Homework 1

Dhruvil Patel, Brandon Yu

February 28, 2019

Abstract:

In this project, we have implemented Forward A*, Backward A* and Adaptive A* path-planning in relation to video game technologies to observe their behaviors in several maze configurations. Python has been used as a primary language to write the code and to "pygame" framework to generate the environment. We have used threads in this project to maintain the execution and implementation of algorithms.

How to Use:
Options: 1=Forward A* ; 2=Backward A* ; 3=Adaptive A*
Run: python3 main.py 1

1 UNDERSTANDING THE METHODS

1.1 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.

In the first search of repeated A* search, the agent only knows that its neighbor cells (E1, D2, and E3 in figure 1(a)) are not blocked. The f-values of neighbor cells of start cell are f(E1) = 5, f(D2) = 5, f(E3) = 3. A* algorithm always chooses the neighbor which has smallest f-value, thus it will expand cell E3. The final path found by the first search is E2→E3→E4→E5. Therefore, the agent will move to the east first.

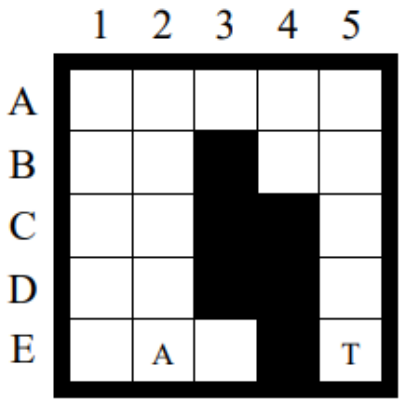


Figure 8: Second Example Search Problem

1.2 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.

Give a convincing argument that the agent in finite gridworlds indeed either reaches the target or discovers that this is impossible in finite time.

- Case 1: The target is NOT separated by blocked cells. Each time the agent finds a blocked cell, A* search algorithm can find a path from the current cell to target cell because the algorithm is complete. Since the grid-world is finite, it will reach the target.
- Case 2: The target IS separated by blocked cells. Each time the algorithm finds a path, the agent will at least move one step before it finds another blocked cell and knows more about the grid-world. In the worst case, the agent will know the entire grid-world in the finite time since the grid-world is finite. If A* search algorithm can't find a path to the target, the agent will discover that it is impossible to reach the target.

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.

- Case 1: The target is NOT separated by blocked cells. If there is one path to the target state, the path has at most n - 1 states, where n is the number of unblocked cells. In the worst case, every time the agent moves, it finds a block in the intended path, which implies that the A* is required to run n-1 times at most. In each A* search, the intended path won't include the same cell because of the optimality of search. Thus, the number of total moves is at most $(n - 1)^2$ and less than n^2.
- Case 2: The target IS separated by blocked cells. Here, there isn't a path to the target state. The agent should find all the unblocked states that the start state can reach and the number of which is bounded by n - 1 before discovering that the target state cannot be reached. The number of moves required to find an

unblocked state is at most n - 2. Since (n - 1)*(n - 2) < n^2, the number of moves is bounded by the number of unblocked states squared.

2 The Effects of Ties

2.1 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. [Hint: For the implementation part, priorities can be integers rather than pairs of integers. For example, you can use c × f(s) - g(s) as priorities to break ties in favor of cells with larger g-values, where c is a constant larger than the largest g-value of any generated cell. For the explanation part, consider which cells both versions of Repeated Forward A* expand for the example search problem from Figure 9.]

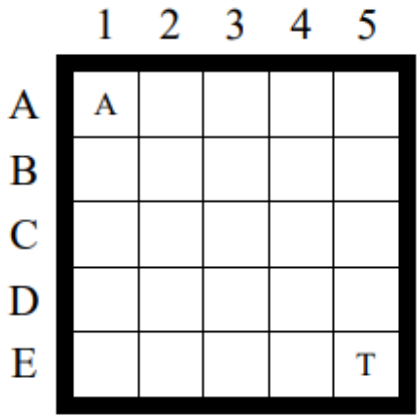
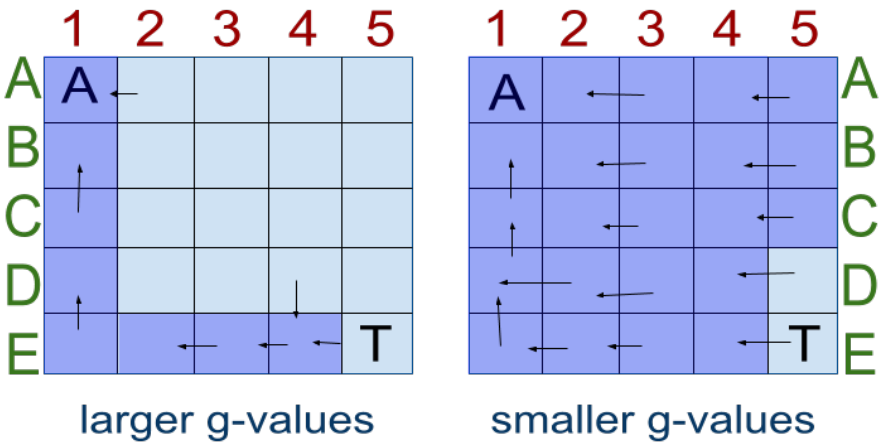


Figure 9: Third Example Search Problem

On average, breaking ties by choosing the applicant with a greater g-value expands fewer cells than the approach of choosing the one with a smaller g-value. Below figures give the searching results of A* in favor of larger and smaller g-values. Let us expand the neighbors in the order of south, east, north and west (counter-clockwise) when values of both f and g are equal. So, smaller g-values will make A* go back many times to expand the cells closer to the start cell, which costs expansions of much more additional cells oppose to smaller g-values A* search. Therefore, the A* in support of larger g-values is better.



3 FORWARD VS. BACKWARD

3.1 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.

During each A* search, the agent only knows some blocks lying in the field near the start cell and it assumes there are no blocks in other areas between the start and target cells. For backward A* , before it reaches the blocks near the start cell, the f-values of cells in the open list are all equal to the Manhattan distance between the start and target states, which is also the smallest f-value during this search. The backward A* will keep expanding all the states with the smallest f-value and then it will expand states with larger f-values. However, forward A* will apparently increase its f-value in the beginning and then keep the same f-value until the target cell. There will be much fewer states with smaller f-values in the open list when the f-value of the current cell would increase and let blocks in near area to be unsearched. Therefore, forward A* will save several more expansions than backward A*

4 HEURISTICS IN THE ADAPTIVE A*

4.1 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 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.

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.

Let's say, h(m) is defined as the Manhattan distance between state m and mgoal. Now, xm and ym are the indexes of rows and columns for state m.

- If m = mgoal, then h(mgoal) = |xmgoal - xmgoal | + |ymgoal - ymgoal | = 0
- If m != mgoal, let n = s(m, a) for simplicity, then h(m) - h(n) = |xm - xmgoal | + |ym - ymgoal | -|xn - xmgoal | - |yn - ymgoal |

Since the agent can only move in four main compass directions and m and n are happens to be the neighbors in grids, ⇒ xm = xn or ym = yn

- If xm = xn, then |ym - yn| = 1
  Thus, h(m) - h(n) = |ym - ymgoal | - |yn - ymgoal | ≤ |ym - yn| = 1
- If ym = yn, then h(m)-h(n) ≤ |xm -xn| = 1

In fact, in both cases, h(m)-h(n) ≤ 1 = c(m, a). For instance, h(m) ≤ h(s(m, a))+c(m, a). Hence, it proves that the Manhattan distances are consistent in gridworlds in which the agent can move only in 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.

- Case 1: m and n were expanded: Here, both heuristics of state m and n are as follows: hnew(m) = g(mgoal) - g(m) hnew(n) = g(mgoal) - g(n)
  With cost g(m) + c(m, a) A* search finds the current state via state m to state n. In fact, there's no guarentee that the cost will be the final g-value of state n because there might be a shorter path. Hence,
  ⇒ g(n) ≤ g(m)+c(m, a)
  ⇒ hnew(m) = g(mgoal)-g(m) ≤ g(mgoal)- g(n)+c(m, a) = hnew(n)+c(m, a).
- Case 2: m was expanded: Here, the h-value of m is the new defined one, i.e. hnew(m) = g(mgoal) - g(n). However, state n is still the Manhattan distance to the target state [hnew(n) = h(n)]. Besides, state n was generated but not expanded, which means the f-value of n is not smaller than that of m,
  ⇒ hnew(m) = g(mgoal) - g(m) = f(mgoal) - g(m) ≤ f(m) - g(m) ≤ f(n)-g(m) = g(n)+h(n)- g(m) = g(n)+hnew(n)-g(m) ≤ g(n)+hnew(n)-g(n)+c(m, a) = hnew(n)+c(m, a)
- Case 3: m was not expanded: Here, hnew(m) = h(m). Besides, the heuristics of the same state won't decrease over time, so we have h(n) ≤ hnew(n) ⇒ hnew(m) = h(m) ≤ h(n) + c(m, a) ≤ hnew(n) + c(m, a). Therefore, the new h-values are consistent even if the action costs can increase.

5 HEURISTICS IN THE ADAPTIVE A*

5.1 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.

The adaptive A* algorithm always outperforms the forward A* algorithm, in terms of the total number of expanded cells during the search. In the nth adaptive A*, the h-values of states in the closed list of the (n-1)th adaptive A* are changed to the real path lengths from the target state. The fewer expansions in the adaptive A* benefit from the usage of the new h-values which would reflect their true distances to the target. If the possible paths with the lengths of Manhattan distance are all blocked then a state with a small Manhattan distance to the target state might have a larger new h-value. Also, the adaptive A* expands no more states than the repeated forward A*. The block distribution along the path of adaptive A* might cause adaptive A* to repeat more times afterward than forward A*

6 MEMORY ISSUES

6.1 You performed all experiments in gridworlds of size 101 × 101 but some real-time computer games use maps whose number of cells is up to two orders of magnitude larger than that. It is then especially important to limit the amount of information that is stored per cell. For example, the tree-pointers can be implemented with only two bits per cell. Suggest additional ways to reduce the memory consumption of your implementations further. Then, calculate the amount of memory that they need to operate on gridworlds of size 1001 × 1001 and the largest gridworld that they can operate on within a memory limit of 4 MBytes.

One can only keep the information of the open and/or blocked cells in memory to save some memory. Also, a linked list or hash mapped array can be used to do insert, delete, search, etc operations for each cell. In this case, we can do 62-64 bytes per cell which is 120-126 MB for a grid size of 1001*1001.