Heuristic Function Analysis | Adversarial Search

Udacity Artificial Intelligence Nanodegree
Brittany Martin | 12/26/2017

Abstract

We developed an adversarial search agent to play the game "Isolation". Agents have a fixed time limit each turn to search for the best move and respond. We were tasked with creating three of our own heuristics. The final step was to test the relative performance of our agents in a round-robin tournament against several other predefined agents.

Heuristics

- *Custom Score 1* - weights active player's available moves by 1.5, then multiples that by the opponents available moves at a weight of 2.5, the distance equal to the square of the distance from the center of the board to the position of the active player.

- *Custom Score 2* - finds the difference between the the active player's moves and the weighted opponent's moves to locate next best move. The opponent is weighted by 2.

- *Custom Score 3* - finds the difference between the the active player's moves and the weighed opponent's moves to locate next best move. The opponent is weighted by .5.

Hypothesis

I would expect out of the custom heuristics, Custom Score 3 would perform the worst since it minimizes the opponent's move count. I would also assume that Custom Score 2 would perform the best since it considers both the active and opponent's available moves.

I hypothesize all of my custom functions to outperform Random and only for Custom Score 3 to be beaten by AB_Improved. Custom Score 2 and Custom Score 3 should have an upperhand on all MM challengers since they utilize alpha-beta pruning.

Results

Number of Matches Against Each Opponent: 20

Time to Complete Entire Tournament: 59 minutes and 37 seconds

Figure 1. Benchmarked Tournament Computational Time %

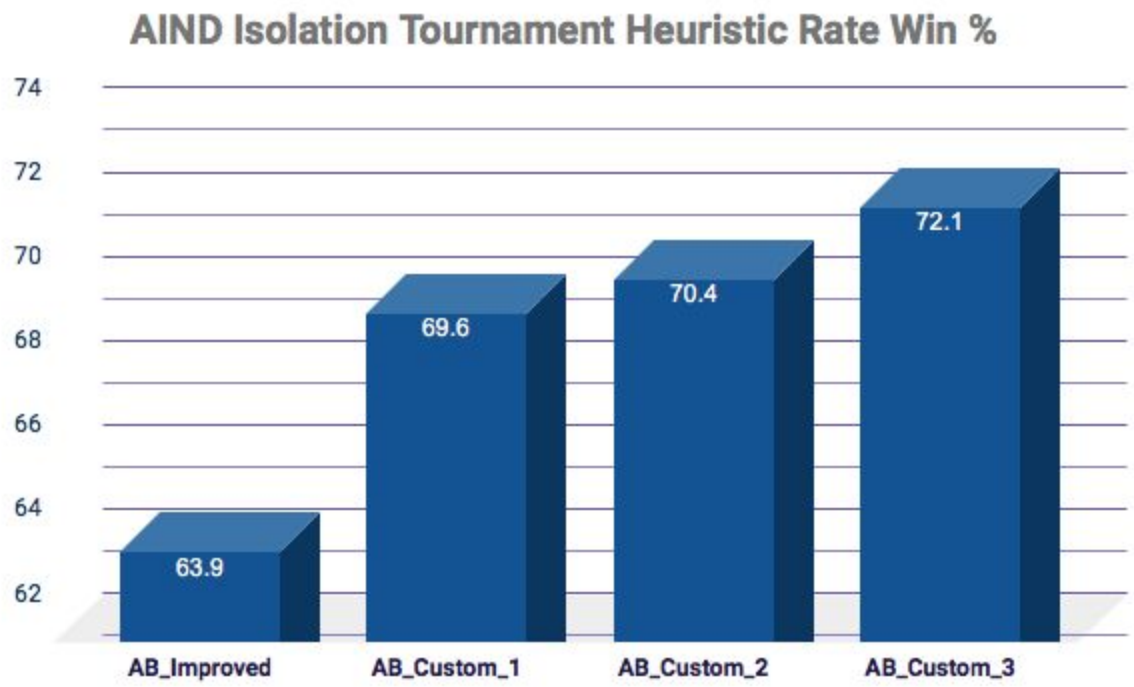
Machine Specs:

Figure 2. Tournament Results

```
                      ************************
                          Playing Matches
                      ************************

Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                      Won | Lost    Won | Lost   Won | Lost   Won | Lost
   1       Random      34  |  6      38  |  2     39  |  1     39  |  1
   2       MM_Open     27  |  13     28  |  12    31  |  9     32  |  8
   3      MM_Center    32  |  8      38  |  2     35  |  5     36  |  4
   4     MM_Improved   21  |  19     34  |  6     31  |  9     31  |  9
   5       AB_Open     22  |  18     23  |  17    20  |  20    17  |  23
   6      AB_Center    24  |  16     19  |  21    21  |  19    24  |  16
   7     AB_Improved   19  |  21     15  |  25    20  |  20    23  |  17
--------------------------------------------------------------------------
          Win Rate:      63.9%        69.6%        70.4%        72.1%
```

Figure 3. Win Rate % for Tournament



As you can see from Figure 1, 2 and 3, the hypotheses matched the following:

1. All three custom score functions consistently beat the Random player.
2. Custom Score 1 won the majority of its matches (69.6%), also beating AB_Improved at 63.9%.
3. Custom Score 3 was the most performant at 72.1%, beating AB_Improved at 63.9%.

4. Custom Score 2 and 3 easily beat the MM challengers as predicted.
5. Computational resources percentage consumed were almost the same across all three heuristics.

Analysis

This tournament proved that Alpha-Beta pruning is a reliable optimization technique for minimax algorithm since it allowed the algorithm to search faster and go into deeper levels in the gametree. Since the computational resources were statistically equivalent to one another, the preferred algorithm is Custom Score 3.