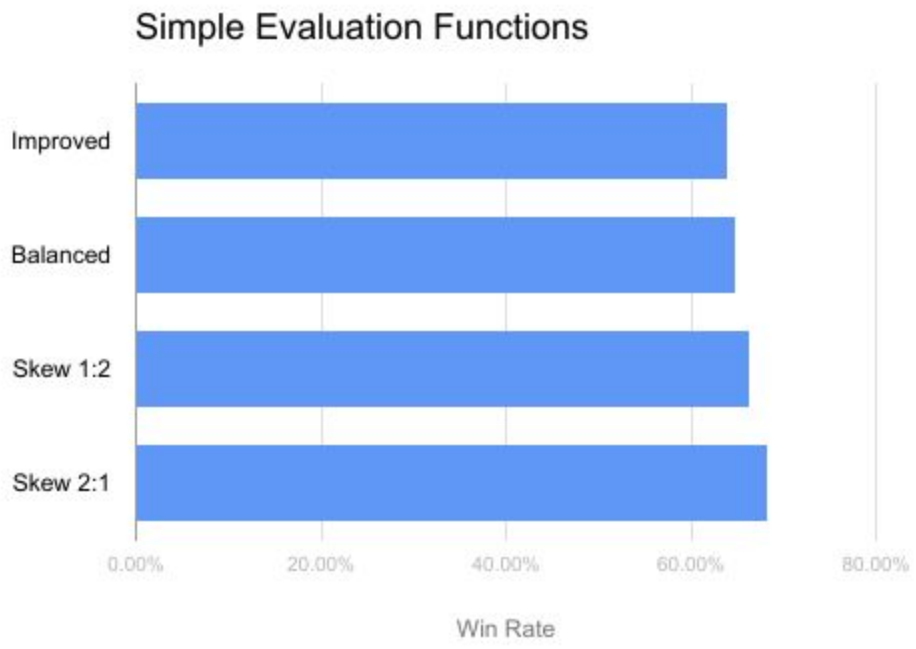
Heuristic Analysis

In the game Isolation an evaluation function is an heuristic that estimates a utility for any game state. I select three simple evaluation functions to explore two easily computed metrics and their combination. The number of my moves and my opponent's moves relate directly to mobility, and the game ends when mobility goes to zero. As a utility function, they are shortsighted since one move can open up the available moves while another can close them down, but being fast will allow the iterative deepening to look farther in compensation.

I begin with the balanced evaluation of my moves and my opponent's moves. To that I add two skewed variants, one weighing my moves twice my opponents', and the other weighing my opponent's moves twice mine. Out of 700 trials over a variety of opponents, the worst performer was the balanced function but the other functions were close. In a test of statistical significance, none of these simple evaluation functions perform significantly better than Improved. All of them (including Improved) perform significantly better than Random and are sufficient to pass as an 'intelligent' agent. These are the functions I will submit, although I did explore the performance of other functions, and I describe that exploration here. I also describe the steps I would take if I were to continue working on this assignment and apply machine learning and optimization methods.



Clearly, the Improved evaluation function and my set of simple evaluation functions must be similar in nature. It turns out that Improved is actually identical to Balanced in my set. There are two other evaluation functions being used by the tournament, Open and Center. Open is the precursor to improved, which is essentially a skew where my opponent's moves have zero weight. It makes sense then, that they should score similarly to my functions as they are composed of the same factors. Center is the only one with a different feature, player distance from center. Given that varying the weight of my opponent's moves makes no significant difference, we can conclude that the dominant factor is my moves. It might be improved by incorporating the new feature from center, but center performs poorest of all the functions by a wide margin. A

quick change to Center inverting the value of the distance makes it a stronger contender than Open and Improved.

I set up the new evaluation function, Combined, as the linear combination of my moves and my distance from center. I ran the tournament again, this time giving them equal weights and skewed weights as I did the previous set of candidates. I tested them against the Improved function, and unfortunately, none performed well. I then considered that the optimal weights for each feature may change as the game progresses, with centrality being more important in early game than the late game. I added a decay parameter to centrality and tried a few combinations of my moves plus decaying centrality, but found no decay rate performed better than Improved.

I next collect data from simulated gameplay, where moves are made by an agent that mixes random exploration and strategic exploitation to better explore new states. Each visited state is evaluated in feature space, and the outcome of the game for the player (0 or 1) recorded on all visited states at the end of each game. This would produce a dataset in feature space with a win/loss response. Using this data, I computed a utility function of being in a position in a turn for each position, and to shrink it down, where I had fewer than 10 visits to a cell, I substitute the Combined heuristic. Unfortunately, even compactly encoded this utility function could not be submitted for review due to being larger than the maximum acceptable file size.

I recommend a learned heuristic function that estimates the value of a state from history of wins and losses reached in that state. I chose to encode turn and position in a three level nested dictionary, and found this to be time efficient, if not space efficient enough for submission.