**Project Report: Autonomous Delivery Agent**

**CSA2001**

1. **Introduction**

**This project covers the implementation of an autonomous agent for package delivery in a simulated 2D city. The key challenge is efficient pathfinding in a complex environment containing static, variable-cost, and dynamic obstacles, all under a strict fuel constraint. The implemented agent is rational, using informed and uninformed search algorithms to find paths and a local search strategy to react to unforeseen environmental changes.**

1. **Environment & Agent Model**

**The environment is a 2D grid parsed from `.txt` files. Cells are categorized as start, goal, wall, or standard terrain with integer costs. Dynamic obstacles are defined in a `.json` file, specifying their coordinates at each time step. The agent's state includes its position, current fuel, and the current time step.**

**3. Algorithm Implementation**

**3.1 Uninformed Search (UCS):**

**Uniform-Cost Search was implemented as a baseline for static environments. It explores paths purely on their accumulated cost from the start (`g(n)`), guaranteeing an optimal path by expanding the cheapest node first. Its state is simply the agent's `position`.**

**3.2 Informed Search (A\* ): The primary planner is a time-aware A\* search, critical for dynamic environments. Its state is a `(position, time)` tuple to avoid dynamic obstacles. It uses the evaluation function `f(n) = g(n) + h(n)`, where `h(n)` is the Manhattan distance heuristic. This heuristic is admissible, ensuring A\* finds a cost-optimal path.**

**3.3 Local Search Replanning: For unexpected obstacles, a fast local search is used. It attempts a constrained random walk from the agent's current position to a viable "rejoin point" on its original path, providing a rapid "good enough" patch.**

**4. Experimental Results**

**UCS and A\* were compared on three static maps.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Map** | **Algorithm** | **Path Cost** | **Nodes Expanded** |
| **small** | **UCS** | **5** | **10** |
| **small** | **A\*** | **5** | **8** |
| **medium** | **UCS** | **12** | **44** |
| **medium** | **A\*** | **12** | **86** |
| **large** | **UCS** | **66** | **1423** |
| **large** | **A\*** | **66** | **4355** |

**5. Analysis**

**The experimental results confirm that both UCS and A\* find the optimal path, returning the same Path Cost. For the `small` map, A\* behaved as expected, expanding fewer nodes than UCS due to its goal-oriented heuristic. However, for the `medium` and `large` maps, A\* expanded significantly more nodes than UCS. This counter-intuitive result stems from the implementation details: the A\* search is time-aware, meaning its state is `(position, time)`. In a static environment where time is irrelevant, this creates a much larger search space. The algorithm explores redundant nodes, treating arrival at the same cell `(x, y)` at `t=10` as a different state from arriving at `(x, y)` at `t=11`. UCS, with its simpler `position`-only state, does not explore these redundant time-based states and is therefore more efficient on these specific static maps. This highlights a critical trade-off: a specialized algorithm (time-aware A\* ) can be less efficient when applied to a simpler problem (static pathfinding) it was not optimized for.**

**6. Conclusion**

**The implemented agent successfully navigates complex environments using appropriate algorithms. The project confirms the theoretical optimality of both UCS and A\* . The experimental results provide a crucial insight into algorithmic overhead, demonstrating that a more complex planner (time-aware A\* ) can be less efficient than a simpler one (UCS) when its specialized features (time-awareness) are not relevant to the problem.**