A-Star

Joseph E. Sutton

July 5, 2013

Question 1

In a finite grid world A* is guaranteed to find the target or determine it's unreachable in finite time. This is because for each expanded cell it adds at most four states that have not already been expanded or blocked to the OPEN list. In a worst case it will expand all cells with the exception of the cell that are blocked. In a grid world that is finite this number is finite and will execute in finite time.

Question 2

Based on the results from running searchers on the inputfiles and the example in figure 8, it suggest that choosing the larger g-value is a better choice. You can see that in all cases the larger g-value executes in less time and expands fewer nodes. The reason the smaller g-value performs worse is in the fact that it chooses the cell closer to the current cell. The larger g-value, does the opposite, choosing the state farther from the current cell and thus should be closer to the target.

	Forward A* (larger g-value)		Forward A* (smaller g-value)	
	Runtime/search	Cells Exp./search	Runtime/search	Cells Exp./search
Inputfile0	0.000066	11.5	0.000075	13.0
Inputfile1	0.000063	9.3	0.000090	13.0
Inputfile2	0.000060	7.5	0.000125	17.5
Inputfile3	0.000033	6.7	0.000038	7.6
Inputfile4	0.000069	8.6	0.000105	15.2
Inputfile5	0.000131	17.6	0.000397	43.6
Figure 8	0.000157	8.0	0.000222	23.0
Average	0.000070	10.2	0.000138	18.3

Question 3

Based on the results from running searchers on the inputfiles, it suggest that choosing the Repeated Forward A* is a slightly better choice. You can see that in some cases it expands more state in other it expands less. In all cases the runtime was faster in Repeated Forward A* this is because to implement Repeated Backward A* I swapped the current state and target state, then needed to update h-values according. So this processing takes more time. To conclude there is no real advantage of using Repeated Backward A* over Repeated Forward A*.

	Forward A*		Backward A*	
	Runtime/search	Cells Exp./search	Runtime/search	Cells Exp./search
Inputfile0	0.000066	11.5	0.000104	8.5
Inputfile1	0.000063	9.3	0.000083	13.5
Inputfile2	0.000060	7.5	0.000094	10.4
Inputfile3	0.000033	6.7	0.000048	6.0
Inputfile4	0.000069	8.6	0.000137	15.3
Inputfile5	0.000131	17.6	0.000176	19.7
Average	0.000070	10.2	0.000107	12.2

Question 4

A heuristic $h(n)$ is consisten if, for every node n and every successor n' of n generated by an action a, the estimated cost of reaching the goal from n is no greater than the step cost of getting to n' plus the estimated cost of reaching the goal from n': $h(n) \leq c(n, a, n') + h(n')$[1]. Therefor, if unchanged costs are consisten then increasing costs would also be consistent because it still remains greater then the $h(n)$.

Question 5

Based on the results from running searchers on the inputfiles, it is not clear which method is better, Repeated Forward A* or Adaptive A*. Repeated Forward A* has a faster run time however Adaptive A* is slightly lower number of expanded cells. The runtime can be explained in the fact that the Adaptive A* needs to maintain a CLOSED list in order to update the h-values after a search. This can not be avoided, so if I had to make a choice I would go with Repeated Forward A* because it is slightly faster and lesses memory.

	Forward A*		Adaptive A*	
	Runtime/search	Cells Exp./search	Runtime/search	Cells Exp./search
Inputfile0	0.000066	11.5	0.000070	10.0
Inputfile1	0.000063	9.3	0.000068	8.7
Inputfile2	0.000060	7.5	0.000063	7.5
Inputfile3	0.000033	6.7	0.000103	5.5
Inputfile4	0.000069	8.6	0.000083	8.8
Inputfile5	0.000131	17.6	0.000149	14.6
Average	0.000070	10.2	0.000089	9.2

[1]Artificial Intelligence - A Modern Approach, pg. 99