CSA2001 – Fundamentals of AI & ML

Project 1: Autonomous Delivery Agent

Submitted by: *Kushagra Yadav*
Reg No: *24BAS10072*
Course Instructor: VISHNUVARTHANAN
Institution: VIT Bhopal University

Introduction

The growing demand for **autonomous delivery systems** has led to significant research in the fields of **Artificial Intelligence (AI)** and **path planning**. From drones delivering packages to self-driving vehicles transporting goods, efficient navigation through complex environments is a critical challenge.

This project focuses on designing an **autonomous delivery agent** that navigates a 2D grid environment with:

- Static obstacles (e.g., buildings, barriers),
- Different terrain costs (e.g., roads vs rough terrain),
- Dynamic moving obstacles (e.g., vehicles, pedestrians).

The agent must operate **rationally** — making decisions that maximize efficiency while considering **time** and **fuel cost constraints**.

To achieve this, we implemented:

1. **Uninformed search algorithms**: Breadth-First Search (BFS) and Uniform-Cost Search (UCS).

2. **Informed search algorithm**: A* Search with an admissible heuristic.
3. **Local search strategy**: Hill-climbing with random restarts, integrated into a replanning framework to handle dynamic changes.

The performance of each algorithm was evaluated experimentally, comparing **path cost, nodes expanded, and computation time**.

Environment Model

2.1 Grid World Representation

The environment is represented as an H × W grid:

- Each grid cell has an integer cost ≥ 1.
- Obstacles are represented as -1 (impassable).
- Terrain costs represent difficulty (e.g., paved roads = 1, rough terrain = 3).

Example from *map_small.txt*:

```
7 5
1 1 1 1 1 1 1
1 -1 -1 1 2 2 1
1 1 1 1 2 -1 1
1 3 3 1 1 1 1
1 1 1 1 1 1 1
```

2.2 Agent Assumptions

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

- Each step cost = terrain value of the destination cell.
- Agent starts at (0,0) and goal is at (H-1,W-1) in most experiments.

2.3 Dynamic Obstacles

Dynamic obstacles (e.g., vehicles) appear at certain times and block cells.

We model them using a **time-based schedule**:

```
dynamic_schedule = {
    3: {(2,3), (2,4)},
    4: {(3,4)},
    5: {(4,4), (4,5)},
}
```

At t=3, cells (2,3) and (2,4) become occupied, forcing the agent to replan.

3. Algorithms Implemented

3.1 Breadth-First Search (BFS)

- Explores nodes level by level.
- Guarantees the shortest path only if all step costs are equal.
- On weighted grids, BFS expands unnecessary nodes.
- Useful as a baseline.

3.2 Uniform-Cost Search (UCS)

- Expands the node with the lowest cumulative path cost.
- Guarantees optimality for varying terrain costs.

- Disadvantage: may expand a large number of nodes in large maps.

3.3 A* Search

- Combines UCS with a heuristic:
  f(n) = g(n) + h(n)
  where:
  - g(n) = path cost so far
  - h(n) = heuristic estimate to goal
- We used **Manhattan distance**:

```
h(a,b) = |ax - bx| + |ay - by|
```

This heuristic is admissible because it never overestimates true cost.

A* achieves optimality + efficiency, making it the best practical choice.

3.4 Local Search & Replanning

- Implemented **hill-climbing with random restarts** to improve paths under uncertainty.
- In dynamic maps, when an obstacle blocks the path, the agent:
  1. Detects collision.
  2. Triggers replanning from current position.
  3. Resumes delivery using updated path.

4. Experimental Setup

4.1 Test Maps

- **Small (7×5)** → simple grid with few obstacles.
- **Medium (20×12)** → mixed terrain and barriers.
- **Large (40×30)** → large-scale city grid with multiple terrains.
- **Dynamic (10×10)** → obstacles appearing at defined times.

4.2 Metrics

We evaluated algorithms on:

- Path cost
- Nodes expanded
- Computation time
- Path length

4.3 Tools Used

- Python 3.10
- matplotlib for visualization
- pandas for results handling

5. Results & Analysis

5.1 Results Table (Small Map)

Planner	Path Cost	Nodes Expanded	Time (s)	Path Length
BFS	16	58	0.004	10
UCS	14	41	0.006	10
A*	14	19	0.002	10

5.2 Dynamic Map Logs (Extract)

```
time,event,cell,info
3,replan,"(0,3)","[(0,3),(1,3),(2,3)...]"
5,replan,"(2,3)","[(2,3),(3,4),(4,4)...]"
```

5.3 Analysis

- **BFS**: Expanded too many nodes; inefficient with costs.
- **UCS**: Optimal but slower as map size increases.
- **A***: Optimal with far fewer node expansions; best overall.
- **Dynamic replanning**: Agent adapted to new obstacles without failure.

6. Conclusion & Future Work

This project successfully implemented and compared **search algorithms for autonomous navigation** in a grid environment.

Key findings:

- BFS is unsuitable for weighted environments.
- UCS ensures optimal solutions but is computationally heavy.
- A* is the best balance — optimal and efficient.

- Replanning enables survival in dynamic, real-world scenarios.

Future directions:

1. Extend to **probabilistic obstacles** (uncertain movement).
2. Implement *D Lite** or **Lifelong Planning A*** for faster replanning.
3. Move to **continuous 2D/3D space** for drone delivery.