COMP3608 AI Assignment 1 Report

Introduction

This paper reports the implementation of an Artificially Intelligent (ai) player in the game of Connect-4. Described are three different types of strategies the ai can use in this implementation. These strategies are empirically evaluated via the matching of human and ai players to produce results, which are discussed and evaluated.

Implementation

Heuristic 1

For strategies A and B, the same evaluation function was used. This function receives a configuration of the board, representing the current state, as an input and returns a utility value for that state. To calculate the utility, the function counts the number of potential “winning lines” for a player. The total utility value is the difference between the number of winning lines for the current player and the number of winning lines for the opponent.

Strategy A – Minimax algorithm

The Minimax algorithm recursively calculates the Minimax decision of nodes throughout the game tree. Since the entire game tree is too large to traverse through, a cut off is defined, which specifies the deepest level to be reached before evaluating the Minimax values. The algorithm searches in a depth-first manner, going as deep as it can before it starts calculating the Minimax values. When a terminal state is reached, it will call the heuristic evaluation function, which then returns the heuristic value for the given state. As the recursion unwinds, the Minimax values are passed up through the tree, eventually reaching the root. The action returned by the Minimax algorithm is the optimal move for the current player, assuming that its opponent plays optimally. In the case of connect 4, the column in which to place the next piece is returned.

Strategy B – Alpha-Beta pruning algorithm

The Alpha-Beta pruning algorithm is very similar to the Minimax algorithm. It follows the same principle of traversing the game tree and calculating the Minimax decisions from the heuristic evaluations at terminal states. However, Alpha-Beta is able to reduce the number of nodes to be searched through by keeping track of two parameters:

Alpha - the highest Minimax value so far along the path for MAX

Beta - the lowest Minimax value so far along the path MIN

By comparing the Minimax value of the current node with alpha or beta (depending on whether it is MAX’s turn or MIN’s turn), the subtree below the current node may be ignored, or “pruned”, if it turns out that the current node’s value is worse than alpha or beta, for MAX

or MIN respectively. Consequently, the search space could be reduced significantly, giving improvements in the running time and allowing for deeper searches into the tree.

Strategy C - Alternate evaluation function

The difference between strategy C and the others is that a different evaluation function is used. The new evaluation function was built up on top of the original heuristic in an effort to make it stronger. The theory used in creating a stronger heuristic is to use several discrete calculations of heuristic values and then combining them together to form the complete value. Each of these calculations utilises a certain “rule”. In the original function, only one rule was used, i.e. the “number of winning lines” for a given state. For the evaluation function in this strategy, another two rules were used to determine the the total heuristic value.

The first rule added was “number of pieces in a row” in a given state, which counts the number of player pieces that would end up together in a formation of four adjacent spaces (for all rows, columns and diagonals). In addition, these values were weighted according to whether the pieces were adjacent to each other or not. For example, for a line of three player pieces, a larger weight was given to three adjacent pieces as opposed to two adjacent pieces and one standalone piece. This rule is very simple in its reasoning because, the more pieces a player has together, the higher the chance of winning.

The second rule added was “blocking opponent pieces”, which detects vulnerable positions for the player. This rule looks at the position of opponent pieces and puts a value on whether it would be beneficial for the player to “block” the opponent. For example, if it is found that there are three opponent pieces in a row, then a high value would be assigned to the case where the player’s piece is placed amongst the three opposing pieces, in order to prevent the opponent from winning. The reasoning behind this rule is that it is able to prevent vulnerable situations for the player before they actually arise.

One problem with using multiple “rules” to construct a heuristic is that it is hard to know whether any of them conflict with each other. Consequently, when assigning values to the different states, it may be the case that the heuristic value could in fact be higher than the actual calculated value. This is a problem that was not addressed when developing the evaluation function and hence, is a weak point of the heuristic.

Evolution of Strategy C

To reach our final Strategy C, we went through various strands of heuristics that we improved on over time.

The evolution of the efficiency of these different heuristics can be observed in the appendix. Using various metrics and visual cognition, version 3 of heuristic 2 (H2_v3) was the most optimal, and thus was chosen as the final heuristic to be used with strategy c.

Empirical Evaluation

Our results are presented in table form in the appendix.
They consist of results from games where Humans versed AIs, and AIs versed AIs.
In Human versus AI games, we tested the Artificially Intelligent Players which composed of Mini-max or Alpha-Beta, at depths ranging from 3 to 6.

Human players were queried after playing two different AIs, which game they believed to be harder. The vast majority guessed correctly (when verified with results from AIs' versus AIs').

There was insufficient time to fully test these opponent ai's over all their different dimensions. Hence, the accuracy of the data presented could definitely have been improved given more user data.

The below refers to tests regarding only Heuristic 1.

> A total of 24 players were tested when comparing Min-Max, Alpha-Beta, and tree-cut-off depths.
>
> The response time of the computer had no noticeable effect on the overall outcomes of human-played games. At a depth of 4, when Humans played Alpha-Beta, they won 8 out of 15 games (53%), and drew one. When Humans played Min-Max at the same depth, they won 4 out of 8 games (50%).
>
> No human players won any games where the ai had a search depth greater than 4.
> At depths of 3, Humans won 5 out of 15 games (33%).
> At depths of 4, Humans won 12 out of 24 games (50%).
> At depths of 5 and 6, no Humans won any games (0%).
>
> It is interesting that more humans won at depths of 4 than 3, in that a depth of 4 is theoretically meant to be 'harder' for a human-player, than a depth of 3. This 'theory' is also backed by the fact that when an AI with a depth of 4 versus an AI with a depth of 3, the AI with a depth of 4 will win no matter which AI began. Hence, this may be a statistical anomaly.
>
> The fact that no players won at games with depths of 5 and 6, suggests that most human players can not win in this situation, where the search-depth is greater than 4.

In testing Strategy C:

> A total of 12 players were tested, at a search-depth of 4.
> Humans won 4 out of 12 games (33 %).
> This is a 17% decrease from when humans played at the same depth against an ai with

Heuristic 1 (winning 50%).

This shows, that at least at a depth of 4, this heuristic is an improvement (therefore harder), over the original one.

Of interesting note, there was one participant who played three games, and lost, drew, and won to this ai (in that order). It was stated that he observed and learnt the patterns of the ai and adapted to it. Such a factor, the abilitiy to learn, was considered, but not implemented with this ai due to time constraints, though it would definitely be an avenue for future exploration.

Conclusions

To conclude, the statistics presented will definitely benefit from more user data. Minimax takes a longer amount of time than Alpha-Beta, but at low depths (less than 5), it does not affect game outcomes. Using both Minimax and Alpha-Beta (at the same depths), produces the same results. Most humans cannot beat an ai with a basic heuristic (Heuristic 1) at depths greater than four.

The final heuristic is definitely an improvement over the original, even though it does not always best the original in every scenario.

Reflection

In the process of doing this assignment, I have gained knowledge into the principles of game playing. I felt that the most important concept that I have learned is the use of heuristics to play a game. The strength of the AI is highly dependent and controlled, in large part, by the heuristic evaluation function. The difficulty of finding the most effective evaluation function made the implementation challenging. By having the most effective evaluation function, the AI will be able to win every time.

I have learnt much in the undertaking of this assignment. Heuristics being definitely key to determining a good AI, and to try and always start of with the simplest Heuristics, as they are often the better than convoluted ones (and easier to understand). It was also quite fun to get other people to test out the program, and evolve heuristics accordingly. Also of interesting note, was the method of statistical analysis derived to quickly judge a heuristic as it was developed (the use of visual cognition to quickly judge the performance of a heuristic versus another one across different depths, as can be observed in the appendix).

Appendix

Testing methods

Strategy A

- To test correctness of the algorithm, the tree structure was printed out as the algorithm runs to ensure that indeed, the correct Minimax values are assigned. The min and max values for the nodes were printed out in an indented manner so that it could be easily seen which level the algorithm was up to.
- To test the correctness of the evaluation function, board states with corresponding utility values were printed out at shallow depths. These were then checked manually by hand.
- Games with AI vs. AI were run using different depths to test whether players using deeper searches are stronger.

Strategy B

- As with Minimax, the game tree structure with Minimax values was printed out. Since Alpha-Beta pruning was being used in this strategy, it was expected that there would be fewer nodes displayed, which was indeed the case. Also, the results should be exactly the same as the Minimax algorithm. If there were any differences, this would've indicated that the implementation of one of them could be incorrect.
- Setting up games with strategy A vs. strategy B allowed us to check whether the pruning method was indeed more efficient. By setting the same depth for both algorithms, it could be seen that strategy B took a shorter time in taking its turn than strategy A.
- The evaluation function was also tested in the same manner as with strategy A to make sure that the results were consistent.

Strategy C

- Since strategy C uses a different evaluation function, we were able to test which one was stronger by setting up games with AI vs. AI with each player using a different evaluation function.

Table of collected user data.

User							Key
	MinMax - D3 H1	AlphaBeta - D4 H1	MinMax - D4 H1	AlphaBeta - D5 H1	AlphaBeta - D6 H1	D4 - H2	Player 1 = Human
u1	1	1					Player 2 = AI
u2	1	1					
u3	1	1					1 = Player 1 Won
u4	1	1					2 = Player 2 Won
u5	1	1					0 = Draw
u6	-1	1					
u7	-1	1					
u8	-1	1					
u9	-1	0					
u10	-1	-1					
u11	-1	-1					
u12	-1	-1					
u13	-1	-1					
u14	-1	-1					
u15	-1	-1					
u16	-1	-1					
H			1	-1			
H2			-1	-1			
E			-1	-1			
J			-1	-1			
K			1		-1		
R			1		-1		
K2			1		-1		
R2			-1		-1		
u25						1	
u26						1	
u27						1	
u28						1	
u29						0	
u30						-1	
u31						-1	
u32						-1	
u33						-1	
u34						-1	
u35						-1	
u36						-1	

Example table of quickly judging the efficiency of a heuristic.