Part 5 - Forward A vs. Adaptive A Analysis*

Implementation

We implemented both Repeated Forward A* and Adaptive A* for the Fast Trajectory Replanning problem. Both implementations break ties among cells with the same f-value in favor of cells with larger g-values. Adaptive A* dynamically updates its heuristic values based on previously computed paths to improve efficiency in future searches.

Experimental Results

We ran experiments on 50 randomly generated mazes of size 101×101 with approximately 30% blocked cells. The results show how Adaptive A* compares to Forward A* in terms of efficiency:

1. **Expanded Cells:**
    - Forward A*: Average of 7,916.52 cells expanded
    - Adaptive A*: Average of 8,042.46 cells expanded
    - Ratio (Adaptive/Forward): 1.016x (1.6% more expanded cells)

Contrary to expectations, Adaptive A expanded slightly more nodes than Forward A*. This suggests that the heuristic updates may not be significantly reducing search effort in subsequent iterations for mazes of size 101x101.

2. **Path Length:**
    - Forward A*: Average path length of 161.58
    - Adaptive A*: Average path length of 170.00
    - Difference: Adaptive A* paths were 5.21% longer

The Adaptive A* paths were slightly longer, which contradicts the typical assumption that improved heuristics should guide the search toward more efficient routes, but was not unexpected, as the larger the mazes become, the better adaptive A* will run.

3. **Runtime:**
    - Forward A*: Average runtime of 0.100888 seconds
    - Adaptive A*: Average runtime of 0.113215 seconds
    - Ratio (Adaptive/Forward): 1.12x (12.22% slower)

Adaptive A* was slower than Forward A* despite its intent to reduce search effort. This suggests that the overhead of heuristic updates was not offset by a reduction in node expansions.

Detailed Analysis

1. **Consistency of Performance**
    - Adaptive A was slower than Forward A* in most test cases.
    - The additional heuristic updates did not consistently reduce node expansions.

- In some mazes, Adaptive A expanded fewer nodes, but in others, it expanded significantly more, making its performance inconsistent.

2. **Impact of Maze Complexity**
    - The performance gap increased in larger and more complex mazes.
    - In some cases, Adaptive A expanded more than 10% additional nodes compared to Forward A*.
    - In simpler mazes, the difference was negligible, suggesting that Adaptive A* is less beneficial in randomly structured environments.

Explanation of Observations

1. **Unexpected Increase in Expanded Nodes**
    - Adaptive A* updates heuristic values dynamically, intending to reduce future search efforts.
    - However, if the heuristic updates do not significantly improve search efficiency, the additional computational overhead may lead to more expanded nodes instead of fewer.
    - In these experiments, the heuristic updates may not have been effective enough to improve the overall search efficiency.
2. **Overhead of Heuristic Updates**
    - Each time Adaptive A* completes a search, it modifies the heuristic values for future searches.
    - This additional computation does not pay off unless the environment structure allows for significant reuse of the updated heuristics.
    - In these mazes, the adaptive updates did not meaningfully reduce search effort, leading to increased runtime.
3. **Trade-off Between Path Length and Computation**
    - Theoretically, Adaptive A should reduce node expansions and improve path efficiency.
    - However, in this case, it found longer paths while still expanding more nodes.
    - This suggests that the updated heuristics were not effectively guiding the search toward better paths.

Conclusion

For repeated A* search in a fog-of-war scenario, Adaptive A underperformed compared to Forward A*:

- Expanded 1.6% more nodes on average.
- Increased path lengths by 5.21%.
- Ran 12.22% slower despite being designed to improve efficiency.

These results indicate that Adaptive A may not be beneficial in highly random environments, where the heuristic updates do not sufficiently improve search efficiency. Forward A* remains more computationally efficient in this setting of 101x101 grids.

When to prefer Forward A*:

- When computational resources are limited.
- When search time is critical.
- When navigating environments where heuristic improvements have little impact.

When Adaptive A would be more efficient:

- When solving highly structured environments where updated heuristics can be reused effectively.
- When paths reuse previous computations rather than requiring new searches from different starting locations.

Key Takeaways

- Forward A was overall more efficient than Adaptive A* in this set of tests.
- The heuristic updates in Adaptive A* did not significantly reduce node expansions.
- The additional overhead of updating heuristics resulted in slower runtimes.
- In structured environments, Adaptive A may perform better, but in random environments, Forward A remains the better choice.