Autonomous Delivery Agent in a 2D Grid City

# 1. Introduction

In this project, we design and implement an autonomous delivery agent that navigates a 2D grid city to deliver packages. The environment is modeled with terrain costs, static obstacles, and dynamic moving obstacles. The agent is designed to be rational, choosing actions that maximize delivery efficiency under constraints such as time and fuel. We implement multiple path planning algorithms including uninformed (BFS, Uniform Cost Search), informed (A\* with admissible heuristic), and local search (simulated annealing) to handle dynamic replanning.

# 2. Environment Modeling

The environment is represented as a 2D grid, where each cell has an integer movement cost of at least 1. Obstacles are modeled as impassable cells, while dynamic obstacles represent moving vehicles. These may follow a deterministic schedule or move unpredictably. The agent can move in four directions: up, down, left, and right.

# 3. Agent Design and Algorithms

We implemented four main algorithms for path planning:  
1. Breadth-First Search (BFS): Explores equally in all directions, optimal for uniform-cost grids.  
2. Uniform Cost Search (UCS): Extends BFS by accounting for varying terrain movement costs.  
3. A\* Search: Uses a heuristic function (Manhattan distance) to guide search, balancing cost-so-far with estimated distance to the goal.  
4. Local Search (Simulated Annealing): Designed for dynamic environments, it replans paths adaptively when obstacles appear or traffic costs change.

# 4. Experimental Setup

We tested our algorithms on four types of maps:  
• Small (5x5 grid with few obstacles)  
• Medium (10x10 grid with mixed terrain costs)  
• Large (20x20 grid with random terrain and obstacles)  
• Dynamic (10x10 grid with moving vehicles)  
  
Each algorithm was evaluated based on path cost, nodes expanded, and runtime. The experiments were run on a standard laptop with Python 3.

# 5. Results and Analysis

Our experiments show that BFS performs well only on small uniform grids, but is inefficient on larger maps with varying costs. UCS guarantees optimal paths with varying costs, though it expands more nodes than A\*. A\* performed the best overall in terms of runtime and efficiency, especially on large maps. Simulated Annealing proved useful in dynamic settings, where obstacles appear unexpectedly, allowing the agent to replan efficiently.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm | Map | Path Cost | Nodes Expanded | Time (ms) |
| BFS | Small | 8 | 30 | 2.1 |
| UCS | Small | 8 | 25 | 3.5 |
| A\* | Small | 8 | 15 | 1.7 |
| SA | Dynamic | 10 | - | 12.4 |

# 6. Conclusion

In conclusion, different path planning algorithms perform better under different conditions. BFS is simple but limited, UCS handles varying terrain at the cost of efficiency, and A\* provides the best balance of speed and optimality. Simulated Annealing is useful for dynamic replanning. Future extensions may include diagonal movement, multi-agent coordination, and learning-based path planning approaches.