## Heuristics I tried but didn't use

1. Ratio of my open moves to their open moves
2. Adapting weights over the course of the game. More aggressive or less aggressive.
3. Improved heuristic with opening book. First move is always center position.

# Explanation of the three heuristics I submitted

### `custom_score` - Improved heuristic with 2/1 weights

This was one of the weighted versions of the Improved heuristic, described above. I settled on the weights of `my_weight = 2` and `their_weight = 1` for the reasons described at the end.

### `custom_score_2` - Warnsdorf's Rule

When brainstorming heuristics, one of the things I realized is that Isolation with knights is just an adversarial version of a Knight's Tour. One of the approaches listed on Wikipedia for solving a Knight's Tour with a computer is by repeatedly applying something called Warnsdorf's Rule. Warnsdorf's Rule is that when selecting the next board position for your knight you should always select the move that results in the fewest subsequent moves from that new position. I realized that this was essentially the opposite heuristic to the Open moves heuristic. So for my implementation of this heuristic I just returned the negation of the number of open moves from any given board position.

### `custom_score_3` - Minimize opponent's open moves

This was inspired by the Open and Improved heuristics. After testing those heuristics, an obvious question seems to be "what happens if you only care about minimizing your opponent's possible moves?" To me this seemed equally valid a strategy as maximizing your own possible moves, and I was curious how it would compare to Open and Improved.

# How I chose my best heuristic

`custom_score`, my best heuristic, was inspired by my observation that the Open and Improved algorithms seemed to perform about equally well when I ran them against each other using `tournament.py`. From this observation, I hypothesized that maximizing the number of possible own moves was more important than minimizing the number of possible moves for your opponent, although both are important. Based on this, I created a heuristic that treated the number of own moves as twice as important as the number of opponent moves, then maximized the spread between these two values just like the Improved heuristic. To test this, I modified `tournament.py` to run 100 games instead of 10 games, and I only compared my heuristics with AB_Open, AB_Center, and AB_Improved since they were the only agents that posed a significant challenge for any of the alpha-beta based agents. The results of this test are below:

```
Match #   Opponent    AB_Improved    AB_Open     AB_Custom   AB_Custom_2  AB_Custom_3
                       Won | Lost   Won | Lost   Won | Lost   Won | Lost   Won | Lost
   1       AB_Open     48  |  52    54  |  46    54  |  46    43  |  57    49  |  51
   2      AB_Center    59  |  41    53  |  47    59  |  41    46  |  54    56  |  44
   3     AB_Improved   42  |  58    49  |  51    53  |  47    38  |  62    44  |  56
----------------------------------------------------------------------------------
          Win Rate:      49.7%        52.0%        55.3%        42.3%        49.7%
```