CS 520: Introduction to Artificial Intelligence **(Due: 02/25/20)**

Assignment #1: Fast Trajectory Replanning

Instructor: Abdeslam Boularias
TA: Aravind Sivaramakrishnan

Name: Laxman Pottimuthi, *Netid:* lp642
Name: Sreeram Madineni, *Netid:* sm2323

Question 0: Maze Creation

- Maze was generated using a python library matplotlib

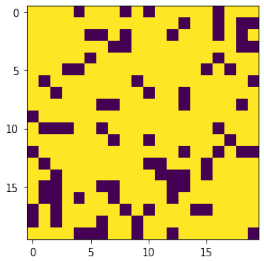

Figure 1: Sample Maze

- According to requirement, maze was generated with 30% probability that are marked. This can be seen in the method. Following is the code snippet:

```
def get_rand_value():
    rand_value = random.randint(0,9)
    if rand_value in [0,2]:
        return -1
    else:
        return 1
```

Question 1: Understanding Methods

Read the chapter in your textbook on uninformed and informed (heuristic) search and then read the project description again. Make sure that you understand A* and the concepts of admissible and consistent h-values.

(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.

Agent first move towards east rather than north:

- While calculating path, A* chooses optimal path based on the calculated f-value, where f = g + h.
- The cell on the east has an f value = 3 [1(g)+2(h)] and cell on the north has a f-value = 5 [1(g)+4(h)]
- Cell on the east will be chosen as it has least f value among both neighbors.
- As the cell doesn't know anything about the status of any blocked cells other than its neighbours it chooses to move in the east direction initially.

(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.

Agent is guaranteed to reach the target if it is not separated from it by blocked cells. Reasons:

- Assume that there is a path from source to destination. Initially A* algorithm calculates the path from source to destination and the agent traverses along that path. If there is a blocked cell along the path then a fresh A* algorithm runs with the new starting point adjacent to the blocked cell. Then the path is computed and the agent traverses along the new path. This can go on until the agent reaches the destination. It cannot go in an infinite loop. The reason is as follows . Assume that there is a node n which has K paths to the destination. When the agent starts at node n it takes any of the k paths to the destination. If it is unable to reach the destination(due to a blocked cell) and if at any later point it reaches the node n then it takes any of the remaining k- 1 paths to the destination. It cannot take the path it traversed before as the agent remembers that there are blocked cells along that path , therefore its new path would be different. So the agent will traverse each path only once between a given source and destination. So it will keep traversing through every path between any starting cell and the destination until it reaches the destination.
- Assume that a path doesn't exist between the source and destination due to various blocked cells . There would a particular iteration in Repeated A* when the agent cannot compute the path from the source(which can be different than the starting source) to destination. During this iteration the open list becomes empty. This means there is no node left through which the agent can travel to destination. Then the program exits and the the agent is not able to destination.
- Assume that there are K unblocked nodes in the maze. Any of the K nodes can be starting point and each of the nodes can be a starting point of an A* iteration only once. This is true as the agent will be blocked at a node only once throughout its entire journey. It cannot be blocked at that node again as when it was blocked for the first time it will know about its neighbours and it will make sure to not get blocked there again when it is computing path. So from each of the K nodes as starting points of A* algorithm the agent can make at most K moves (since the total number of unblocked nodes is K). So the total number of moves the agent makes in its entire journey has upper bound of K*K. SO the total number of moves it makes is O(K^2)

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

- Assume the tie breaks' are done using a priority, where priority = constant*f -/+ g.
- If larger g-value tie break is used, then the priority is constant*f - g and if tie break is based on smaller g-value, priority is constant*f+g.
- If the tie break is based on larger g value, the cell is away from the starting point and if the g value is less, it means the cell is closer to start point.
- If the ties are broken based on smaller g value, we end up expanding more number of unnecessary cells near the starting point whereas if ties are broken using larger g value we expand cells that are away from the starting point and closer to destination which would let us find the destination quickly.
- As the performance is dependent on number of expanded cells, expanding cells with greater g-value is considered for better performance.

Figure 2: Repeated Forward A^*: Tie with larger G v smaller G

Question 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.

- Repeated Backward A* expands more number of nodes and thus it runs slower than Repeated Forward A*.
- The reason is in the case of Repeated Backward A* the algorithm computes the path from destination to the source.
- But the agent knows more information(whether the cells are blocked or not) about the cells which are near the source and it has less information about cells which are near destination. So more cells near the destination are likely to be expanded while computing the path.
- Also whenever the agent travels from the source to a blocked cell it only gathers information about the neighboring cells along the path, it is very unlikely that it would gather any information about cells which are near the destination.
- So this will cause unnecessary expansion of cells near the destination during path computation for every iteration of Repeated Backward A*.

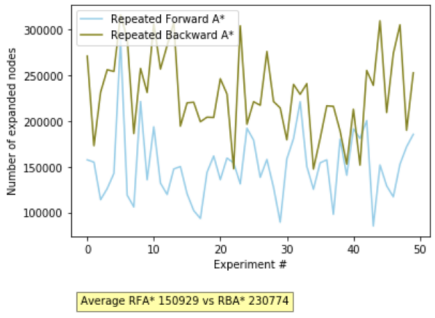

Figure 3: Repeated Forward A^* vs Repeated Backward A^*

Question 4: Heuristics in the Adaptive A*: Consistent and admissible

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. Manhattan Distance consistent

- h-values h(s) are consistent iff they satisfy the following triangle inequalities for all states s with s $\neq s_{goal}$ and all actions a that can be executed in state s.

$$h(s_{goal}) = 0 \tag{0.1}$$

$$h(s) \leq c(s,a) + h(succ(s,a)) \tag{0.2}$$

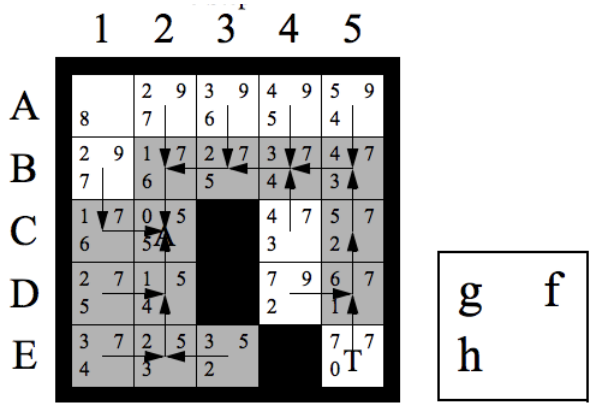

Figure 4: Forward A*

- Assume our goal is to move from A5 to E5. We do this by moving one step at a time with following pattern $A5- > B5- > C5- > D5- > E5$. In the gridworld, movement can happen only in the four compass directions and the cost from moving from one gridcell to any immediate cell is same.

$$c(s,a) = c(s+1,a) = c(s+2,a) \tag{0.3}$$

- We can prove Manhattan distance is consistent using induction. From 0.2, let us assume it is true for s, we will get the following

$$h(s) \leq c(s,a) + h(s+1) \tag{0.4}$$

And we need to show this inequality holds for s+1

$$h(s+1) \leq c(s+1,a) + h(s+2) \tag{0.5}$$

- Cost of moving from s to s+1 is c(s,a) and cost of moving from s+1 to s+2 is c(s+1,a). In the gridworld Manhattan distance between two points is sum of unit distances.

- Let us assign coordinates to points to calculate distances.
A5 - (i,j), B5 - (i, j+1), C5 - (i, j+2), E5 - (i, j+4)

- Hence,
Distance from A5 to E5 is $h(s) = abs(i-i) + abs(j-j-4) = 4$
Distance from B5 to E5 is $h(s+1) = abs(i-i) + abs(j+1-j-4) = 3$
Distance from C5 to E5 is $h(s+2) = abs(i-i) + abs(j+2-j-4) = 2$

- From above, we can say

$$\text{h(s)} = 1 + \text{h(s+1)} \text{ and } \text{h(s+1)} = 1 + \text{h(s+2)}$$

From 0.4

$$\begin{aligned} & h(s) \leq c(s,a) + h(s+1) \\ \implies & 1 + h(s+1) \leq c(s,a) + 1 + h(s+2) \\ \therefore & h(s+1) \leq c(s+1,a) + h(s+2) \end{aligned}$$

Hence proving 0.5, Manhattan distance consistency is proved by induction.

b. The h-values h_{new}(s) ... are not only admissible but also consistent.

- It is mentioned that initially Manhattan distance is consistent. We need to prove that new h values are also consistent
- We have the below from how we calculate h_{new}

$$h_{new}(s) = g(s_{goal}) - g(s) \tag{0.6}$$

$$h_{new}(s+1) = g(s_{goal}) - g(s+1) \tag{0.7}$$

- Subtracting 0.6 from 0.7,

$$\begin{aligned} h_{new}(s) - h_{new}(s+1) &= g(s_{goal}) - g(s) - (g(s_{goal}) - g(s+1)) \\ h_{new}(s) - h_{new}(s+1) &= g(s_{goal}) - g(s) - g(s_{goal}) + g(s+1) \\ h_{new}(s) - h_{new}(s+1) &= g(s_{goal}) - g(s_{goal}) + g(s+1) - g(s) \end{aligned} \tag{0.8}$$

- As s+1 and s are neighboring cells,

$$g(s+1) - g(s) = 1 \tag{0.9}$$

- Substituting 0.9 in 0.8,

$$h_{new}(s) = 1 + h_{new}(s+1) \tag{0.10}$$

- In 0.10, it is of the form,

$$h(s) \leq c(s,a) + h(s+1)$$

- In the considered maze we assumed cost from s to s+1 is one. So c(s,a) = 1
- Therefore new Manhattan distance(h_{new}) is also a consistent heuristic.

Question 5: Heuristics in the Adaptive A*

Implement and compare Repeated Forward A* and Adaptive A* with respect to their run-time. 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.

- On an average, Adaptive A* is much more efficient than A*.
- 50 different experiments are tested for both the algorithms under similar conditions, i.e, maze, start and goal.
- Performance of the algorithms are calculated by number of expanded nodes in each algorithm and averaging in the sample data.
- If the number of nodes expanded are less, it means algorithm performed better.
- The result of the sample experiments: the average number of nodes expanded for Adaptive A* is 134181 nodes and of repeated forward A* is 140608 nodes.
- Adaptive A* on average is better than Repeated forward A*
- This performance difference is due to updating the h-value of previous iteration's expanded nodes in the current iteration.
- Adaptive A* h-value heuristic for the cells is a much better estimate of distance to the destination when compared to Forward A*.
- While calculating the heuristic in the case of Forward A*(is Manhattan distance and) it doesn't take data about blocked cells into consideration(since it doesn't not have it) to update the heuristic of any cell.
- But Adaptive A* updates the heuristic function of every expanded cell in a particular iteration and use these heuristic values in the next iteration.
- These new heuristic values give a better estimate of distance to destination leading to less number of expanded nodes.

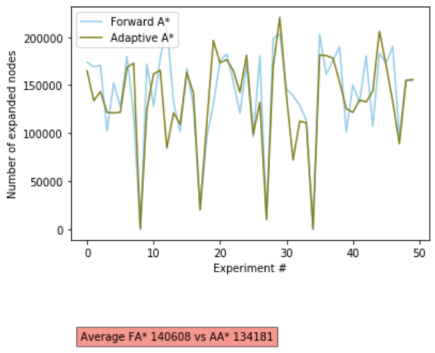

Figure 5: Adaptive A^* vs Forward Repeated A^*

Question 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.

- The following is the procedure to determine the statistical significance of Repeated A* and Adaptive A* algorithms.
- In order to perform, we need to choose a statistical hypothesis test to determine the statistical significance of the experiment. We can compare the performance difference of both the search algorithms based on number of expanded cells in the map.
- As we are considering two different experiments with 50 random sample data set, two sample tests could be chosen. Different type of hypothesis test statistics such as t-test and z-test can be used to determine the two sample test.
- Considering T-test as example, two sample T-test is used validating the hypothesis. As a comparison between experiments are made, one tailed test is used to accept/reject the hypothesis.
- In the following, AA* is adaptive A* test sample and RA* is Repeated forward A* test sample.

$$H_0 : AA^* = RA^* \tag{0.11}$$

$$H_1 : AA^* > RA^* \tag{0.12}$$

$$T_{stat} = \frac{AA^* - RA^*}{\sigma_{AA^*-RA^*}} \tag{0.13}$$

$$\sigma_{AA^*-RA^*} = \sqrt{\frac{(n_{AA^*}-1)s^2_{AA^*} + (n_{RA^*}-1)s^2_{RA^*}}{n_{AA^*}+n_{RA^*}-2}(\frac{1}{n_{AA^*}} + \frac{1}{n_{RA^*}})} \tag{0.14}$$

$$df = n_{AA^*} + n_{RA^*} - 2 \tag{0.15}$$

where

H_0: Null Hypothesis - Both the experiments perform equally good
H_1: Alternative Hypothesis - Adaptive A* gives better performance compared to Repeated A*
$\sigma_{AA^*-RA^*}$: Standard error of difference between AA* and RA^*
s_{AA^*} : Standard error of Adaptive A*
s_{RA^*}: Standard error of RA^*
n_{AA^*} : No. of samples in first algorithm
n_{RA^*}: No. of samples in second algorithm
df : Degrees of freedom

- A confidence interval should be chosen to perform the test. Using the confidence interval, significance of the test is calculated. Based on the type of the t-test(In this case, it's a 1-tailed T-test as mentioned above), critical values are calculated from the T-test statistic table.
- For example, test is conducted at 95% confidence. Critical values are calculated based on the confidence level and degrees of freedom. If the T_{stat} is greater than critical value, Null Hypothesis is rejected. Thus it can be confirmed that the alternative hypothesis is accepted with 95% confidence. It means that AA* will be outperforming RA*.

REFERENCES

- **Adaptive A* by Sven Koenig and Maxim Likhachev**
- **Consistent heuristic**
- **Admissible Heuristic**