Build a Game Playing Agent - Heuristics Analysis

The implementations of the following search algorithms were tested in similar conditions –

- Minimax search
- Alphabeta search
- Iterative deepening with both MM and AB

The differences between the Student and the Improved ID players were - the randomness of the start of the play and the heuristics functions.

The results from the multiple runs averaged out across the four heuristics functions are as follows –

Heuristic Name	ID Improved Score (%)	Student Score (%)	Depth	Complexity	Heuristics Performance	Heuristics Details
A	72.14	70.71	Deep	Low	Low	Worst Performance of the lot. If the Student has fewer move options than the opponent, the function returns: - (# moves the opponent has)/(# moves the computer has)**2. If the Student has more move options than the opponent, the function returns (# moves the opponent has)/(# moves the computer has)**2. If a move causes the computer to have greatly proportionally fewer moves than the opponent, it is a bad move.
B	72.14	73.57	Deep	Low	Low	Aggressive play in the first half of the game. Active player will try to choose the most aggressive move. Heuristic calculates number of players move vs 3.5 of value of an opponent's moves. In the second half of the game heuristic will calculate number of players move vs number of an opponent's moves.
C	72.14	76.43	Deep	Low	Medium	Similar to "B", Most aggressive initially, then drop aggressiveness at 1/3 moves and further more at 1/4 moves remaining.
D	72.86	75	Deep	Low	Medium	Similar to "B", Least aggressive initially, then increase aggressiveness at 1/3 and further more at 1/4 moves remaining.
E	75.71	76.43	Medium	Medium	Medium	own_moves ^2 / opp moves
F	75.71	77.14	Medium	Medium	High	Best Heuristic from the test results, Similar to "E" : own_moves ^3 / opp moves
G	72.86	74.29	Medium	Medium	Low	Similar to "E": own_moves ^4 / opp moves^2
H	74.29	76.43	Medium	Medium	Medium	(own_moves - opp_moves) / cells_left

After the inputs from the reviewer, I ran the above tests. They have been grouped (under similar categories) as follows –

Heuristic Group	Analysis	Prediction
A	Low complexity, goes deep in terms of layers but doesn't seem to have a good win rate based on the tests	Less likely to win
B, C and D	Average complexity, shallow in terms of layers but some of the specific cases seem to have a good win rate based on the tests	Less likely to win
E, F and G	Average complexity, shallow in terms of layers but some of the specific cases seem to have a good win rate based on the tests. One has the highest (Heuristics F)	Most likely to win "Heuristics F"
H	Average complexity, shallow in terms of layers but doesn't seem to have a good win rate based on the tests	Less likely to win

Tests were run with minor variations as seen above with B, C, D and E, F, G. Any visible impact with minor variations to the heuristics were to be measured. In some cases, there were some variations observed in others not so much.

If we increase the complexity of the heuristics although there is a tradeoff of higher performance to shallow searches, it may prove to be beneficial in some case to not go deeper, as is observed with Heuristics F.

For larger problems with higher time and space complexity, going deeper may not be an option most of the times, so having a complex heuristics function may help in such cases with some approximation of the score value. After submission of the project, When I find some spare time, I plan on using a neural network on training the evaluation function. The idea would be for the features to be fed into the NN and single score returned that will be used as the score for the node in question.