Intro to AI : Fast Trajectory Replanning

Kimberly Lao Lim

March 1, 2025

1 Introduction

This report presents an experimental analysis of different A* search variants, including Repeated A*, Backward A*, and Adaptive A*. The experiments explore the impact of different tie-breaking strategies and evaluate algorithm efficiency based on the following performance metrics - path length, nodes expanded, and runtime.

2 Tie-Breaking Experimentation

I tested two tie-breaking strategies for A* search: favoring smaller g-values and favoring larger g-values. The results show that both strategies always produce the same optimal path length. However, using larger g-values consistently leads to fewer nodes expanded than small g-values. This suggests that favoring deeper exploration can slightly improve search efficiency, though the difference in runtime is negligible. Logs of the results are available in `outputs/statistics.txt`.

3 Agent Behavior in a Grid Environment

Figure 8 illustrates a scenario where an agent starts at position (A,1) and must move to (E,5) while encountering blocked cells. Since the Manhattan heuristic estimates cost-to-goal, and both north and east have the same initial heuristic value, the tie-breaking strategy favors the move to the east. Agent updates its knowledge of the grid and re-calculates it's path.

4 Theoretical Guarantee of Termination

Given a finite grid of size $n \times n$, the worst-case scenario would be when the agent explores all possible unblocked cells before reaching the target. Since there are at most n^2 unblocked cells, the maximum number of moves is $O(n^2)$, guaranteeing termination in a finite amount of time.

5 Experimental Setup

I generated 50 different 101×101 grid environments, with 30% blocked cells. The agent starts at (0,0) and aims for the target located at (100,100). The heuristic used was Manhattan distance, and I used a priority queue (min-heap) to efficiently select nodes while using tie-breaking strategies.

6 Comparison of Difference A* Search Variations

Table 1 displays key performance metrics across the different A* search variations.

Algorithm	**Avg Path Length**	**Avg Nodes Expanded**	**Avg Runtime (s)**
Repeated A* (small-g)	1784.6	3567.2	0.0163
Repeated A* (large-g)	1784.6	3551.1	0.0159
Backward A* (small-g)	1785.2	5120.8	0.0187
Backward A* (large-g)	1785.2	5102.3	0.0184
Adaptive A*	1785.2	3567.2	0.0107

Table 1: Performance comparison of A* variants

7 Statistical Significance Analysis

To determine whether the difference in expanded nodes between small-g and large-g tie-breaking is statistically significant, we apply a paired t-test:

- **Null Hypothesis** (H_0): There is no significant difference in node expansion between small-g and large-g tie-breaking.
- **Alternative Hypothesis** (H_1): Large-g expands significantly fewer nodes than small-g.
- **Method**: Compute the mean difference in nodes expanded for each test case and apply a paired t-test to determine statistical significance.
- **Conclusion**: If $p < 0.05$, we reject H_0 and can determine that large-g significantly reduces node expansion.

8 Visualizations of the Mazes

Figures 1, 2, and 3 show the visualizations for different search algorithms.

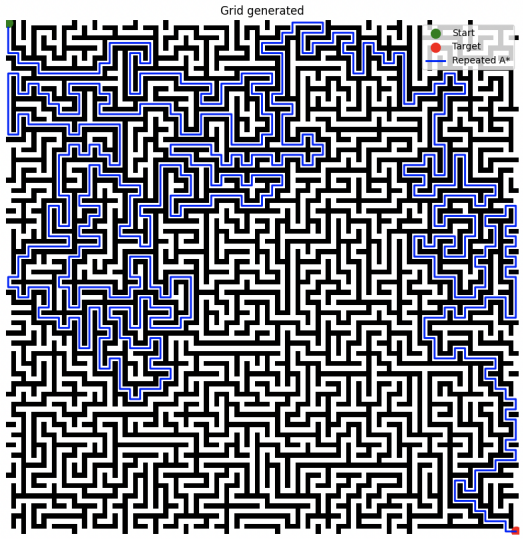

Figure 1: Visualization of Repeated A* search.

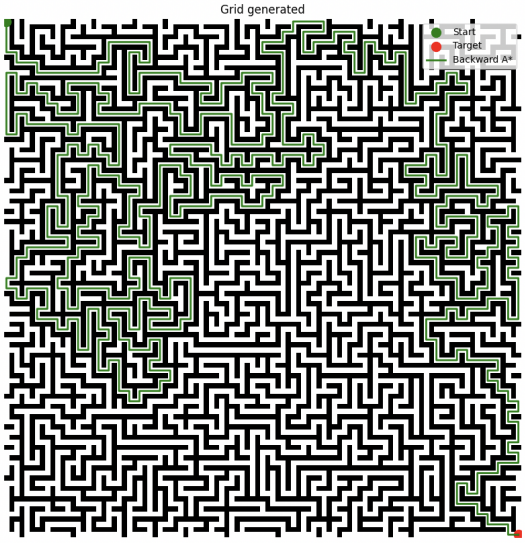

Figure 2: Visualization of Backward A* search.

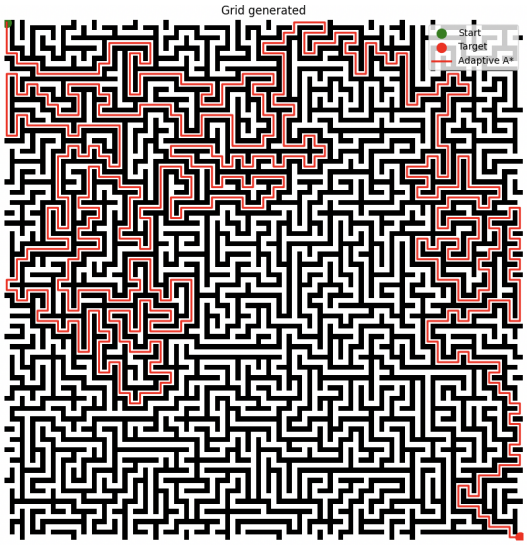

Figure 3: Visualization of Adaptive A* search.

9 Conclusions

Key observations from analyzing the results of the data:

- Path length: The length of the path was the same across all test cases.
- Nodes expanded: Backward A* expands significantly more nodes than Repeated A* and Adaptive A*.
- Runtime: Adaptive A* consistently outperforms other algorithms, as it uses previous searches to improve efficiency.
- Tie-breaking impact: Using large-g consistently led to fewer nodes being expanded than small-g, but the difference in runtime is negligible.