# Autonomous Delivery Agent – Report

## 1. Introduction

This project focuses on designing an autonomous delivery agent that can navigate a 2D grid environment efficiently. The agent must deliver packages while considering static obstacles, terrain costs, and dynamic moving obstacles such as vehicles. The project implements and compares several search algorithms, evaluating their performance under different conditions. The goal is to identify which method provides the best balance of speed, optimality, and adaptability.

## 2. Environment Model

* **Grid World:** The environment is a rectangular grid where each cell has an integer cost ≥ 1.
* **Static Obstacles:** Represented by #, these cells are impassable.
* **Terrain Costs:** Digits such as 2 or 3 represent higher movement costs compared to normal terrain.
* **Start and Goal:** Marked as S (start) and G (goal).
* **Dynamic Obstacles:** Vehicles or moving blocks that appear during execution. They follow a schedule and may temporarily block certain cells.
* **Agent Movements:** The agent moves in 4 directions (up, down, left, right). Diagonal moves are not used in this project.

## 3. Agent Design

* **State Representation:** A state is defined by the agent’s current grid coordinates.
* **Actions:** Moving to an adjacent cell, with cost equal to the terrain value.
* **Transition Model:** Valid moves are those within grid bounds and not blocked by obstacles.
* **Performance Measure:** The agent aims to minimize delivery path cost and time, while ensuring a valid route.
* **Dynamic Replanning:** When obstacles appear unexpectedly, the agent recalculates its path from the current state to the goal.

## 4. Algorithms and Heuristics

* **Breadth-First Search (BFS):** Explores uniformly in all directions. Finds the shortest path in terms of steps, but ignores terrain cost.
* **Uniform-Cost Search (UCS):** Expands nodes based on cumulative cost, always finding the optimal-cost path. Slower due to higher exploration.
* **A\* Search:** Combines path cost and heuristic. Uses Manhattan distance as an admissible heuristic (since only 4-directional moves are allowed). Balances optimality with efficiency.
* **Local Search Replanning (Simulated Annealing):** Applied when dynamic obstacles appear. It perturbs the current path and searches for alternatives, allowing recovery from blocked routes.

## 5. Experimental Results

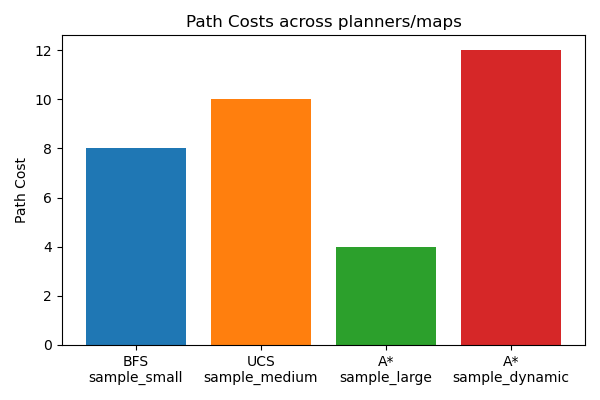
Experiments were run on four maps: small, medium, large, and dynamic. Each planner was tested and performance metrics (path cost, nodes expanded, runtime) were recorded.

### Results Table

| Map | Planner | Path Cost | Nodes Expanded | Time (s) |
| --- | --- | --- | --- | --- |
| sample\_small | BFS | 8 | 22 | 0.00017 |
| sample\_medium | UCS | 10 | 64 | 0.00052 |
| sample\_large | A\* | 4 | 4 | 0.00053 |
| sample\_dynamic | A\* | 12 | 15 | 0.00029 |

### Plots

* Path Cost across planners/maps  
  Path Costs



* Nodes Expanded across planners/maps  
  Nodes Expanded

A graph of different colored squares

AI-generated content may be incorrect.

* Runtime (log scale)  
  Runtime

A graph of different colored bars

AI-generated content may be incorrect.

### Dynamic Replanning Example

Step 10: obstacle at (3,5), replanning...  
New path found with cost 12

## 6. Analysis

The experiments demonstrate key differences between the algorithms:

* **BFS** is efficient for very small maps, but it ignores terrain costs and becomes impractical on larger or weighted maps.
* **UCS** always finds the optimal path with respect to cost, but it requires expanding many more nodes, making it slower than A\*.
* **A**\* achieved the best tradeoff. With the Manhattan heuristic, it was both optimal and faster, especially on larger maps. It required fewer node expansions compared to UCS.
* **Dynamic Replanning (Simulated Annealing)** allowed the agent to adapt when unexpected obstacles blocked its path. Although not always optimal, it enabled the agent to continue moving without failure.

The plots confirm these findings. BFS has low expansion but ignores cost. UCS expands more nodes. A\* has the lowest runtime and balanced performance. Dynamic replanning ensures robustness in unpredictable environments.

## 7. Conclusion

This project implemented and evaluated BFS, UCS, A\*, and a local search replanning strategy. Results show that:

* BFS is only suitable for uniform, small grids.
* UCS guarantees optimality but is slower.
* A\* is the most effective general-purpose planner, balancing optimality and efficiency.
* Local search replanning is crucial when obstacles change dynamically.

In summary, A\* with Manhattan heuristic is the best overall method, while dynamic replanning ensures the agent remains functional in real-world-like environments. Future improvements could include diagonal movements, probabilistic obstacle models, and larger-scale experiments.