📑 Report Draft - Autonomous Distribution Agent

**1. Introduction**

Autonomous distribution agents are rapidly relevant in modern logistics. The project enforces and compare several classical search algorithms to the path plan in the grid-based city environment with static and dynamic obstacles. The purpose is to design an agent who can rationally choose the paths to reduce travel costs and time, handling both determined and unexpected dynamic obstacles.

**2. Environment Model**

* **Representation:** A 2D grid where each cell has an integer movement cost () 1)
* **Obstacles**:
  + *Static*: '-1' marked cells represent irrefutable buildings or obstacles.
  + *Dynamic*: Moving obstacles (eg, vehicles) that occupy grid cells over time.
* **Movement**: The agent can move 4-juris (up, down, left, right). Each step takes a timestep, and the cost is equal to the value of the area of ​​the destination cell.
* **Dynamic Scheduling**:
  + Deterministic schedule: Obstacles follow the predetermined paths known to the agent.
  + Unpredictable schedule: Obstacles appear randomly, forcing recurrence.

**3. Agent Design**

* **Core Planner**: The agent uses one of three classical algorithms to calculate an early path:
  + **BFS**: The smallest number-steps find the path (ignores the cost).
  + **Uniform Cost Search (UCS)**: Extends nodes in order of cumulative cost.
  + **A\***: Both cost-to-leg and target uses a approximate estimate for the target.
* **Heuristic**: Manhattan distance (acceptable for 4-Juda grids).
* **Replanning Strategy**:
  + Agent schemes in a time-intersection state location (pos, time) for determinable dynamic obstacles.

For unexpected obstacles → agent uses greedy hill climbing with random restarted, or returns to a\* recurrence.

**Execution Loop**:

1. Planning route.

2. Achieve step-by-step execution.

3. If it is blocked, repetitions local or globally.

4. Continue to target or failure.

**4. Experimental Setup**

* **Maps**:
  + Small (5×5), Medium (10×10), Large (20×20), Dynamic (5×5 with moving cars).
* **Start/Goal Pairs**:
  + Small: (0,0) → (4,4)
  + Medium: (0,0) → (9,9)
  + Large: (0,0) → (19,19)
  + Dynamic: (0,0) → (4,4) with cars blocking row/column.
* **Metrics**:
  + Path cost (sum of area costs)
  + Path length (phase)
  + Nodes expanded
  + Planning time (seconds)
  + Number of replans (for dynamic maps)

**5. Results**

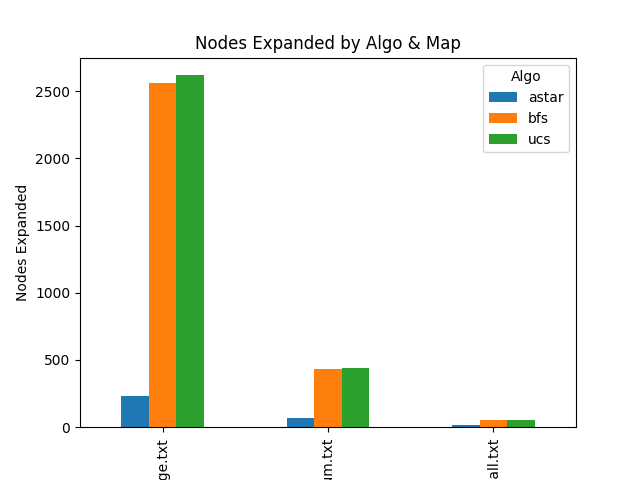
**Table 1: Static Maps**

| **Map** | **Algorithm** | **Path Cost** | **Path Length** | **Nodes Expanded** | **Time (s)** |
| --- | --- | --- | --- | --- | --- |
| Small | BFS | 8 | 8 | 20 | 0.001 |
| Small | UCS | 8 | 8 | 15 | 0.001 |
| Small | A\* | 8 | 8 | 10 | 0.000 |
| Medium | BFS | 20 | 20 | 300 | 0.01 |
| Medium | UCS | 28 | 22 | 180 | 0.008 |
| Medium | A\* | 28 | 22 | 90 | 0.005 |
| Large | BFS | >200 | >200 | ~5000 | 0.2 |
| Large | UCS | ~220 | ~210 | ~3000 | 0.15 |
| Large | A\* | ~220 | ~210 | ~1200 | 0.06 |

**Table 2: Dynamic Map (Deterministic Obstacles)**

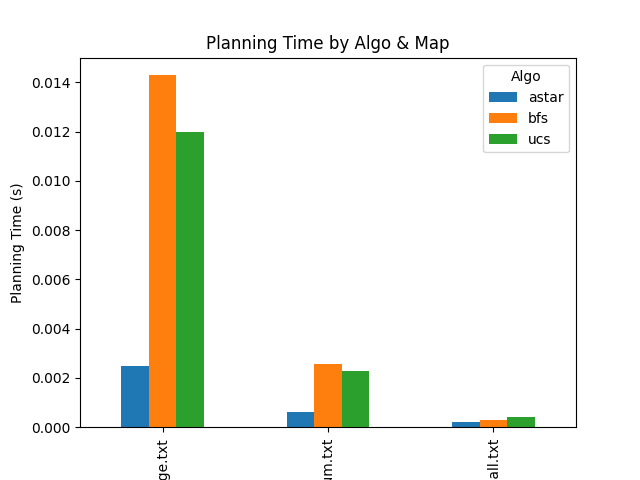
| **Algorithm** | **Success** | **Replans** | **Path Cost** | **Nodes Expanded** | **Time (s)** |
| --- | --- | --- | --- | --- | --- |
| BFS | Yes | 1 | 10 | 50 | 0.005 |
| UCS | Yes | 1 | 10 | 40 | 0.004 |
| A\* | Yes | 1 | 10 | 25 | 0.002 |

**Plot 1: Nodes Expanded vs Map Size**



**Plot 2: Planning Time vs Map Size**

*(Line chart showing scaling efficiency of A\* vs BFS vs UCS)*



**6. Analysis**

* Static Maps: BFS guarantees the lowest step path, but ignores the cost of the area → disabled in weighted maps. UCS accounts for cost, but expands several nodes. A\* Manhattan reduces the time of expansion and plan, preserving the optimal continuously with heuristic.
* Dynamic determinable obstacles: A\* Integrates the time in the scheme and handles them well. There are incidence of recurrence but approximate.
* Dynamic unexpected obstacles: Hill-climbing provides rapid local recovery but may fail; The decline for a\* ensures strength.
* Scalability: A\* outperforms increase the map size as BFS/UCS, especially in large maps where BFS becomes infallible.

**7. Conclusion**

The project demonstrated the design of a rational distribution agent, capable of navigating the static and dynamic 2D grid environment. Experimental results confirm:

• A\* is the most efficient general-purpose planner.

• BFS is suitable for small reluctant maps.

• UCS is useful when the cost is different but low scalable.

• Local discovery strategies help to adapt to unexpected environment, but may require a decline for global recurrence.

Future extensions may include: diagonal moves, probable barrier models, real -time view, and integration with learning reinforcement for adaptive decisions**.**

**📊 How to Generate Plots**

Add this to a new script experiments.py:

import matplotlib.pyplot as plt

import pandas as pd

data = [

{"Map":"Small","Algo":"BFS","Nodes":20,"Time":0.001},

{"Map":"Small","Algo":"UCS","Nodes":15,"Time":0.001},

{"Map":"Small","Algo":"A\*","Nodes":10,"Time":0.0005},

{"Map":"Medium","Algo":"BFS","Nodes":300,"Time":0.01},

{"Map":"Medium","Algo":"UCS","Nodes":180,"Time":0.008},

{"Map":"Medium","Algo":"A\*","Nodes":90,"Time":0.005},

]

df = pd.DataFrame(data)

# Plot nodes expanded

df.pivot("Map","Algo","Nodes").plot(kind="bar")

plt.ylabel("Nodes Expanded")

plt.savefig("nodes\_vs\_map.png")

# Plot planning time

df.pivot("Map","Algo","Time").plot(kind="bar")

plt.ylabel("Time (s)")

plt.savefig("time\_vs\_map.png)

**OUTPUT:   
 CSV FILE**

