Assignment 1 Report

Gautam Agarwal (–), Tanvi Thigle (–), Riddhi Patel (–)

February 2023

1 Part 1 - Understanding Methods

A.) Explain in your report why the first move of the agent for the example search problem from Figure 8 is to the east rather than the north given that the agent does not initially know which cells are blocked.

The first move of the agent will be east because the goal of the agent is to get the shortest path to the target when running A*. The agent's initial vision does not detect any blockage toward its assumed shortest path (which is 3 blocks away from the target in Figure 8). However, once it does detect a blockage then it will reroute and move North.

B.) This project argues that the agent is guaranteed to reach the target if it is not separated from it by blocked cells. Give a convincing argument that the agent in finite grid worlds indeed either reaches the target or discovers that this is impossible in finite time. Prove that 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 squared.

Within finite time the agent will reach the target or discovers it is not possible because if the agent isn't separated from the target by blocked cells then it can move step-by-step by exploring its neighbors respectively that are not blocked until it reaches the target. It will be guaranteed.

However, if the agent is separated from the target with blocked cells, then the agent will explore all the other unblocked neighboring cells step-by-step without finding the target. When it explores every unblocked cell, that's when the agent will realize that it is not possible to reach the target.

Proof:

Overall, this depends on the number of unblocked cells n, found in the grid: We know that the agent when trying to reach the target, has 4 moves that it can make: left, right, up, and down. This implies that there can be at most 4 unblocked cells that the agent can move to when running A*. An agent can make m steps depending on the path it takes to reach the target. Therefore, to represent that the maximum number of m steps is represented as $4\mathrm{m}^2 + 4m + 1$.

If we have a case where the m-steps it takes are greater than n unblocked cells in the grid:

$4m^2 + 4m + 1 \geq n$

$4m^2 + 4m \geq n-1$

$m^2 + m \leq (n-1)/4$

$\sqrt{m^2 + m} \leq \sqrt{((n-1)/4)}$

$m + \sqrt{m} \leq \sqrt{((n-1)/4)}$

As a result, the number of moves the agent makes until it either reaches the target or doesn't because it's blocked is bounded above by the total number of unblocked cells ($m^2 \leq (n - 1) / 4$)

---

2 Part 2 - The Effects of Ties

Repeated Forward A needs to break ties to decide which cell to expand next if several cells have the same smallest f-value. It can either break ties in favor of cells with smaller g-values or in favor of cells with larger g-values. Implement and compare both versions of Repeated Forward A* with respect to their runtime or, equivalently, the number of expanded cells. Explain your observations in detail, that is, explain what you observed and give a reason for the observation.*

-Repeated Forward A-star in favor of smaller g-values is more efficient in terms of runtime and expanding cells as compared to A star breaking ties in favor of larger g-values.

This is due to the fact that breaking in favor of smaller g-values means that the agent is still close to the start state and is expanding near the start instead of near the goal state. Since the agent is close to the start state it has more flexibility to find the optimal path. Therefore even if it expands a lot of cells near the start state it helps towards finding the optimal path. However, if the tie is broken in favor of large g-values the agent is basically expanding more cells near the goal states most of which end up being dead-ends. It is also important to note that the difference between these two is not very significant and both can find the path. The heuristics, search space, and other factors can also play an important role in this. In general, breaking in favor of smaller g-values is more efficient as it expands fewer cells as compared to breaking ties in favor of larger g-values

---

3 Part 3 - Forward VS. Backwards

-Implement and compare Repeated Forward A and Repeated Backward A* with respect to their runtime or, equivalently, the number of expanded cells. Explain your observations in detail, that is, explain what you observed and give a reason for the observation. Both versions of Repeated A* should break ties among cells with the same f-value in favor of cells with larger g-values and remaining ties in an identical way, for example randomly.*

Repeated Forward and repeated backward give us similar results in terms of runtime and expanded cells. The results can also be attributed to the forward and backward algorithm since the tie-breaking is controlled by one type of method. This insignificant difference can be attributed to small-scale mazes and randomly generated mazes.

However, it seems as though the backward repeated A star is more focused since it starts from the goal state and has more knowledge about the surroundings of the goal state as compared to the repeated forward A-star. Repeated forward A star, however, has more visibility from the start state and therefore is not stuck at obstacles that the A star path did not foresee like what happens in repeated backward A star. It is difficult to tell which is better since each of them might have pros and cons in different circumstances with different goals. There might be cases where forward A star unnecessarily expands nodes that are not closer to the goal but closer to the Start state and have no promise of expanding the optimal path while the repeated backward A star might be able to focus the search to nodes closer to the goal state and find the result faster with expanding fewer nodes.

---

4 Part 4 - Heuristics in the Adaptive A*

-The project argues that "the Manhattan distances are consistent in grid worlds in which the agent can move only in the four main compass directions." Prove that this is indeed the case.

Proof: First of all, the Manhattan distance is consistent in the grid world as the agent moves in the four compass directions. This is because the Manhattan distance satisfies the triangle inequality which proves consistency. This says that there are three points, let's say points 1, 2(point somewhere in the middle), and 3, a path from 1 to 3 would be less than the sum of the path from 1 to 2 and 2 to 3.

d= distance function

$d(1,3) \leq d(1,2) + d(2,3)$

The Manhattan distance satisfies this. Let's say that we have points on the grid (x1,y1), (x2,y2), and (x3, y3) that form a triangle. The absolute difference of the x and y coordinates between two points, let's say 1 and 3, would be less than the sum of the absolute difference between 1-2 and 2-3.

$$|x3 - x1| + |y3 - y1 \leq |x2 - x1| + |y2 - y1| + |x3 - x2| + |y3 - y2|$$

As the agent can only move in the four directions on a compass(North, South, East, West), the absolute difference of each coordinate(x,y) summed can be used to represent the distance between the points as you add the horizontal distance plus the vertical distance, which is what the Manhattan distance does as it counts by each grid in such directions. The equation above can be simplified to fit the triangle inequality.

d(1, 3) $\leq$ d(1,2) + d(2,3)

Thus, the Manhattan distance is consistent when moving in the four compass directions.

-Furthermore, it is argued that "The h-values hnew (s) ... are not only admissible but also consistent." Prove that Adaptive A leaves initially consistent h-values consistent even if action costs can increase.*

Proof: Initially, the Adaptive A* uses the Manhattan distance to compute the initial h value for the path. As the Manhattan distance is consistent thanks to the triangle inequality, the h values are initially consistent.
As g(s) increases by 1 for every step the agent takes, the g(s) is not consistent. However, the heuristic changes to become

hnew(s) = g(sgoal) - g(sl)

where the g(sgoal) is the distance of the goal from your current state, and g(s) is the cost from the start state to your current state.)

In order to prove that the Adaptive A* to leave consistent hnew(s) values, we need to show that the new heuristic satisfies the triangle inequality and that when it goes from one state to the next the h value never overestimates the actual step cost

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

With c is the cost of the action from state s to s' the successor state. The initial h value is calculated by the Manhattan distance which is consistent.

In order to show that satisfies the triangle inequality let's use s1, s2, s3 as nodes in succession with a path from s1 to s3 and s2 and we want to prove the

following inequality then:

h(s1) $\leq$ c(s1, s2) + h(s2) and h(s2) $\leq$ c(s2, s3) + h(s3)

Implies that h(1) $\leq$ c(s1, s2) + c(s2, s3) + h(s3)

We want to prove the above inequality to show consistency. Thus, we can manipulate this to show that the hnew(s) values are consistent. We can first substitute for h(s) = g(sgoal) - g(s) for both of the h(s1) and h(s3)

g(sgoal) - g(s1) $\leq$ c(s1, s2) + c(s2, s3) + g(sgoal) - g(s3)

This equation be further simplified to:

g(sgoal) - g(s1) + c(s3, goal) $\leq$ c(s1, s2) + c(s2, s3)

We can then add g(s1) to both sides of the equation:

g(sgoal) + c(s3, goal) $\leq$ c(s1, s2) + c(s2, s3) + g(s1).

From this equation we see that the cost of a step from s3 to goal is less than or equal to the cost of the path from s1 to s3. Thus we see that the heuristic is consistent as fits this equation and the cost of the path to the next is less than or equal to the cost of the path from a starting state s1 to s3 with s2 in between. This satisfies the triangle inequality.

---

5 Part 5 - Heuristics in the Adaptive A*

-Implement and compare Repeated Forward A and Adaptive A* with respect to their runtime. Explain your observations in detail, that is, explain what you observed and give a reason for the observation. Both search algorithms should break ties among cells with the same f-value in favor of cells with larger g-values and remaining ties in an identical way, for example randomly.*

When implementing The Repeated Forward A* compared to the Adaptive A*, we found that the runtime often varied between the two. Sometimes the runtime for repeated A* would be faster than Adaptive* and sometimes it was the other way around. A lot of the time we found that they had similar runtimes. There was not a significant improvement when comparing the two. We believe this is because, while the Adaptive A* uses a more accurate heuristic of h(s) = g(sgoal) - g(s), since it is closer to the actual cost of the path compared to repeated, the adaptive A* does require more bookkeeping. We have to continually record and update the heuristic values for states or positions that have been expanded, even though we do expand nodes more efficiently. This can cause the runtime of adaptive A* to maybe take a similar amount of time

to repeated A*.

---

6 Part 6 - Statistical Significance

-Performance differences between two search algorithms can be systematic in nature or only due to sampling noise (= bias exhibited by the selected test cases since the number of test cases is always limited). One can use statistical hypothesis tests to determine whether they are systematic in nature. Read up on statistical hypothesis tests (for example, in Cohen, Empirical Methods for Artificial Intelligence, MIT Press, 1995) and then describe for one of the experimental questions above exactly how a statistical hypothesis test could be performed. You do not need to implement anything for this question, you should only precisely describe how such a test could be performed.

We can do a statistical hypothesis test when comparing two algorithms for a significant difference. We can perform this hypothesis test when comparing repeated forward A star and adaptive A star.

We want to perform sample t-tests to see if there is a significant difference between the two algorithms. Our null hypothesis will be that both the adaptive A star and repeated forward A star have the same mean performance. The alternative hypothesis would be the complete opposite, implying that there would be a difference between the means. In other words, a significant difference.

To do so, the data that we will record/collect would be based on the runtime between the two algorithms when executed on the randomly generated mazes. To calculate the t-values:

t = (mean(A) - mean(B)) / (sqrt($SA^2/nA + SB^2/nB$))

mean(A) and mean(B) are the means of the performance for algorithms Repeated Forward A star and Adaptive A star. SA and SB are the standard deviations of the respected algorithms, and nA and nB are the sample sizes (which are the number of times we ran the algorithms to calculate runtime on randomly generated mazes).

So we will need to calculate the mean and standard deviation of both A star algorithms to calculate the t-value for both adaptive and repeated A star. From there, we can calculate the p-value which helps determine the probability of whether or not the t-value is extreme, assuming that the null hypothesis is true. If p is less than the significance level (which is 0.05), the null hypothesis will be rejected because there is a significant difference between the two A* algorithms. However, if the p-value is greater than the significance level, we can reject the null hypothesis. This means that there is not enough evidence to conclude there is a significant difference between the two A star algorithms.
