Smart Delivery Robot
Path Analysis

1. Introduction

This report presents a comprehensive performance analysis of the search algorithms implemented for the Smart Delivery Robot problem. The goal is to evaluate and compare Breadth-First Search (BFS), Uniform Cost Search (UCS), and A Search based on their ability to find the most cost-efficient path, while also considering nodes explored and computational time.

2. Performance Metrics & Comparison

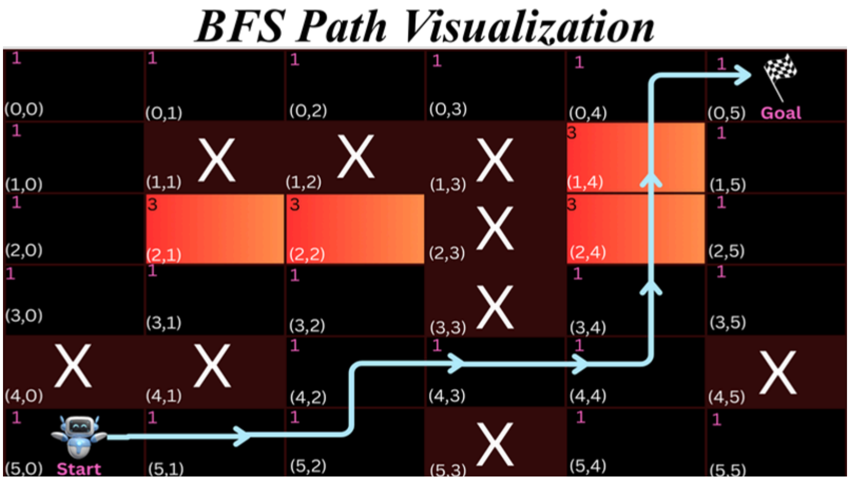
After implementing the three search algorithms on the delivery grid, the results in the table show a clear distinction in performance:

Algorithm	Path Cost	Nodes Explored	Time (ms)
UCS	10	41	0.1
A*	10	21	0.12
BFS	14	29	0.16

- **Path Cost & Optimality:** Both **UCS** and **A*** successfully identified the optimal (cheapest) path with a total cost of **10**. In contrast, **BFS** resulted in a higher cost of **14** because it focuses on the number of steps rather than the road traffic costs.
- **Search Efficiency:** **A*** proved to be the most efficient algorithm, exploring only **21 nodes** to find the goal. This is a significant improvement over **UCS**, which had to explore **41 nodes** to reach the same result.
- **Time Performance:** All algorithms executed in under **0.2 ms**, confirming their suitability for real-time path planning in this environment.

3. Path Visualization & Behavioral Analysis

- ## *BFS Algorithm*



The Breadth-First Search (BFS) algorithm is designed to find the shortest path in terms of the number of steps (edges) between the start and the goal. It explores the grid level by level, expanding all possible directions equally. However, BFS is 'cost-blind,' meaning it does not take into account the different weights assigned to specific cells.

Performance Based on Results:

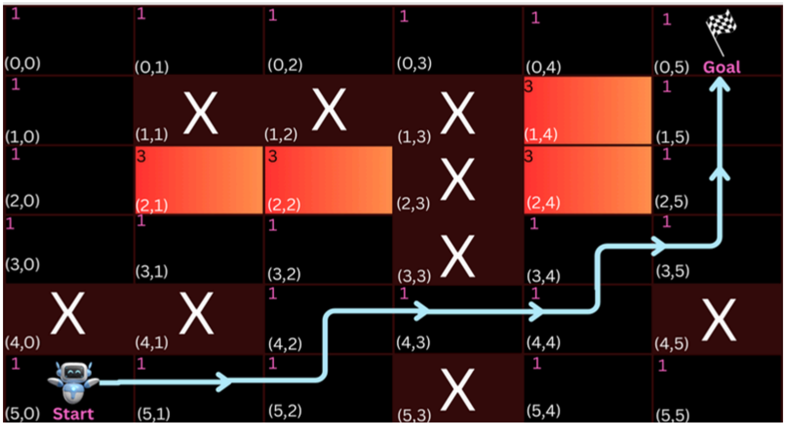
- **Path Cost (14):** **As shown in our results, BFS recorded the highest path cost of 14. This happened because the algorithm chose a direct route that passed through high-traffic cells (colored orange) with a cost of 3 each, rather than taking a slightly longer but cheaper detour.**

- **Nodes Explored (29):** **The algorithm explored 29 nodes before reaching the goal. While it found a path quickly, the lack of cost-awareness makes it inefficient for a Smart Delivery Robot that aims to minimize Total Cost.**

- **Execution Time (0.16 ms):** **BFS had the highest execution time among the three, as it spent more resources expanding nodes that did not necessarily lead to an optimal cost path.**

- ## *UCS Algorithm*

UCS Path Visualization



The Uniform Cost Search (UCS) algorithm is an optimal search strategy designed to find the path with the lowest total cost between the start and the goal. Unlike BFS, UCS is 'cost-aware'; it prioritizes exploring paths based on their cumulative cost rather than the number of steps. It ensures that the robot avoids high-traffic areas if a cheaper alternative exists.

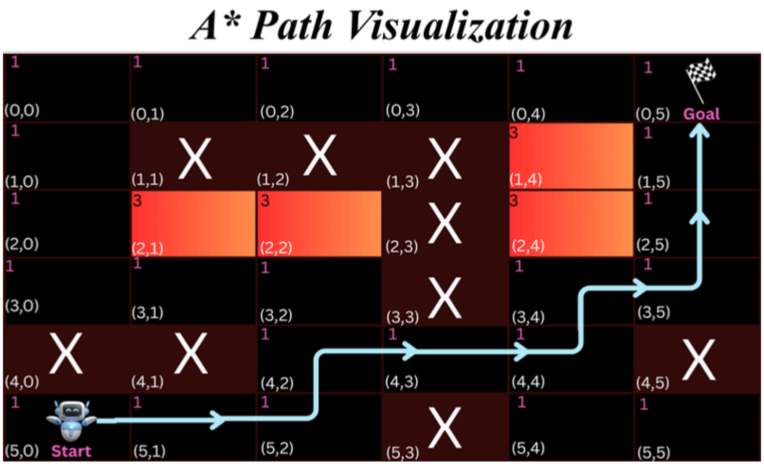
Performance Based on Results:

- **Path Cost (10):** **UCS successfully identified the optimal path with a total cost of 10. By considering cell weights, it intelligently navigated around the high-traffic orange cells (cost 3) to minimize delivery expenses.**

- **Nodes Explored (41):** **The algorithm explored 41 nodes, which is the highest among the three algorithms. This is because UCS expands in all directions equally based on cost, making it less focused compared to heuristic-based searches.**

- **Execution Time (0.1 ms):** **Despite exploring more nodes, UCS achieved the fastest execution time of 0.1 ms in this specific grid, showing its reliability in finding optimal solutions.**

• *A* Algorithm*



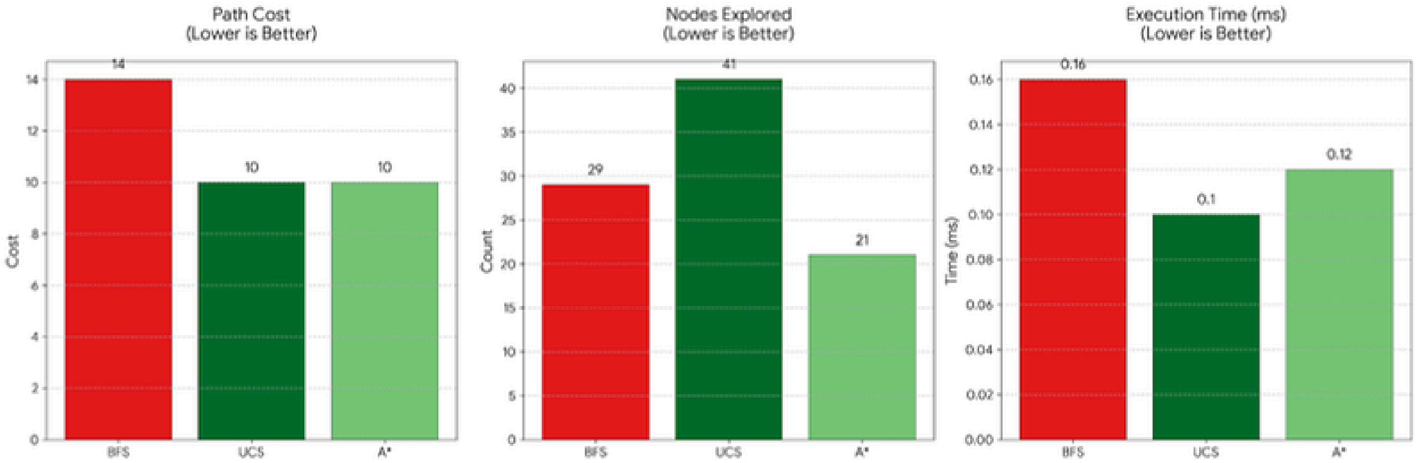
The A* Search algorithm is the most advanced and intelligent pathfinding strategy used in our study. It is an informed search that combines the actual cost from the start (like UCS) with an estimated cost to the goal, known as a Heuristic. By using the Manhattan Distance as a heuristic, A* focuses the robot's search directly towards the target, avoiding unnecessary exploration of the grid.

Performance Based on Results:

- **Path Cost (10):** Like UCS, A* is an optimal algorithm that successfully found the cheapest path with a cost of 10.
- **Search Efficiency (21 Nodes):** This is the key strength of A*. It reached the same goal as UCS but explored only 21 nodes instead of 41. This 50% reduction in explored nodes proves that A* is much more efficient and "smarter" in managing resources.
- **Execution Time (0.12 ms):** A* performed exceptionally fast, nearly matching the speed of UCS while doing much less computational work.

4.Statistical Performance Comparison

- **To provide a rigorous evaluation of the three algorithms, we compared their performance across three primary metrics: Path Cost, Search Efficiency (Nodes Explored), and Execution Time. The following charts illustrate the quantitative trade-offs between each approach.**



- **The statistical results confirm that A* Search is the most efficient algorithm, finding the optimal path (Cost: 10) with the fewest nodes explored (21 nodes). While UCS also finds the optimal path, it requires double the computational effort (41 nodes). BFS is unsuitable for this traffic-heavy environment as it resulted in the highest path cost (14).**

5. Conclusion and Final Recommendation

Conclusion

This project involved the design and implementation of an intelligent navigation system for a delivery robot. By comparing BFS, UCS, and A* algorithms, we demonstrated how different search strategies impact the robot's ability to navigate a cost-sensitive environment (traffic and road conditions). The results showed that while blind search (BFS) is simple, it is inefficient for real-world scenarios where costs vary.

Final Recommendation

Based on the quantitative analysis conducted in the previous section, we strongly recommend the *A* Search Algorithm* for the robot's production environment.

Why A*?

- **Optimality: It consistently finds the shortest and least-cost path, ensuring timely deliveries.**
- **Efficiency: It explores the minimum number of nodes (21 nodes), which translates to lower CPU usage and faster decision-making compared to UCS (41 nodes).**
- **Resource Management: For a mobile robot, this efficiency is crucial for preserving battery life and operating effectively in real-time**