Evaluation of the three custom heuristic functions for the isolation game

Since it is not possible to explore the entire game tree because of the exponential increase in size with increase in depth, we search only up to a limited depth. We approximate the value of a branch at a specified depth to ascertain its goodness compared to other branches. We identify the most promising branch and explore it further. Since we are considering zero-sum adversarial games, both players attempt to optimize their gains. The goodness of a node in the game tree is evaluated using heuristic evaluation functions.

We have considered three heuristic functions for the isolation game and compare their performance against a basic agent that uses iterative deepening and a basic heuristic function: $(\#\ own\ moves\ -\ \#\ opponent\ moves)$
The heuristic functions that we have taken are described below:

1. $\boldsymbol{custom\_score = len(own\_moves) / len(opp\_moves)}$
   We calculate the number of moves available to both agents and divide the moves available to 'Student' by the number of moves available to opponent. When there is Zero Division Error, we add 1 in the denominator to handle it. This way, we will tend to prefer branches where our agent has greater number of available moves compared to the opponent. One thing to observe is that in the beginning of the game, both own_moves and opp_moves will have large values, which will make their proportion small. While near the end of game, their smaller sizes will give larger scores. Thus initially, custom_score will have smaller variability, while in the end of game, the it will have larger variability. This is realistically true and helps provide more promising paths, reducing chances of opting misleading paths near the leaves. Following tables helps to visualize the results. We see that 'Student' has slightly better performance than ID_Improved. Though for both agents, two matches are tie and rest are wins, but the average of 'Student' is more than the 'ID_Improved'.

Match ID	Opponent Player	ID_Improved	Student
Match 1	Random	15 to 5	15 to 5
Match 2	MM_Null	14 to 6	14 to 6
Match 3	MM_Open	10 to 10	12 to 8
Match 4	MM_Improved	10 to 10	10 to 10
Match 5	AB_Null	14 to 6	16 to 4
Match 6	AB_Open	13 to 7	10 to 10
Match 7	AB_Improved	11 to 9	14 to 6
Overall Percentage		**62.14 %**	**65.00 %**

2. $\boldsymbol{custom\_score = (weighted\_own\_moves) / (weighted\_opp\_moves)}$
   In this heuristic, we try to weigh each legal move with the number of subsequent legal moves possible if we take that move. So, it's like peeking one step further, though it does not consider the move by adversary in between. So, it's like differentiating between all available moves and not considering them equal. Though one subsequent move will bring

the player back to its previous location. But we ignore that here. This seems better as some moves will put the player in the corner of board, thus limiting his options. Finally, we divide the weighted_own_move by weighted_opp_move to get the custom score. Following tables show that this scheme indeed performs better. We also observe that 'Student' wins all matches with large margins, and there is no tie. So, the performance with this heuristic is quite good compared to ID_Improved, which ties two of its matches.

Match ID	Opponent Player	ID_Improved	Student
Match 1	Random	18 to 2	18 to 2
Match 2	MM_Null	15 to 5	12 to 8
Match 3	MM_Open	10 to 10	14 to 6
Match 4	MM_Improved	10 to 10	13 to 7
Match 5	AB_Null	14 to 6	17 to 3
Match 6	AB_Open	11 to 9	12 to 8
Match 7	AB_Improved	15 to 5	15 to 5
Overall Percentage		**66.43 %**	**72.14 %**

3. $\boldsymbol{custom\_score = (weighted\_own\_moves - weighted\_opp\_moves)}$

This heuristic is similar to the second heuristic except that we find difference between the weighted moves of both players to calculate the heuristic score. Though, it also performs better than the basic agent, the results for the second heuristic are still better compared to this heuristic, because the division operator provides more variance in scores down the tree, and hence reduces mistakes in choosing right branch. Following tables help in the visualization and comparison:

Match ID	Opponent Player	ID_Improved	Student
Match 1	Random	17 to 3	18 to 2
Match 2	MM_Null	16 to 4	16 to 4
Match 3	MM_Open	9 to 11	11 to 9
Match 4	MM_Improved	10 to 10	11 to 9
Match 5	AB_Null	14 to 6	12 to 8
Match 6	AB_Open	15 to 5	14 to 6
Match 7	AB_Improved	11 to 9	14 to 6
Overall Percentage		**65.71 %**	**68.57 %**

So, we see that 'Student' performs better than 'ID_Improved' in all the three custom heuristics that are defined above. But still we would like to choose one, and compare their performances among themselves.

Comparison of above heuristic scores

To compare the above three custom heuristic scores, I would like to summarize them in the table given below:

Match ID	Players	Heuristic 1	Heuristic 2	Heuristic 3
Match 1	Student vs Random	15 to 5	18 to 2	18 to 2
Match 2	Student vs MM_Null	14 to 6	12 to 8	16 to 4
Match 3	Student vs MM_Open	12 to 8	14 to 6	11 to 9
Match 4	Student vs MM_Improved	10 to 10	13 to 7	11 to 9
Match 5	Student vs AB_Null	16 to 4	17 to 3	12 to 8
Match 6	Student vs AB_Open	10 to 10	12 to 8	14 to 6
Match 7	Student vs AB_Improved	14 to 6	15 to 5	14 to 6
Overall Percentage		**65.00 %**	**72.14 %**	**68.57 %**

As we can visualize from the above table, by using heuristic 1 we get tie against two opponents, while for Heuristic 2 and heuristic 3, all matches are wins. We also note that the overall average score of heuristic 2 is more than the average of heuristic 3. So, I would like to choose Heuristic 2, and support it on the basis of following reasons:

1. Heuristic 2 has highest score among all three heuristics that we have considered.
2. Heuristic 2 wins all its matches against all the seven opponents.
3. Though heuristic 3 is comparable to heuristic 2, but as we have noted, heuristic 2 will give more variability of scores corresponding to competing branches near the end of game, and thus more likely to not select suboptimal branch. Hence, heuristic 2 is preferred.