Project 1 Answers

Edward Wang, Atishay Mitra, Kyle Love

2020-06-29

1 Part 1

a. Since the agent can only observe its 4 adjacent cells, it will choose to move to the east in the first step since the cell to the east is unblocked and is currently the first step on the shortest path to the target state. After the agent would move to position E3, it would observe that the north and east cells are blocked and adjust its path.

b. In a finite gridworld, there will be a limited number of cells which the agent must check and therefore the time required to find the path or conclude that a path to the target does not exist will be finite. The time it takes to search is linked to the finite number of cells. In contrast, a search algorithm applied on an infinite, constantly expanding gridworld could take an infinite amount of time to conclude whether an agent can reach its target or not.

The number of moves the agent can make is bounded from above by the number of unblocked cells squared because if an adjacent cell is blocked then the agent will backtrack to the same cell at most one time meaning that they will have visited the cell at most two times within the algorithm before either realizing that there is another path or that there is no path at all to the target.

2 Part 2

3 Part 3

To test and compare the performance of Repeated Forward A* against Repeated Backward A*, we conducted ten runs of both algorithms. Firstly, we applied Forward A*, randomly choosing nodes for both the goal and start, then swapped those same nodes and ran it again to simulate Backward A*. We then took the resulting values for the number of expanded nodes and compared them to determine their efficiency.

Percent Difference	Forward A*	Backward A*
60.6%	816	508
62.8%	482	296
-20.8%	19	24
-44.2%	829	1487
12.8%	2024	1793
-36.5%	453	713
246%	1974	517
56.4%	341	218
20.6%	262	330
-42.2%	1783	3086

On average, we determined that Forward A* had a runtime 31.55% greater that that of Backward A* for the same grids and nodes. Overall, the data seems to suggest that Forward A* is significantly less efficient that Backward A*. However, it is also important to note that, in theory, the two algorithms should have similar performance, given that the obstacles/blocked nodes are spaced largely randomly. The results we have found are may largely be due to our small sample size. As such, it is hard to conclude off of our data that Forward A* is in fact slower than Backward A*

4 Part 4

4.1 Manhattan distances

The manhattan distance is calculated with the following formula

$$h(s) = |X_s - X_{goal}| + |Y_s - Y_{goal}| \tag{1}$$

To be considered a consistent heuristic, it must prove true for the following formula

$$h(s) <= h(s') + c(s, a, s') \tag{2}$$

We substitute all instances of h(s) and h(s') with the corresponding manhattan distance definition.

$$|X_s - X_{goal}| + |Y_s - Y_{goal}| <= |X'_s - X_{goal}| + |Y'_s - Y_{goal}| + c(s, a, s') \tag{3}$$

Then simplify

$$|X_s - X_{goal}| + |Y_s - Y_{goal}| - |X'_s - X_{goal}| + |Y'_s - Y_{goal}| <= c(s, a, s') \tag{4}$$

Say s and s' are neighboring cells, s' being the immediate successor of the initial state. Furthermore, let us say that s' is vertically adjacent to s. Hence, the following proves true.

$$|X_s - X_{goal}| - |X'_s - X_{goal}| = 0 \tag{5}$$

$$|Y_s - Y_{goal}| - |Y'_s - Y_{goal}| = 1 \tag{6}$$

$$|Y_s - Y'_s| = 1 \tag{7}$$

$$|Y_s - Y_{goal}| - |Y'_s - Y_{goal}| = c(s, a, s') \tag{8}$$

$$h(s) - h(s') = |Y_s - Y_{goal}| - |Y'_s - Y_{goal}| \tag{9}$$

Then, given the following inequality

$$|A| - |B| <= |A - B| \tag{10}$$

we say

$$h(s) - h(s') <= |(Y_s - Y_{goal}) - (Y'_s - Y_{goal})| = 1 = c(s, a, s') \tag{11}$$

Therefore, manhattan distances can be said to be consistent heuristics.

4.2 Adaptive A*

To prove conclusively that Adaptive A* leaves heuristics consistent over iterations, we must prove it's validity for the following three cases

4.2.1 In the case both s and s' are expanded

The new heuristics for both of the nodes are as follows

$$h_{new}(s) = g(s_{goal}) - g(s) \tag{12}$$

$$h_{new}(s') = g(s_{goal}) - g(s') \tag{13}$$

We then substitute them into the consistency inequality like so

$$h_{new}(s) <= h_{new}(s') + c(s, a, s') \tag{14}$$

$$g(s_{goal}) - g(s) <= g(s_{goal}) - g(s') + c(s, a, s') \tag{15}$$

$$g(s') <= g(s) + c(s, a, s') \tag{16}$$

Using the above equation, we can conclusively state that the heuristic remains consistent because the algorithm calculates the g value of the successor state based off of the g value of its predecessor with the cost to transition added, so the inequality will never be false.

4.2.2 In the case s, but not s' is expanded

The consistency inequality we want to prove in this case is as follows

$$h_{new}(s) <= h(s') + c(s, a, s') \tag{17}$$

Firstly, considering that case 1 holds true, we can similarly say that case 2 holds true, because the inequality would still be valid if the node were to be expanded.

Nothing changed with regards to the situation, only the amount of information we have has changed. Furthermore,

$$f(s) <= f(s') \tag{18}$$

because the algorithm prioritizes the smallest f-value to expand nodes. Therefore, if we substitute into the equation above as follows, splitting the f-value into h+g values,

$$h_{new}(s) + g(s) <= h(s') + c(s, a, s') + g(s) \tag{19}$$

we find that

$$h_{new}(s) <= h(s') + c(s, a, s') \tag{20}$$

proving the inequality true

4.2.3 In the case neither are expanded

Unlike the other cases, the adaptive A* algorithm modifies neither s or s', so considering the fact that the consistency inequality should hold true in normal A*, we can conclude that this case holds true as well.

5 Part 5

6 Part 6