CS 440: Introduction to Artificial Intelligence: Assignment 1

Due on October 15, 11:55pm, 2021

Part 0

We used the algorithm provided in the assignment instruction to randomly construct the maze of size 101×101. To start point of the maze is randomly sampled at the upper-left corner of the maze, whereas the goal point is randomly sampled at the bottom right corner. To visualize the maze in the terminal, please refer to the `part_0()` function in the source file *run_exp.py*.

Part 1

Question a)

Initially, since the agent does not observed the blocked cell at `E4`, to agent will move to the east cell (i.e., `E3`) whose f value is smaller that the f value of the north cell `D2`. The f value of these two cells are given by:

$$f(\texttt{E3}) = 2 + 1 = 3 \tag{1}$$
$$f(\texttt{D2}) = 4 + 1 = 5 \tag{2}$$

Question b)

In the repeated forward A* algorithm, the agent will repeatedly run the A* algorithm every time it was blocked by a cell on it ways to the goal. Therefore, the agent will adjust its planned path according to its latest observation when running the A* algorithm. Thus, the agent will ultimately find a path to the goal or figure out that it is impossible to reach the goal. In the worst case, the agent needs to explore all unblocked cell to find the path to the goal, or discover it is impossible. Thereby, the number of moves of the agent until it reaches the target or discovers that this is impossible is bounded from above by the number of unblocked cells.

Part 2

The experimental results are shown in Table , where we run the repeatedly forward A* with each tie-breaking preference over all 50 mazes generated in Part 0, and calculate the averaged runtime over the 50 trails.

Table 1: Averaged Runtime of Repeated Forward A*

Method	Averaged Runtime [sec]
Larger g-values	0.8001
Smaller g-values	0.0336

From the experimental results, we can observe that the averaged runtime of the smaller g-values tie-breaking preference is much smaller than the tie-breaking preference with larger g-values. One explanation is that in the unobserved world, the agent needs to repeatedly explore the grid world. In the larger-g-values preference, the search algorithm will choose the cell that far from the start position (larger g), and thus have the agent more likely to move directly to the goal until blocked by a cell somewhere, whereas the smaller g-values preferences may have the agent explore more grid cell while heading to the goal in each round of A* search, and thus observe more previously unseen blocked cell after each round. With more blocked cell being observed, the overall repeatedly search algorithm may need less rounds to finally find the path to the goal. Thereby, the averaged runtime of the smaller g-values tie-breaking preference is much smaller than the tie-breaking preference with larger g-values.

Part 3

The experimental results are shown in Table , where we run the repeatedly forward A* and repeatedly backward A* with same tie-breaking preference (larger g-values) over all 50 mazes generated in Part 0, and calculate the averaged runtime over the 50 trails.

Table 2: Averaged Runtime of Repeated Forward/Backward A*

Method	Averaged Runtime [sec]
Forward g-values	0.7977
Backward g-values	0.8055

From the experimental results in Table , we can observe that there is no significant difference between repeated forward and backward A* search. The runtime of the backward method is slightly slower than the forward method. The reason is that the forward and backward is almost symmetric in our case as the mazes are randomly generated on a uniform basis.

Part 4

Firstly, we prove the Manhattan distance is a consistent heuristic for this search problem.

Proof. Since the agent can only go up, down, left, and right direction in each step, the Manhattan distance heuristic is naturally admissible.

$$h(s) \leq h^*(s), \ \forall \ s$$

where the $h^*(s)$ is the optimal distance from s to the goal.

Every time when the agent makes a movement, i.e., from a state s to a successor state s', there are two possibilities

1. The agent is move one step farther away to the goal, then we have $h(s') = h(s) + 1$, thus $h(s) \leq h(s') + c(s, s')$, where $c(s, s')$ is the cost from s to s', and $c(s, s') = 1$.
2. The agent is move one step closer to the goal, then we have $h(s') = h(s) - 1$, thus $h(s) \leq h(s') + c(s, s')$.

No matter which possibilities happened, we always have $h(s) \leq h(s') + c(s, s')$ hold. Thereby, the Manhattan distance heuristic is consistent. □

Next, we want to prove the heuristic for the Adaptive A* is also consistent.

Proof. Without the lose of generality, we consider any arbitrary state s, and the agent make a movement from s to s' with cost $c(s, s') = 1$. Then, there are four possibilities:

1. s and s' are both un-expanded in the previous A* search. In this case, we directly use the Manhattan distance as the heuristic, and we already proved that the Manhattan distance is consistent, and thus we have $h(s) \leq h(s') + c(s, s')$.
2. s and s' are both expanded in the previous A* search. In this case, we apply the heuristic $h_{new}(\cdot)$ for both states. Then we have

$$\begin{aligned} h_{new}(s) - h_{new}(s') &= [gd(s_{start}) - g(s)] - [gd(s_{start}) - g(s')] \\ &= g(s') - g(s) \leq |g(s') - g(s)| \end{aligned}$$

Since s' is the successor of the state s, we have $|g(s') - g(s)| = 1$, thereby, we have

$$h_{new}(s) - h_{new}(s') \leq 1 = c(s, s')$$

And thus, we have $h_{new}(s) \leq h_{new}(s') + c(s, s')$.

3. s is expanded in the previous A* search, whereas s' is not expanded in the previous A* search. In this case, we apply $h_{new}(\cdot)$ for the state s, and the original Manhattan distance $h(\cdot)$ for the state s'.

   Since the state s' was not expaned in the previous A* search, thereby, in the previous search, we must have

$$h(s') + g(s') \geq gd(s_{start})$$
$$h(s') + g(s') - g(s) \geq gd(s_{start}) - g(s)$$
$$h(s') + g(s') - g(s) \geq h_{new}(s)$$

   Thereby, we have

$$h_{new}(s) \leq h(s') + g(s') - g(s)$$
$$\leq h(s') + |g(s') - g(s)|$$
$$\leq h(s') + 1 = h(s') + c(s, s')$$

   Thus, in this case, we also have $h_{new}(s) \leq h(s') + c(s, s')$.

4. s was un-expanded in the previous A* search, whereas s' was expanded in the previous A* search. If we apply Manhattan distance for both s and s', as Manhattan heuristic is consistent, we have

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

   As we know,for a previously expanded state s', we have $h(s') \leq h_{new}(s')$, thereby, we have

$$h(s) \leq h(s') + c(s, s')$$
$$leq h_{new}(s') + c(s, s')$$

   Thus, in this case, we also have $h(s) \leq h_{new}(s') + c(s, s')$.

Thereby, in summary, under all possibility, the consistency requirements are always hold. Then, we know that the heuristic for the Adaptive A* is also consistent.

□

Part 5

The experimental results are shown in Table , where we run the Repeated Forward A* and Adaptive Forward A* with the same tie-breaking preference (larger g-values) over all 50 mazes generated in Part 0, and calculate the averaged runtime over the 50 trails.
From the experimental results, we can observe that the averaged runtime of the Adaptive A* is slightly slower than that of the Repeated A* algorithm. We carefully examine the reason behind this since it against our assumption. Although, the Adaptive A* explored less states comparing the to Repeated A* algorithm, it takes over time to calculate the $h_{new}(\cdot)$ function. Particularly, we use a hash dictionary to store the h_{new} heuristic value for the previously explored states. The time saved on the state expansion cannot offset the the time it takes to access and write to the hash dictionary.

Table 3: Averaged Runtime of Repeated/Adaptive Forward A*

Method	Averaged Runtime [sec]
Repeated g-values	0.8303
Adaptive g-values	0.8762

Part 6

We can perform a statistical hypothesis test to verify whether the performance difference between two algorithms are systematic in nature or only due to sampling noise.

1. Define the initial research hypothesis, i.e., the performance difference are due to sampling noise.
2. The null hypothesis is that performance difference between two algorithms are systematic in due to sampling noise.
3. Run the experiment for many times to gather sufficient statistical data samples.
4. Derive the distribution of the test statistic under the null hypothesis from the assumptions.
5. Select a significance level, a probability threshold below which the null hypothesis will be rejected.
6. Compute from the observations the observed value of the test statistic, and decide to either reject the null hypothesis in favor of the alternative or not reject it.