

Tournament Analysis

By

Kenneth R. Farr III

www.kennethfarr.com

Introduction:

This report is to fulfill the tournament analysis report portion of Udacity's AI Nano Degree program, Project: Build a Game-Playing Agent. During this project a custom Isolation Game Playing Agent was written with multiple 'scoring' algorithms. The goal is to develop a scoring algorithm that is superior to the built-in, provided algorithms.

Test 1

The first test was constructed with a simple hypothesis; that a variable weighting of (#OUR_MOVES) - (#THEIR_MOVES) would produce a viable scoring algorithm. This tournament was ran with the following weighting.

```
custom_score: #OUR_MOVES * 2 - #THEIR_MOVES  
custom_score_2: #OUR_MOVES * 4 - #THEIR_MOVES  
custom_score_3: #OUR_MOVES * 8 - #THEIR_MOVES
```

This resulted in **Fig 1**. At first this looks as if AB_Custom_3 resulted in a much better win/lose rate. To be certain, a test for statistical significance is required though. I conducted a 2-sample T-Test between each pair of Opponents. While ANOVA could have been performed, the six T-Tests were not cumbersome and there was high hopes that at least one pair would be statistically significant.

***** Playing Matches *****									
Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	10	0	10	0	9	1	10	0
2	MM_Open	7	3	8	2	9	1	9	1
3	MM_Center	8	2	10	0	8	2	9	1
4	MM_Improved	8	2	7	3	8	2	10	0
5	AB_Open	5	5	6	4	3	7	6	4
6	AB_Center	5	5	6	4	8	2	5	5
7	AB_Improved	5	5	6	4	5	5	5	5
Win Rate:		68.6%		75.7%		71.4%		77.1%	

Fig 1) Initial Tournament

The results of the T-Tests can be found in this [Tournament Stats Google Sheet](#).

T-Tests Results

The results of the T-Tests can be viewed in **Fig 2**. With an Alpha of 0.05, and performing a 2-tailed test our T-Critical value is 2.179, well outside of the range of any of the T-Tests. This confirms our default Null Hypothesis and there is no statistical significance between any of these tests.

A perfect 'false' game was constructed under the heading AB_Perfect to see what kind of win/lose rate would be significant. A scoring algorithm would need a win rate of 85.7% to be significantly better than the worst performing AB_Improved.

T Stats	AB_Improved	AB_Custom	AB_Custom_2	AB_Custom_3	AB_Perfect
AB_Improved		-0.7094756548	-0.2526455763	-0.7539370349	-2.241262103
AB_Custom			0.3905667329	-0.1294595556	
AB_Custom_2				-0.4692371322	
AB_Custom_3					

Fig 2) T-Tests

Test 2: Searchable Space

Not content with a losing algorithm, I constructed my own tournament that would search for a Theta0, Theta1, and Theta2 such that

$$\text{Score} = \text{Theta0} + \text{Theta1} * \# \text{OUR_MOVES} + \text{Theta2} * \# \text{THEIR_MOVES}$$

This tournament executed from [-11, -11, -11] to [11, 11, 11] skipping every other value, so that -11, -9, -7... were used. This helped reduce the search space while still providing a gage on the direction of the algorithm. This data is available in the 'Solution Search' tab of the Tournament Stats SpreadSheet linked above. Sorted for minimal loses, **Fig 3** shows the best performing Thetas. These values were then put into the custom_score, custom_score_2, and custom_score_3 functions and the tournament.py application ran again, as seen in **Fig 4**. Resulting in even worse performance.

Theta 0	Theta 1	Theta 2	Loses
-11	5	-1	0
-11	11	1	0
-7	11	-7	0
-11	3	1	1

Fig 3) Searched Thetas

***** Playing Matches *****									
Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	9	1	9	1	9	1	9	1
2	MM_Open	7	3	8	2	7	3	6	4
3	MM_Center	9	1	9	1	8	2	7	3
4	MM_Improved	6	4	6	4	8	2	7	3
5	AB_Open	5	5	4	6	7	3	5	5
6	AB_Center	6	4	6	4	6	4	3	7
7	AB_Improved	5	5	5	5	4	6	5	5
Win Rate:		67.1%		67.1%		70.0%		60.0%	

Fig 4) Theta Tournament

Test 3: Minimax V AlphaBeta

One addition test was to compare the Minimax matches against the AlphaBeta matches. A pivot table was created to group all MM matches in one dataset and all AB matches in another.

Fig 5 shows the results of the T-Test, predicting that the difference between Minimax and AlphaBeta with greater than 99.9% probability. This is consistent with our intuition that an AlphaBeta prune will allow for a deeper search in the same amount of time, more accurately predicting each branches success.

	Wins against MM	Wins against AB
	7	5
	8	5
	8	5
	8	6
	10	6
	7	6
	9	3
	8	8
	8	5
	9	6
	9	5
	10	5
	8	9
	8	9
	9	9
Mean	8.4	6.133333333
Std Dev	0.9102589898	1.807392228
deg freedom	14	14
t-stat	4.190946314	
prob	0.001	

Fig 5) Minimax v AlphaBeta

Conclusion

It appears as if the algorithms constructed and tested in this experiment and used to score each tournament has little bearing on the outcome of the game. It is possible that there are scoring algorithms that are statistically significant from the provided ones, but that must be left for further research. The search implementation however has a much greater impact on the winning outcome of the game, with AlphaBeta pruning surpassing naive MiniMax with 99.9% confidence.