I also tried increasing these weights, but that didn't seem to make a difference, as expected.

Below is a sample tournament:

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

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      10  |   0    10  |   0    10  |   0    10  |   0
    2       MM_Open      7  |   3     8  |   2     6  |   4     7  |   3
    3      MM_Center     9  |   1     8  |   2    10  |   0    10  |   0
    4     MM_Improved   10  |   0     7  |   3     6  |   4     5  |   5
    5       AB_Open      5  |   5     9  |   1     6  |   4     5  |   5
    6      AB_Center     7  |   3     5  |   5     6  |   4     6  |   4
    7     AB_Improved    5  |   5     7  |   3     4  |   6     6  |   4
--------------------------------------------------------------------------
           Win Rate:      75.7%        77.1%        68.6%        70.0%

There were 5.0 timeouts during the tournament -- make sure your agent handles search
timeout correctly, and consider increasing the timeout margin for your agent.
```

While a formal and more systematic evaluation should be conducted to confirm these findings, these results suggest that playing aggressively, i.e., as to minimize your opponent's moves, is a more successful strategy. Take, for instance, the board below (X indicated available moves for me, the next to act):

```
     0   1   2   3   4   5   6
0  | X | X | X | X | X | X | X |
1  | X | X | X | X | X | X | X |
2  | X | X | X | X | X | X | X |
3  | X | X | X | X | X | X | X |
4  | X | X | X | X | 1 | X | X |
5  | X | X | X | X | X | X | X |
6  | X | X | X | X | X | X | X |
```

Here, my opponent was the first to move and has 8 available positions for their next turn. Any heuristic favoring number of available moves would choose one of the positions in the middle, but we're faced with the option to either block one of our opponents next move, reducing both our available moves for the next turn, or move to a space both of us remain with 8 available moves. The table below describes the function evaluation for all 4 heuristics comparing 3 sample moves, one that blocks our opponent and two that doesn't:

	AB_Improved own_m - opp_m	AB_Custom own_m - 2*opp_m	AB_Custom_2 2*own_m - opp_m	AB_Custom_3 Weighted own_m-opp_m
(2,2) doesn't block	**8 - 8 = 0**	8 - 16 = -8	**16 - 8 = 8**	3*2+4*2+6*2+8*2 = 42
(3,2) blocks	7 - 7 = 0	**7 - 14 = -7**	14 - 7 = 7	4*3+6*2+8*3 = 48
(3,3) doesn't block	8 - 8 = 0	8 - 16 = -8	16 - 8 = 8	**6*4+8*4 = 56**

The moves that would be picked appear highlighted (note that, although some evaluate to the same result, AlphaBeta picks the one on the left). `AB_Custom_3` successfully picks the most central option, while `AB_Custom` is the only one that chooses an option that provides a lower number of possible moves for ourselves because it prioritizes minimizing the number of moves for our opponent.

I had never played Isolation before, but this strategy of trying to block our opponent does sound more like some of the strategies presented by Thad, which leads me to conclude that playing aggressively, as to minimize our opponents moves even at the cost of our own seem to be generally more successful. Thus I chose `own_m - 2*opp_m` as my preferred heuristic.