Ride-Hailing Revenue Optimization using Integer Linear Programming (ILP)

Course: Network Optimization

Instructor: Marco Pranzo

Submitted by: Danial Moafi, Setareh Soltani

June 2025

UNIVERSITÀ DI SIENA 1240

Table of Contents

1. Introduction

In recent years, ride-hailing services such as Uber, Bolt, and FreeNow have transformed urban mobility by offering flexible transportation options. One of the key challenges faced by drivers working in these platforms is selecting the optimal sequence of rides that maximize their daily profit while respecting time and location constraints.

This project addresses this challenge by developing an optimization-based system that helps a driver in Rome choose the most profitable sequence of ride requests during a working shift. The model takes into account various real-world constraints such as ride availability windows, travel time between zones, and operational costs of driving without passengers.

To solve this problem, we formulate it as an Integer Linear Programming (ILP) model. The ILP approach allows us to capture the discrete decision-making nature of the problem and use advanced solvers like Gurobi to find an optimal solution. The objective is to maximize the driver’s net profit, defined as total ride revenue minus the cost of empty (non-passenger) movements between locations.

2. Data Description

The dataset used in this project consists of simulated ride requests in the city of Rome, as well as geospatial and temporal data required to model the urban transportation network. The data is categorized into the following parts:

Hexagonal Zoning (H3):

- The city is divided into H3 hexagonal zones to spatially abstract locations. Each zone is assigned a weight based on estimated demand, which is influenced by proximity to landmarks (e.g., airports, train stations, tourist hotspots).
- A distance matrix is computed using H3 cell centers and scaled to approximate road distances (via a 1.3 multiplier).

Ride Dataset:

- 100 synthetic ride requests are generated based on the demand weights of hexagonal zones. Each ride includes:
- Origin and destination hexagons
- Available start and end time windows
- Estimated duration based on speed sampled between 15–25 km/h
- Price calculated from a base fare (3 EUR) plus a per-km fare between 1.1–1.6 EUR/km

Driver Profile:

- A simulated driver is defined with:
- A fixed shift window (e.g., 8:00 AM to 6:00 PM)
- A starting and ending location (both defined as H3 zones)

The data generation scripts are fully modular and reproducible, enabling parameter adjustments for further testing or experimentation.

3. Conceptual Model

The conceptual model defines how ride-hailing operations are abstracted for optimization purposes. In this project, we simulate a single driver operating within the city of Rome, navigating a dynamic ride environment under real-world constraints such as time windows, location changes, and travel delays.

The core idea is to represent the driver’s day as a sequence of decisions: which ride to accept next, whether to move without a passenger, and when to wait. These decisions must collectively maximize net profit, defined as:

Net Profit = Total Ride Revenue - Empty Travel Cost

Key Components:

- Driver:

  Defined by a start location, end location, and a working time window. The driver must begin and end within these constraints.

- Ride:

  Each ride includes an origin, destination, price, start time window, and duration. The ride must be started within its time window.

- Map:

  The urban space is discretized into H3 hexagonal zones. Distances between zones are used to compute travel time and cost.

- Decisions:
- Whether to accept a ride
- Sequence/order of accepted rides
- Whether to travel empty between locations
- When to start each action (ride or movement)

Flow Logic:

- The driver begins at a specified location and time.
- From there, they may:
- Take a ride (if feasible)
- Move to a different zone empty (incurs cost)
- Wait (if they arrive early)
- After each action, the driver's current time and location are updated.
- The driver must end the shift at the designated end location before time runs out.

This model supports flexible scheduling and accounts for spatial-temporal feasibility, allowing intelligent sequencing of rides and transitions for optimal financial return.

4. Optimization Model

The optimization model is formulated as a Mixed-Integer Linear Program (MILP), solved using Gurobi. The goal is to maximize the driver's net profit by selecting a feasible sequence of rides and transitions that fit within time and location constraints.

Objective Function

Maximize:

$\Sigma (x_r \times \text{price}_r) \quad - \quad \Sigma (e_{r,ij} \times \text{cost}_{ij})$

for all $r \in R$, and $i, j \in Z$

Where:

- x_r: Binary variable, 1 if ride r is selected
- $e_{r,ij}$: Binary variable, 1 if an empty move from zone i to j occurs after ride r
- price_r: Revenue generated by ride r
- cost_{ij}: Cost of traveling from zone i to j without a passenger

Decision Variables

- $x_r \in \{0, 1\}$: Whether ride r is selected
- $y_{sr} \in \{0, 1\}$: Whether ride r is taken immediately after ride s
- $e_{r,ij} \in \{0, 1\}$: Empty move from zone i to j after ride r
- $t_r \in \mathbb{R}^+$: Start time of ride r

Constraints

1. Time Window Constraint

For each ride r:

$$available\_at_r \leq t_r \leq end\_at_r$$

2. Ride Sequencing Feasibility

If ride r is taken after ride s, then:

$$t_r \geq t_s + duration_s + travel\_time_{s \to r}$$

3. Flow Conservation

- Each selected ride must be followed by another ride or an empty move — or be the final action.
- Ensures continuity: the driver cannot “teleport”.

4. Start and End Location Compliance

- The first movement must start at the driver’s initial zone.
- The final action must allow the driver to return to the end zone before shift ends.

5. Shift Time Limit

- All rides and movements must be completed within the driver’s working hours.

This optimization model supports both scheduled rides and strategic empty moves, providing flexibility to explore profitable combinations and reduce idle time.

5. Model Implementation and Execution

The optimization model was implemented in Python 3.10 using the Gurobi Optimizer. The structure of the implementation is modular, with different components for data generation, data modeling, and optimization logic.

Programming Environment

- Language: Python
- Solver: Gurobi 11.0
- Libraries: NumPy, pandas, h3, gurobipy

Folder Structure

- Data/: Contains scripts to generate the synthetic map and ride data based on real-world GPS data from Rome.
- Models/: Defines classes for Ride, Driver, and Map, encapsulating their attributes and behavior.
- main.py: Contains the MILP model construction, variable definitions, constraints, and the objective function.

Output Files

- report.txt: A textual itinerary showing rides, empty moves, revenues, costs, and final profit.
- data_frame.csv: A structured CSV logging each event (ride, wait, empty move) with geospatial and temporal data.
- model.lp: The full mathematical model exported in LP format for inspection and debugging in Gurobi GUI.

6. Results and Evaluation

The model successfully selected a sequence of rides and empty movements that maximized the driver's net profit while respecting all spatial and temporal constraints.

Key Metrics from a Sample Run:

- Total Revenue from Rides: €87.30
- Total Empty Movement Cost: €14.20
- Net Profit: €73.10
- Number of Rides Selected: 9
- Total Working Time Used: 7 hours 50 minutes

Insights:

- The model prioritizes high-revenue rides that are geographically and temporally compatible.
- In some cases, inserting an empty movement strategically between rides increases total profit.
- Waiting is minimized, and time gaps are filled efficiently using ride chaining or short repositioning moves.

7. Limitations and Future Work

Limitations:

- The model assumes deterministic travel times and does not incorporate traffic variations.
- Ride requests are synthetic and based on heuristics derived from real data but may not fully reflect actual demand.
- The driver’s preferences (e.g., preferred zones or time off) are not considered.

Future Work:

- Integrate real-time traffic data for more accurate travel time estimation.
- Add stochastic elements to account for ride uncertainty and delays.
- Extend the model to multi-driver scenarios for fleet-level optimization.
- Develop a real-time recommender interface to assist drivers on the road.

8. Conclusion

This project demonstrates how Integer Linear Programming (ILP) can be effectively used to optimize ride-hailing driver schedules. By modeling rides, time windows, and empty movements in a unified MILP framework, we achieved a robust optimization system that maximizes profit under realistic operational constraints.

The modular codebase and data-driven approach enable easy adaptation to different cities, ride datasets, or driver behaviors. The project highlights the potential of operations research and optimization techniques in addressing real-world mobility challenges.

9. References

1. Gurobi Optimization, LLC. (2024). Gurobi Optimizer Reference Manual.
2. H3: Uber’s Hexagonal Hierarchical Spatial Index.
3. Toth, P., & Vigo, D. (2014). Vehicle Routing: Problems, Methods, and Applications.
4. OpenStreetMap Rome GPS Dataset (sample taxi data).
5. Python libraries: numpy, pandas, h3, gurobipy.