Homework #1

Wenzheng Zhang, Qi Wu

March 1, 2020

Part 0

We implemented a class $Maze$ to generate and visualize the maze. This class is initialized with two integer number which is the number of rows and columns of the maze. In class $Maze$, we implemented a function $generate\_maze()$. This function use depth first search approach to mark the blocked and unblocked cells and set $start = (1, 1)$ and $goal = (row - 1, col - 1)$. To visualize the maze, we implemented a function $maze.visualize()$ using the third-party library $matplotlib$ to plot the maze. We use a loop to generate the maze 50 times. All the mazes are initialed with size 101×101.

Part 1

a

The estimated cost of cheapest solution through the east neighbor of the agent is

$$f(east) = g(east) + h(east) = 1 + h(east) = 1 + 2 = 3$$

But the estimated cost of cheapest solution through the north neighbor of the agent is

$$f(north) = g(north) + h(north) = 1 + h(north) = 1 + 4 = 5$$

According to the rule of A* search, because $f(east) < f(north)$, we choose to move the agent to the east at the first step given that the agent doesn't know initially which cells are blocked.

b

A* algorithm always expand the cell with smallest f-value. Once a cell is expanded, it will never be put back the open list and be expanded again. This can be proved by contradiction:
Suppose cell a was expanded. Cell b is expanding at current iteration, and b is adding a to open list. Because b is adding a to open list, we have

$$g(a) > g(b) + c(b, a)$$

Also, because a was expanded before b,

$$f(a) = g(a) + h(a) < f(b) = g(b) + h(b)$$

In addition, since heuristic is consistent,

$$g(a) + h(a) < g(b) + h(b) < h(a) + g(b) + c(a, b)$$

$$g(a) < g(b) + c(a, b)$$

This is a contradiction. Therefore, if a was expanded, it will never be added to open list. And we can say a is put into a closed list.
So, because the grid world is finite, we can put at most finite number of cells into closed list. This means in searching, we can only find the target or report that it is impossible to reach the target.
For moving part, suppose there are n unblocked cells in the grid world. For each iteration, the agent can move at most n cell and find at least one blocked cell neighboring one unblocked cell. Since there are at most 4n blocked cells neighboring unblocked cell, the algorithm can only run at most 4n times. Therefore, the number of moves of the agent until it reaches the target or discovers that this is impossible is $O(n \times 4n) = O(n^2)$.

Part 2

We set 2 different modes of priority queue for the open list in A* search. For both modes, we first take f-values as priorities in favor of smaller f-value. When we encounter two cells with the same f-values, if we set mode=0, we use $g(s)$ as priorities to break ties in favor of cells with smaller g-values. If we set mode=1, we use $-g(s)$ as priorities to break ties in favor of cells with larger g-values. In test, we run A* search using two different modes of priority queue. Fig. 1 is the results of 50 tests on 101×101 maze experiment:

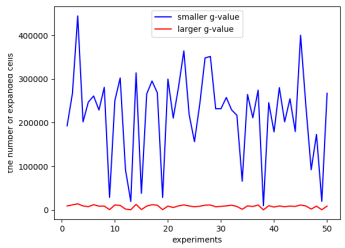


Figure 1: Smaller g-value and larger g-value

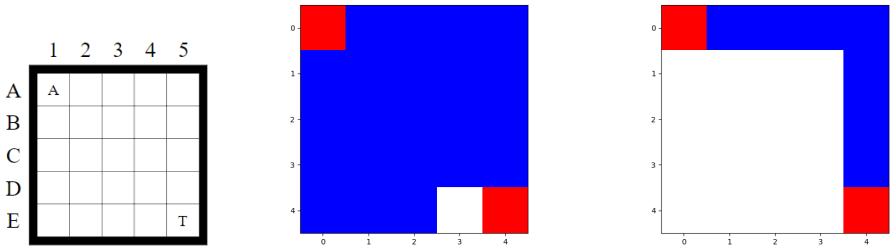
According to the result, we can find that the number of expanded cells of mode=0 is always bigger than the number of expanded cells of mode=1, which means the Repeated Forward A* using larger g-value tie breaking strategy is faster than Repeated Forward A* using smaller g-value tie breaking strategy.
The reason is that the agent will choose the point closer to the target to expand if we choose larger g-value to break ties. For two cells s_1, s_2 with f-value $f(s_1) = f(s_2)$, if

$$g(s_1) > g(s_2)$$

then

$$h(s_1) = f(s_1) - g(s_1) \leq f(s_2) - g(s_2) = h(s_2)$$

Therefore, if the A* use larger g-value to break tie, it will choose a cell with smaller h-value, which is closer to the goal. In A* searching, all cells with f smaller than $gd(start)$ will be expanded. So, the breaking ties strategy only affect how many cells with $f = gd(start)$ will be expanded. If A* choose to expand a cell with smaller g-value, it will expand almost all the cells with $f = gd(start)$ before it add $goal$ to the open list. Let's consider the following example to see which cells the two modes will choose to expand.



(a) maze (b) smaller (c) larger

Figure 2: Expanded map

For mode 0, see Fig. 2(c). The cells in blue means that the cells have been expanded through the whole search process.
For mode 1, see Fig. 2(b).
From these two results, we can see that we expand the cells closer and closer to the target when we choose larger g-value but we expand all the cells when we choose smaller larger g-value.
For mode 0(smaller g-value to break tie), here is the order of expansion,
(we always choose the point that is put into the open list earlier to expand when both f-value and g-value are equal).

$$A1$$
$$\rightarrow A2 \rightarrow B1$$
$$\rightarrow A3 \rightarrow B2 \rightarrow C1$$
$$\rightarrow A4 \rightarrow B3 \rightarrow C2 \rightarrow D1$$
$$\rightarrow A5 \rightarrow B4 \rightarrow C3 \rightarrow D2 \rightarrow E1$$
$$\rightarrow B5 \rightarrow C4 \rightarrow D3 \rightarrow E2$$
$$\rightarrow C5 \rightarrow D4 \rightarrow E3$$
$$\rightarrow D5$$

All the cells in this maze have same f-value. The level from high to low correspond the g-value from large to small.
For mode 0, we can see that the algorithm first expand all cells with smallest g-values, and then the second smallest and so on. Before it reach the goal, the algorithm have expand almost all cells in maze. That is far inefficient than mode 1. Here is the order of the expansion of the cells for mode 1,

$$A1 \rightarrow A2 \rightarrow A3 \rightarrow A4 \rightarrow A5 \rightarrow B5 \rightarrow C5 \rightarrow D5$$

We can see that the algorithm only expand 1 cell in each level. The number of total expended cells is less than mode0.

part 3

The Repeated Forward A* and Repeated Backward A* can share same A* searching algorithm. Therefore, to implement these 2 algorithm, we add a parameter $decode_{mode}$ to our A* function. When $decode_{mode} = 1$, the function will swap the position of *start* and *goal* and invert the *path* to

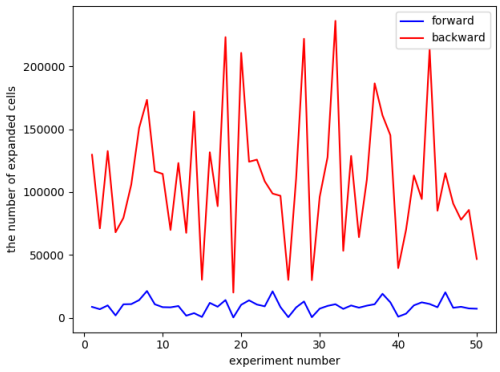


Figure 3: Forward A* search and Backward A* search

do Backward A* searching.
Fig. 3 shows the result of 50 test on 101×101 maze. According to the result, Repeated Forward A* expanded much less nodes than Repeated Backward A*. On average, Repeated Forward A* only expand 9113 nodes in each test, whereas Repeated Backward A* expand 111250 nodes. In addition, the variance of Repeated Forward A* is 24296811, and the variance of Repeated Backward A* is 2758697079. So, Repeated Forward A* is also more stable than Repeated Backward A*.
The reason is that the known world is closer to *start* than *goal*. In general, for any two points, if there are more blocked cells around one point, the cost of the path between them will be expectedly higher. Since the known world is closer to *start*, the blocked cells are relatively close to *start*. Therefore, for any two cells a and b respectively in forward and backward search, if $h(a) = h(b)$, there are higher probability that $g(a) > g(b)$. So, in general, there are more cells s with $f(s) < gd(start)$ in backward a* search than in forward a* search. Since all cells with $f(s) < gd(start)$ will be expanded in A* search, Backward A* search will expand more cells.

Part 4

Proving Manhattan distances are consistent:

According to triangle inequality, for any three point a, b and c, we have

$$h(a, c) \leq h(a, b) + h(b, c)$$

$h(a, c)$ is the Manhattan distance between a and c.
Then, since the agent can move only in the four main compass directions, the shortest distant between any two points is their Manhattan distances. So,

$$h(a, b) + h(b, c) \leq c(a, b) + h(b, c)$$

$c(a, b)$ is the real distance between a and b. So, any two point a and b satisfy

$$h(a, c) \leq c(a, b) + h(b, c)$$

Therefore, Manhattan distance are consistent.

Proving Adaptive A* is constances

Proving by induction.
Since the initially h-values are consistent, we assume the h-values in first $i-1$ searching are consistent, and need to prove the h-values in i^{th} searching are consistent. Separating the situation of any two cell a and b into three cases and prove

$$h^i(a) \leq c^i(a,b) + h^i(b)$$

1. Both a and b were expanded in $i-1^{th}$ search.
   Because both a and b were expanded, their h-values in i^{th} search are

   $$h^i(a) = g^{i-1}(goal) - g^{i-1}(a) \text{ and } h^i(b) = g^{i-1}(goal) - g^{i-1}(b)$$

   According to triangle inequality, we know

   $$g^{i-1}(b) \leq g^{i-1}(a) + c^{i-1}(a,b)$$

   Therefore, because the action cost is nondecreasing

   $$\begin{aligned} h^i(a) &= g^{i-1}(goal) - g^{i-1}(a) \\ &\leq g^{i-1}(goal) - g^{i-1}(a) + c^i(a,b) \\ &\leq h^i(b) + c^i(a,b) \end{aligned}$$

2. a was expanded but b was not.
   Because a was expanded but b wasn't, their h-values in i^{th} search are

   $$h^i(a) = g^{i-1}(goal) - g^{i-1}(a) \text{ and } h^i(b) = h^{i-1}(b)$$

   According to triangle inequality, we know

   $$g^{i-1}(b) \leq g^{i-1}(a) + c^{i-1}(a,b)$$

   Because b wasn't expanded, according to the expanding rule of A*, we have

   $$g^{i-1}(goal) \leq g^{i-1}(b) + h^{i-1}(b)$$

   Therefore,

   $$\begin{aligned} h^i(a) &= g^{i-1}(goal) - g^{i-1}(a) \\ &\leq g^{i-1}(b) + h^{i-1}(b) - g^{i-1}(a) \\ &\leq h^i(b) + c^i(a,b) \end{aligned}$$

3. b was expanded but a wasn't expanded.
   In this case,

   $$h^i(a) = h^{i-1}(a) \text{ and } h^i(b) = g^{i-1}(goal) - g^{i-1}(b)$$

   Because the heuristic action cost keeps nondecreasing, we can know

   $$h^i(a) = h^{i-1}(s) \leq h^{i-1}(b) + c^{i-1}(a,b) \leq h^i(b) + c^i(a,b)$$

Therefore, we prove that Adaptive A* leaves initially consistent h-values consistent

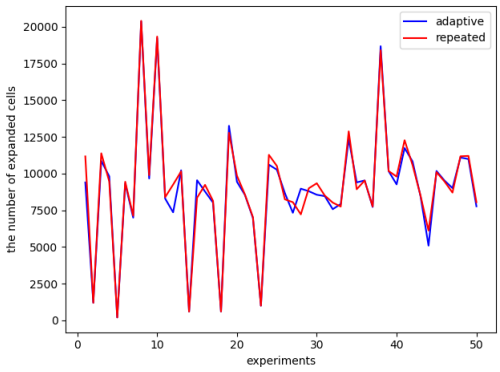


Figure 4: Repeated A* and Adaptive A*

Part 5

Fig. 4 shows the result of 50 independent tests for repeated A* search and adaptive A*.
From this result figure, we can find the total expanded states of these two different searching algorithm is nearly equal but adaptive A* search expands a little less states than repeated A* search. The reason is that adaptive A* search use $h_{new}(s) = g(goal) - g(s)$ to update the h-value of all expanded cell s. Since we know that $gd(s) \geq h_{new} \geq h(s)$, the h_{new} is more informed than the Manhattan distance $h(s)$. Therefore, Adaptive A* searching will expand less wrong cells (the cells out of the path) to find the path from *start* to *goal*.

part 6

Take the experiment question in part 2 to do a statistical hypothesis test.
Assume that performance differences between two search algorithms are due to sampling noise(null hypothesis). The probability of one algorithm performs better than another due to sampling noise is 50%. Set the significance level of the test $\alpha = 0.01$. The probability of getting the result in part 2 that the performance of breaking tie with larger g-value is better than the performance of breaking tie with smaller g-value shows up for 50 times in 50 experiments is $p = 0.5^{50} < \alpha$, which is in the region of rejection, so we need to reject the null hypothesis. Therefore, performance differences between two search algorithms are systematic in nature but not due to sampling noise.