Heuristic Analysis

I ran the tournament test three times for each of my three evaluation heuristics, for a total of 9 runs. Here are the results.

The first heuristic I used was a simple one: giving the number of opposing player number of legal moves double the importance -> return(own_moves – 2*opp_moves) This will make the agent play more aggressive and prioritize leavening your opponent with less moves.

The three scores that I got with this heuristic were:

ID_Improved			
1	2	3	Average
62.86%	72.86%	67.14%	67.62%
Student			
1	2	3	Average
72.14%	65%	66.43%	67.856%

The first thing that caught my attention was the variability between the scores. In the first test, ID_Improved did pretty bad at 62.86%, and Student did very well with 72.14. On test 2 though, the results were the opposite: ID_Improved performed much better with 72.86% compared to the 65% of Student. The third test didn't say much, both agents performing similarly. Overall, the Student agent managed to come out on top with an average of 67.856%, a 0.236% difference, which judging by the variability of the results, doesn't say too much.

The second heuristic I used implements the default heuristic with a twist, it has a tie-breaker feature, which gives a bit more weight to moves that take the player to a state where the number of blank spaces is an odd number, which from experience from playing the game, I believe it may be advantageous in most cases ->

```
if len(game.get_blank_spaces()) % 2 == 0:
        bonus = 0
else:
        bonus = 0.5
return(own_moves - opp_moves + bonus)
```

Here are the results for the three tests.

ID_Improved			
1	2	3	Average
68.57%	69.29%	65.71%	67.856%
Student			
1	2	3	Average
70%	70%	65.71%	68.57%

This second set of trials resulted in much more stable performances, which makes me wonder if there are other variables at play, like the stability of my computer for example. At the time I ran the tests, I didn't use the computer, but I didn't close other applications I had running. Maybe there was a process running on the background that interfered on the test results.
Nonetheless, this heuristic function seemed to perform very well, with consistent wins on every test made, and a difference of 0.714% of the average of the results compared to the ID_Improved agent.

The third heuristic I used implements a combination of the first two, it gives double the importance to the number of opponent legal moves, and it also has the tie-break feature->

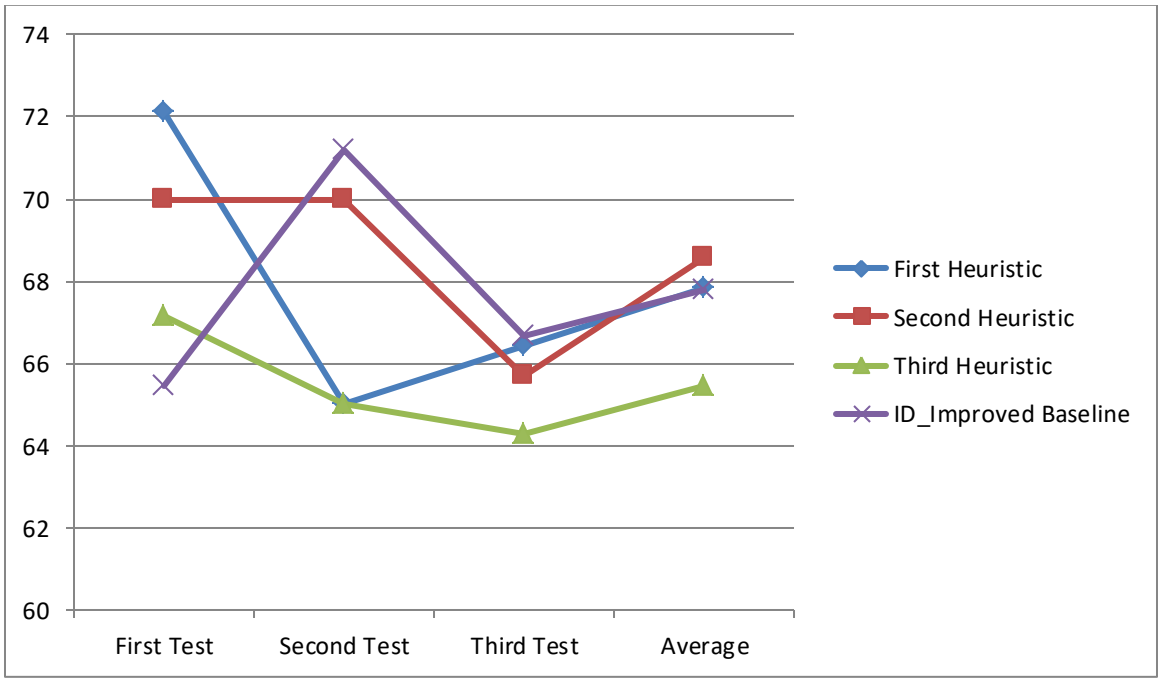
```
if len(game.get_blank_spaces()) % 2 == 0:
        bonus = 0
else:
        bonus = 0.5
return(own_moves – 2*opp_moves + bonus)
```

Here are the results for the three tests.

ID_Improved			
1	2	3	Average
65%	71.43%	67.14%	67.856%
Student			
1	2	3	Average
67.14	65%	64.29%	65.476%

We can see that this was the only heuristic function that did not perform better than the ID_Improved agent. It only performed better during the first run, but on average, it stayed behind for -2.38% which is significantly lower compared to the advantage that the other heuristics managed to get.

Here is a graph to make it easier to digest the data.



Although the first heuristic had the best overall performance for a specific test, it got outperformed on average by the second heuristic, the latter was more consistent in its performance.

For the ID_Improved Baseline, I computed the averages of the three tests for each of the three heuristic tests, and finally, an average of all of these tests. It's interesting to note that the first heuristic and the ID_Improved agent had almost the same average performance, and it is good to know that considering the average of all computed values for the ID_Improved agent (a total of 9), the second heuristic still came on top.

Ultimately, I decided to keep the second heuristic, because it had the best overall advantage over the ID_Improved agent, because its results were the most consistent and it implements an additional feature (tie-break) compared to the second best heuristic. I do think that more tests would be needed to ensure a best pick, and many more tests to find a significantly better heuristic function. It was interesting to note that although the first two heuristics performed better than the ID_Improved agent (although by a small margin), a heuristic made up of a combination of these two heuristics performed worse than its standalone versions. This makes me think that designing efficient heuristics can be pretty complex and may not be intuitive in many cases.