CS440 (Intro to AI) - HW1

Janine Yanes, Aadi Gopi, Jason Cai

February 28, 2025

0 Environment setup

To build our environments, we used the assignment description's method:

1. Initially set all of the cells as unvisited.
2. Start from a random cell; mark it as visited and unblocked. Add this cell to the stack.
3. Visit a random unvisited neighbor. Mark it as blocked with 30% probability; otherwise, mark it as unblocked and add it to the stack.
   (a) If a dead-end (a cell with no unvisited neighbors) is reached before all nodes are visited, backtrack through the previously visited cells (pop from the stack) until a cell with an unvisited neighbor is reached.
      i. If the stack becomes empty, visit a random unvisited node. Mark it as unblocked and add it to the stack.
4. Repeat step 3 until every cell has been visited.

For visualization, we used black for blocked cells and white for unblocked cells.

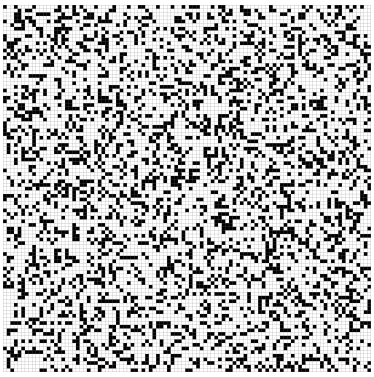

Figure 1: Example of a generated 101x101 grid world

1 Understanding the Methods

a) Using A* search, the agent moves to the unblocked cell s with the smallest f-value $f(s) = g(s) + h(s)$, where $g(s)$ is the distance between the start and s and $h(s)$ is the estimated distance between s and the goal (based on the Manhattan distance).
$g(s) = 1$ for each of the start's neighbors, so their f-values depend on $h(s)$. Since the east cell has the lowest Manhattan distance, the agent moves east instead of north.

b) The agent systematically explores the grid using the A* search algorithm, which means that any reachable unblocked cells are visited. Since the grid worlds are finite, there are a limited amount of cells that are both unblocked and blocked; this means that there can't be an infinite search time as long as no loops are apparent. We prevent the possibility of loops through the implementation of closed lists to track the whereabouts of the agent. If the goal is blocked off, the agent will eventually explore all possible ways to reach unblocked cells and then determine that there is no path to said goal.
Let n be the finite number of unblocked cells. Since the agent has limited knowledge of which cells are unblocked, it can travel down dead ends, forcing it to revisit cells. However, even if at each cell, the agent was forced to go through every other cell, the total number of moves would be $n(n-1)$. Therefore, the upper bound for the total number of moves is n^2.

2 The Effects of Ties

We found that the version of Repeated Forward A* that broke ties in favor of cells with larger g-values generally had a faster runtime and expanded less cells than the version that broke ties in favor of cells with smaller g-values.
This has to do with the cells' h-values: the true distance to the goal from cell s is always $\geq h(s)$ (since $h(s)$ doesn't account for blocked cells), so a lower $h(s)$ value indicates a lower minimum distance to the goal. If two f-values $f(s_1)$ and $f(s_2)$ are equal, but $g(s_1) > g(s_2)$, that must mean that $h(s_1) < h(s_2)$ since $f(s) = g(s) + h(s)$. Therefore, ties should be broken in favor of cells with larger g-values since there's a possibility of traveling less distance/exploring less cells on the way to the goal.

3 Forward vs. Backward

We found that between Repeated Forward A* and Repeated Backward A*, one was not consistently faster than the other. To be more specific, the algorithms' relative speeds were highly dependent on which cells were blocked. With Forward A* going from start to goal, it starts with more information on the cells around the start than Backward A*. As a result, it can avoid obstacles around

the start more easily than Backward A*. Conversely, since Backward A* starts with more information on the cells around the goal, it might easily avoid some obstacles at the start of its search that Forward A* struggles with later on.

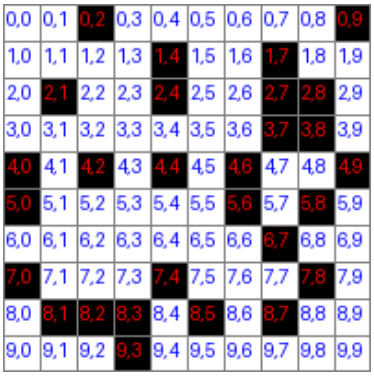

Figure 2: A 10x10 grid world, with coordinates labeled

```
------------
start: (9, 5) end: (4, 7)
Larger g-values forward path: No path found
65 cells expanded: [(9, 5), (9, 6), (8, 6), (7, 6), (6, 6), (7, 7), (9, 7), (6, 5), (5, 5), (4, 5), (7, 5), (9, 8),
 (8, 8), (9, 4), (8, 4), (3, 5), (3, 6), (5, 4), (6, 4), (8, 9), (7, 9), (6, 9), (5, 9), (6, 8), (9, 9), (2, 6), (2
, 5), (3, 4), (5, 3), (4, 3), (6, 3), (1, 6), (1, 5), (3, 3), (5, 2), (6, 2), (7, 3), (0, 6), (0, 7), (0, 5), (2, 3
), (3, 2), (5, 1), (4, 1), (6, 1), (7, 2), (0, 8), (1, 8), (0, 4), (1, 3), (2, 2), (3, 1), (6, 0), (7, 1), (1, 9),
(2, 9), (3, 9), (0, 3), (1, 2), (3, 0), (1, 1), (2, 0), (0, 1), (1, 0), (0, 0)]
Larger g-value forward runtime : 0.0005064010620117188 seconds
Larger g-values backward path: No path found
3 cells expanded: [(4, 7), (5, 7), (4, 8)]
Larger g-value backward runtime: 5.3882598876953125e-05 seconds
------------
```

Figure 3: Comparison of Forward A* and Backward A* runtime for Figure 2

A clear example of this is shown using Figures 2 and 3: since the cells close to the goal are blocked, Backward A* discovers the obstacles and determines that there is no path much more quickly than Forward A*.

4 Heuristics in the Adaptive A*

4.1 Manhattan distance consistency

The heuristic $h(s)$ is consistent iff:

- $h(s_{\text{goal}}) = 0$
- For every successor s' (generated by action a) of every cell $s \neq s_{\text{goal}}$, $h(s) \leq c(s, a, s') + h(s')$, where $c(s, a, s')$ is the step cost of getting to s'

In this case, $h(s)$ is the cell's Manhattan distance: the sum of the absolute difference of the x coordinates and the absolute difference of the y coordinates of the cell and the goal, $|x_s - x_{\text{goal}}| + |y_s - y_{\text{goal}}|$. This means that $h(s_{\text{goal}}) = |x_{\text{goal}} - x_{\text{goal}}| + |y_{\text{goal}} - y_{\text{goal}}| = 0$, fulfilling the first condition.
Since the only actions the agent can take at cell s are singular steps in one of

the four cardinal directions to one of the cell's unblocked neighbors (successors), $c(s, a, s')$ will always equal 1. Also, the coordinates of every successor s' will either be $(x_s - 1, y_s), (x_s + 1, y_s), (x_s, y_s - 1)$, or $(x_s, y_s + 1)$. In other words, $h(s') = h(s) \pm 1$, since s' is either one coordinate unit closer to or one coordinate unit further from the goal than s. $h(s) \leq 1 + (h(s) \pm 1)$, fulfilling the second condition.
Therefore, the Manhattan distances are consistent in grid worlds where the agent can move only in the four cardinal directions.

4.2 Adaptive A* h-value consistency

The heuristic $h_{\text{new}}(s)$ is consistent iff:

- $h_{\text{new}}(s_{\text{goal}}) = 0$
- For every successor s' (generated by action a) of every cell $s \neq s_{\text{goal}}$, $h_{\text{new}}(s) \leq c(s, a, s') + h_{\text{new}}(s')$, where $c(s, a, s')$ is the step cost of getting to s'

In this case, $h_{\text{new}}(s) = g(s_{\text{goal}}) - g(s)$, so $h_{\text{new}}(s_{\text{goal}}) = g(s_{\text{goal}}) - g(s_{\text{goal}}) = 0$, fulfilling the first condition.
Since the only actions the agent can take at cell s are singular steps in one of the four cardinal directions to one of the cell's unblocked neighbors (successors), $c(s, a, s')$ will always equal 1. Also, the coordinates of every successor s' will either be $(x_s - 1, y_s), (x_s + 1, y_s), (x_s, y_s - 1)$, or $(x_s, y_s + 1)$. In other words, $g(s') = g(s) \pm 1$, since s' is either one coordinate unit closer to or one coordinate unit further from the start than s. This means that

$$c(s, a, s') + h_{\text{new}}(s') = 1 + [g(s_{\text{goal}}) - (g(s) \pm 1)] = 1 + (h_{\text{new}}(s) \pm 1)$$

. $h_{\text{new}}(s) \leq 1 + (h_{\text{new}}(s) \pm 1)$, fulfilling the second condition.
Therefore, $h_{\text{new}}(s)$ is consistent, even if action costs can increase.

5 Heuristics in the Adaptive A*, cont.

When comparing the two programs, we can see that Adaptive A* runs more consistently faster and expands less cells than Repeated Forward A*.
This is because Adaptive A* uses the cost from the current cells to the goal, based on the pathfinding from the previous searches. In other words, it takes into account obstacles encountered in previous searches. On the other hand, Repeated Forward A* doesn't account for any obstacles, as its heuristic, Manhattan distance, is the cost of the best path from start to goal if there are no unblocked cells; this leads to more searches and exploration of dead ends.

6 Statistical Significance

Let there be two search algorithms, $f_1(s)$ and $f_2(s)$, each with their own runtime frequency distributions. Let μ_1 be the average runtime for algorithm $f_1(s)$ and

μ_2 be the average runtime for algorithm $f_2(s)$. We can use a two-sided t-test to determine if there is a statistically significant difference between the two average runtimes.

1. $H_0 : \mu_1 = \mu_2,\ H_A : \mu_1 \neq \mu_2$
2. Determine a sufficient cutoff α; a common α is 0.05.
3. Test both algorithms using the same number of 101x101 grid worlds n.
4. Find the sample means (the mean runtime for the algorithms when applied to the grid worlds), $\bar{x}_1$ and $\bar{x}_2$, and the sample standard deviation $s_p = \sqrt{\frac{s_1^2+s_2^2}{n}}$, where s_1^2 and s_2^2 are the sample variances for their respective algorithms.
5. Calculate the observed test statistic value $t^{obs} = \frac{\bar{x}_1-\bar{x}_2}{s_p}$.
6. Using the t-distribution with $2n-2$ degrees of freedom, find the probability $p = P(|t^{obs}| > t_{\alpha,2n-2})$
    - If $p < \alpha$, we reject the null hypothesis: the test indicates the difference in runtime between the two algorithms is systematic in nature, rather than sampling noise.
    - If $p \geq \alpha$, we fail to reject the null hypothesis: we do not have sufficient evidence that any difference in runtime between the two algorithms is systematic in nature.