AI-DRIVEN DELIVERY ROUTE USING ANT COLONY OPTIMIZATION AND MACHINE LEARNING

1. INTRODUCTION

This project focuses on optimizing delivery routes through a combination of Ant Colony Optimization (ACO) and Machine Learning (ML). The goal is to minimize delivery times while ensuring adherence to strict time windows at various delivery locations. The document provides an overview of the mathematical principles involved, explains the code implementation, offers practical examples, and explores how AI and ML can enhance the overall efficiency of the optimization process. By integrating real-time traffic data and historical route performance, the solution aims to create more adaptive and efficient delivery strategies.

2. MATHEMATICAL CONCEPTS AND PROBLEM FORMULATION

2.1 Orienteering Problem (OP)

- The project draws inspiration from the Orienteering Problem (OP), where the goal is to find a path that maximizes a score (in our case, minimizes time) while adhering to certain constraints, such as a time limit.
- Vertices (locations): In our problem, delivery addresses are treated as vertices.
- Scores: While the OP maximizes scores, we aim to minimize delivery time, so we treat travel time between locations as the key optimization variable.
- Path Constraints: The path needs to respect time windows at each location and ensure the total travel time is within acceptable limits.

2.2 Mathematical Formulation

Objective Function:

$$\min \sum_{i}^{N} Tij$$

T_{ij} is the travel time between location i and location j

2.3 Time Matrix

The time matrix essentially provides the travel time between each pair of delivery locations, which the algorithm uses to determine the optimal path. Let's break down the time matrix mathematically.

2.3.1 Definition of the Time Matrix

The time matrix T is a square matrix where each element T_{ij} represents the travel time between location i and location j. Mathematically, the time matrix is defined as:

$$T = \begin{bmatrix} 0 & \cdots & T_{1n} \\ \vdots & \ddots & \vdots \\ T_{n1} & \cdots & 0 \end{bmatrix}$$

Where:

I. T_{ij} is the time it takes to travel from location i to location j.

II. The diagonal elements $T_{ii} = 0$ because there is no travel time between a location and itself.

Given that we fetch real-time data for travel times, the matrix can be thought of as dynamically updating during different runs of the algorithm.

Each delivery location also has a time window $[T_{start}, T_{end}]$ meaning the delivery must occur within this window. The travel time T_{ij} must ensure that the arrival time at location j is within the allowable time window:

$$T_{arrival_j} = T_{departure} + T_{ij}$$

$$T_{start_j} \leq T_{arrival_j} \leq T_{end_i}$$

Where:

- $T_{arrival_j}$ is the time the vehicle arrives at location j.
- $T_{departure_i}$ is the time the vehicle departs from location i.
- T_{start_j}, T_{end_j} represent the start and end of the delivery time window at location j.

```
Time Matrix: [[ 0. 21.  6. 18.  6.  3.  4.]
 [23.  0. 19.  6. 25. 20. 20.]
 [ 4. 21.  0. 18.  8.  1.  8.]
 [19.  4. 15.  0. 24. 16. 21.]
 [ 7. 26. 10. 22.  0.  8.  7.]
 [ 3. 20.  6. 16.  9.  0.  7.]
 [ 8. 23. 10. 23.  9.  8.  0.]]
```

Figure 1 - Example of time matrix output for 7 addresses

2.4 Time Calculation

The element T_{ij} is determined by querying an external API (e.g., Google Maps) AI based that predicts travel time. The travel time between two locations is usually affected by factors such as:

1. Distance between locations**:** The geographic distance between i and j.
2. Traffic conditions**:** Congestion or time of day.
3. Mode of transportation**:** Driving, walking, cycling, etc. The chosen one was “driving”, but if we wanted to simulate it like a motorcycle, we could apply a multiplication factor to adjust key variables.
4. Other delays**:** Weather conditions or road construction.

```
Best Path: [0, 4, 6, 5, 2, 3, 1]
Best Time: 66.0
```

Figure 2 - Example of best route output

2.5 Ant Colony Optimization (ACO)

ACO is a nature-inspired algorithm that mimics the behavior of real ants (*or bees…*) searching for food. It is particularly useful for solving optimization problems, such as finding the shortest path in graphs (optimal path). The key mathematical components are:

- Pheromone Levels T_{ij} Represent the desirability of traveling between two locations i and j. The pheromone level is updated based on the performance of solutions.
- Heuristic Information: The desirability of choosing a particular path is influenced by a heuristic value based on travel time between locations.
- Probability of Choosing Next Vertex: Given the current location i, the probability of moving to location j is:

$$\boldsymbol{P_{ij}} = \frac{[T_{ij}].[N_{ij}]^{\beta}}{\sum k \notin visited [T_{ik}]^{\alpha}.[N_{ik}]^{\beta}}$$

where:

T_{ij} is the pheromone level between i and j,

N_{ij} is the heuristic information (e.g., inverse of travel time),

α controls the influence of pheromone, and β controls the influence of the heuristic.

2.5.1 The pheromone Update:

The pheromone trail is updated after each iteration. Pheromones evaporate over time (evaporation rate ρ) and are reinforced based on the quality of solutions:

$$T_{ij} = (1 - \rho).T_{ij} + \Delta_{ij}$$

Where $\Delta T_{ij} = \frac{Q}{L}$ if the edge ij is part of the best solution, and 0 otherwise. Q is a constant, and L is the total travel time of the path.

3. CODE EXPLANATION

The code uses Ant Colony Optimization to find an optimal delivery route for a set of addresses, considering time windows for deliveries.

3.1 Key Components

Google Maps API Integration: A *generate_time_matrix* computes the travel times between each pair of delivery locations using the Google Maps API. These travel times are stored in a matrix for further use by the ACO algorithm.

3.2 Ant Colony Optimization Setup:

- Pheromone Initialization: All pheromones are initially set to 1. Each iteration of the algorithm updates the pheromone levels based on the paths explored.
- Probability Calculation: Ants choose the next location to visit based on the pheromone levels and travel time using the formula for P_{ij} (as described above in the section 2.5).
- Pheromone Update: After each iteration, pheromones are updated to reinforce good paths and evaporate older paths.

3.3 Simulation of Ants:

The ants explore paths based on the probabilities, visiting each location within the constraints of time windows. The best path found by the ants is stored and refined over iterations.

3.4 Pheromone Update Mechanism:

The pheromones are updated after all ants complete their paths. Shorter delivery times receive a higher pheromone update, ensuring that paths with lower delivery times are reinforced.

3.5 Example of a delivery scenario

Given a set of delivery locations and their respective time windows:

- **A.** Location A (8:00 AM - 8:15 AM)
- **B.** Location B (9:00 AM - 9:15 AM)
- **C.** Location C (9:45 AM - 10:00 AM)

The algorithm will compute the optimal path and the sequence in which the deliveries should be made, minimizing the travel time between them while ensuring each location is visited within its time window.

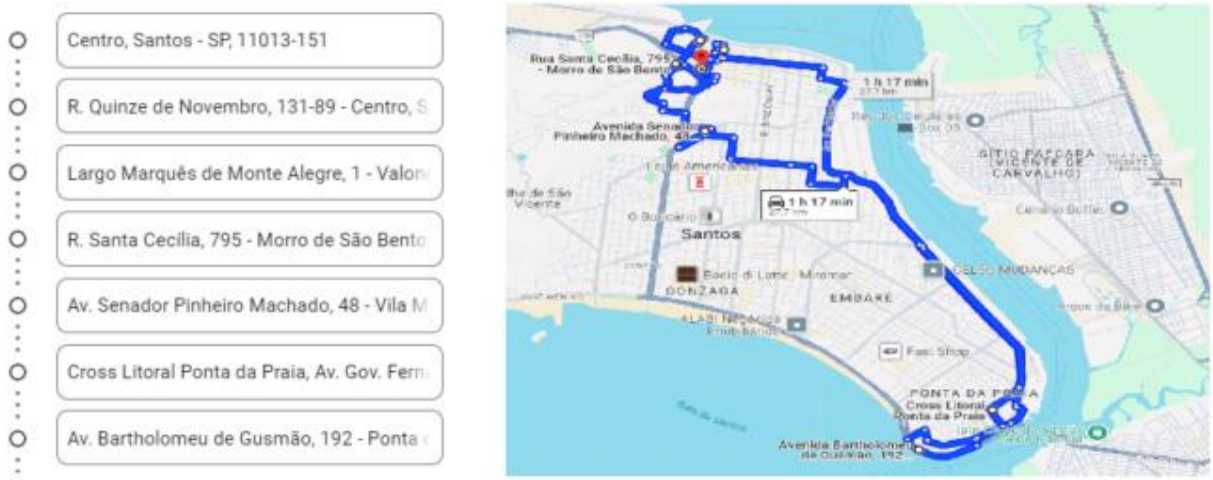


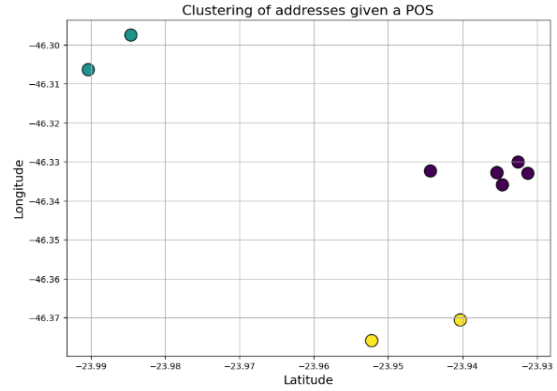
Figure 3 - Example of best route generated for 7 addresses in Santos

4. APPLYING AI

The idea of using AI in this project is to optimize the computational complexity of the algorithm, as well as ensure that each *PDV* (POS or point of sale, in English) has maximum delivery efficiency based on historical data.

4.1 Clustering for Efficient Grouping of Addresses

Applying unsupervised learning techniques like K-Means clustering or Gaussian Mixture Models to group nearby addresses before solving the routing problem. This reduces the complexity by tackling smaller clusters of addresses in separate optimization runs.



	latitude	longitude	cluster
0	-23.935368	-46.332835	0
1	-23.944316	-46.332403	0
2	-23.934640	-46.335945	0
3	-23.931205	-46.332993	0
4	-23.932514	-46.330103	0
5	-23.984628	-46.297525	1
6	-23.990365	-46.306415	1
7	-23.935368	-46.332835	0
8	-23.952223	-46.375928	2
9	-23.940299	-46.370607	2

Figure 4 - Clustering of addresses example

4.2 Optimizing pheromones

Using a machine learning classifier (e.g., Decision Trees, XGBoost) to predict the best pheromone update rules in Ant Colony Optimization based on past route performance.

5. DELIVERY CASE EXAMPLE: AI-ENHANCED DELIVERY LOGIC

A POS in Santos, Brazil, needs to deliver packages to seven locations, each with a strict time window. Real-time traffic conditions influence the route, and AI is used to optimize both the grouping of deliveries and the overall route.

5.1 Key Information

1. Addresses**:** The delivery locations are spread across Santos, each having a 15-minute delivery window.
2. Real-Time Traffic Data**:** We retrieve live traffic data using the Google Maps API to estimate travel times between locations.
3. Goal**:** Use AI to group orders efficiently, ensuring the best routes are chosen while respecting time windows and minimizing travel time.

5.2 AI-Enhanced Delivery Process

- Order Grouping with AI**:** Before optimizing the route, AI-based clustering algorithms (e.g**.,** K-Means or DBSCAN) are used to group orders based on location proximity, traffic patterns, and time windows. This ensures that the vehicle visits clusters of nearby locations that share similar time constraints. For instance, locations in close geographic proximity (e.g., *Pr. dos Andradas* and *Largo Marquês de Monte Alegre*) might be grouped together and scheduled for delivery within similar time slots.
- Route Planning with AI-Optimized Pheromones**:** Ant Colony Optimization (ACO) is used to plan the routes for each group of deliveries. In this AI-enhanced version, Reinforcement Learning (RL) adjusts the pheromone levels dynamically based on real-time data. As the system collects more information from past deliveries (e.g., traffic patterns at specific times), it learns to optimize pheromone deposits for better future route decisions. Routes that consistently perform well under certain traffic conditions leave stronger pheromone trails

6. GAINS FROM AI-DRIVEN DELIVERY ROUTE OPTIMIZATION

1. Customer Satisfaction: Faster, more predictable deliveries increase trust and improve the overall customer experience.
2. Continuous Improvement: The system learns from data, continuously refining routes for better future performance, and enabling predictive capabilities.
3. Efficiency: Reduced delivery times and better adherence to time windows, improving overall delivery throughput and customer satisfaction.

7. CONCLUSION

By grouping nearby deliveries using AI clustering algorithms and applying Ant Colony Optimization (ACO) for route planning within each group, the delivery system ensures minimal travel time and on-time deliveries. The AI-enhanced pheromone optimization further improves route efficiency over time, making the system more intelligent and adaptive for each POS.