Autonomous Delivery Agent — Short Report

Project goal (one paragraph)

Design and implement an autonomous delivery agent that navigates a 2D grid city to deliver packages while handling static obstacles, varying terrain costs, and dynamic moving obstacles. The agent should be rational (optimize delivery efficiency under constraints) and implement uninformed (BFS/UCS), informed (A* with admissible heuristic), and a local search replanning strategy (simulated annealing) to handle dynamics. Experimental comparison and a short demo are provided.

Environment model

- Grid: each cell has integer movement cost ≥ 1 or is impassable (`#`).

- Start `S` and Goal `G`.

- Movement: 4-connected (up, down, left, right).

- Dynamic obstacles: JSON schedule files listing `(time, row, col)` positions. Occupancy is time-indexed.

Agent and planners

- **BFS**: shortest number of steps (ignores terrain cost).

- **Uniform-Cost Search (UCS)**: optimal w.r.t. summed terrain cost; uses a priority queue.

- **A***: heuristic = Manhattan distance × minimum terrain cost (admissible because min cost × distance ≤ true minimal remaining cost). Supports time-expanded search when dynamic schedule is known.

- **Local search (Simulated annealing)**: quick path repair by attempting local re-solves between two nodes in the path and accepting improvements or occasional worse moves to escape local minima.

Dynamic replanning strategy

- Agent executes plan step-by-step with a global clock `t`.

- If a planned next cell is occupied at arrival time, agent:

1. Attempts **local repair** (fast).

2. If repair fails, **replans** using time-aware A* (if schedule exists) or time-agnostic A* otherwise.

Heuristic admissibility

- Heuristic `h(n) = Manhattan(n, goal) × c_min`.

- Because each step costs at least `c_min`, the heuristic never overestimates true remaining cost — admissible.

Experimental setup (how to reproduce)

1. Ensure Python 3.8+ and `matplotlib` installed for plotting/gif generation.

2. In project folder, run:

```
python delivery_agent.py --write-sample-maps

python delivery_agent.py --map maps/small_map.txt --algo astar --show
```

3. To run the provided experiment runner:

```
python delivery_agent.py --run-experiments
```

This creates `results.csv` with planner comparisons.

Results summary (example)

- Small map: A* finds a low-cost path with fewer node expansions than UCS in many cases due to heuristic guidance.

- Dynamic map: Local search often repairs quickly but may produce suboptimal reroutes; time-aware A* gives safer, schedule-aware plans when obstacles' schedules are known.

Analysis & conclusion (short)

- **BFS**: useful when steps matter, environment uniform cost; fails to account for terrain cost.

- **UCS**: optimal for cost, but can be slow in large maps.

- **A***: best trade-off when heuristic is informative; significantly reduces nodes expanded.

- **Local repair**: fast for short-term obstacle avoidance, good for unpredictable dynamics; combine with time-aware A* for best reliability.

- Future work: richer local-search neighborhoods, diagonal moves, visualization/animation improvements, multi-objective optimization (time vs fuel).