Isolation Heuristics

In testing my heuristics, I've found that minimax fairly consistently outperforms alphabeta. This is most likely due to the iterative deepening taking a long time to reach a depth significant enough to be able to predict as well as the minimax. While alphabeta prunes branches deemed unnecessary, it still eliminates possibilities nonetheless. It's possible that the heuristic evaluation at an earlier branch that caused alpha beta to prune turned out to be a min or max when further explored by the depth limited minmax function.

```
                         *************************
                              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      8  |   2    10  |   0     7  |   3     9  |   1
    3      MM_Center    10  |   0     9  |   1     9  |   1    10  |   0
    4     MM_Improved    8  |   2     6  |   4     8  |   2     6  |   4
    5       AB_Open      5  |   5     7  |   3     5  |   5     5  |   5
    6      AB_Center     8  |   2     6  |   4     6  |   4     7  |   3
    7     AB_Improved    5  |   5     5  |   5     4  |   6     3  |   7
--------------------------------------------------------------------------

        Win Rate:      77.1%        75.7%        70.0%        71.4%
```

I noticed that the heuristic *custom_score_3* performed well in general, but terribly against the AB_improved. This is likely because it was based on the number of legal moves each player had at any given board state, but each move was weighted for being closer to the edges of the board. A player's distance from the edges may be relevant all game, but the sum of all legal moves weighted by their distance from the edge is virtually useless in the very beginning of the game. What probably happened

here, is the opponent's mins were wrongly assumed early on, causing alphabeta to prune the wrong branches.

The function *custom_score_2* performed pretty well, even against the *open_moves* evaluation, despite this specific test case. I expected this function to soar over *open_moves,* since it's evaluation includes the number of open spaces within 2 spaces of each player. The reasoning behind this was that all open spaces within 2 spaces can be future legal moves for the agent. This also makes this heuristic slightly volatile, since two moves or an increase in three to four depths will almost completely void any previous evaluation as the pieces could be four or more spaces away, making all those open spaces unreachable.

The final function *custom_score_1* had slightly better results than the previous functions. This may seem strange, since it only compares the players distances from the corners of the board, but I think it works because iterative deepening is so time consuming that it doesn't get a chance to evaluate several branches ahead, where it would likely be much less indicative to the board's end state.

I tested this theory by doubling the timeout from 150ms to 300ms. The results can be viewed on the last page. Although each game lies on a randomly chosen first move, that could be determining the way the match sways, I think there is still a clear trend. *custom_score_3* performs much better when it is given enough time to go deeper for each move. The reason for this is because legal moves near the edge of the board means that the player is a lot more likely to lose the game than those same moves towards the beginning of the game. This makes this function much more reliable than the other two. I didn't want to change the parameters of the scoring function, but I think that allowing depth to weigh the score as well could make this function even more successful.

Perhaps using a more general heuristic like the first one towards the beginning of the game, then switching to *custom_score_3* after the first few moves could increase its usefulness. I think that *custom_score_3* most closely mimics a real person's behaviors. This function was thought up when I played a couple of practice games, and realized that if legal moves were near the edge, I would do anything to avoid playing them, at least towards the end of the game. Of course if there is a hard time limit of 150ms, then I would choose *custom_score_1* whose scoring is the least affected by how many depths into the future it is allowed to explore.

****
Playing Matches with TIME_LIMIT
increased to 300ms
****

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	9	1	10	0	9	1	10	0
2	MM_Open	9	1	10	0	9	1	9	1
3	MM_Center	9	1	9	1	10	0	10	0
4	MM_Improved	7	3	8	2	8	2	8	2
5	AB_Open	7	3	6	4	6	4	6	4
6	AB_Center	5	5	3	7	5	5	9	1
7	AB_Improved	5	5	4	6	7	3	5	5
	Win Rate:	72.9%		71.4%		77.1%		81.4%	