440 Assignment 1

Xiaowei Zhang, 204007085
Xinyuan Chen, 197001668

October 2021

Part 1

a. Because the agent supposes all the cells are unblocked in the beginning. The path is determined by the evaluation function $f(s) = g(s) + h(s)$. After expanded the start node $E2$, we got $\{E1, D2, E3\}$ in the open list. $g(s)$ is the path cost and equal to 1 for each cell in the list. We choose Manhattan distance for $h(s)$, in this case, so $E3$ has the smallest h-value, thus the smallest f-value. So firstly moving to the east gives the shortest path from the start point to the target.

b. Since the number of grids is finite, even in the worst scenario that the agent needs to transverse the whole maze, it can still either find the path or find it is impossible to reach the target within finite time. Also, according to the A^* algorithm, the agent will not expand the cells that have already expended, so it will not resulting in an infinite looping. Suppose there are n unblocked cells, in the worst case, the program will need to expand all the unblocked cells with in one call of A^*, so the maximum moves within one call is $a \leq n$. Also suppose after each move, the agent encounter the blocked cell and need to call A^* again, the total times of calling A^* is $b \leq n$. Thus, the total number of moves needed is $a \cdot b \leq n^2$, so it is bounded by n^2.

Part 2

- Observation: We randomly ran the different versions of A^* on the same map 50 times (see Figure 1, red: smaller g-values vs blue: bigger g-values). By comparing the results, we find that the version with larger g-values is better in terms of the run time in most of the cases, and it also expands less cells.
- Explanation: This is because since $f(s) = g(s) + h(s)$, for the two cells with the same $f(s)$, a larger $g(s)$ means a smaller $h(s)$, which means it is closer to the target.

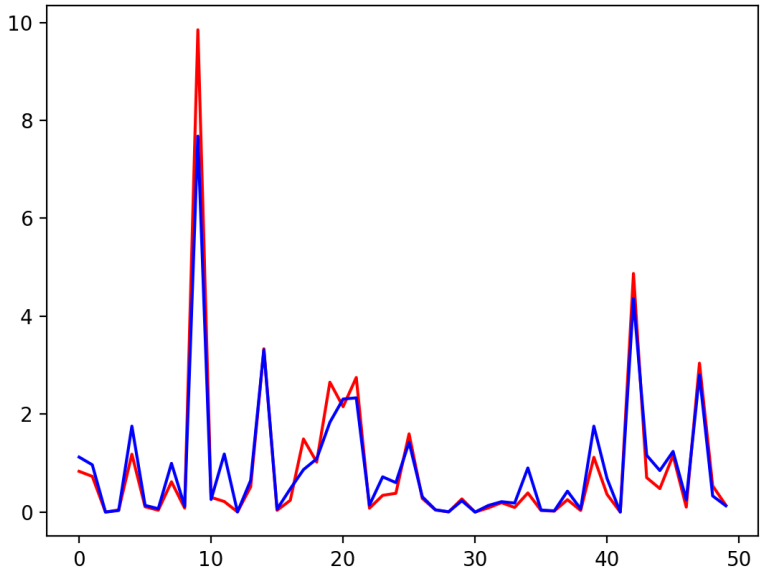

Figure 1: x axis: case number - y axis: runtime

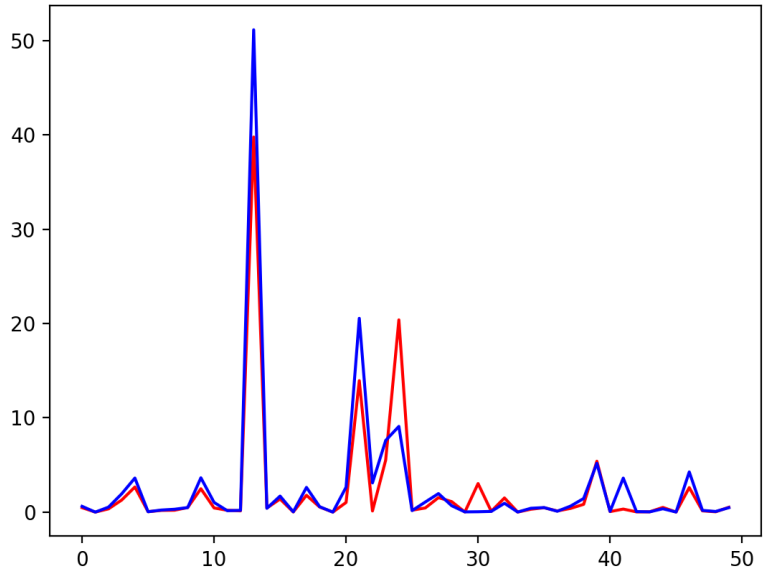

Figure 2: x axis: case number - y axis: runtime

Part 3

- Observation: After both choosing larger g-values to break ties, we randomly ran the forward A^* and backward A^* 50 times (see Figure 2, red: Forward A^* vs blue: Backward A^*). By comparing the results, we find that it is hard to tell which one is better. Sometimes backward A^* is better in terms of the run time and expanded cells, sometimes it is not. It depends on the location of the blocked cells. We found that when the blocked cells are located more around the start point, the the forward is better, similarly, if the blocked cells are located more around the target, then backward is better.

- Explanation: This is because it is better for us to find the blocked cells earlier so that we don't need to redirect the path again and again later. So if the blocked cells are more around the start cell, then using forward method can help us find blocked cells earlier. Similarly, if the blocked cells are more around the target cell, then using backward method can help us find blocked cells earlier.

Part 4

- By the definition, we know a heuristic function is said to be consistent if for any $(n, a, n') : h(n) \leq c(n, a, n') + h(n')$, where $h(n)$ is the estimated cost from n to target and $c(n, a, n')$ is the step cost from n to n'. We use Manhattan distance here as h-value, it is defined as the sum of the absolute lengths of paths between the two cells horizontally and/or vertically, which is consistent with how the agent moves (the agent can not move diagonally, otherwise Manhattan distance would overestimate).
  Denote $h(s) = |x_s - x_{goal}| + |y_s - y_{goal}|$, for $s \neq goal$, we want to prove $h(s) - h(s') \leq c(s, a, s')$ where either $x_s = x'_s$ or $y_s = y'_s$ since they are neighbours:
  If $x_s = x'_s$, $h(s) - h(s') = (|x_s - x_{goal}| + |y_s - y_{goal}|) - (|x'_s - x_{goal}| + |y'_s - y_{goal}|) = |y_s - y_{goal}| - |y'_s - y_{goal}| \leq |(y_s - y_{goal}) - (y'_s - y_{goal})| = |y_s - y'_s| \leq 1 = c(s, a, s')$.
  If $y_s = y'_s$, $h(s) - h(s') = (|x_s - x_{goal}| + |y_s - y_{goal}|) - (|x'_s - x_{goal}| + |y'_s - y_{goal}|) = |x_s - x_{goal}| - |x'_s - x_{goal}| \leq |(x_s - x_{goal}) - (x'_s - x_{goal})| = |x_s - x'_s| \leq 1 = c(s, a, s')$.
  Above all, $h(s) - h(s') \leq c(s, a, s')$, so we can say the Manhattan distances are consistent in the grid worlds.

- Since the h-values are initially consistent, we know that $h(n) \leq c(n, a, n') + h(n')$. The new h-value is defined as $h_{new}(s) = g(s_{goal}) - g(s)$, where $g(s)$ is the distance from the start state to s and $g(s_{goal})$ is the distance from the start state to goal. Since we only update h-values of the expanded nodes in Adaptive A^*, if both s and s' are not expanded, we still use Manhattan distance which has proved above. Now consider at least one node has been expanded, we need to prove $h_{new}(s) \leq h_{new}(s') + c(s, a, s')$:
  If both s and s' have been expanded, then $h_{new}(s) = g(s_{goal}) - g(s)$, $h_{new}(s') = g(s'_{goal}) - g(s')$. Since s' is a successor of s, so $g(s') \leq g(s) + c(s, a, s')$, thus $g(s_{goal}) - g(s) \leq g(s_{goal}) - g(s') + c(s, a, s')$, which is $h_{new}(s) \leq h_{new}(s') + c(s, a, s')$.
  If s has been expanded, but s' has not, then $h_{new}(s) = g(s_{goal}) - g(s)$, $h_{new}(s') = h(s')$. s' is a successor of s, so $g(s') \leq g(s) + c(s, a, s')$. Also since $f(s) \leq f(s')$, $g(s) + h_{new}(s) \leq g(s') + h(s')$. Add the two equations above, we get $h_{new}(s) \leq h_{new}(s') + c(s, a, s')$.

Part 5

- Observation: After both choosing larger g-values to break ties, we randomly ran the forward A^* and backward A^* 50 times (see Figure 3, red: A^* vs blue: Adaptive A^*). By comparing the results, we find that Adaptive A^* is slightly better in terms of the runtime, it also expands less cells.

- Explanation: Since the Adaptive A^* changes the h-values of the expanded

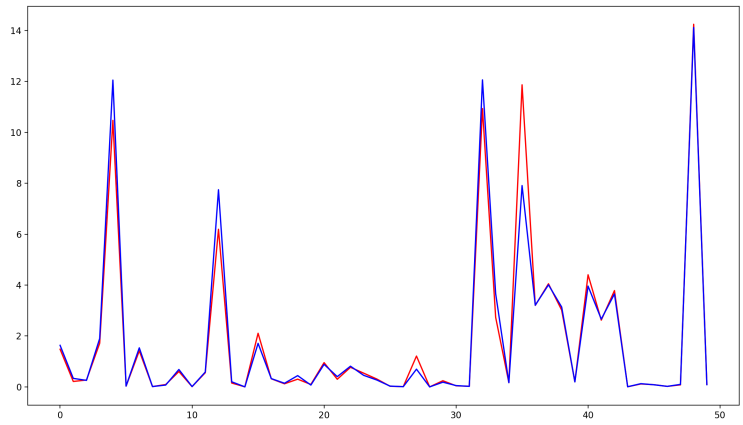

Figure 3: x axis: case number - y axis: runtime

nodes into a higher value, so the agent keeps learning and updating through its way to the target, thus it can expand less nodes and reduces the unnecessary moves of the agent. However, since only the h-values of expanded nodes will be updated, there are still many nodes remain unknown in the maze. Also, we have to transverse the h-values table and update it after each move, these also take time, so the overall actual runtime of Adaptive A^* is just slightly smaller than the normal A^*.

Part 6

For example for Part3, we first use forward and backward A^* to solve for random problems several times, say 50 times, so sample size $n1 = n2 = 50$. Then calculate the mean value of runtime of each approach, denote the sample mean runtime of forward A^* as $\bar{x_1}$, the sample mean runtime of backward A^* as $\bar{x_2}$. Denote the runtime of forward A^* as μ_1, the runtime of backward A^* as μ_2. Assume the variances of two approaches are equal, denote the two sample variances as s_1^2 and s_2^2. Since we don't know the population variance, we use t-test.

H0: $\mu_1 = \mu_2$, H1: $\mu_1 \neq \mu_2$

$s_p^2 = \frac{(n_1-1)s_1^2+(n_2-1)s_2^2}{n1+n2-1}$, $t = \frac{(\bar{x_1}-\bar{x_2})}{\sqrt{s_p^2(\frac{1}{n_1}+\frac{1}{n_2})}}$, we use $\alpha = 0.05$.

If $|t| > t_{\alpha/2}$, we have enough evidence to reject H0 and conclude that there are significant differences between forward and backward A^*. Otherwise, we do not have enough evidence to reject H0 and conclude that there are not significant differences between forward and backward A^*.