Homework #1

Wenzheng Zhang, Qi Wu

February 28, 2020

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

We keep a closed list and the agent never comes back to the cells in the closed list. Because the grid world is finite, we can put at most finite number of cells into the closed list, which means we can only move finite steps to find the target or report that it is impossible to reach the target if we cannot find unblocked neighbors at one cell.
Suppose there are n unblocked cells in the grid world. According to the repeated A* algorithm this project use, the length of the path we compute every time is at most n. And we can compute path at most for n times, which means we need to compute path after each move. Therefore, that the number of moves of the agent until it reaches the target or discovers that this is impossible is $O(n \times n) = O(n^2)$.

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. For the same maze we generate, we run A* search using two different modes of priority queue. Figure 1 is the results of 50 different experiment:

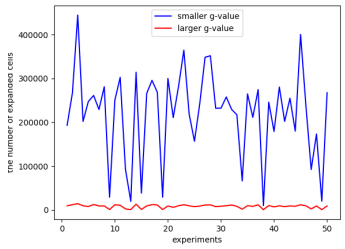
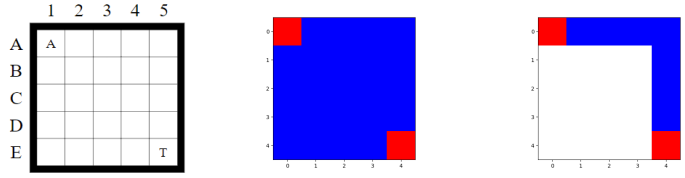


Figure 1: Smaller cost and larger cost

The y-axis is the total expanded cells that the agent starts from the start until it reaches the target or finds no way to the target in each experiment, the x-axis is each experiment. From the Figure we can know 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* with larger g-value tie break strategy is much faster than Repeated Forward A* with smaller g-value tie break 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 but the agent will choose the point closer to the start to expand if we choose smaller g-value to expand and we will expand many useless cells in this case. Take two states s_1, s_2 with the same f-value $f = f(s_1) = f(s_2)$ but $g(s_1) < g(s_2)$ as an example. We know $f(s_1) = g(s_1) + h(s_1)$ and $f(s_2) = g(s_2) + h(s_2)$, so $h(s_1) > h(s_2)$. Because $f(s_1) = f(s_2)$ and for mode 0 we choose smaller g-value to break tie, which means we will choose s_1 to expand since $g(s_1) < g(s_2)$. Thus, we choose the state with larger h-value to expand, which means we choose to expand the state that is closer to the start states and expand all other states with the same f-value and g-value as s_1 before we can expand the states with the same f-value but with smaller h-value in this case. Similarly,

for mode 1 we choose larger g-value to break tie, which means we choose s_2 to expand. Because $h(s_1) > h(s_2)$, we choose the point that is closer to the target in this case and don't need to expand any other states that have the same f-value and g-value with s_2 before we expand the state with the same f-value but smaller h-value. 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: result of larger cost and smaller cost. the blue cells is the expanded cells

For mode 0, see Figure 2(c). The cells in blue means that the cells have been expanded through the whole search process.
For mode 1, see Figure 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 1(larger g-value to break tie), here is the order of the expansion of the cells

$$A_1 \to A_2 \to A_3 \to A_4 \to A_5 \to B_5 \to C_5 \to D_5 \to E_5$$

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).

$$\begin{aligned} & A_1 \\ & \to A_2 \to B_1 \\ & \to A_3 \to B_2 \to C_1 \\ & \to A_4 \to B_3 \to C_2 \to D_1 \\ & \to A_5 \to B_4 \to C_3 \to D_2 \to E_1 \\ & \to B_5 \to C_4 \to D_3 \to E_2 \\ & \to C_5 \to D_4 \to E_3 \end{aligned}$$

$$\rightarrow D_5 \rightarrow E_4$$

$$\rightarrow E_5$$

We can see that we expand cells level by level ordering by the distance to the start point and expand many useless points when we choose smaller g-value to break tie.

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 do Backward A* searching.

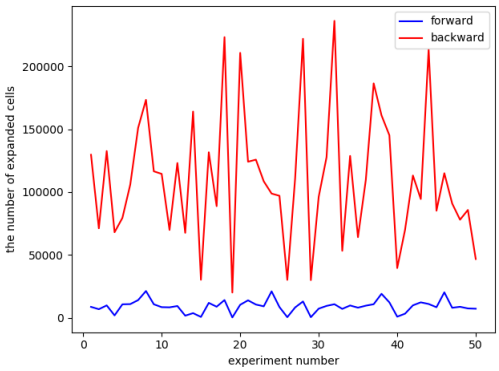


Figure 3: Forward and Backward

Figure3 show 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 more stable than Repeated Backward A*.
We believe the reason is that in general the known world is closer to *start* than *goal*. In each searching, for any unblocked cell s, the A* algorithm will expand it if $f(s) < gd(start)$, where $f(s) = g(s) + h(s)$ and $gd(start)$ is the real path cost form *start* to *goal*. For any two points, in general, the cost of

the path between them will be relatively higher, if there are more blocked between them. Since the known world is closer to *start*, the blocked cells are relatively close to *start*. Therefore, in general, there are more cells s with $f(s) < gd(start)$ in backward a* search than in forward a* search. So, Backward A* search will expand more cells.

Answer

Let $x_{ij} = 1$ if number i is hashed to bucket j. Let n_j denotes the number of elements in bucket j. For bucket j, we know the bubble sort algorithm of sorting this bucket j is $O(n_j^2)$. Thus, the total expected running time is

$$E(\sum_{j=1}^{n} O(n_j^2)) = \sum_{j=1}^{n} O(E(n_j^2))$$

We know

$$E(n_j^2) = \sum_{i=1}^{n}\sum_{k=1}^{n} E(x_{ij}x_{kj} = \sum_{i=1}^{n} E(x_{ij}^2)) + 2\sum_{i=1}^{n}\sum_{k=i+1}^{n} E(x_{ij}x_{kj}) = n\times\frac{1}{n} + 2\times\binom{n}{2}\frac{1}{n^2} = O(1)$$

Therefore, the expected running time is

$$E(\sum_{j=1}^{n} O(n_j^2)) = \sum_{j=1}^{n} O(E(n_j^2)) = n \times O(1) = O(n)$$

Part 4

a

let (i, j) denote the position of the state that is in i-th row and j-th col of the maze. Suppose the target point is (x, y). Take any state s (i, j), let $succ(s, a)$ denote the state that taking action a on s. Because we only have 4 choices of actions, so $succ(s, a)$ can be the following 4 different states:

$$(i + 1, j), (i - 1, j), (i, j + 1), (i, j - 1)$$

The manhattan distance between the goal to each of these 4 states can only have 2 possible results:

$$|x - i| + |y - j| + 1 \; and \; |x - i| + |y - j| - 1$$

and we know

$$|x - i| + |y - j| = h(s)$$

Therefore, $h(succ(s, a)$ can only have two possible value: $h(s)+1$ and $h(s)-1$ Thus, $h(s) - 1 \leq h(succ(s)) \leq h(s) + 1$. By induction for any two different states s,s' and $c(s, a, s') = k = c(s', a, s)$, we can know

$$h(s) - k \leq h(s') \leq h(s) + k$$

Therefore,

$$h(s) \leq h(s') + c(s', a, s)$$

and

$$h(s') \leq h(s) + c(s, a, s')$$

. Thus, the Manhattan distances are consistent in gridworlds in which the agent can move only in the four main compass directions.

b

We prove that Adaptive A* leaves initially consistent h-values consistent even if action costs can increase by induction. The initial heuristic h is supplied by user and is consistent. $h(goal) = 0$ is always true because we never expand the target. And $h(succ(a)) \leq h(s) + c(s, a)$. If actions cost increases from $c(s, a)$ to $c'(s, a)$, $h(succ(a)) \leq h(s) + c'(s, a)$ is still true because $c(s, a) \leq c'(s, a)$. Now consider when heuristic updates from h to h'. Let's consider the following 3 cases:
(i) both s and succ(s,a) were expanded in last search.
Thus, $h'(s) = g(goal) - g(s)$ and $h'(succ(s, a)) = g(goal) - g(succ(a))$. We can know $g(succ(a)) \leq g(s) + c(s, a)$ because the A* search discovers a path from the current state via state s to state succ(s, a) of cost $g(s) + c(s, a)$ during the expansion of state s. Therefore,

$$h'(s) = g(goal) - g(s) \leq g(goal) - g(succ(s, a)) + c(s, a) = h'(succ(s, a) + c(s, a)$$

(ii) s was expanded but $succ(s, a)$ was not.
In this case we can know $h'(s) = g(goal) - g(s)$ and $h'(succ(s, a)) = h(succ(s, a))$. And $g(succ(a)) \leq g(s) + c(s, a)$ for the same reason in case (i). Also, $g(goal) \leq f(succ(s, a))$ because $succ(s, a)$ is generated but wasn't expanded. Thus,

$$h'(s) = g(goal) - g(s) \leq f(succ(s, a)) - g(s)$$

$$= g(succ(s, a)) + h(succ(s, a)) - g(s) = g(succ(s, a)) + h'(succ(s, a)) - g(s)$$

$$\leq g(succ(s,a)) + h'(succ(s,a)) - g(succ(s,a)) + c(s,a) = h'(succ(s,a))$$

(iii) succ(s,a) was expanded but s wasn't expanded. In this case we can know $h'(s) = h(s)$ and $h(succ(s,a)) \leq h'(succ(s,a))$ because the heuristic of the same state keeps nondecreasing. Thus, we can know

$$h'(s) = h(s) \leq h(succ(s,a)) + c(s,a) \leq h'(succ(s,a)) + c(s,a)$$

From above we can know Adaptive A* leaves initially consistent h-values consistent even if action costs can increase

Part 5

We ran 50 independent experiments of repeated A* search and adaptive A* search and get the result as following:

g-value.png g-value.png

From this result figure we can know the total expanded states of these two different search algorithm is nearly equal but adaptive A* search expands a little less states than repeated A* search. I think the reason is that adaptive A* search is more informed, that is, the heuristic of adaptive A* search is closer to the real heuristic.

From this result figure we can know the total expanded states of these two different search algorithm is nearly equal but adaptive A* search expands a little less states than repeated A* search. I think the reason is that adaptive A* search is more informed, that is, the heuristic of adaptive A* search is closer to the real heuristic.

From this result figure we can know the total expanded states of these two different search algorithm is nearly equal but adaptive A* search expands a little less states than repeated A* search. I think the reason is that adaptive A* search is more informed, that is, the heuristic of adaptive A* search is closer to the real heuristic.

From this result figure we can know the total expanded states of these two different search algorithm is nearly equal but adaptive A* search expands a little less states than repeated A* search. I think the reason is that adaptive A* search is more informed, that is, the heuristic of adaptive A* search is closer to the real heuristic.