Quentin Thomas

AIND-Isolation Heuristic

11 September 2017

Heuristic Analysis

In order to build the most optimal game playing AI agent I experimented with several custom score functions to see which one gave the best performance. This report will be a brief summary of each custom score function as well as the reasoning behind why the score functions were chosen during testing of the AI game playing Agent. A visual will also be available at the end of the report to show the final results as well.

CUSTOM_SCORE_1 ALGORITHM

$$\forall a \in A \sum_{i=m}^{n} a + i$$

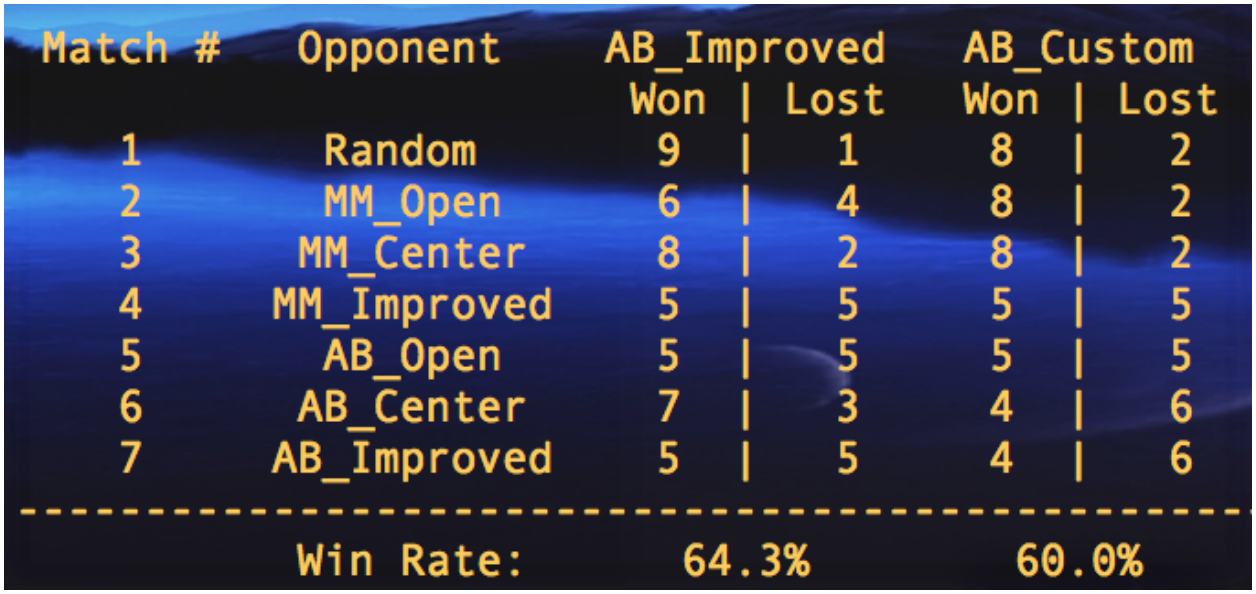
$$\forall b \in B \sum_{i=m}^{n} b + i$$

$$\gamma \times A - B$$

For custom score one we represent our two players are represented by sets A, and B. A will represent the set of moves of the first player to make a move in the game and B will represent the second player that makes a move in the game. Before we can execute the heuristic we need to get the sum of all the move values in both sets A and B. Once we have those we implemented our chosen heuristic by choosing a small gamma rate multiplying it by the difference between the values of Both A and B. In order to find the optimal gamma score we chose random values between 1 and 2 to see which one would perform the best. After several tries, it seems we

converged on 1.5 performing with the best win percentage. The following values were tried for this particular problem.

Gamma Values	Win Rate
1.0	58.2
1.3	59.02
1.5	60.0



Match #	Opponent	AB_Improved Won	AB_Improved Lost	AB_Custom Won	AB_Custom Lost
1	Random	9	1	8	2
2	MM_Open	6	4	8	2
3	MM_Center	8	2	8	2
4	MM_Improved	5	5	5	5
5	AB_Open	5	5	5	5
6	AB_Center	7	3	4	6
7	AB_Improved	5	5	4	6
	Win Rate:	64.3%		60.0%	

One thing to notice is our first custom score algorithm did pretty well against the Random opponent, and the MM_Open, and MM_Center strategies. It won 80% of those matches against those two strategies, however, didn't do so well against the MM_Improved agent as the win total was split.

CUSTOM_SCORE_2 ALGORITHM

The next option was to try to utilize a minimizing loss heuristic. This idea came from researching the way that Neural Networks deal with minimizing loss while trying to learn a specific function. We found several ways of minimizing loss after referencing the resource here

We finally settled on creating decay function of one of the players moves and leveraging the decay values for the player we wanted to win the game. This epsilon decay function is used in reinforcement learning, so we decided to give it a try here [source]. Our algorithm can be defined as..

$$\forall a \in A \sum_{i=m}^{n} a + i$$

$$\forall b \in B \sum_{i=m}^{n} \varepsilon \times a \hat{} b + i$$

$$-A \div B$$

Because we are maximizing our first player we will only get the sum of all the move totals that player can make. Next when we calculate the next player moves we will multiply the the sum of each of the opposite players move in the set by gamma. Finally, we will take the negative version of all the moves of A divided by the moves in set B in order to carry out our decay function. The idea here is to keep minimizing loss until we get to 0. Once we hit zero the algorithm will have no need to go lower. Our results as compared to the first customer score function with an epsilon value of 1.0 can be found below…

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2	
		Won	Lost	Won	Lost	Won	Lost
1	Random	9	1	8	2	7	3
2	MM_Open	6	4	8	2	5	5
3	MM_Center	8	2	8	2	9	1
4	MM_Improved	5	5	5	5	7	3
5	AB_Open	5	5	5	5	6	4
6	AB_Center	7	3	4	6	9	1
7	AB_Improved	5	5	4	6	6	4
	Win Rate:	64.3%		60.0%		70.0%	

We noticed here we got a better Win Rate percentage after 7 matches which improved from our first custom score algorithm. This strategy got better wins against strategies like MM_Improved, and all the AB strategies than the first custom score did.

CUSTOM_SCORE_3 ALGORITHM

Our final custom score experiment involved testing out what would happen if we multiplied our top gamma score by the opponent player's total moves in the set. This was more of the inverse of what we did in the first custom score example because before we multiplied gamma by the difference of both player scores. This was completely experimental and we were not sure whether it would create any improvements from our top scores but we tried it anyway.

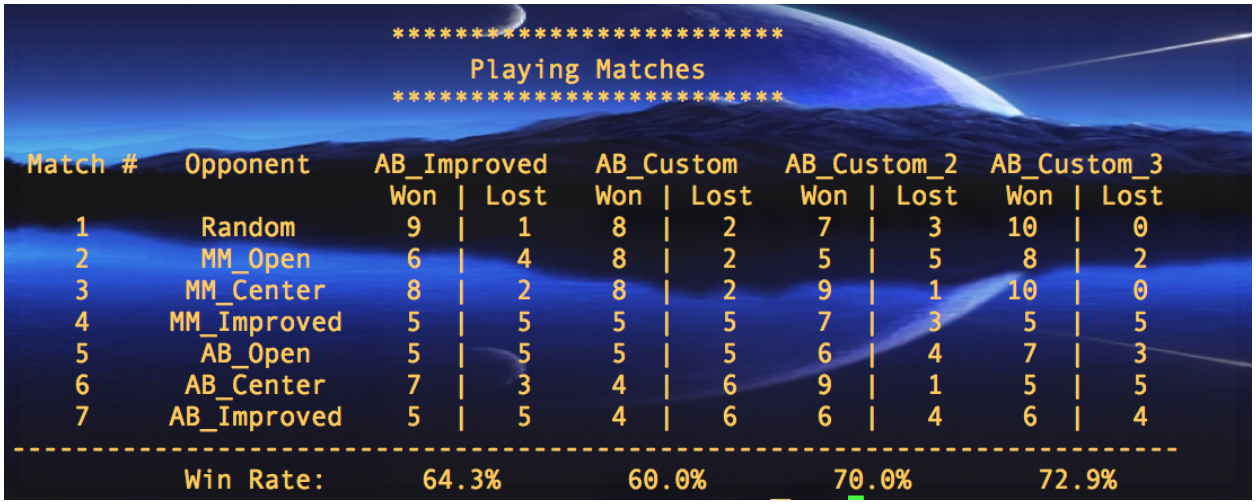
Our algorithm can for custom score 3 be expressed as….

$$\forall a \in A \sum_{i=m}^{n} a + i$$

$$\forall b \wedge a \ \in A \wedge B \sum_{i=m}^{n} \gamma \times a^{\wedge} b \ + \ i$$

$$A - \gamma B$$

This is to say that we first get the total number of moves for A and we multiply gamma by each one of the elements in either A or B depending on who's playing, we perform the gamma operation on the opposite players moves. Finally, we get the difference of A and Gamma multiplied by the set B. The results from this custom score as compared to the others was



****
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	8	2	7	3	10	0
2	MM_Open	6	4	8	2	5	5	8	2
3	MM_Center	8	2	8	2	9	1	10	0
4	MM_Improved	5	5	5	5	7	3	5	5
5	AB_Open	5	5	5	5	6	4	7	3
6	AB_Center	7	3	4	6	9	1	5	5
7	AB_Improved	5	5	4	6	6	4	6	4
	Win Rate:	64.3%		60.0%		70.0%		72.9%	

CONCLUSION:

After trying all 3 score functions we found that there were clear improvements from function to function. Each time we tried a new heuristic our win rate increased. Choosing either of those two functions 2 or 3 would result in a pretty good Win rate against the main opponents given. We can't say that either of these will create the most perfect AI, but we can conclude that using either of these 3 functions would result in successfully beating your opponent at least 7 out

of 10 times. In a tournament that would be 5 matches, we could surmise that our system would be unbeatable with these numbers in 70% of the games it plays regardless of the strategy of the opponent.

Works Cited

Loss Function wikipedia [ref]

Epsilon Decay [ref]

Weighted A* Search Carnegie Mellon [ref]