Isolation Custom Heuristic Analysis

Udacity Artificial Intelligence Nanodegree

Will Trott

Results

Match #	Opponent	AB_Improved	AB_Custom	AB_Custom_2	AB_Custom_3
		Won|Lost	Won|Lost	Won|Lost	Won|Lost
1	Random	8|2	5|5	9|1	10|1
2	MM_Open	8|2	9|1	4|6	7|3
3	MM_Center	8|2	7|3	6|4	8|2
4	MM_Improved	3|7	6|4	5|5	8|2
5	AB_Open	4|6	5|5	7|3	6|4
6	AB_Center	5|5	5|5	5|5	7|3
7	AB_Improved	3|7	4|6	4|6	5|5
	Win Rate:	55.7%	58.6%	57.1%	72.9%

Heuristic 1

Score = player_moves / opponent_moves

The idea behind this heuristic was that performing a normalized version of the improved heuristic would cause the minimization nodes to choose different moves because large differences would be reduced. While the win rate of this heuristic was higher than that of the improved version, it wasn’t by enough to make much of a difference.

Heuristic 2

Score = - opponent_moves / player_moves

The second heuristic was designed as a variation of the first, preserving the fact that the player will want to avoid moves which will cause the opponent to have a lot of moves or the player to have relatively few moves. It came about as an attempt to improve the performance of heuristic 1 by preserving the spirit by changing the math. Unfortunately, it behaves slightly worse than heuristic 1, although again the difference is so slight that the two heuristics might as well be considered the same.

Heuristic 3

Score = 1.5 * player_moves – opponent_moves

This heuristic came about after seeing that changing the spirit of the improved heuristic didn't improve the situation much, so I should stick with the same basic structure, but introduce a weight on the player_moves term. Doing this will cause the strategy to favor positions with high player moves, and if the constant were to be increased further, it would eventually disregard the number of moves that the opponent has entirely. I chose 1.5 as the initial weight, since it improves the improved heuristic and guarantees that it doesn't push it too far. Using this weight caused the bot to win almost 20% more of its games. It's possible that introducing a weight to the opponent_moves term would improve the performance further. To determine the optimal weights, Monte Carlo simulation should be performed to find the weight pair that maximizes the win percentage.

Recommendation

Of the three custom heuristics, **Heuristic 3** is the obvious choice for which function to use. The main reason is that its use leads to a 17.2% improvement in the win rate of the isolation agent. This improvement comes at basically no extra cost, since the only change to the baseline algorithm is an additional multiplication by a constant. So this heuristic should maintain similar complexity and be able to traverse the game tree to a similar depth while giving better results in comparison to the baseline method.