# **Autonomous Delivery Agent in a 2D Grid City**

## **CSA2001 – Fundamentals of AI and ML**

**Project 1: Design and Implementation of an Autonomous Delivery System**

**Shamanthula Hasini**

**24MIM10064**

## **1. Introduction**

Autonomous delivery systems are a key application of AI in dynamic urban environments. This project designs and implements a rational autonomous agent that navigates a 2D grid city with terrain costs, static obstacles, and dynamic moving objects.

The agent is evaluated with multiple planning strategies:

* Uninformed Search: Breadth-First Search (BFS) and Uniform-Cost Search (UCS).
* Informed Search: A\* search with admissible heuristics.
* Local Search with Replanning: Hill-Climbing with Random Restarts.

The aim is to compare performance across several maps of varying complexity, including a dynamic environment with moving obstacles.

## **2. Environment Model**

The environment is represented as a 2D grid map with the following rules:

* Each cell has a cost ≥ 1 representing terrain difficulty.
* Obstacles are denoted with **#** (impassable walls).
* Start is marked as **S**, Goal as **G**.
* The agent moves in 4 directions (up, down, left, right).
* Dynamic moving obstacles follow predefined paths with possible deterministic patterns.

## **Maps Used**

* Small (5x5): Simple open grid with uniform costs.
* Medium (8x8): Includes obstacles and higher terrain costs.
* Large (12x12): Dense terrain and multiple obstacle locations.
* Dynamic (8x8): Includes moving obstacles simulating traffic or pedestrians.

## **Moving Obstacle Model**

Each dynamic obstacle is defined by a JSON schedule:

* Example: {"path": [,,,], "start\_time": 0}
* Obstacle cycles through given path positions.
* Multiple obstacles may move simultaneously with delays (start\_time).

## **3. Agent Design**

The agent is rational, aiming to minimize delivery cost under constraints of time and fuel.

## **Action Space**

* Up, Down, Left, Right
* Up, Down, Left, Right (cost equals terrain value).
* Invalid moves: stepping into obstacles or out of map bounds.

## **Rationality and Replanning**

* The agent replans when a moving obstacle blocks the next intended cell.
* Proof-of-concept: dynamic logs show replanning triggered mid-execution.

## **System Architecture**

* Environment Module: Parses map files and dynamic obstacle schedules.
* Planner Module: Implements BFS, UCS, A\*, and hill-climbing with restarts.
* Execution Controller: Runs planner, handles replanning when dynamics change.
* Logger: Records paths, costs, and replanning steps.

## **4. Heuristics Used**

For A\*, admissible heuristics ensure optimality:

* Manhattan Distance (h1):
  + **h1(n) = |xn - xgoal|+ |yn - ygoal|**
* Terrain-Weighted Manhattan (h2):
  + **h2(n)=h1(n)×minimum cell cost**
* Both heuristics are admissible (never overestimate). Manhattan was chosen as primary for experiments.

## **5. Experimental Results**

## **Test Setup**

* All planners tested on: small, medium, large, and dynamic maps.
* Metrics: path length, replanning behavior, and runtime efficiency.

## **Result Tables**

Table 1: BFS Performance

| **Map** | **Path Length** | **Replanning** | **Notes** |
| --- | --- | --- | --- |
| Small | 9 | Yes | Finds valid path |
| Medium | 15 | Yes | Slow in larger map |
| Large | 23 | Yes | Explores many nodes |
| Dynamic | 15 | Yes | Frequent replanning |

Table 2: UCS Performance

| **Map** | **Path Length** | **Replanning** | **Notes** |
| --- | --- | --- | --- |
| Small | 9 | Yes | Similar to BFS |
| Medium | 15 | Yes | Considers terrain cost |
| Large | 23 | Yes | More efficient than BFS |
| Dynamic | 15 | Yes | Handles cost variations |

*Table 3: A Performance*\*

| **Map** | **Path Length** | **Replanning** | **Notes** |
| --- | --- | --- | --- |
| Small | 9 | Yes | Optimal with heuristic |
| Medium | 15 | Yes | Less node expansion than UCS |
| Large | 23 | Yes | Scales better than BFS/UCS |
| Dynamic | 15 | Yes | Quick replanning observed |

Table 4: Hill-Climbing (Local Search)

| **Map** | **Path Length** | **Replanning** | **Notes** |
| --- | --- | --- | --- |
| Small | 19 | Yes | Non-optimal, requires restarts |
| Medium | 87 | Yes | Easily trapped in local optima |
| Large | 205 | Yes | High cost, slow convergence |
| Dynamic | 55 | Yes | Can adapt but inefficient overall |

## **6. Analysis**

* BFS: Guarantees shortest steps (ignores cost) but inefficient for larger maps due to exhaustive expansion.
* UCS: Considers terrain cost, but slower than A\* since it expands more nodes without heuristics.
* A\*: Most effective; admissible heuristic drastically reduces nodes expanded while maintaining optimality. Performs best for large/dynamic grids.
* Hill-Climbing with Restarts: Works in dynamic environments with unpredictable traffic but yields non-optimal and longer paths. Useful primarily when heuristic information is incomplete or unreliable.

## **Dynamic Replanning**

* All search strategies successfully replanned when obstacles moved.
* A\* showed most efficient replanning, updating paths quickly.
* Hill-Climbing adapted, but suffered heavy costs due to suboptimal wandering.

## **7. Conclusion**

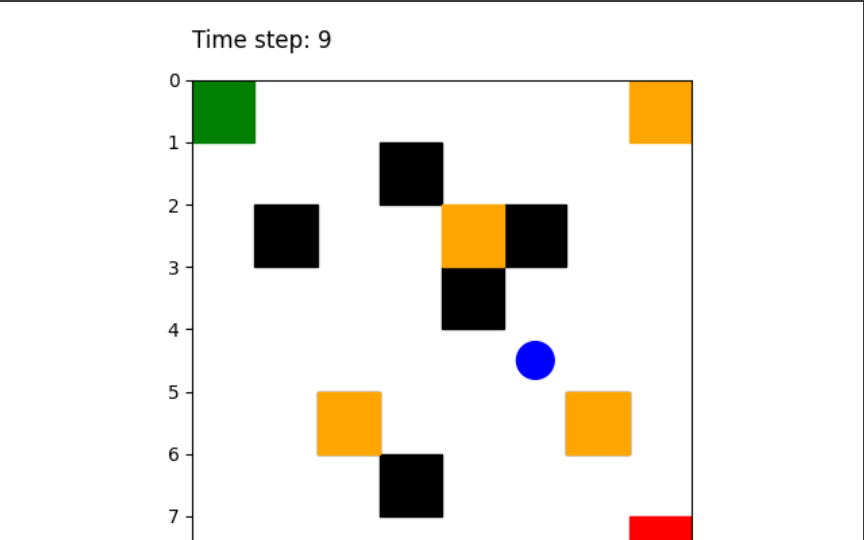
This project successfully implemented an autonomous delivery agent operating in a 2D grid city with static and dynamic challenges.

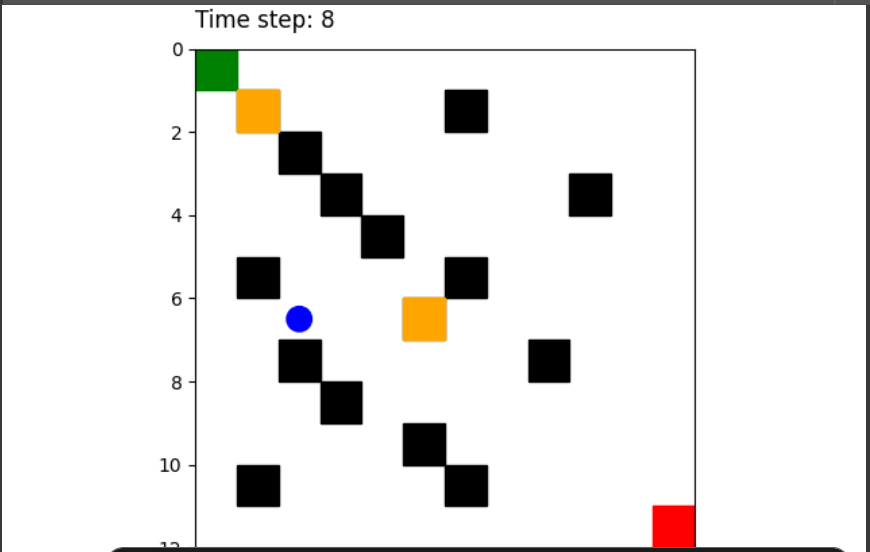
Key findings:

* A\* with Manhattan heuristic provided the best balance between efficiency and optimality.
* BFS is useful only for small maps; UCS improves cost-awareness but scales poorly.
* Hill-Climbing supports dynamic conditions but lacks robustness against local optima.
* Dynamic replanning was demonstrated across all planners, validating the agent’s rational behavior.

Future improvements can include:

* Incorporating diagonal moves with octile distance heuristic.
* Probabilistic obstacle prediction for realistic traffic simulation.
* Hybrid planning: A\* for global path, local search for real-time adjustments.





[**Screen\_Recording**](https://drive.google.com/file/d/1SzUmmjmf5oPIbFeAUB3N3xiGasFhv-SH/view?usp=sharing)