Assignment1
Fast Trajectory Replanning

Nicholas-Schenk
Charles Li
Vibhu Ramanujam

October 22, 2021

Introduction

This assignment is about implementing variants of of A* algorithm in gridworld environment. This involves generating a 101 X 101 grid with maze like structure. This is created using depth-first approach. Start from a random cell, then go to a random neighbor that is not visited and mark them blocked with 30% probability and continue till all the cells are visited. The following variants of A* are implemented.

- Repeated Forward A* is implemented. In one version, if the f-value is tied, the tie is broken in favor of the larger g-value and in another version the tie is broken in favor of the smaller s-value. The performance of both these versions are compared.
- Implemented and performance compared Repeated Forward A* and Repeated Backward A* . The f-value tie is broken in favor of the larger g-value.
- Implemented Adaptive A*, an improved version of A* wherein the experience of the earlier searches is used to update the h-value. This is compared with Repeated Forward A*.

Part 1

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 know initially which cells are blocked.*

The agent doesn't know which cells are blocked when it starts to find the path. If it has full knowledge of the maze, then the best path is to the North and then go around the obstacle.

Since the agent doesn't know, it has four options South, West, North and East. The first step in A* will be towards the shortest unblocked path.

It cannot go South as E2 is the bottom cell.

The manhatten distance from the neighbors are:

From North (D2) = 4

From West (E1) = 4

From East (E3) = 2 (shortest path)

So, the first move from E2 the agent will make is to the East.

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 gridworlds 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.*

The agent will reach the goal as the number of grids is finite. In A*, there is a close list, so once it is expanded it will never go back to the open list. This will avoid infinite looping. If a cell is reachable from the start, the agent will visit them in a finite number of steps unless it is surrounded by blocked cells.

The algorithm terminates when it reaches the goal or when all the cells are visited. So, the algorithm is guaranteed to find a solution if it is not blocked. The worst-case scenario would be if all the cells are unblocked, and the agent start from an unblocked cell and visit and expand all the other unblocked cells. By visiting every cell, it is guaranteed to find the target if it is possible.

If the number of unblocked cells is 'n' in a certain scenario: In the worst case, A* will expand all n nodes. If A* start search from every unblocked cell (n) and the agent visits every other unblocked cell from every starting position it will expand n nodes n times.

So, the upper limit on the moves by the agent is number of unblocked cells squared (n2)

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, number of expanded cells. Explain your observations in detail, that is, explain what you observed and give a reason for the observation.*

Our Repeated Forward A* with larger g-values see on average 10.2% more steps to reach the target with 154% less cells expanded compared to lower g-values. In terms of runtime, larger g-values are 174% faster than smaller g-values. Larger g-values perform better because cells that have higher g-values encourage the agent to travel away from its current position. Of the set of cells with higher g-values it is often the case that when an agent moves to these cells, they get closer to reaching the target, which decreases those cells' heuristic cost. Lower g-value cells by comparison do not always encourage movement that brings the agent closer to the Target.
The majority of the performance differences are a result of how cells are stored in the min heap. In the case of larger g-values, cells that bring the Agent closer to the Target take greater priority in the min heap. In the case of smaller g-values, many cells that do not necessarily bring the agent closer may take precedence.

Part 3

Forward vs. Backward: Implement and compare Repeated Forward A and Repeated Backward A* with respect to their runtime or, equivalently, 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.*

In a large majority of generated grids, Repeated Forward A* sees better performance than Repeated Backward A* by a huge margin. On average Forward A* sees 6.9% greater steps to reach the target than Backward A*. However, Forward A* expands 191% fewer cells and runs in 198% less time. Backward A* involves so many more expansions than Forward A* because it expands cells from the Target location to the Agent and typically computes a path that naively assumes that areas around the Target are unblocked. This is mostly due to the agent not having reached near the target to inform the path compute function of blocked cells that would normally invalidate the computed path at each step. It is only until the Agent nears the Target and knows about unblocked cells throughout the grid when the expanded cells result in more valid paths for the Agent to follow. Forward A* does not face this issue as much since the expanded cells are relative to nearby cells of the agent in its current position on the grid. Thus, the compute path function can account for blocked cells that would invalidate a path much better compared to Backward A*.

Part 4

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

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

a A heuristic h(n) is consistent if, for every node n and every successor n' of n generated by any action a, the estimated cost of reaching the goal from n is no greater than the step cost of getting to n' plus the estimated cost of reaching the goal from n'. In other words, the heuristic function must underestimate the actual path cost and the following triangular inequality must hold.

h(n) $\leq$ step cost (to neighbor n') + h(n')

Manhattan distance is calculated by adding the horizontal and the vertical path. In the gridworld, Manhattan distance is always the same and it is the fastest path possible to reach the goal as the agent can move only up, down, left, or right. Here, the agent cannot move diagonally and so the agent will always find the shortest path. So, Manhattan distance is consistent. If the agent can go diagonally, the heuristic can overestimate the distance to reach the target.

To Prove that Adaptive A* is consistent:

Since, h-values are consistent, and h(s) follows Manhattan Heuristics.

h_{new}(s) = g(s_{goal}) – g(s). —— 1

c(s,a,s') – Step cost of going from s to s' using action a is ONE.

The triangular inequality says:

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

We have to prove h_{new}(s) $\leq$ h_{new}(s') + c(s,a,s')

Substituting (from 1) h_{new}(s) = g(s_{goal}) – g(s) and

h_{new}(s') = g(s_{goal}) – g(s') above ,

We get g(s_{goal}) – g(s) $\leq$ g(s_{goal})– g(s') + c(s,a,s')

Simplifying we get:

=> g(s) $\geq$ g(s') – c(s,a,s')

This is TRUE because:

- In Manhattan Heuristics c(s,a,s') is always ONE
- If g(s') is smaller than g(s), subtracting ONE from it will make it still smaller by ONE
- If g(s') is greater than g(s), subtracting ONE from it will make them equal

So, h-values h_{new}(s) are consistent.

b *Now, to prove that Adaptive A* leaves initially consistent h-values consistent even if action costs can increase:*

Let us again consider the triangular inequality:

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

Let there be a cost increase and c(s,a,s') be the cost before and c'(s,a,s') be the cost after the increase.

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

So, we can see that the heuristic is consistent with action cost increase.

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.*

Across 100 grids with dimensions 101 X 101, our Adaptive A* algorithm took, on average, 5255.8 expansions to reach the target. Forward A*, on the other hand, took an average of 43,915 expansions to reach the target, showing just how much more efficient Adaptive A* is.

It is unsurprising that Adaptive A* is much faster because it improves the h value after each iteration of A* by making a more informed guess. Adaptive A*'s estimate of the goal distance for a state is found by taking the goal distance of the start state and subtracting the distance from the start state to state s. As proven in the assignment write-up, this is admissible. Adaptive A* updates the heuristic value for all states expanded during an A* search and increases them, therefore makes them less likely to be visited in future A* searches. In all, this makes the Adaptive A* much more focused and more efficient. This updated heuristic also can only ever increase the h value of a cell, not decrease it. This guarantees that the algorithm never makes the agent more likely to revisit a cell during A* and only can help it or be neutral.

That being said, the agent does not know the entire grid at the start and therefore occasionally paths that seem better based on Adaptive A* actually end up being significantly worse. What we did notice when testing on smaller grids, however, was that often even if the agent took more steps to reach the target using adaptive A*, the algorithm still generally expanded less cells (i.e was still faster).

Part 6

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.

Null Hypothesis: H0: Repeated Forward A* = Adaptive A*: Meaning there is no systematic difference between the two algorithms
Alternate Hypothesis: H1: There is systematic difference. (inverse of the Null Hypothesis)
Steps for statistical hypothesis test:

1. Under the null hypothesis, the test assumes that two samples were drawn from the same experiment setup (population), meaning identical grid with same start and target.

2. Run experiments and record results. compute respective sample mean and sample standard deviation. Let delta (Δ) be the difference in their means.

3. One can choose either two sample z test (Reference) (for large sample size) or two sample t test (for small sample size) to test the significance of the difference between the two samples.

4. Find the distribution of Δ under the assumption that the null hypothesis is true, Forward A* = Adaptive A*

5. Use this distribution to find the probability p of Δ, assuming Forward A* = Adaptive A*

6. If the probability is very low (if p <.01), meaning 99% confidence, reject null Hypothesis H0: Forward A* = Adaptive A*

   (p value is the probability of incorrectly rejecting H_0)

7. p<.01 is the residual uncertainty that Forward A* might equal Adaptive A*

If experimental setup (population) standard deviation is unknown and samples are small, we can use the t test.

In this case we assume the sampling distribution to be t, not normal, but approaches normal as samples size increases. The test statistic has very similar form as normal, but probabilities of the test statistic are obtained by consulting tables of the t distribution, not the standard normal distribution.

$$t = \frac{(mean(forwardA^*) - mean(adaptiveA^*))}{\text{weighted average of the two sample standard deviations}}$$

The Z test (Reference) involves nothing more than standardizing the difference between (μ), the mean of the sampling distribution under the null hypothesis and the sample mean $\overline{x}$.

$$z = \frac{\overline{x} - \mu}{\sigma \overline{x}}$$

References

Empirical Methods for Artificial Intelligence. USC Information Sciences Institute © Paul Cohen, 2006
url = http://www.eecs.harvard.edu/cs286r/courses/spring08/reading6/CohenTutorial.pdf