Heuristic Analysis on Game-Playing Agent

Written by Yuen Pok Henry. Ho, AI Nano Degree Term 1, AIND-Isolation.

Introduction

AIND-Isolation is the second project for Udacity AI Nano Degree, covering knowledge and technique introduced in lessons on advanced game playing and adversarial search. The main goal is to win in an Isolation game, which is a deterministic, two-player game of perfect information in which the players alternate turns moving a single piece from one cell to another on a board.

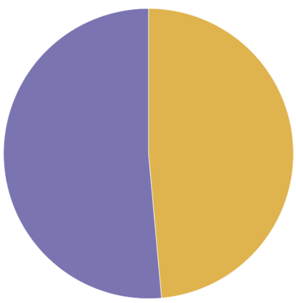
The Minimax, AlphaBeta pruning technique and Iterative Deepening were implemented as suggested in the video tutorials and instructions. Whereas the Iterative deepening function was slightly modified to allow the AlphaBeta agents to search deeper towards the end of the game, contributed to the increased of overall win percentage when running the agent tournament.

The heuristic functions were implemented where two of them used a similar statistical approach, and the third is a modified hybrid of improve_score and center_score functions. An overall average between 55% - 78% were achievable with the default time limit of 150ms and a significantly higher 60% - 80% were achieved when given 500ms.

Learning & Understanding

The win results for each agent and heuristic function although varied when accumulated, they appeared sporadic when observed in real-time, hence a method of data collection and aggregation was required. ElasticSearch and its Python module was used to collect and aggregate data for analysis and later used to construct the dataset in two heuristic functions to weigh the legal moves available at each step.

The success rate of each heuristic is depended upon several factors. In the case with timeout limits is in place the efficiency and execution time of the heuristic functions are more significant, especially with iterative deepening where the average depth of each search is inversely proportional to the execution time.



Player 1	Player 2
1039 (51.44%)	981 (48.56%)

Fig 1.0: Player 1 against Player 2 winning rates.

It is also observed that Player 1 indeed has a slight advantage over Player 2 with a winning rate of approximately 52% to 48%. (As shown in Fig 1.0 above.)

By running the tournament we can compare the winning rate on how the Custom functions play against Improved function as shown in Figure 2.0 below:

	Opponents	AlphaBeta Improved			AlphaBeta Custom 1			AlphaBeta Custom 2			AlphaBeta Custom 3		
		WIN	LOSE	RATE	WIN	LOSE	RATE	WIN	LOSE	RATE	WIN	LOSE	RATE
	Random	8	2	0.8	10	0	1.0	10	0	1.0	8	2	0.8
	MiniMax open position	7	3	0.7	8	2	0.8	9	1	0.9	10	0	1.0
	MiniMax center position	9	1	0.9	9	1	0.9	9	1	0.9	7	3	0.7
	MiniMax Improved	7	3	0.7	7	3	0.7	6	4	0.6	7	3	0.7
	AlphaBeta open position	5	5	0.5	6	4	0.6	4	6	0.4	6	4	0.6
	AlphaBeta center position	5	5	0.5	6	4	0.6	5	5	0.5	3	7	0.3
	AlphaBeta Improved	7	3	0.7	5	5	0.5	5	5	0.5	5	5	0.5
	WIN RATE	**68.6%**			**72.9%**			**68.6%**			**65.7%**		

Fig 2.0: Tournament Win Rate.

Conclusion

Although the winning rate in tournament is close to 100% against random player and above 80% against MiniMax Agent with Open and Center function. It is still difficult to win against AlphaBeta with Improved function. The statistical distribution gave a 10%-15% improvement against AlphaBeta Improved player, largely due to the better positioning provided at the beginning. This advantage became insignificant towards the end game where a better heuristic is probably required.

In average, the AlphaBeta Custom function is observed to have 70% winning rate, with the biggest winning margin against Random and player unable to analyse deeper into the game tree.

Custom function 1 is the best choice, as it has the highest win rate compared to the other two custom functions. It is also the only function that has a higher win rate against the AlpahBeta Improved function which is considered a more advanced player. The balanced use of pre-calculated statistical data lookup table together with move count and improved score can be also be efficient and fast, allowing the agent to search deeper into the game tree before it timeout.