Vaibhav Kumar
Udacity - AIND

Project 2: Build a Game-Playing Agent
Heuristic Analysis

Heuristics selected

The three custom heuristics chosen for this project are described below. For consistency, they will be referred to by the name of their functions through this report (i.e. `custom_score`, `custom_score_2`, and `custom_score_3`).

`custom_score`
This was a simple improvement on `AB_Improved`, by implementing a higher penalty on the opponent's available moves, thus making our own player much more aggressive. The difference in available legal moves is considered, while weighting the opponent's moves by a factor of 3.

`custom_score_2`
Here, our player gradually becomes more aggressive as the game progresses. This is done by computing the percentage of remaining blank spaces on the board, and weighting the opponent's available legal moves by the proportion of board covered so far.

`custom_score_3`
This heuristic produces a higher value the further away the players are from each other. The distance between the two players is calculated using the Manhattan distance, which is just the sum of the X and Y distances.

Results

The results from executing tournament.py with the custom functions and `AB_Improved` are below.

```
                    *************************
                         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     9  |   1     9  |   1
    2       MM_Open      7  |   3     8  |   2     6  |   4     8  |   2
    3      MM_Center    10  |   0    10  |   0     9  |   1     9  |   1
    4     MM_Improved    5  |   5     9  |   1     7  |   3     8  |   2
    5       AB_Open      8  |   2     6  |   4     5  |   5     3  |   7
    6      AB_Center     6  |   4     6  |   4     6  |   4     6  |   4
    7     AB_Improved    6  |   4     3  |   7     6  |   4     3  |   7
--------------------------------------------------------------------------
           Win Rate:      74.3%        74.3%        68.6%        65.7%
```

Analysis

From the results, it is seen that best performing heuristics were that by `AB_Improved` and `AB_custom` (`custom_score()`), with both having an average win rate of 74.3%. The other 2 custom heuristics did not perform as well, but are not too far behind at 68.6% and 65.7%, respectively.

If we break down the results into the selected heuristics competing against the opponent of different heuristic types, we see the following average win rates.

Opponent groups	(Group description)	AB_Improved	AB_custom	AB_custom_2	AB_custom_3
All	**Same as win rate produced by tournament.py across all opponent types**	74.3%	74.3%	68.6%	65.7%
Random	**Only random**	100%	100%	90%	90%
MM	**Average across all MM**	73.3%	90%	73.3%	83.3%
AB	**Average across all AB**	66.7%	50%	56.7%	40%
Open	**Average across all Open**	75%	70%	55%	55%
Centered	**Average across all Centered**	80%	80%	75%	75%
Improved	**Average across all Improved**	55%	60%	65%	55%

With this breakdown, it is possible to see that our heuristics, in general, perform better against a Minimax agent, with the average win rate all being above 70% (3rd row), whereas against AlphaBeta all our win rates score below 67% and as low as 40% (4th row).

Similarly, it appears that the centered strategy (row 6) applied by the opponent, wherein the score is based on the distance from the board center, is not as good as the others, as our player is generally able to secure a higher win rate against this opponent (75-80%).

The random opponent is, unsurprisingly, the easiest opponent to win against (row 2), with win rates ranging from 90-100%. This is because a random move is selected from the available moves by the opponent, without consideration for the opponent, position on board, or anything else.

Except for against the AlphaBeta opponents and agents using ‘Improved’ function, `AB_custom` (`custom_score`) fares better than the other custom heuristics everywhere else.

Recommendation

From the custom evaluation functions defined in this project, my recommendation would be to choose `AB_custom` (`custom_score`), which makes our player aggressive and attempts to minimize the available moves for the opponent. The reasons for this are mainly that it is the most consistently high performing of the lot, particularly against Minimax agents as discussed above. It is quite straightforward to compute for a high efficiency. It is able to provide the best results while being valid across any part of the board and the opponent's location and game progress are irrelevant.