RUTGERS

Fast Trajectory Planning

submitted in fulfillment of the requirements

for

FALL 22 Assignment 1

in

CS520: INTRODUCTION TO ARTIFICIAL INTELLIGENCE

by

KOWNDINYA BOYALAKUNTLA 219002814

ANIRUDHA SARMA TUMULURI 219000093

Supervisor(s)

Dr. Abdeslam Boularias

DEPARTMENT OF COMPUTER SCIENCE

RUTGERS UNIVERSITY - NEW BRUNSWICK

OCT 2022

PART 0: SETTING UP THE ENVIRONMENT

Maze Generation

We have created 50 grid worlds of size **101 x 101**. As outlined in the instructions, we have initially set all the cells as unvisited. We then chose a random cell, marked it as visited, and we unblock the initial cell. Every time we choose a random unvisited neighbor, we account for a probability of 0.3 to block it; otherwise, the cell is unblocked and will be added to the *dfs_stack*. During the exploration, if exploration reaches a dead-end, we backtrack and visit a new unvisited cell. Eventually, the algorithm will visit each cell of the grid, and the random maze is now ready for experimentation.

We repeat the above-described algorithm 50 times to generate 50 random mazes and we store them inside the folder named *mazes*. A random grid world is shown below in Figure 1.

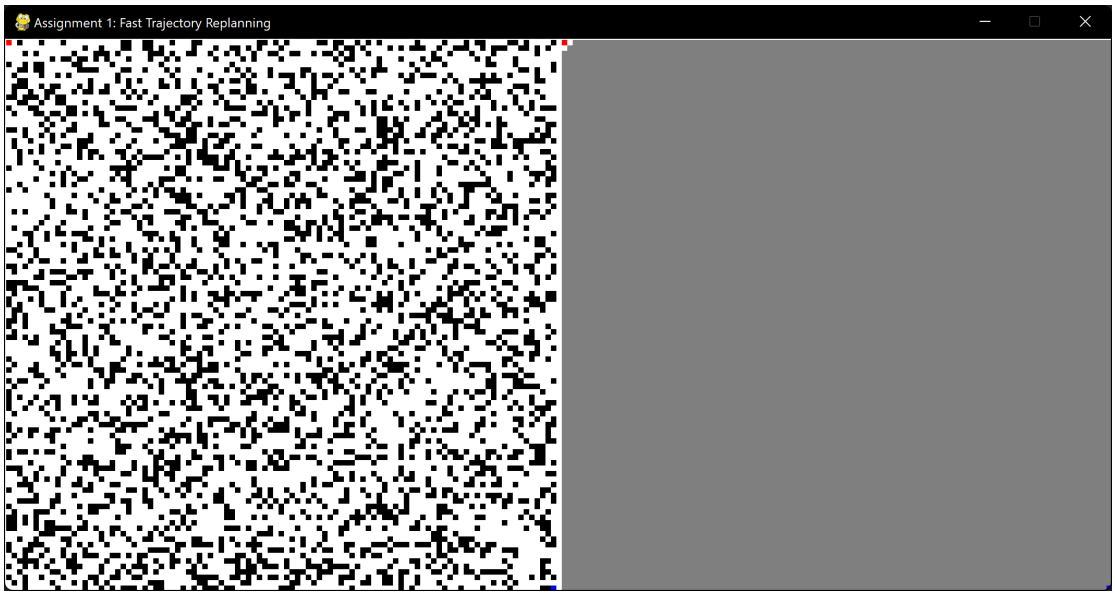

Figure 1: Random Maze

PART 1: UNDERSTANDING THE METHODS

a) Agent moves to the east

In Figure 8, the agent initially has no knowledge of blockage at E4 and assumes it to be open; hence while doing A^*, it assumes that it can traverse through the square, which would have a lower h value, and thereby leading to a lower f value.

b) No. of agent moves <= No. of unblocked cells**2

i) One A^* search visits at most all the unblocked cells

If the agent is not separated from the target by blocked cells, then at any iteration of A^*, there exists at least one move in the open set that has been unexplored. This is true until the last move leads to the target, and after that, it has already reached its destination.

If it is blocked, then it reaches a point where the open set is empty, and there are no more unexplored moves since we keep track of visited nodes, there is no reason for it to get stuck in an infinite loop as well, and it has not reached the destination.

The agent keeps moving in the path planned by the previous iteration of A^* until the path is blocked or reaches the destination. When the path is blocked, it then computes the path using A^* again and takes a different

route. Every time it hits a roadblock, it explores a different path, i.e., adds unexplored nodes into the open set. This is done till the target is reached or exhausted all options. The maximum it can explore at one point if it is blocked is all the unblocked nodes that can be reached from the starting node.

ii) A^* **search can repeat for all unblocked cells**

In the worst case, it would visit all unblocked cells in each search and this search can be repeated for all unblocked cells thereby leading to an upper bound for agent moves to be unblocked cells squared in the worst case.

PART 2: EFFECT OF TIES

Repeated Forward A^* maxG vs minG

Repeated forward A^* favoring maximum G values runs faster than that of one favoring minimum G values. This is because for cells with the same f value the algorithm is preferring cells with lower heuristics (which is given by larger Gs) and hence making lesser steps in reaching the goal. Conversely, for the algorithm preferring smaller G values, the cells with relatively higher heuristic cost are preferred and therefore picking cells with higher h(estimates).

As shown below in Figure 2, the blue line graph shows the runtime for the repeated forward A^* algorithm in favor of maximum G values (a.k.a maxG) and correspondingly, the red line shows runtime for the algorithm favoring minimum G values (a.k.a minG). Additionally, Figure 3 shows the path visualization on a random maze in Forward A^* maxG and minG algorithms.

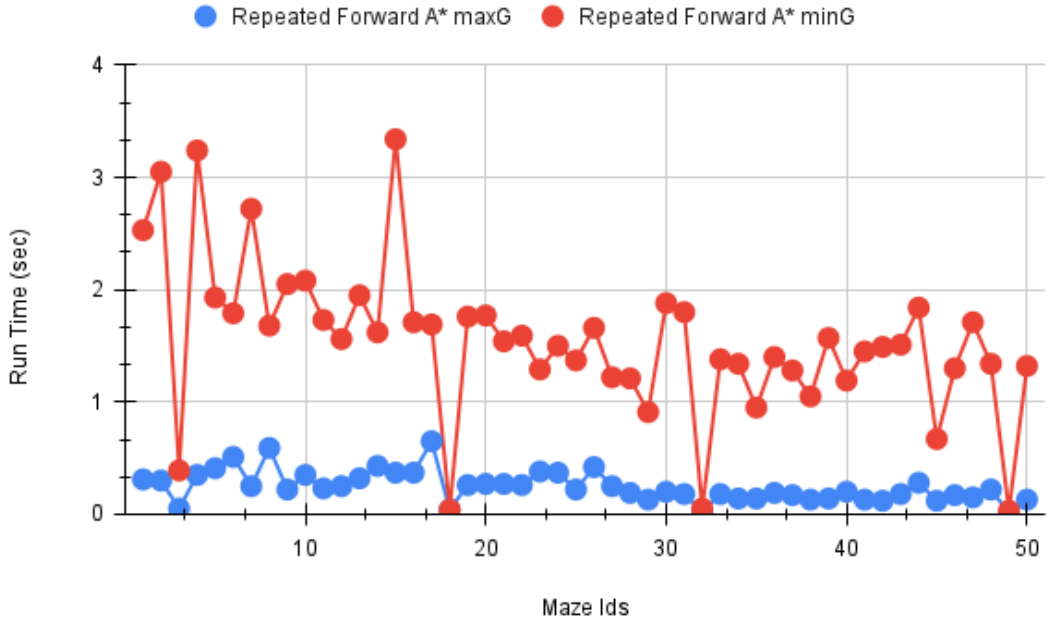

Figure 2: Repeated Forward A* maxG vs minG

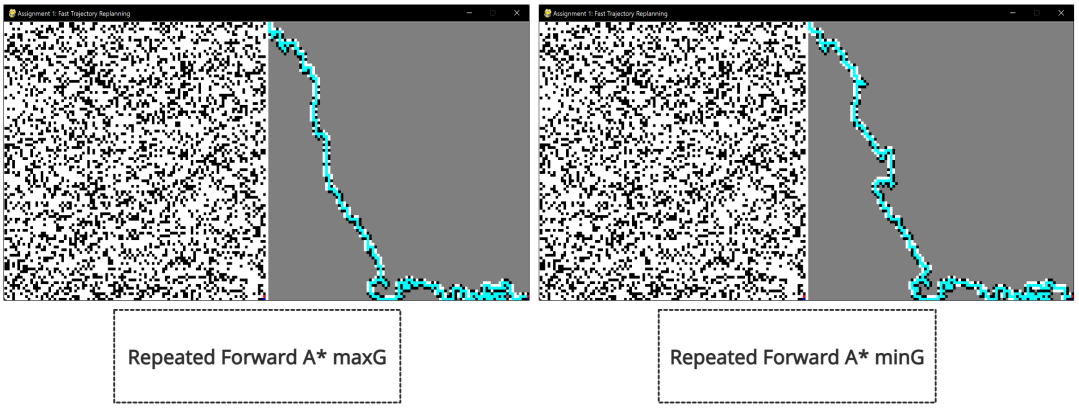

Figure 3: Path visualization in Repeated Forward A* maxG and Repeated Forward A* minG

PART 3: Forward vs Backward

Repeated Forward A^* maxG vs Repeated Backward A^* maxG

We see that from running the algorithms on various mazes, repeated forward A^* works faster than repeated backward A^* in almost every situation. This is because backward A^* starts exploring from the endpoint, which leads to it having limited information during the initial part of the search.

It gives more weight to the path it initially starts exploring and keeps doing so in repeated instances until it definitively proves that it cannot reach the target through that path which only occurs late into the search when the agent moves and updates its knowledge of the environment.

On the other hand when forward A^* runs, we see that it can easily avoid routes that don't lead to a favorable outcome since it starts from the agent where we already have information about its immediate environment which leads to better decisions and fewer routes being explored.

The observed runtimes for repeated forward A^* and repeated backward A^* are shown below in Figure 4. Additionally, Figure 5 shows the path visualization on a random maze in Forward A^* maxG and Backward A^* maxG algorithms.

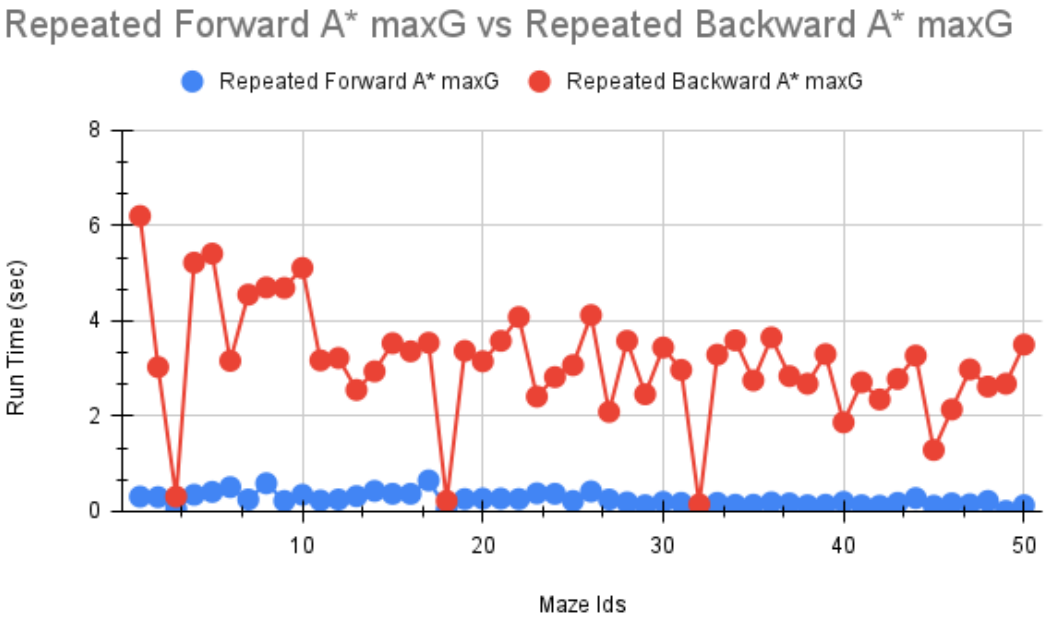

Figure 4: Repeated Forward A* maxG vs Repeated Backward A* maxG

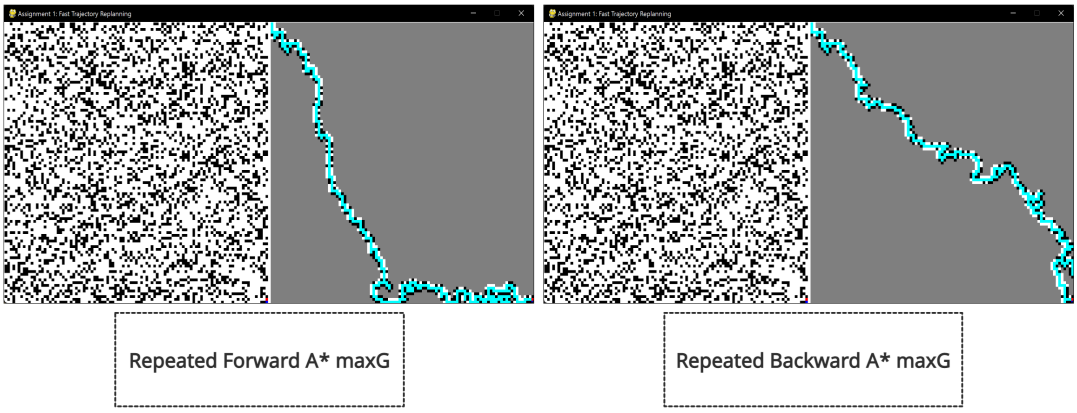

Figure 5: Path visualization in Repeated Forward A* maxG and Repeated Backward A* maxG

PART 4: Heuristics in Adaptive A^*

a) Proving Manhattan distances are consistent

Starting with the definition of the Manhattan distance: it is the 1-norm distance between the two points in the grid, and it is the shortest distance between any points. Assume we are at location *(i,j)* in the grid world. This location has a heuristic value $h_{(i,j)}$. The next step can be in any of the four compass directions that are, the next location when the agent moves can only be *(i+1, j)*, *(i-1, j)*, *(i, j+1)*, or *(i, j-1)* assuming they are valid steps. For our grid world problem, the step cost to move from one node to another is always 1. When a step happens, the heuristic of the new location either increases by one or decreases by one from the previous location. Since the step cost is always 1, this means that the sum of the step cost and new heuristic will always be equal to or one greater than the previous state heuristic. This means that $h_{(i,j)} <= step\_cost_{(new\_location)} + h_{(new\_location)}$. Hence, by the definition of consistency, we can say that Manhattan distances are consistent in grid worlds where the agent can only move in the four compass directions.

b) Proving Heuristic in Adaptive A^* is consistent

The heuristic in question is after the first iteration of A^*, we update the heuristic based on the following equation: h(location) = g(Goal) - g(location).

Consider the moves that an agent can make in Adaptive A^*. It is still the four compass directions. But this time, one of the moves would be a part of the path it came from. This means that the step cost is one, but the heuristic at that location would be one more than the heuristic at the current location since we are on a path that was achieved through that location and since the heuristic represents the steps remaining on the path calculated in the previous iteration. i.e $h_{\text{location}} <= step\_cost + h_{\text{new\_location}}$.

At least one of the other three moves left will take the agent farther from the goal, which was either not explored by the previous iteration or was explored and deemed to be more expensive. Either case, it would still lead to a higher h cost. The remaining one or two moves would be taking the agent closer to the goal and could have been on the previously calculated path. Again, in either case, the h cost would reduce by one, and since the step_cost is 1, it would remain equal to the current location heuristic cost.

PART 5: Heuristics in Adaptive A^*

Adaptive A^* vs Repeated Forward A^*

Adaptive A^* runs faster than Repeated forward A^* favoring maximum G values. We have observed that Adaptive A^* runs in almost half the time that Forward A^* runs for the same maze. This is because, after each iteration, the heuristic values for each previously explored location are better than when compared to Forward A^*.

Both start with the same path but diverge a lot during instances where the agent is blocked in multiple directions. Forward A^* explores all the locations around the blockage and starts to expand on the first location that gives a path, favoring the downward and rightward directions since they give the most optimal direction in our tests where the start point is the top left and the endpoint is the bottom right. Adaptive A^* on the other hand focuses solely on the number of moves it could possibly take to reach and during one iteration when it finds that a location is blocked, it ignores that location by assigning a higher heuristic value (since g goal would be very much high when going through such a location).

This also leads to less computation as the locations being expanded by Adaptive A^* have a higher likelihood of reaching the goal faster even in settings with partial information.

As shown below in Figure 6, the blue line graph shows the runtime for the adaptive A^* algorithm in favor of maximum G values and correspond-

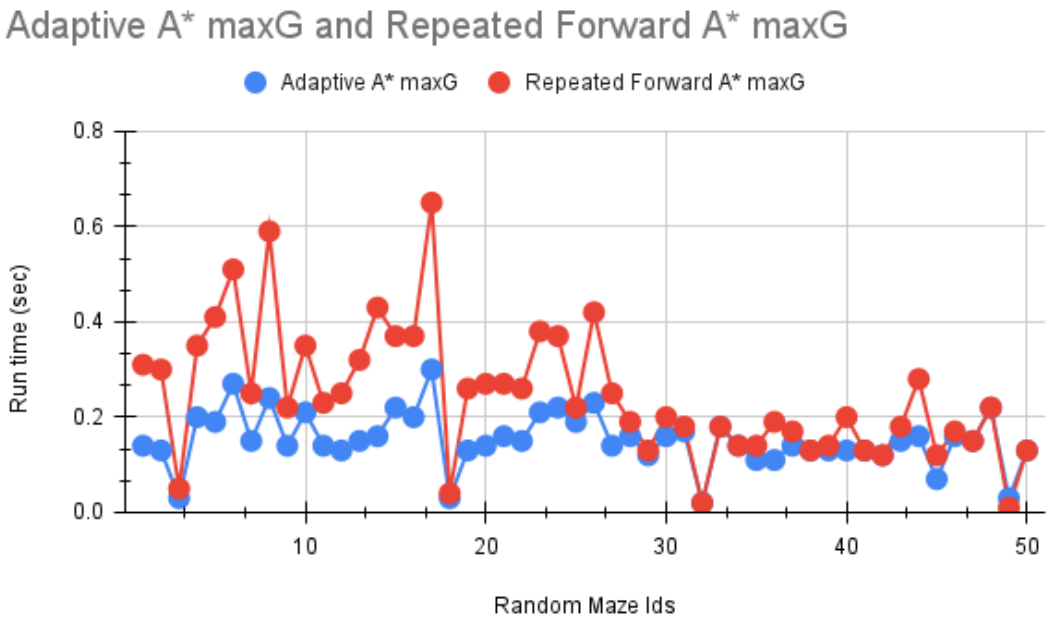

Figure 6: Adaptive A* vs Repeated Forward A*

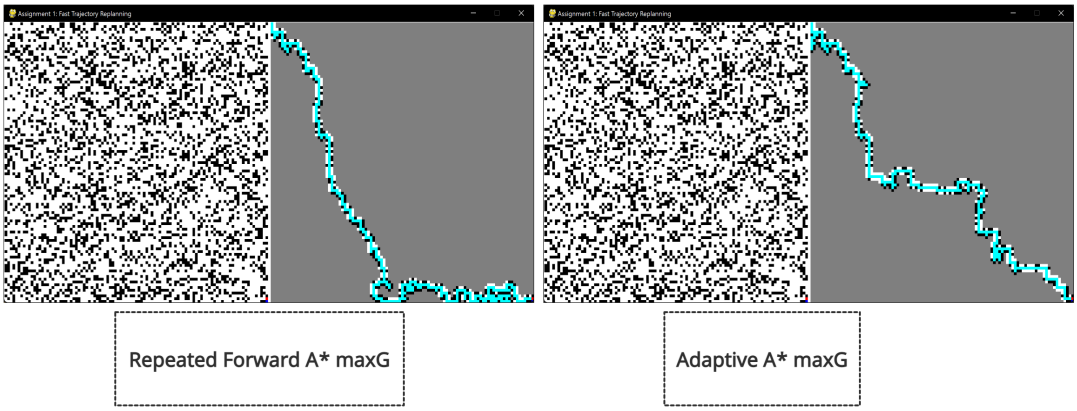

Figure 7: Path visualization in Repeated Forward A* maxG and Adaptive A* maxG

ingly, the red line shows runtime for the repeated forward A^* algorithm favoring maximum G values. Additionally, Figure 7 shows the path visualization on a random maze in Forward A^* maxG and Adaptive A^* maxG algorithms.

PART 6: Statistical Significance

Statistical Hypothesis Test for Adaptive A^* vs Repeated Forward A^*

We can perform a statistical hypothesis test for comparing two algorithms by collecting the mean (μ) and standard deviation (σ) for the runtimes obtained from the respective algorithms. Here, in this case, the number of samples is 50, as we are running the search algorithms on the 50 randomly generated mazes. Additionally, we can collect the number of expanded cells along with runtime and calculate the mean (μ) and standard deviation (σ) for the same. Later, the p-value (while performing the p-test) can be calculated to understand the degree of statistical significance, and therefore, the validity of the statistical hypothesis can be measured. We have performed a two-sample t-Test for the adaptive A^* and repeated forward A^* algorithms, and our null hypothesis (i.e., both algorithms have similar runtime) got rejected, and the p-value obtained was 2.64e-05. Hence, the difference in performance between the selected two algorithms is statistically significant. Similarly, this can be observed in the number of cells explored.

ACKNOWLEDGMENTS

- Picture credits for Rutgers Logo is taken from online search[1].
- Some LATEX libraries have been used from the IIT Madras Overleaf template available here[2].

[1] https://logos-world.net/wp-content/uploads/2022/01/Rutgers-University-Emblem.png

[2] https://www.overleaf.com/latex/templates/iitmdissertation-the-official-latex-tpbrrtcygybkj