Heuristic analysis

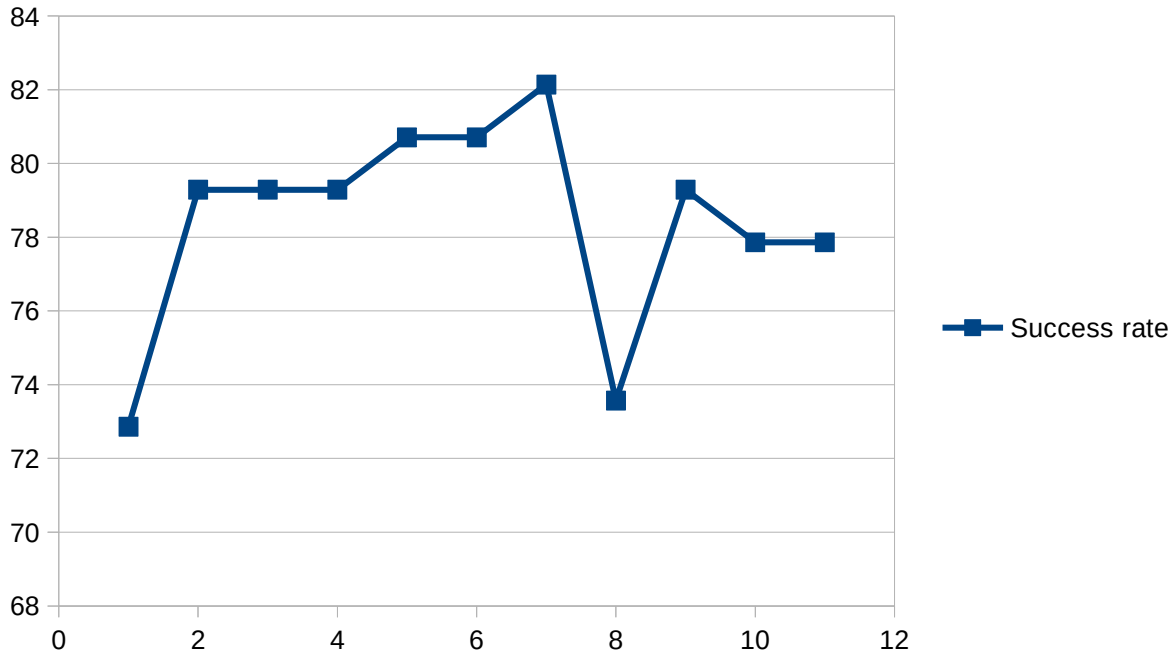
Please note for the following summary the scores listed next to heuristics were achieved using the original tournament code and agent skeleton. However, the conclusion in the end uses the new tournament code and agent skeleton required for project assistant.

For the game of isolation the following heuristics were tested:

Heuristics #1:
#player moves – k * #opponent moves

The rationale behind this heuristics is to provide incentive to go after moves that favor number of remaining player moves compared to remaining opponent moves. The rate is controlled by the *k* parameter, telling us how much we weight opponent moves.

The graph below suggests the optimal k=7 with 82.14% success rate. This rate was obtained using the original skeleton code; using the new skeleton code for project assistant the success rate was 70%.



Heuristics #2:
#player moves - #previous player moves - (#previous opponent moves - #opponent moves)
The success rate of this heuristics was 76.43% in the original tournament code.

This heuristics tries to use delta between available player moves in current and previous turn and opponent moves in previous and current turn. It then subtracts opponent delta from player delta, expecting to favor outcome where the expectation is that it would favor moves that result in smaller decrease of available player moves and larger decrease of opponent moves.

Heuristics #3:
#player moves / #previous player moves - #previous opponent moves / #opponent moves
The success rate of this heuristics was 73.57% in the original tournament code.

This heuristics tries to use ratio between available player moves in current and previous turn and opponent moves in previous and current turn. It then subtracts opponent ratio from player ratio, with the hope that moves resulting in higher remaining moves player ratio than opponent ones.

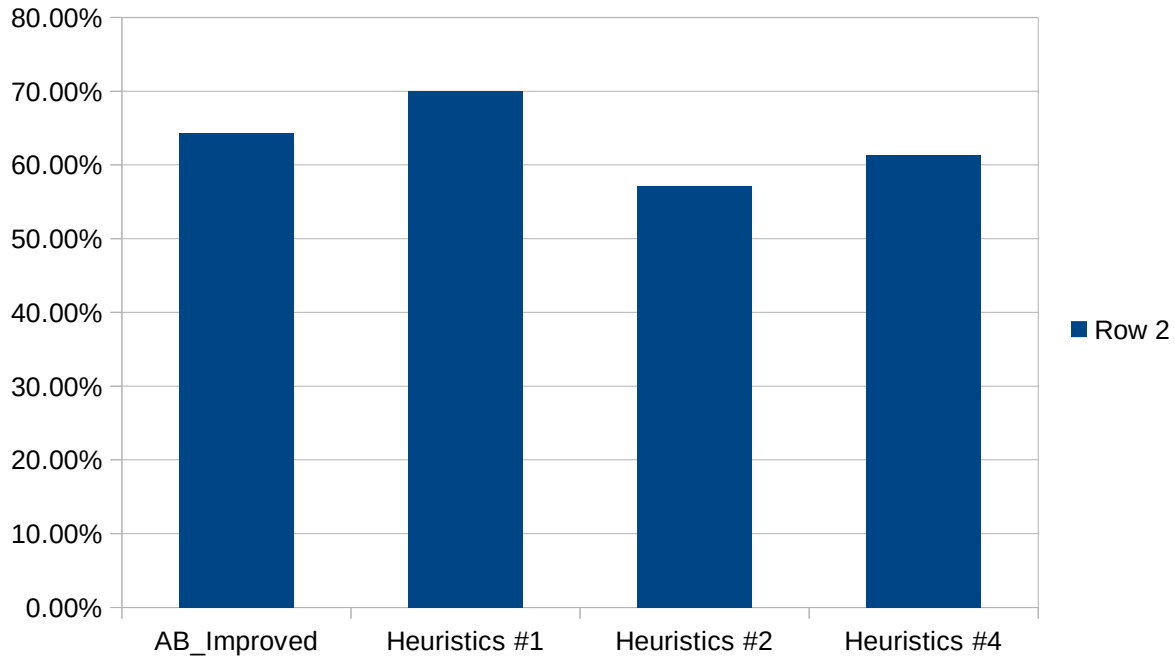
Heuristics #4:
(#player moves / #previous player moves) / (#previous opponent moves / #opponent moves)
The success rate of this heuristics was 80.71% in the original tournament code.

This heuristics is based on the previous one; instead of subtracting move ratios it tries to divide them, hoping this introduced non-linearity might bring better results than just subtraction.

Heuristics #5:
case #player moves >= #opponent moves:
#player moves / #opponent moves - (#previous opponent moves - #opponent moves)
else
-(#opponent moves / #player moves) - (#previous opponent moves - #opponent moves)
The success rate of this heuristics was 75.00% in the original tournament code.

This heuristics takes into account ratio between player and opponent moves as well as a delta between previous and current opponent moves. It hopes to maximize player / opponent moves ratio while forcing opponent moves that result in highest drop of available moves. It splits the ratio computation depending on whether player or opponent has more remaining moves to make sure it is not < 1.0 and thus negligible.

Conclusion



Taking the top three best performing heuristics (#1, #2, #4) and running them with the new tournament code, the results look as follows (the first column is the reference ID Improved):
As the graph shows, the **best performing** heuristics is ***#1***, **#player moves – 7 * #opponent moves** with a tournament success rate of 70.00%, hence it was the chosen one. Moreover, this heuristics is **trivial to implement** comparing to other ones that need to maintain a stack of previous available moves, and its **execution speed is very fast**, requiring only subtraction and a multiplication by a constant.