Heuristic Review

To complement our Alpha-Beta pruning algorithm implementation in the game of Isolation we have been tasked to create three custom scoring functions that score specific moves on a game board of isolation. The aim is to beat the already implemented AB_Improved heuristic which simple takes the number of possible moves for each player and subtracts to create a heuristic.

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	10	0	10	0
2	MM_Open	9	1	9	1	9	1	10	0
3	MM_Center	10	0	10	0	10	0	10	0
4	MM_Improved	8	2	9	1	9	1	10	0
5	AB_Open	5	5	4	6	6	4	5	5
6	AB_Center	6	4	5	5	4	6	5	5
7	AB_Improved	6	4	3	7	5	5	6	4
	Win Rate:	77.1%		71.4%		75.7%		80.0%	

Figure 1 - An example of the default tournament.py results

Heuristic explanations

Heuristic_1:

Takes the number of legal moves and multiplies it by the number of moves so far for each player. This is based on the logic that the further into the game the number of possible moves left is a better indicator of winning. Then takes the opponent players value away from the users value to create the score.

Heuristic_2:

Is an attempt to create a more intelligent heuristic that takes the board size into account so not to overfit the importance of number of moves left and number of moves taken. The below equations explains the logic.

$$Calibration = \frac{movesTaken + (boardSize - spacesLeft)}{boardSize}$$

$$ownMoves = numOfLegalMoves * (Calibration * NormalisingFactor)$$

$$oppMoves = numOfLegalMoves * (Calibration * NormalisingFactor)$$

$$Score = ownMoves - opp\_moves$$

This aim here is to make the factor relevant to board size and to still affect the score to a significant level. Prior to the normalising factor being added I found there was not a large enough impact being created: I.E. the values were not showing clearly enough to be always chosen.

Heuristic 3:

Is a different approach that finds the current position of each player and calculates the distance between them and rewards players for being further away. Calculate distance using Pythagoras theorem. Initially I was worried about processing time especially as square rooting is a very intensive function however it seems to not be an issue – perhaps in very large boards this could become a problem. The distance is then added to the number of legal moves so it rewards moves that have lots of options and are further away from the opposite player

Results

To gather the results I ran the default tournament script 10 times and calculated the average of these results vs all opponents. Ideally the script would be ran many more times however I am time limited.

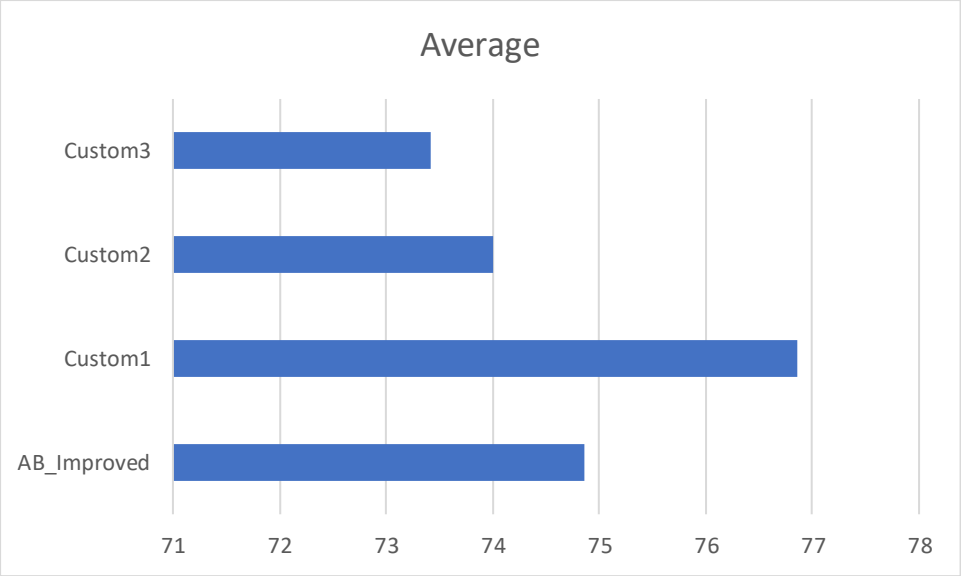

Figure 2- Average win % vs all other players

In Figure 2 you can see that Heuristic 1 (Custom1) out performs all the other algorithms slightly in this small sample size.

To test the performance of the heuristics against specifically AB_Improved I found the average wins per 10 games played (Figure 3). This again shows that Heuristic 1 is again slightly superior.

It is likely that Heuristic 1 is the best as it remains very simple computationally while also adding to AB improved logic taking into account progress throughout the game. It seems my "More intelligent" heuristic 2 is just worse possible due to the normalizing value and the need for further testing to improve this.

Heuristic 3 is very stable and does not vary much – perhaps a good way to make it better would be to instead encourage mirroring the opponents moves (as suggested in the Udacity course) would be an interesting way to take it forward.

Based on the data I would recommend using Custom Heuristic 1 as it seems to be the best performing heuristic from the data. Given greater threshold testing heuristic 2 may be superior but based on the data I cannot recommend it.

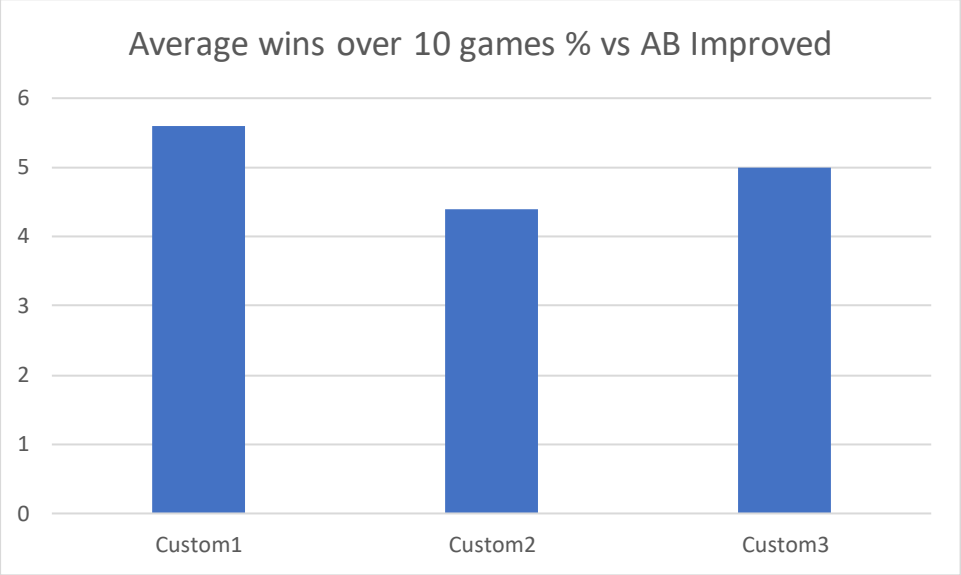

Figure 3 - Average wins over 10 games vs AB improved (10 sets)

Appendix:

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	8	2	9	1	10	0	10	0
2	MM_Open	9	1	9	1	9	1	9	1
3	MM_Center	9	1	9	1	10	0	10	0
4	MM_Improved	7	3	9	1	8	2	9	1
5	AB_Open	4	6	4	6	6	4	4	6
6	AB_Center	6	4	6	4	5	5	6	4
7	AB_Improved	5	5	5	5	4	6	6	4
	Win Rate:	68.6%		72.9%		74.3%		77.1%	

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	10	0	10	0
2	MM_Open	9	1	9	1	7	3	8	2
3	MM_Center	9	1	10	0	10	0	9	1
4	MM_Improved	10	0	7	3	7	3	10	0
5	AB_Open	8	2	5	5	5	5	4	6
6	AB_Center	6	4	3	7	6	4	6	4
7	AB_Improved	4	6	4	6	5	5	5	5
	Win Rate:	80.0%		68.6%		71.4%		74.3%	

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	10	0	9	1
2	MM_Open	8	2	8	2	9	1	5	5
3	MM_Center	8	2	10	0	8	2	10	0
4	MM_Improved	8	2	8	2	9	1	8	2
5	AB_Open	5	5	8	2	7	3	6	4
6	AB_Center	5	5	5	5	7	3	6	4
7	AB_Improved	5	5	5	5	5	5	5	5
	Win Rate:	70.0%		77.1%		78.6%		70.0%	

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	10	0	10	0
2	MM_Open	8	2	9	1	9	1	8	2
3	MM_Center	10	0	10	0	9	1	10	0
4	MM_Improved	10	0	10	0	7	3	9	1
5	AB_Open	7	3	5	5	7	3	6	4
6	AB_Center	7	3	7	3	5	5	4	6
7	AB_Improved	5	5	8	2	3	7	5	5
	Win Rate:	81.4%		84.3%		71.4%		74.3%	

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	10	0	8	2
2	MM_Open	8	2	10	0	10	0	9	1
3	MM_Center	10	0	9	1	9	1	10	0
4	MM_Improved	6	4	10	0	8	2	8	2
5	AB_Open	5	5	6	4	4	6	5	5
6	AB_Center	7	3	6	4	6	4	6	4
7	AB_Improved	6	4	6	4	5	5	4	6
	Win Rate:	74.3%		81.4%		74.3%		71.4%	