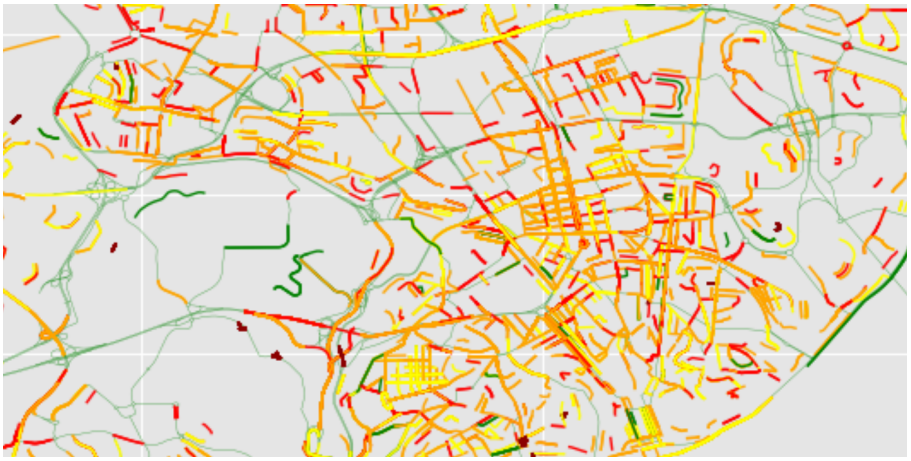
TÉCNICO LISBOA



# Road Traffic Flow Inference based on Aggregated Mobile Phone Data

Lisbon Case Study

**André Tiago Ferreira Barreiros**

Thesis to obtain the Master of Science Degree in

## Electrical and Computer Engineering

Supervisor(s): Prof. João Pedro Gomes
Prof. Filipe Moura

### Examination Committee

| | |
|---|---|
| Chairperson: | Prof. Full Name 1 |
| Supervisor: | Prof. Full Name 2 |
| Member of the Committee: | Prof. Full Name 3 |

**July 2025**

Dedicated to someone special...

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

# Resumo

Inserir o resumo em Português aqui com um máximo de 250 palavras e acompanhado de 4 a 6 palavras-chave.

**Palavras-chave:** palavra-chave1, palavra-chave2, palavra-chave3,...

## Abstract

Insert your abstract here with a maximum of 250 words, followed by 4 to 6 keywords.



**Keywords:** keyword1, keyword2, keyword3,...

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

# Chapter 2

# Background and State-of-the-Art

The objectives of this chapter are to frame the topic of the work within the context of Urban Mobility, Road Traffic Engineering, and Big Data Processing, provide a brief overview of the current landscape of solutions for the problem of road traffic flow prediction, and present the literature that will serve as the foundation for all the work to be carried out.

## 2.1 Background

This work seeks to model and infer road traffic volumes from Vodafone mobile-phone data, positioning the work squarely within the domain of Road Traffic Engineering. Accurate volume estimates are indispensable for computing Level-of-Service (LOS) and broader performance metrics, as thoroughly detailed in the *Highway Capacity Manual* [1] and *Traffic Flow Fundamentals* [2].

Figures 5.9 and 2.2 illustrate classic speed–flow and speed–density relationships for a Chicago (USA) corridor; each point links measured speed to concurrent flow or density levels. Beyond LOS analysis, reliable traffic counts underpin geometric design decisions (e.g., lane requirements, shoulder widths), safety assessments that normalize crash frequencies by exposure, signal-timing and other ITS operations that allocate green time where demand is highest, and both environmental and economic evaluations that forecast emissions, noise, travel-time savings, and cost–benefit ratios. Pattern recognition in the data can also reveal origin–destination matrices within the city—indispensable inputs for strategic traffic modeling and urban mobility management.

Another critical challenge of this work is the problem of processing data with large dimensions, as we were dealing with data that amounts to several gigabytes. In this sense, it falls within the areas of Data Science and Big Data processing, as using techniques already studied and deepened in these areas helped to optimize the results obtained. Data preparation, parallel processing and knowledge discovery were essential procedures when approaching the problem and doing exploratory data analysis.

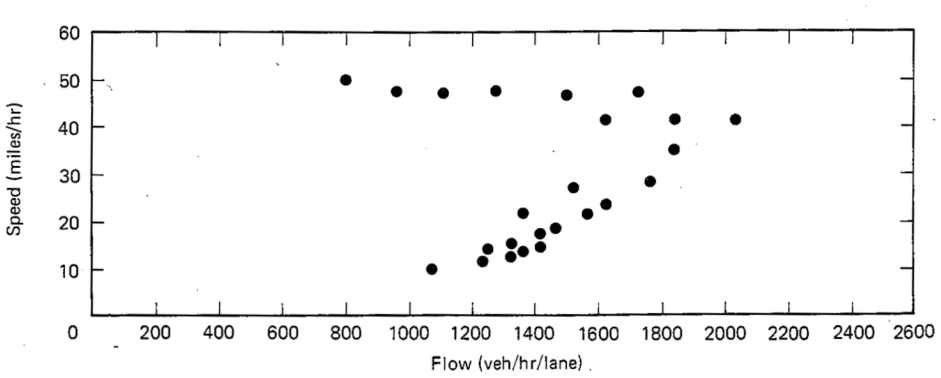



Figure 2.1: Points from a relationship between speed and flow. Source: [2]

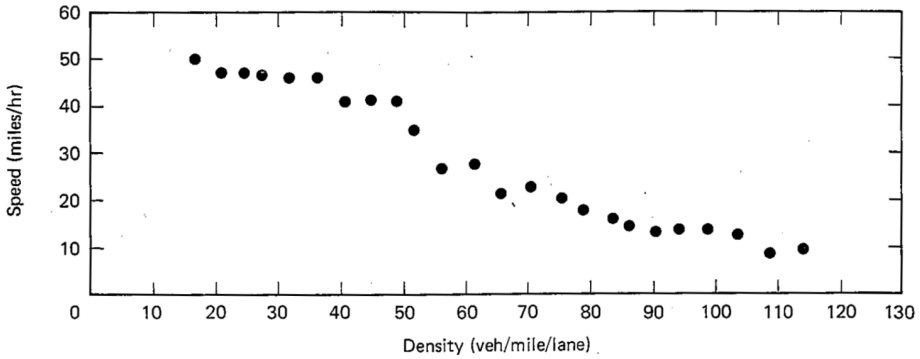



Figure 2.2: Points from a relationship between speed and flow density. Source: [2]

## 2.2 State of the Art

This section presents some works related to the topic of this study. The issue of road traffic prediction has been studied for many decades, but with the new technologies and techniques available in recent years, the state of the art is constantly changing.

### 2.2.1 Previous Results for Challenge 70

LxDATALAB's Challenge 70 is an initiative created in 2021, and since that year, several projects have been carried out to address the challenge. All these works always use at least part of the provided data, mostly from Vodafone. Table 2.1 presents the works shared briefly.

Although these works use the same data and many of them even have the code available, almost all the proposed objectives and interpretations differ from the objectives intended in this work. Many characterizations were made that are not relevant for road traffic inference. The only exception is the work by researcher Mafalda Zúquete, where she creates the flow map 2.3 that infers traffic intensity on almost all roads in Lisbon.

This last project was based on the methodology used in the paper [3], where the data is very similar to Vodafone's data, but in the city of Milan. It focuses on extracting information about people's movements by calculating the Wasserstein distance (2.2.2) between consecutive density profiles to determine the main directions followed by people and the associated volumes that optimize movements.

| Name | Year | Institution | Description |
|---|---|---|---|
| MOURATO, Rodrigo | 2024 | ISCTE-IUL | Group work focusing on data characterization, focusing on the airport |
| SIMÕES, Rodrigo | 2024 | ISCTE-IUL | Group work with descriptive analysis of concentration and movement of people, using prediction models |
| COUTINHO, Bruno | 2024 | ISCTE-IUL | Paper with data analysis and predictions of arrivals and departures from Lisbon using LSTMs |
| CHERNVAEVA, Anastacia et al. | 2023 | ISCTE-IUL | Group work with data characterization during large events, using data analysis and visualization techniques. Predictions of data using machine learning models |
| DIAS André et al. | 2022 | ISCTE-IUL | Group work with presentation slides, various data characterizations but without much depth |
| FRANCISCO, Bruno | 2022 | ISCTE-IUL | Thesis with large-scale visualization of all received data, aiming to characterize tourist behavior in Lisbon |
| ZÚQUETE, Mafalda | 2022 | NOVA-IMS | Presentation slides of an unfinished research project showing a relevant flow map for this report's work, based on a linear programming problem using Wasserstein distance |
| LEAL, Daniel Romão | 2022 | ISCTE-IUL | Thesis with a data mining approach, performing statistical analysis, visualization, and clustering |

Table 2.1: Previous results shared by LxDATALAB

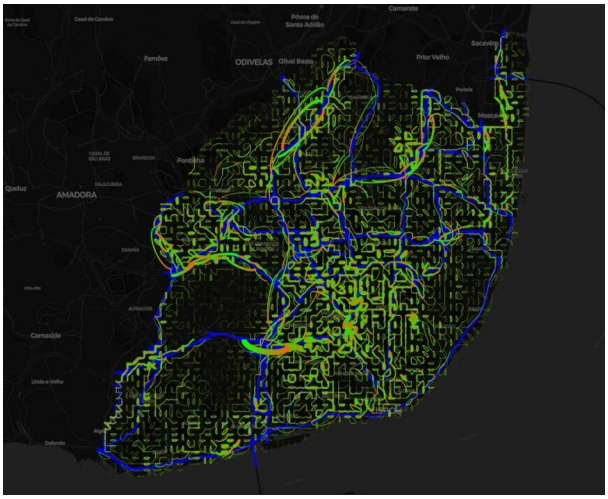



Figure 2.3: Flow map from Mafalda Zúquete's work

### 2.2.2 Wasserstein Distance

The Monge-Kantorovich mass transfer problem, described in [4], can help understand this distance. Given a mound of sand with mass distribution $\rho_0$ and a pit with equal volume and mass distribution $\rho_1$, there is a minimal cost to transfer the sand to the pit. The cost to move the mass depends both on the distance from the origin to the destination point and the amount of mass moved along that path.

This approach uses the notion of Wasserstein distance as an optimization function. In the space $\mathbb{R}^n$ using the Euclidean metric, let $\rho^0$ and $\rho^1$ be two probability density functions such that $\int_{\mathbb{R}^n} \rho^0 = \int_{\mathbb{R}^n} \rho^1$. For all $p \in [1, +\infty)$, the $L^p$-Wasserstein distance between $\rho^0$ and $\rho^1$ is:

$$W_p(\rho^0, \rho^1) = \left( \min_{T \in \mathcal{T}} \int_{\mathbb{R}^n} \|T(x) - x\|^p \rho^0(x) dx \right)^{\frac{1}{p}} \tag{2.1}$$

where:

$$\mathcal{T} := \left\{ T : \mathbb{R}^n \to \mathbb{R}^n : \int_B \rho^1(x) dx = \int_{\{x : T(x) \in B\}} \rho^0(x) dx, \forall B \subseteq \mathbb{R}^n \text{ bounded} \right\}. \tag{2.2}$$

$\mathcal{T}$ is the set of all possible maps that transfer mass from one configuration to another. Knowing the actual value of the Wasserstein distance $W_p$ is not as important as the optimal mass transfer map that finds the $\arg\min$ in equation (2.1), which represents the paths along which the mass is transferred. To discretize the problem, it is abstracted into a graph $G$ with $N$ nodes corresponding to various grids. This procedure gives an approximation of the Wasserstein distance and provides an algorithm to calculate the optimal paths. Starting with an initial mass $m_j^0$ and a final mass $m_j^1$, for $j = 1, \ldots, N$, distributed at the nodes of the graph, we aim to optimally distribute the first mass into the second. We denote by $c_{jk}$ the cost of transferring one unit of mass from node $j$ to node $k$, and by $x_{jk}$ the (unknown) mass moved from node $j$ to node $k$. The problem is then formulated as follows:

$$\text{minimize} \quad H := \sum_{j,k=1}^{N} c_{jk} x_{jk} \tag{2.3}$$

subject to

$$\sum_k x_{jk} = m_j^0 \quad \forall j, \quad \sum_j x_{jk} = m_k^1 \quad \forall k \quad \text{and} \quad x_{jk} \geq 0. \tag{2.4}$$

Defining

$$x = (x_{11}, x_{12}, \ldots, x_{1N}, x_{21}, \ldots, x_{2N}, \ldots, x_{N1}, \ldots, x_{NN})^T, \tag{2.5}$$

$$c = (c_{11}, c_{12}, \ldots, c_{1N}, c_{21}, \ldots, c_{2N}, \ldots, c_{N1}, \ldots, c_{NN})^T, \tag{2.6}$$

$$b = (m_1^0, \ldots, m_N^0, m_1^1, \ldots, m_N^1)^T, \tag{2.7}$$

and the matrix

$$A = \begin{bmatrix} 1_N & 0 & 0 & \dots & 0 \\ 0 & 1_N & 0 & \dots & 0 \\ 0 & 0 & 1_N & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1_N \\ I_N & I_N & I_N & \dots & I_N \end{bmatrix}, \tag{2.8}$$

where $I_N$ is the identity matrix $N \times N$ and $1_N = (1\ 1\ \dots\ 1)$, our problem is written as a linear programming problem: minimize $c^T x$, subject to the conditions $Ax = b$ and $x \geq 0$. The result of the algorithm is a vector $x^* := \arg\min c^T x$ whose elements $x^*_{jk}$ represent how much mass moves from node $j$ to node $k$, applying the minimum cost mass movement.

In the paper by Balzotti et al. *Understanding Human Mobility Flows from Aggregated Mobile Phone Data* [3], the Wasserstein distance was calculated, and results like the one in figure 2.4 were obtained. This methodology, although producing significant results, raises some doubts.


Figure 2.4: Flows for an event area in Milan. Source: [3]

This approach is based on the assumption of several ideas that make it quite questionable. For example, it was assumed that people could move in any direction, ignoring obstacles and that they are all the same. Moreover, it was not possible to distinguish vehicle traffic from pedestrian traffic.

The work in figure 2.3 already uses another approach that avoids assuming the first idea. People can only move on referenced roads and not through the entire space. For the abstraction in the graph $G$ with $N$ nodes, in this case, the travel time between two points was used as the measure of distance instead of Euclidean distance. The travel time depends on the maximum speed at which a car can travel on the roads connecting the nodes. This last idea, however, assumes that cars can always travel at maximum speeds, which is not the case during peak hours, thus opening up room for improvements

in the algorithm. Also, the work was not finished as the project was canceled halfway, leaving few to no results.

### 2.2.3 AI Models

Machine learning and deep learning models have proven to be very important in predicting road traffic. Here are some examples of interest:

- Master's thesis using the same data as Waze: João Vaz (2022)[5]. This work, which answers LxDATALAB's Challenge 50, had three objectives: 1. Develop a predictive model capable of forecasting congestion, using Waze and IPMA data; 2. Create a traffic fluidity indicator; 3. Visualize in a dashboard where predictions can be viewed. The Waze data is indeed the same as for Challenge 70. The traffic indicator is shown by a prediction of delay calculated in an algorithm. The models used were Distributed Random Forest and later, because it obtained better results, XGBoost (algorithm with gradient boosted decision trees). They also created derived features, defining whether a specific data point was from a holiday/weekend, the day before a holiday/weekend, weekdays, peak hours (7h/10h or 18h/20h), and whether it was morning, afternoon, or night. They also included historical data: from speed, level, length, and delay attributes, calculating the median, average, and maximum of values in the last 30 minutes, 1 hour, and 1 hour 30 minutes. These last features significantly improved the results.

- Articles using data similar to Vodafone's: Cecaj et al. (2020)[6], which compares different methods for predictions, stating that deep learning methods are preferable due to simplicity and speed of use and for identifying complex temporal dependencies. Statistical models can sometimes be more accurate, require less data, but are much more complicated to build. In the end, they do not argue that one is better than the other, just that the applicability depends on the situation. Sagl et al. (2013)[7] is an older work but shows how Self-Organizing Maps (SOM), an Unsupervised Neural Network, can also be used for mobility pattern identification and clustering in the data.

- The use of different data types aids prediction: Ming Ni (2014)[8] explores how useful data from social networks can be obtained and its benefits in traffic prediction models. They applied Neural Networks, Auto-regressive Models, Support Vector Regression (SVR), K-Nearest Neighbors (KNN) to traffic sensor data. The SVR model gave the best results. Like this, Da Zhang (2018)[9] used other data, in this case from weather, to optimize their prediction. With data from 39,000 sensors in California (USA), they used the models Deep Gated Recurrent Neural Network, Simple RNN, Long Short-Term Memory (LSTM), AutoRegressive Integrated Moving Average (ARIMA), SVR, and Random Forest. The DGRNN provided the lowest prediction error.

- LSTM models have been widely used for traffic flow prediction: Fu et al. (2016)[10] explored the use of LSTM and GRU neural network methods for traffic flow prediction, highlighting that existing models like ARMA and ARIMA are primarily linear models and cannot describe the stochastic and nonlinear nature of traffic flow. Villarroya et al. (2022)[11] developed a traffic prediction

model for the city of Valencia, using data from electromagnetic loops and also applying two LSTM models. Tian et al. (2018)[12] proposed traffic flow prediction based on LSTM with missing data, where multi-scale temporal smoothing is applied to infer missing data and the prediction residue is learned through their approach.

### 2.2.4 Statistical and Other Models

Statistical models have not seen much development in this area but have considerable weight:

- Statistical models: Min and Wynter (2011)[13] developed a *Multivariate Space-Time Autoregressive Moving Average* (MSTARMA) model for real-time traffic prediction, capturing spatio-temporal correlations and allowing comprehensive traffic pattern analysis by considering spatial and temporal interactions in the data. Kumar (2015) [14] used data from three main roads in Chennai (India) for three consecutive days to predict short-term traffic with the SARIMA model. This model overcomes data availability challenges and shows better performance compared to traditional methods.

- Hybrid model: The paper by Li et al. (2017) presents a hybrid model combining ARIMA and Radial Basis Function Neural Networks (RBF-ANN). The proposed methodology is effective for short-term traffic flow prediction, demonstrating that "the hybrid model can better capture the linear and nonlinear patterns within traffic flow data and improve predictive modeling."

Another type of data found in some works is the *Call Details Record* (CDR), used by Elragal and Raslan (2014) [15], Batran et al. (2018)[16], or Caceres et al. (2012)[17]. This data type is not provided for this report's work but has interesting similarities. The CDR logs all the antennas to which a mobile phone connects and the moment in time it connects to each. Thus, by tracking the mobile phone's path through antennas near roadways 2.5, we can easily determine its speed.

An additional recent work of some relevance is by Carlos Lemonde et al. (2021)[18] which conducts a spatio-temporal analysis of multimodal traffic patterns, covering all major transport modes — road, rail, inland waterways — and active transport modes such as walking and cycling. This uses Waze data and also addresses the Big Data processing area.


Figure 2.5: Antennas near roadways for geolocation

Considering traffic-assignment models driven by observed counts, a recent Portuguese work by Gonçalves and Ribeiro[19], where they did an OD–matrix estimation with *TFlowFuzzy* case–study of Guimarães city, shows how purely stratified traffic counts can be leveraged to build a reliable urban four-step model when no prior origin–destination (OD) survey is available. Their workflow, implemented in PTV VISUM 15, begins with a detailed representation of the physical network (nodes, links, turns and zones), followed by an initial "flat" OD matrix, and then iteratively refines that matrix through the *TFlowFuzzy* algorithm until modeled link volumes match ground counts within accepted GEH and RMSE thresholds. A simplified outline of the procedure is given in Figure 2.6.

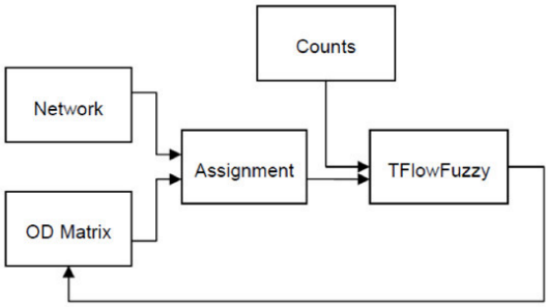



Figure 2.6: Iterative adjustment loop used by *TFlowFuzzy* to reconcile modeled flows with observed counts[19].

Applied to Guimarães (27 traffic zones, morning peak hour, 08:15–09:15), the calibrated model achieved, among other things, 97 % of count locations with GEH $< 5$ and a global RMSE of 7.9 %, comfortably inside the UK DfT/Visum guidelines, and also an OD matrix suitable for scenario testing—e.g. closing six streets around the historic center increased volumes on alternative corridors (Av. Conde de Margaride, Av. D. João IV, etc.) by up to 100 % and pushed several links above capacity.

Despite relying solely on link counts, the study confirms that *count-only* OD estimation can deliver decision-ready traffic models for medium-sized cities, provided a diligent calibration/validation strategy (GEH, RMSE, scatter-plots) is followed. The approach is particularly attractive when mobile-phone or household-survey data are unavailable, expensive, or raise privacy concerns.

### 2.2.5 Vehicle Counting Data Technologies and Solutions

In practice, road–traffic volumes can be obtained *without* running a traffic-assignment model. The three most common families of "direct" volume measurement are summarized below:

1. **Video-based automatic detection**: High-resolution CCTV, ANPR, or drone footage plays a key role in modern traffic monitoring.

   The process works by utilizing on-board convolutional neural networks to process footage, detecting, tracking, and classifying vehicles frame-by-frame, as documented in Siemens Portugal 2024 studies [20].

   Examples of this technology include Siemens SITRAFFIC CONCERT and TRAFFIC AIRNALYTICS,

Vivacity Labs “Smart Cameras”, and AllRead AI edge-units, as noted in references from Vivacity 2023 and AllRead 2022 [21, 22].

The typical output provides turning-movement counts, queue length, per-class speeds, and occupancy, all at a 1–5 second resolution.

- *Pros* – Lane-level accuracy, easy to retrofit on existing poles, provides rich multimodal data (cars, HGV, buses, cyclists, pedestrians).
- *Cons* – Performance can degrade at night or in heavy rain; GDPR constraints require number-plate blurring or on-device anonymisation; Expensive hardware.

2. **Connected-vehicle** / **probe data:** Connected-vehicle and probe data involve various technologies for tracking and analyzing traffic patterns.

   The process works by having on-board navigation units, smartphones, or fleet telematics transmit GPS traces every few seconds, with vendors aggregating millions of pings into speed and flow tiles, as referenced in TomTom 2024 Origin-Destination and INRIX 2024 studies [23, 24].

   Examples include TomTom *Origin–Destination* and *MOVE* API, INRIX Trips, and HERE Probe Data.

   The typical output consists of average speed, travel time, and statistically-expanded data per link, often updated every minute.

   - *Pros* – Wide-area coverage (including rural roads), no roadside hardware, captures real-time congestion propagation.
   - *Cons* – Bias toward newer/connected vehicles, penetration (share of cars connected) varies by country but is often low, vendor license fees apply.

3. **Other non-intrusive spot sensors** include several technologies designed for efficient traffic monitoring. Bluetooth/Wi-Fi re-identification involves solar-powered sniffers that match anonymised MAC addresses at successive sites to derive flow and travel time, as noted in the 2021 study on Bluetooth technology [25].

   Acoustic or radar “side-fire” units, such as kerb-side microphones or microwave sensors, count and classify vehicles without requiring pavement cuts, according to the Heimdall 2019 reference [26].

   Additionally, pneumatic tubes and wireless magnetometers offer low-cost, rapid deployment surveys, providing 24-hour axle counts and basic class and speed statistics, as documented in MetroCount 2020 [27].

   - *Pros* – Quick installation, minimal traffic disruption, useful for calibration of probe or video data.
   - *Cons* – Limited classification detail, expensive and easier to be damaged hardware, may overlook certain vehicles.

These complementary technologies are frequently fused in advanced urban traffic-management platforms (e.g. Siemens SITRAFFIC FUSION) to mitigate individual biases and provide a resilient, city-wide picture of traffic demand.

# Chapter 3

# Exploratory Data Analysis and Knowledge Discovery

To understand and validate the quality of the data before processing it, the need to do an exploratory data analysis is evident. The Python language was initially used and proved increasingly useful due to the GeoPandas library, which offers capabilities comparable to ArcGIS [28].

After receiving the data from LxDATALAB, some format corrections were needed for proper visualization, although most data was already ready. Once cleaned, the main tasks included checking consistency with documentation, identifying anomalies, and studying relationships between exploratory variables.

## 3.1 Vodafone Data

These are the main datasets from which inference will be made. The Vodafone dataset consists of three parts:

- ***Vodafone Quadrículas***: Latitude and longitude of the 3743 grid cells or "grids" (200 by 200 meters), indexed by a "Grid_ID".
- ***Vodafone Grelha***: Counts every 5 minutes of the number of mobile phones detected, remaining, entering, and exiting a grid. Many other metrics are available, but these are the most relevant.
- ***Vodafone Eixos***: Counts every 5 minutes of mobile phones entering and exiting via Lisbon's main traffic axes (A1, A5, Ponte 25 de Abril, etc.).

Several doubts that have arisen about the data and how it was collected were clarified in meetings with the LxDATALAB team and Vodafone, although with Vodafone, there was just one meeting.

### 3.1.1 *Vodafone Quadrículas*

This dataset maps the grids used by the Vodafone Grelha. The cleaned fields are shown in 3.1.

| Shape | Important Fields | | | | |
|---|---|---|---|---|---|
| (3743,5) | Grid_ID | Latitude | Longitude | Nome | Geometry |

Table 3.1: Vodafone Quadrículas Fields

A founded issue was the assumption that every grid cell spanned $200\,\text{m} \times 200\,\text{m}$. A simple calculation ($N_{grids} = 3743$ grids, each of area $A_{grid} = 40 \times 10^3 m^2$) yields a total area of $A_{Total} = 149.72 km^2$, which is higher than Lisbon's area ( $100 km^2$). Table 3.2 shows the results from calculating centroid distances, confirming the assumption was incorrect. According to LxDATALAB, this miscalculation occurred during the project's planning phase and was too late to be revised when discovered.

| Med. \Dist. | Vertical | Horizontal |
|---|---|---|
| Avg | 0.1562 | 0.1556 |
| Max | 0.1563 | 0.1557 |
| Min | 0.1561 | 0.1554 |

Table 3.2: Distance between grid centroids in meters

### 3.1.2 *Vodafone Eixos*

This dataset includes counts of mobile phones entering and exiting the city through the 11 main road axes. The count is performed every 5 minutes, identifying when a device moves from outside the grid area into it, and vice versa. The figure 3.1 shows the Vodafone Eixos polygons in red on top of the grids and OpenStreetMap edges (a filtered version). Data was provided for 2021, 2022 and 2023. Table 3.3 shows the dataset fields for January 2022 (initial version).

| Shape(01-22) | Gran. | Important Fields | | | | | | |
|---|---|---|---|---|---|---|---|---|
| (77723,7) | 5 min. | Eixo | C12 | C13 | Datetime | POINT_X | POINT_Y | Geometry |

Table 3.3: Fields in the Vodafone Eixos dataset

One key detail about the polygons in 3.1 is that if a mobile phone is present both in a grid and a polygon, it is assign as being in the grid and not in the ax's polygon, thus not being considered as outside of Lisbon grids.

Some files showed issues. For example, January 2022 contained many missing values for axis counts, shown in Table 3.4. Each axis, for months with 31 days, should have 12(obs./hour)×24(hours)×31(days)=8928 counts, which is significantly more than recorded.

A related problem is the discrepancy in the ratio of inputs to outputs, $C12/C13$. For the January 2022 file, this discrepancy is more pronounced, as shown in the table 3.2. The ratio values on the A1 and the Vasco da Gama Bridge are quite anomalous, since almost 3 cell phones to 2 entered over the course of the month in the case of the A1 and vice versa for the Vasco da Gama Bridge. In the graph 3.3 is illustrated the "camel's humps", known in the field of traffic management, which refer to the common pattern of variation in daily traffic levels, in this case on the bridge, but the discrepancy between entries and exits is large. The origin of this might have been just the missing data problem.

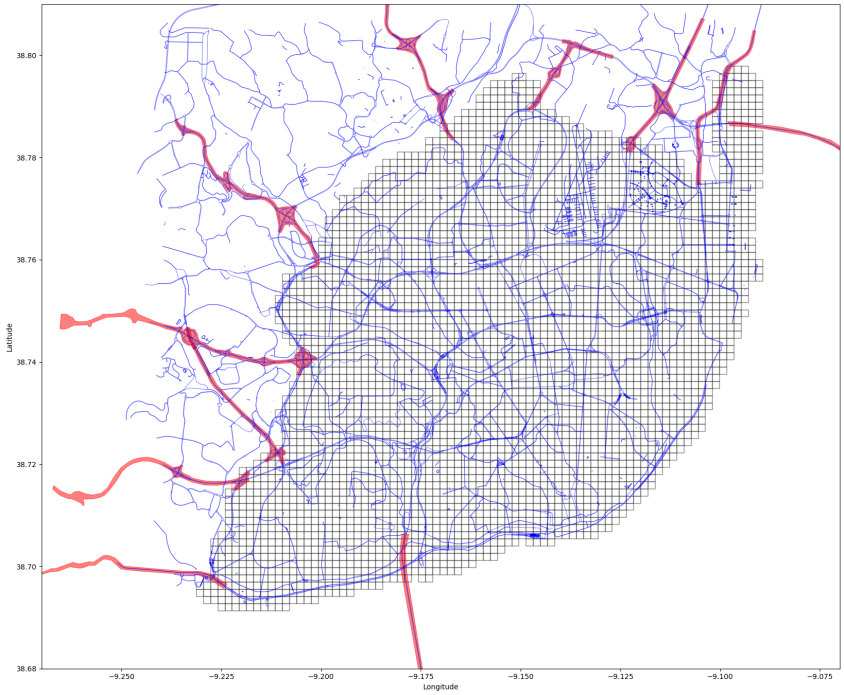



Figure 3.1: Polygons of Vodafone Eixos with Vodafone Quadrículas and OpenStreetMap Edges

| **Eixo** | **Count** |
|---|---|
| IC16 | 7098 |
| A5 | 7096 |
| Ponte 25 Abril | 7090 |
| IC2 (Sacavém) | 7077 |
| Marginal | 7074 |
| A1 | 7072 |
| N117 (Cabos Ávila) | 7057 |
| IC19 | 7054 |
| A36 (Túnel do Grilo) | 7043 |
| Ponte Vasco da Gama | 7035 |
| Calçada de Carriche | 7027 |

Table 3.4: Value counts of January 2022 (initial files)

Initially, according to LxDATALAB, antenna failures sometimes resulted in missing data, and that could be the reason. Later, it was found that the data was being extracted incorrectly. With the errors identified, it was possible to correct the extraction process and, in the new extraction, no missing or duplicate data was identified.

| Eixo | Result | Diff. to 1 |
|---|---|---|
| A1 | 1,463 | 0,463 |
| A36 (T. Grilo) | 0,919 | 0,081 |
| A5 | 1,057 | 0,057 |
| C. de Carriche | 1,042 | 0,042 |
| IC16 | 0,942 | 0,058 |
| IC19 | 1,003 | 0,003 |
| IC2 (Sacavém) | 1,035 | 0,035 |
| Marginal | 1,055 | 0,055 |
| N117 (Ávila) | 0,963 | 0,037 |
| Ponte 25 Abril | 0,964 | 0,036 |
| Pt. Vasco Gama | 0,663 | 0,337 |

Figure 3.2: Ratio between C12 and C13 on each axis

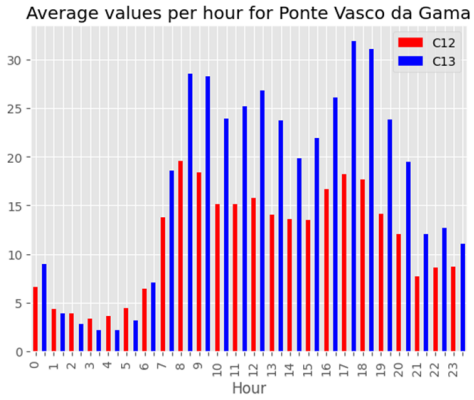



Figure 3.3: Entries and exits per hour on the Vasco da Gama Bridge in 01-2022

### 3.1.3 *Vodafone Grelha*

The analysis of the core data for this work is presented here. In the table 3.5 we find the most important fields that were chosen at start. This includes information such as counts of the number of cell phones entering, remaining and leaving each square of the 3743, among other information. The first files shared by LXDATALAB had periods of 5 min. from 09-2021 to 03-2022 and then 15 min. until 12-2023, but eventually all the files from 2021 to 2023 had periods of 5 minutes.

| Shape(09-21) | Gran. | Important Fields | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| (32627337, 27) | 5 min. | Grid_ID | Datetime | C1 | C3 | C5 | C6 | E7 | E8 | E9 |

Table 3.5: Vodafone Grelha Fields

- **C1** – Number of unique phones detected over 5 minutes;
- **C3** – Number of phones detected at the end of the 5-minute interval;
- **C5, C6** – Number of phones entering and exiting;
- **E7, E8, E9** – Minimum, average, and maximum dwell time.

After removing outliers using z-score and dropping low-importance columns, descriptive statistics (Table 3.6) were computed.

| | C1 | C3 | C5 | C6 | E7 | E8 | E9 |
|---|---|---|---|---|---|---|---|
| **mean** | 270.44 | 178.55 | 152.96 | 147.76 | 68.31 | 23.16 | 134.09 |
| **std** | 365.90 | 252.91 | 215.30 | 199.28 | 43.88 | 33.41 | 118.76 |
| **min** | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| **25%** | 55.47 | 32.10 | 30.15 | 32.74 | 0.00 | 10.81 | 34.00 |
| **50%** | 145.27 | 89.27 | 81.90 | 82.88 | 0.00 | 12.69 | 66.00 |
| **75%** | 336.64 | 218.89 | 188.57 | 183.54 | 15.61 | 19.30 | 300.00 |
| **max** | 11002.38 | 9328.17 | 7408.47 | 5219.50 | 300.00 | 300.00 | 300.00 |

Table 3.6: Descriptive statistics for C1, C3, C5, C6, E7, E8, and E9

To make it easier to understand what these ”C1, C3, C5 and C6” values are, in figure 3.4 is shown an example of how the values vary over time $t$ for 5 cell phones (U,V,X,Y and Z). Figure 3.5b shows

the C1 values on the grid map on 06-11-2023 at 5pm. With bar graphs, the image 3.6 was made to visualize data from 4 different squares on 26-09-2021.

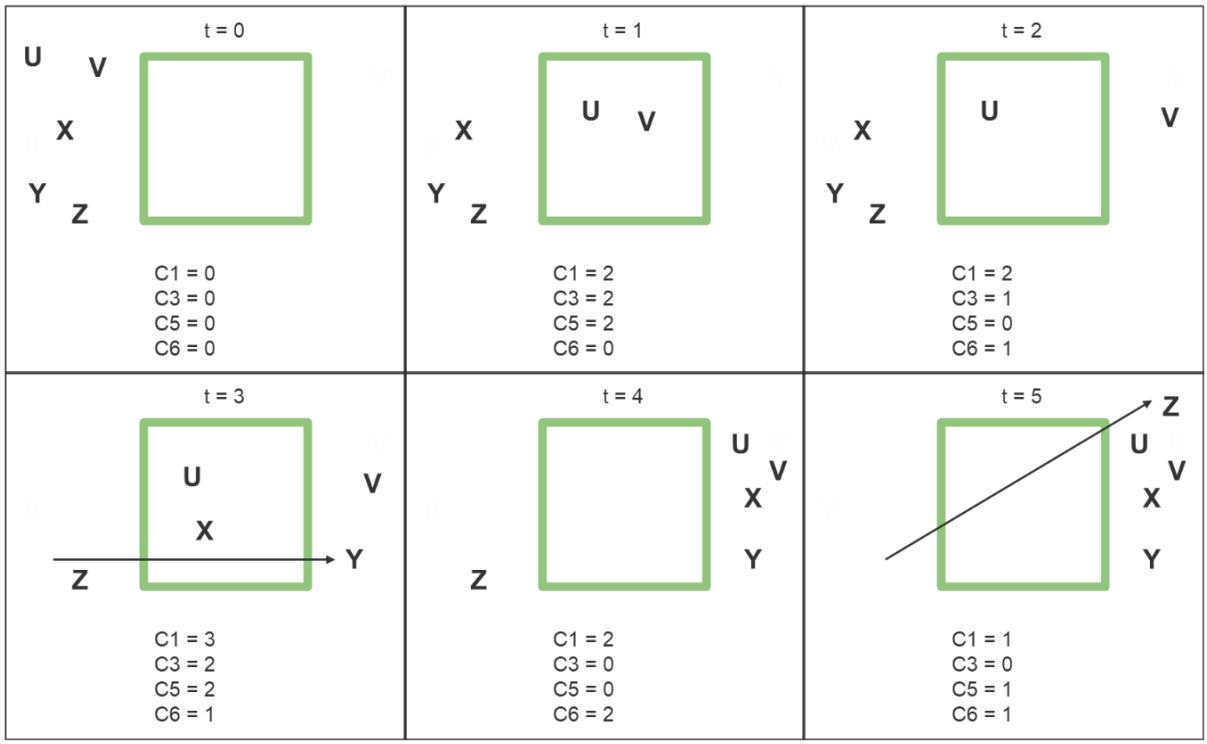
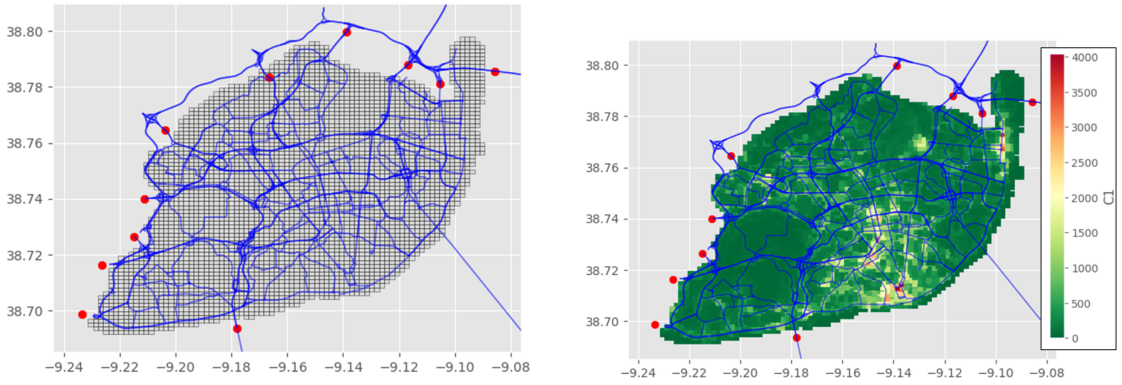



Figure 3.4: Demonstration diagram for values C1, C3, C5 and C6




(a) (b)

Figure 3.5: Map of squares (gray), road network (blue) and axes (red) without C1 values (a) and with C1 values (b). Snapshot of the data from 06-11-2023 at 17h.

Looking closely at the night hours at the “Parque das Nações”, something odd is depicted: the C6 values are bigger than the C1 values. It is natural to question this output. Right away, we can get suspicious about the actual functioning of these indicators. A deeper reflection on this is done in the next sections.

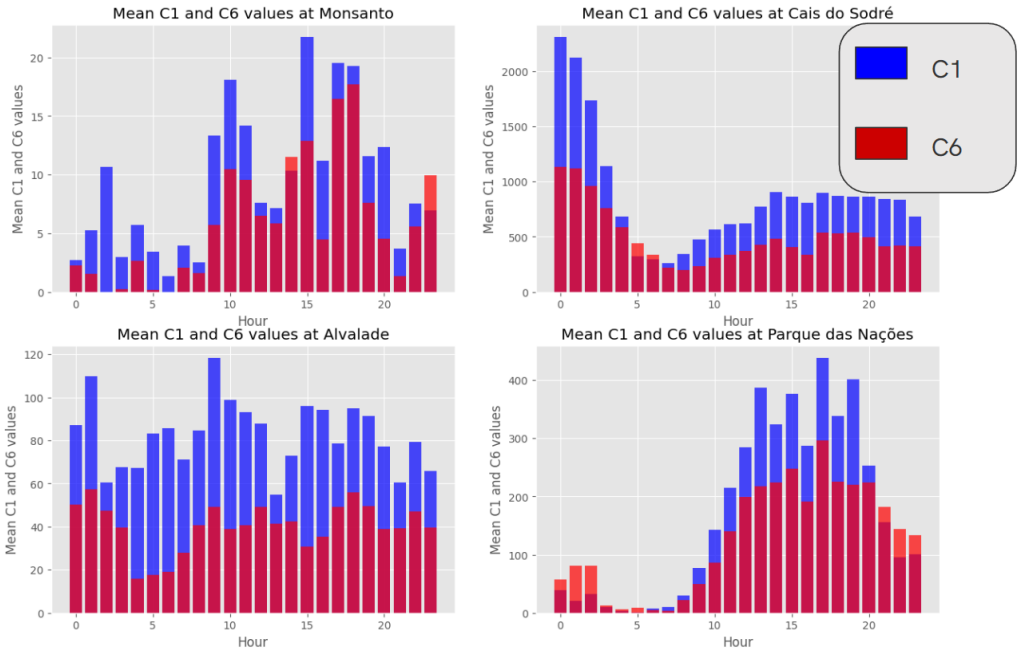



Figure 3.6: Variations of values C1 and C6 in 4 different grids

### 3.1.4 How data is obtained and limitations

A crucial point that has not yet been discussed here is the fact that the cell phone count data has several decimal places. This logically doesn't make sense, but after talking to LxDATALAB, it was possible to partly clear up this doubt. According to the team, Vodafone only has access to the cell phone counts on its network, geolocating via triangulation of antennas (similar to what's done in 2.5) . That's why, before they are shared with LxDATALAB, Vodafone applies an algorithm that makes a prediction of the total number of cell phones in each square based on Vodafone's market share in the localities where the cell phones that are there originate. If a cell phone comes from a certain grid where the Vodafone market share is 0.2, the cell phone will count as 5 wherever it goes, including other grids. This method of data collection may create some confusion and distrust about the consistency of the data. LxDATALAB also explained that the algorithm was more complex but unfortunately, for business reasons, Vodafone didn't provide any more information at first. Later, in a meeting with Vodafone's team responsible for the data, most of our questions and doubts that came up when analyzing the dataset were answered.

A limitation that this data will also have and that generates noise in the data is the fluctuation of cell phone locations on the boundaries of the squares. If a person is at this boundary, the network can often assume rapid movements between the two squares when in fact the phone is stationary or moving little.

## 3.2 Multivariate Analysis and Knowledge Discovery

Following on from the last sections, we now present a preliminary analysis of the possible relationships between the data and a reflection on the results that were obtained.

Figure 3.7 shows the correlation matrix between the data, using the September 2021 file (other months had similar results). In this table it was possible to identify the strong correlation between the values of C1, C3, C5 and C6, which is to be expected. In the case of E7, E8 and E9, however, we see that they have some problems. Not only is the correlation with the 'C' data low, but also the correlation between them. Especially the data from E7, which justified excluding this data in advance from the models and algorithms that were implemented.

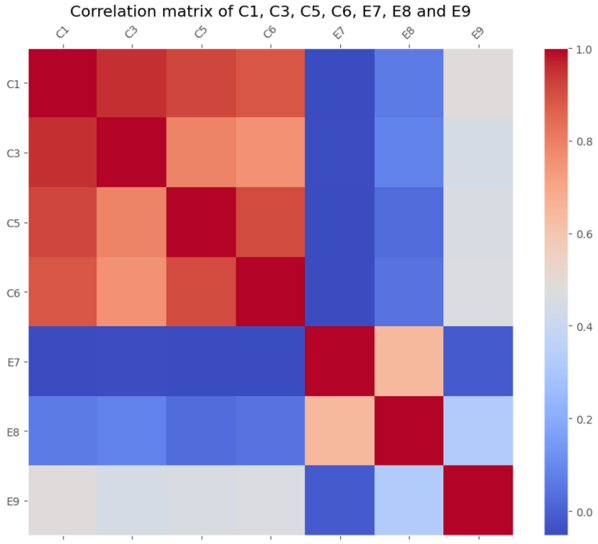



Figure 3.7: Vodafone Grid data correlation matrix

### 3.2.1 Algebraic Relationships

After examining the metadata, considering potential connections and to learn more about the data, the following algebraic relationships were explored:

$$C1(t) = C3(t) + C6(t) \tag{3.1}$$

$$C3(t) = C3(t-1) + C5(t) - C6(t) \tag{3.2}$$

- *Equation 3.1*: the number of cell phones detected is equal to the sum of the cell phones that remained plus those that left the square. Looking at all the times $t$ in the demonstration diagram in figure 3.4, we can see that the formula is always valid.

- *Equation 3.2*: the number of cell phones that remain in a square is equal to those that remained in the previous time $t$ plus the entries and exits in the same period. Like the previous formula, this is always valid in the demonstration of the figure 3.4.

Moving from theory to a direct analysis of the data, we could see that these formulas are surprisingly very incoherent. First, some tests were made to compute the relative error between the predicted $C1(t)$ from 3.1 and $C3(t)$ from 3.2 with their actual values. For the tests in this section, a simple outlier removal was made with an absolute z-score ¡ 4 and removing rows that presented a C3 growth or decrease of 10

times the previous or the next observation. The months used were October, November and December 2023, but in the other months similar results were obtained. The table 3.7 has the values of the relatives errors for both equations.

| Month | Rel. Error Eq.1 | Rel. Error Eq.2 |
|---|---|---|
| 10/23 | 0.299 | 0,279 |
| 11/23 | 0,295 | 0,271 |
| 12/23 | 0.299 | 0.282 |

Table 3.7: Equations 3.1 and 3.2 relative errors

### 3.2.2 Exit, reenter and stay situation

The errors are surprisingly high. Something that can heavily impact the equations is a situation described in the figure 3.8, where the ”Y” cell phone, in the same 5-minute period, enters, leaves and re-enters the square, remaining there. Thus, entering activates C5, leaving activates C6 and staying activates C3. However, C1 will still be 1. This situation will fail in both formulas. Considering this, we can change the equations to inequations because we know that:

- In equation 3.1, the number of cell phones detected is equal to the sum of the cell phones that remained plus those that left the square, but those that left can reenter the square, staying until the end, meaning at the minimum $C1$ has to be equal to $C3$, and at maximum is $C3 + C6$;
- In equation 3.2, the number of cell phones that remain in a square is equal to those that remained in the previous time $t$ plus the entries and exits in the same period, but the exits can be entries as well, so the minimum is $C3(t-1) + C5(t) - C6(t)$ and the maximum $C3(t-1) + C5(t)$.

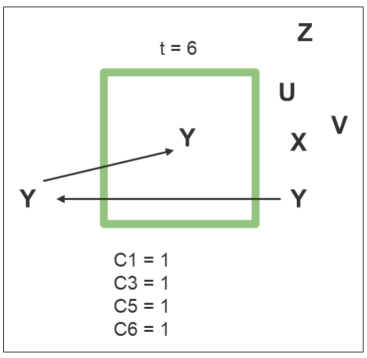



Figure 3.8: Entering, exiting and re-entering a square

When showing these analysis, Vodafone’s team provided further information and clarified a bit more about how the data is being collected and computed. They told that, to be able to maintain a more or less fixed value of the sum of all C3 of all grids in Portugal, they keep changing the coefficients of each actual mobile phone every five minutes. Therefore, it is certain that, at any time and in any grid, the equation 3.2 may not be verified. A simple example is when a grid has no entries or exits but has “stays” in that period, just a small change in the coefficients will result in an error.

### 3.2.3 Inconsistencies related to C6 indicator

Vodafone also elaborated more on a relevant topic regarding what happens when a cell phone disconnects itself from the network. Perhaps it would be expected that the cell phone would leave the grid, but it was said that it stays in the same grid for 3 hours (still activating C3, but it is not certain) and then it leaves (thus increasing C6). A good proof for this, for example, is the fact that in the October 2023 file, the hour with the highest decrease in C3 at night hours is 2 a.m. and not like midnight. This effect, although, is not so prevalent because people usually have their phones connected to the network through the night.

Related to the C6 indicator, we can address here the reason for the incoherence at the end of section 3.1.3, in the figure 3.6. Looking at the data, the C6 being higher than the C1 indicator is something that happens often. From $97 \times 10^6$ observations, $6.7 \times 10^6$ had C6 higher than C1. In the table 3.8, we can even see that the hours with the highest number of rows where $C6 > C1$ are late night hours, inferring that this phenomenon has something to do with the functioning that Vodafone shared about C6.

| **Hour(h)** | **Rows($1 \times 10^5$)** |
|---|---|
| 2 | 4,14 |
| 1 | 4,03 |
| 3 | 3,84 |
| 0 | 3,79 |
| 23 | 3,65 |
| 22 | 3,61 |
| 21 | 3,58 |
| 4 | 3,29 |

Table 3.8: Hours of the day with more number of rows where $C6 > C1$

It's clear that our interpretation of C6 in the data summary sheet is incorrect. In fact, the description of what C6 is is unclear; this is a possible translation to English: "These are the distinct terminals that left the grid. The calculation is made using the previous 5-minute interval as a reference, also considering the grid crossings in the same interval". To conclude, we can say that the C6 indicator won't need a cell phone present in the grid to be incremented. What is unknown is how often and when does that happen exactly. Unfortunately, it was not possible to meet again with Vodafone to clarify this.

Given the inequations formulated in 3.2.2 and what we could know about C6, we remove the rows where $C6 > C1$ and check if an observation fails to fall within the limits and compare the value with the nearest boundary to get an amplitude of the discrepancy. The results of this new approach (also with outliers removed) are in the table 3.9.

Inspection of the results indicates that a larger-than-expected proportion of rows fails to satisfy the 2nd inequality, especially in the lower part. The number of rows per month is like 32 to $33 \times 10^6$ rows, so half of the rows (or even more if we don't count the rows with $C3 = 0$) don't follow a simple intuitive condition. We assume that's because of the strange behavior and misinterpretation of C6 and the fact that the market share of Vodafone and other model inputs they use keeps changing.

A final consistency check, based on the $C_5 - C_6$ difference, reveals that these indicators again deviate from the expected values. If we have the sum of all C6 (exits) and the sum of all C5(entries)

| | **Inequation 1** | | | |
|---|---|---|---|---|
| | $C1 > C3 + C6$ | | $C1 < C3$ | |
| **Month** | Rows $\times 10^6$ | Rel. Error | Rows $\times 10^6$ | Rel. Error |
| 10/23 | 1.41 | 0.067 | 0.09 | $\sim$0 |
| 11/23 | 1.36 | 0.067 | 0.08 | $\sim$0 |
| 12/23 | 1.59 | 0.068 | 0.09 | $\sim$0 |
| | **Inequation 2** | | | |
| | $C3(t) > C3(t-1) + C5(t)$ | | $C3 < C3(t-1) + C5 - C6$ | |
| **Month** | Rows $\times 10^6$ | Rel. Error | Rows $\times 10^6$ | Rel. Error |
| 10/23 | 0.48 | 0.181 | 16.51 | 0.236 |
| 11/23 | 0.41 | 0.195 | 16.73 | 0.229 |
| 12/23 | 0.47 | 0.191 | 17.28 | 0.240 |

Table 3.9: Coherence of the inequalities from 3.2.2

from 1 month, the difference between them is enormous. The result of this test for the 3 months is in the table 3.10. The reason for this to be happening is mostly because of C6 functioning. These results would suggest that every month hundreds of millions of mobile phones would enter Lisbon and somehow would vanish after, which is incoherent with reality.

| **Month** | **C5 - C6 ($\times 10^8$)** |
|---|---|
| 10/23 | 1.17 |
| 11/23 | 1.31 |
| 12/23 | 1.07 |

Table 3.10: C5 sum and C6 sum difference

### 3.2.4 Pings and coherence of data E9

When analyzing various graphs where E9 was a variable, the values were often inconsistent. For this reason, the following test was carried out: if the Vodafone Grid dataset is filtered by the lines that have more than 1000 different mobile phones remaining in the square ($C3 > 1000$), check what is the maximum stay period. Initially, the 09-2021 file was tested and the results show major inconsistencies. The E9 data is the maximum time spent in the square and, with the filter of 1000 cell phones remaining in the square, it was expected that E9 would be 300 seconds in almost all records, which is not the case. On the other hand, in November 2023 the data is more in line with what was expected. This can be seen in tables 3.11 and 3.12, since in the case of 09-2021 there are only entries with 300 seconds in the 75% quartile. The improvement for the 11-2023 data may be due to the fact that the methods and algorithms used by Vodafone to obtain this data have been optimized to better correspond to reality, according to LxDATALAB.

Presenting Vodafone with these findings, they revealed some more important information about the data. When determining the location of a mobile phone, they use frequent pings to it from the network, but these pings are highly sparse. In such a way that sometimes, according to Vodafone, the pings are greater than the period of 5 minutes. But for the purpose of the E9 indicator, the time from when it starts counting is from the first ping of the 5 minutes period, which most of the time is not even close to the beginning. This explains the odd behavior of this indicator, but at the same time, it is a sufficient reason

| Metric | E9 |
|---|---|
| mean | 165.42 |
| std | 97.59 |
| min | 1.00 |
| 25% | 82.00 |
| 50% | 128.00 |
| 75% | 300.00 |
| max | 300.00 |

Table 3.11: E9 values from 09-2021 dataset

| Metric | E9 |
|---|---|
| mean | 269.80 |
| std | 68.55 |
| min | 12.00 |
| 25% | 300.00 |
| 50% | 300.00 |
| 75% | 300.00 |
| max | 300.00 |

Table 3.12: E9 values from 11-2023 dataset

to discard the data as useful for the model.

What is crucial about this, based as well on what Vodafone said, is that when a mobile phone, inside a high-speed transport, goes through multiple grids in the same 5-minute period, the chances of it not being detected there are high. Thus, if it's not detected, no C5 or C6 are incremented with its crossing. This will have a huge impact on the final model implemented in this project.

## 3.3 Lisbon Geodata and GTFS

In this section, we will present some simple analysis on the geodata available for Lisbon that will be used in the model, including the geo-referenced network of roadways and connections (where the train and underground train rail tracks data are present), the dataset of the buildings' geometry, and the GTFS (General Transit Feed Specification) of the bus routes.

### 3.3.1 OpenStreetMap Dataset

OpenStreetMap (OSM) is a collaborative project that provides freely available geospatial data. The dataset used in this project was extracted directly from the official website [29], in the tab "Extract" and delimiting the area of extraction to the Lisbon metropolitan area. This cropping was actually a disturbing factor in the next chapter.

The data contains a high amount of data regarding other aspects of the city and its roads than just the public main roads. To filter the dataset, a multi-step filtering pipeline was applied to obtain the most relevant data for roads geometry, described in the table 3.13.

This filtered dataset provides a more accurate and semantically rich representation of Lisbon's road network, suitable for spatial analysis and modeling tasks that will be carried out. The filtering was done regarding the tags' descriptions available in documentation online. The result can be seen in Figure 3.1, as the plot used this dataset.

From the same OSM dataset, it was possible to find some geometries that had the name of the route lines of the train and underground network of Lisbon. After filtering them by the name, it was possible to have a clean dataset of the rail track geometries, represented in the figure 3.9.

| Step | Description |
|---|---|
| **Remove irrelevant highway values** | Exclude values such as *footway*, *path*, *cycleway*, *construction*, and others not useful for vehicular analysis. |
| **Access restriction filter** | Remove features with *access* values like *private*, *customers*, *delivery*, etc. |
| **Area and structural exclusion** | Remove geometries where *area = yes*, or *service* types like *yard*, *spur*; and *tunnel = culvert*. |
| **Filter landuse with no roads** | Group by *landuse* and drop rows where all values in *highway* are null within the group. |
| **Drop empty columns** | Columns with all values as *NaN* are removed from the dataset. |
| **Exclude metadata-empty rows** | Rows where *highway* and the following are all null are removed: *maxspeed*, *ref*, *junction*, *bridge*, *tunnel*, *access*, *service*. |

Table 3.13: Filtering steps applied to the OSM edges dataset

### 3.3.2 Buildings datasets

The geo-referenced dataset of building outlines was extracted from [30]. The original file contained a substantial number of null values across many attribute columns, making it impractical to load the entire dataset without preliminary filtering. This was necessary to avoid excessive memory usage and overly complex data structures. As a result, a pre-selection of relevant features was performed during the data import stage. Specifically, only features explicitly tagged as buildings were retained, and the following attributes were extracted: the unique identifier (*id*), building type (*building*), number of levels (*building:levels*), and height (*height*). The total number of rows after joining with the Vodafone Quadrículas dataframe was 52,000. The figure 3.10 is the plot of OpenStreetMap edges, Vodafone Quadrículas and the buildings dataset.

One particularly important aspect of this dataset, beyond the building geometries, was the inclusion of height information. Unfortunately, the availability of this attribute was very limited: out of the 52,000 records, only 951 contained non-null values in the *height* field. Similarly, the *building:levels* field had data in just 8,673 entries. Due to this sparsity, it was not possible to obtain a reliable representation of the actual building heights. As a result, this feature was excluded from the modeling process.

Subsequently, an alternative dataset was identified through the Global Human Settlement Layer (GHSL) initiative, which provides comprehensive spatial information on human settlements. Specifically, the GHS-BUILT-H dataset offers estimates of average building heights at a 100-meter resolution, derived from a combination of satellite imagery and digital elevation models. This dataset includes information for Lisbon, presenting average building heights aggregated within 100-meter grid cells [31]. The dataset can be viewed in the figure 3.11.

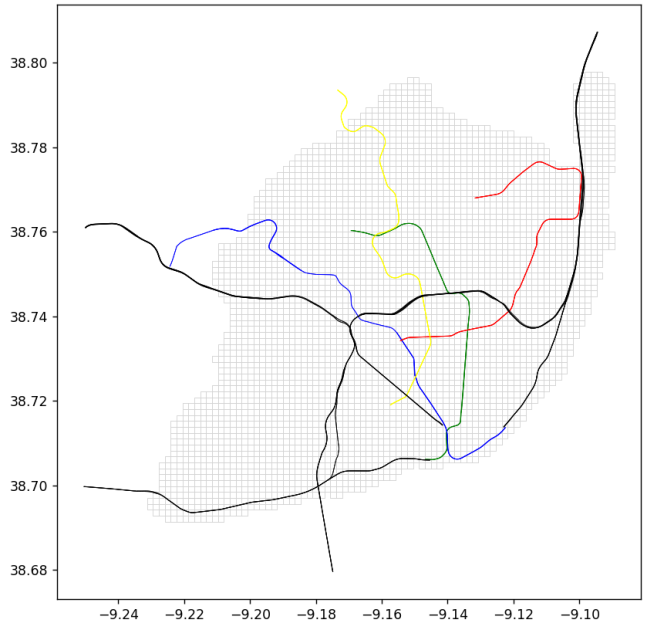



Figure 3.9: Train and underground (Metro) line geometries

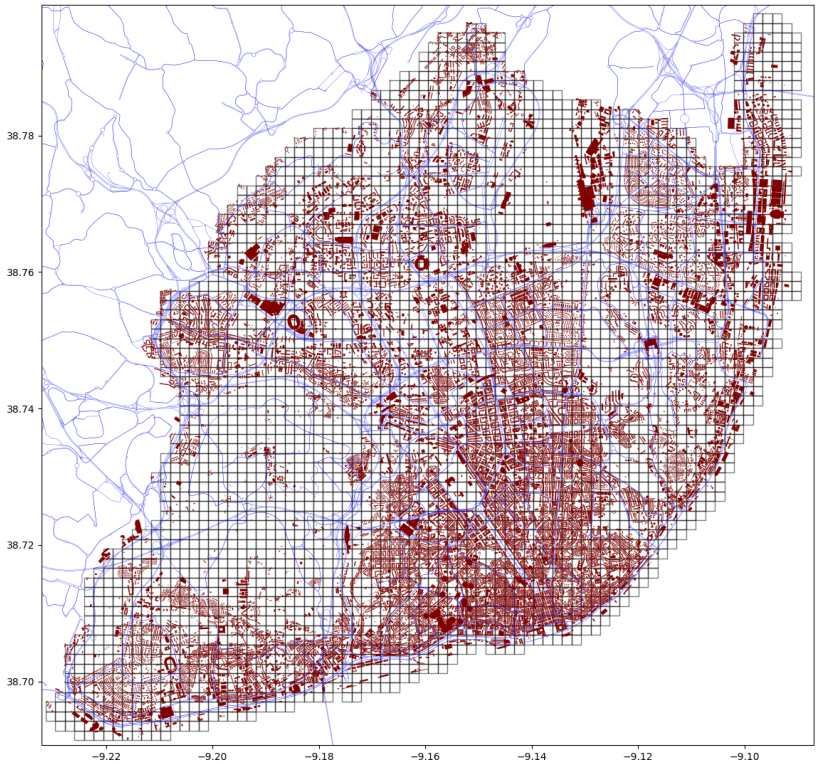



Figure 3.10: Filtered OSM Edges, Vodafone Quadrículas and Buildings of Lisbon (in red)

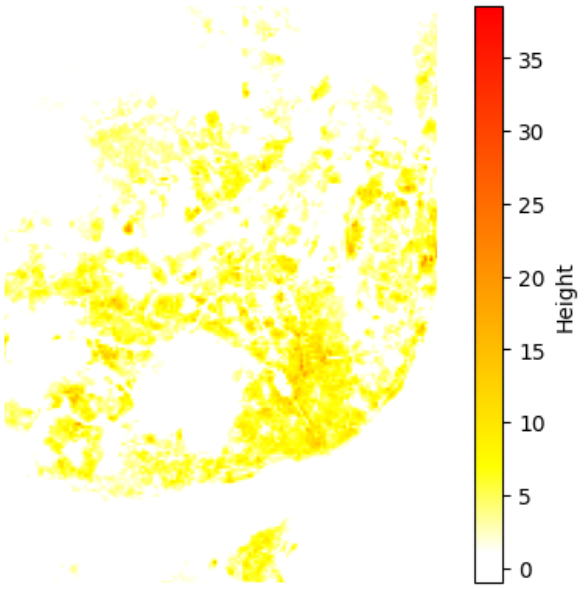



Figure 3.11: Average height of buildings in Lisbon with 100-meter resolution

### 3.3.3 Bus Routes GTFS

The GTFS of the *Carris* bus services was extracted from the TransitLand website [32]. With a sequence of steps of data processing and transformations, it was possible to create geospatial shapefiles that make integration with other datasets easier.

First, the trip and shape datasets were merged to associate each trip with its corresponding geographic coordinates. These coordinates were ordered and converted into LineString geometries, enabling the construction of a spatial representation of each trip. The resulting geometries were enriched by merging them with additional route-level information, such as route names and service identifiers. Next, the dataset was filtered to retain only primary routes—identified by specific route ID patterns—and to exclude school-specific or duplicated routes. The services were then grouped based on their operational periods (e.g., weekday, weekend, holidays), and only routes with directional identifiers (e.g., ascending or descending shapes) were preserved to ensure spatial coherence. Finally, each service subset was exported as an individual shapefile, to be used individually when needed. The figure 3.12 contains all the routes of the Carris bus service on weekdays in Winter.

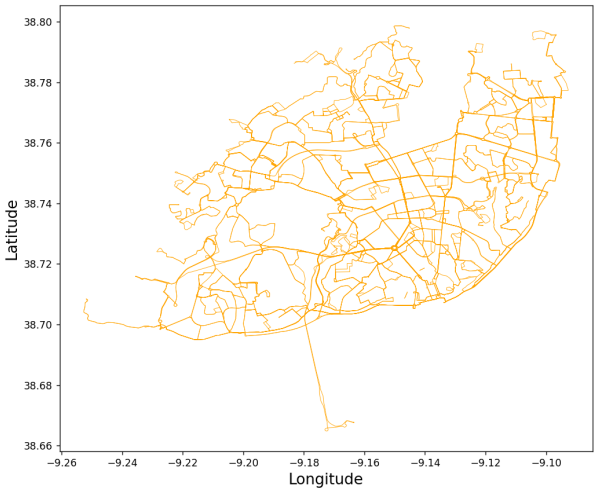



Figure 3.12: Weekdays in Winter bus routes

## 3.4 Traffic data

### 3.4.1 CGIUL road traffic countings

After an extensive search for robust traffic-volume information, the only dataset that met our reliability criteria was the hourly count series published on the CGIUL (*Centro de Gestão e Inteligência Urbana de Lisboa*) *Ambiente* dashboard [33]. The dashboard reports bidirectional car counts from six permanent sensors distributed across the city. Once the raw files were cleaned—removing spurious records, reconciling time stamps, and correcting entry errors—the dataset was ready for analysis.

The full time series shows that, in the early months, the sensors were not configured for a continuous 24-hour schedule. Coverage gradually improved, and by mid-2022 all sites were logging the complete daily cycle. Overall, the dataset spans **14 July 2021** to **15 January 2024**, albeit with occasional multi-day gaps.

Although each record is labelled with a clock hour, CGIUL staff confirmed that the count actually aggregates the preceding *15-minute* interval rather than the full hour. The value is then multiplied by 4 to estimate the hourly counts. This nuance is crucial when aligning the data with other sources operating at the same temporal resolutions.

To obtain an overview of the dataset, Table 3.14 summarizes the main statistics.

| Location | Count | Mean Dir. 1 | Mean Dir. 2 | Min Dir. 1 | Min Dir. 2 | Median Dir. 1 | Median Dir. 2 | Max Dir. 1 | Max Dir. 2 |
|---|---|---|---|---|---|---|---|---|---|
| Calç. Carriche | 18 115 | 1 012 | 1 097 | 0 | 0 | 1 034 | 1 120 | 3 168 | 3 047 |
| Av. Liberdade | 17 489 | 171 | 333 | 0 | 0 | 109 | 380 | 1 739 | 742 |
| Av. D. Pacheco | 17 932 | 1 353 | 1 568 | 0 | 0 | 1 547 | 1 696 | 3 841 | 4 263 |
| Av. República | 5 923 | 542 | 540 | 0 | 0 | 468 | 525 | 1 829 | 1 617 |
| Av. Berlim | 16 593 | 691 | 519 | 0 | 0 | 807 | 605 | 1 369 | 960 |
| Av. F. Armadas | 1 931 | 450 | 854 | 1 | 8 | 463 | 913 | 1 425 | 1 718 |

Table 3.14: Summary statistics of hourly traffic counts for selected locations. Decimal places are removed for readability.

If we focus on the last row of Table 3.14—Avenida das Forças Armadas—the directional means appear counter-intuitive. To inspect this anomaly, Figure 3.13 plots the diurnal profiles for both directions. The chart reveals a marked afternoon excess in the eastbound (west–east) counts relative to the reverse movement, contradicting the expected commuting pattern for this corridor. This inconsistency indicates a possible sensor or data-processing issue that users of the dataset should bear in mind.

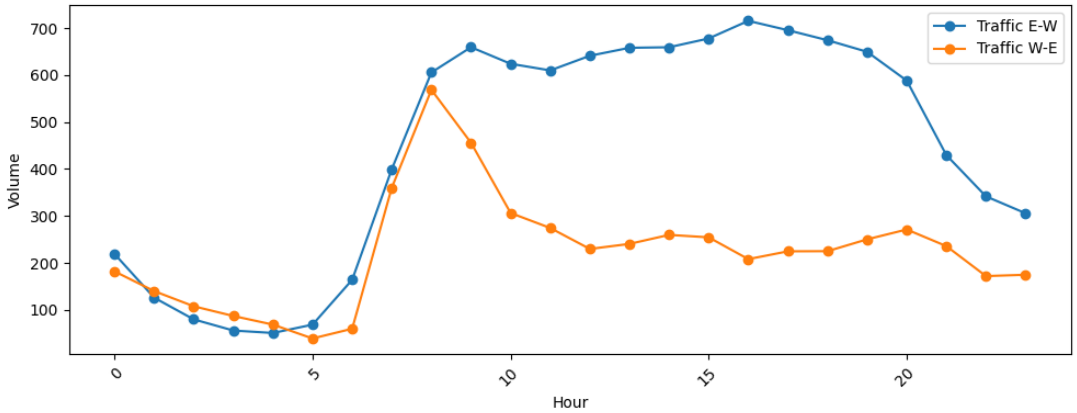



Figure 3.13: Mean hourly traffic volumes in Avenida das Forças Armadas by direction.

### 3.4.2 Waze data

This dataset, Waze *traffic jams*, contains information generated by Waze on traffic jams after processing the GPS location points sent by users' phones and reports generated and shared by them. The data relates to 2021, 2022 and 2023. The data fields can be found in the table 3.15.

| Dim.(02-22) | Gran. | Important Fields | | | | | |
|---|---|---|---|---|---|---|---|
| (1413373, 10) | 1 min. | entity_ts | speed | delay | length | level | position |

Table 3.15: Congestion data generated by Waze

These fields are: *speed* - flow speed in meters/second; *delay* - delay in seconds relative to free flow speed; *length* - length of congestion in meters; *level* - various levels of congestion: 1 = from 80% to 61% of free flow speed, 2 = from 60% to 41%, 3 = from 40% to 21%, 4 = from 20% to 1%, 5 = Road blocked.

The descriptive statistics for this data can be found in table 3.16.

| | Delay | Length | Level | Speed |
|---|---|---|---|---|
| mean | 60.01 | 237.51 | 4.32 | 0.83 |
| std | 129.06 | 370.05 | 1.02 | 1.46 |
| min | -1.00 | 5.00 | 1.00 | 0.00 |
| 25% | -1.00 | 53.00 | 4.00 | 0.00 |
| 50% | -1.00 | 135.00 | 5.00 | 0.00 |
| 75% | 93.00 | 298.00 | 5.00 | 1.42 |
| max | 5482.00 | 10265.00 | 5.00 | 19.56 |

Table 3.16: Descriptive statistics for Delay, Length, Level, and Speed

The delay is -1 when the road is blocked, which also corresponds in the descriptive statistics to congestion level 5 and a speed of 0.

To visualize the distribution of traffic jams throughout the day, the bar graph in figure 3.14 was generated. This graph highlights the fact that there are so many thousands of traffic jams in the early hours of the morning. The answer to this contradiction is simple: most of the traffic jams generated by Waze at that time were not caused by cell phones with people on them. They were made by information from the city council about when a road is blocked, issuing a level 5 congestion signal every 1 minute. In the graph 3.15 we can see how, by removing the level 5 traffic jams, the figures become more coherent.

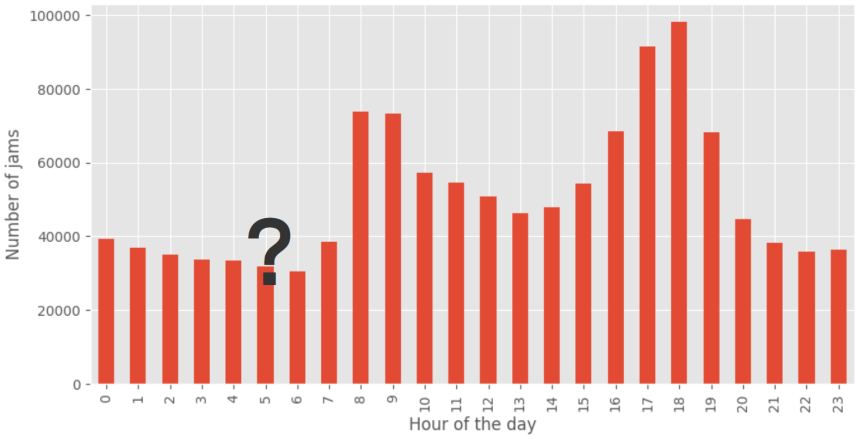



Figure 3.14: Number of traffic jams per hour of the day in January 2023

The map in Figure 3.11a, for the period between 3am and 3.30am on 2023-01-10, shows the location of several of these ”virtual jams”. In the morning of the same day, the congestion pattern seems more

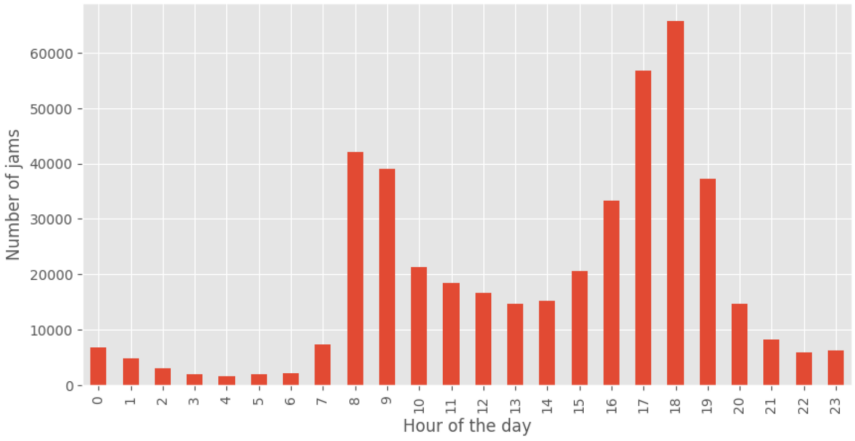
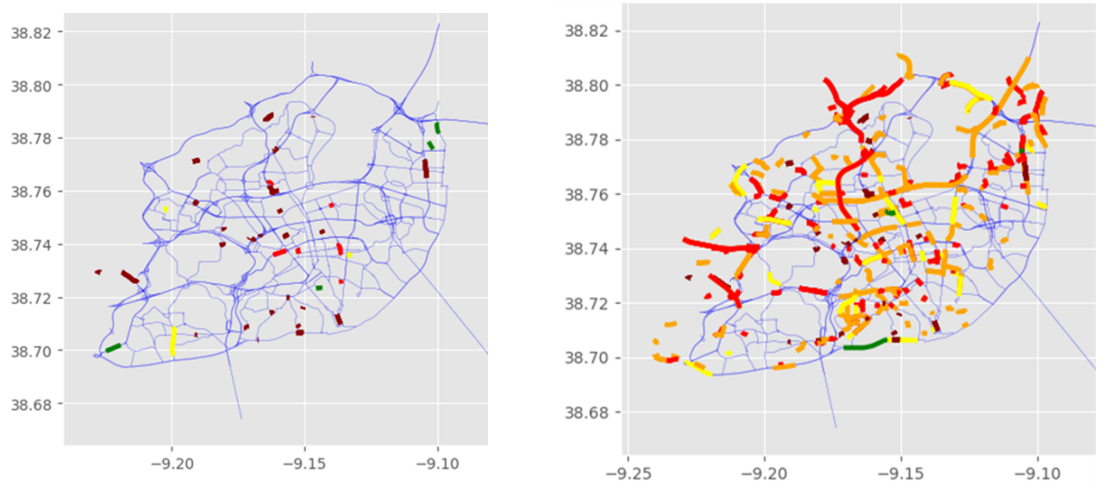



Figure 3.15: Number of traffic jams with level $< 5$ per hour of the day in the month of January 2023

normal.




(a) (b)

Figure 3.16: Congestion levels at (a) 3-3.30 a.m. 2023-01-10 (b) and at 8-8.30 a.m. 2023-01-10.

Looking at the map in figure 3.16b it is not possible to distinguish which flow directions are being used in the bottlenecks. To try to discover them, using the 'geometry' field of each jam, the starting and ending position of the jam was first noted. Then, defining an arbitrary location to serve as the city center, for example, the *Terreiro do Paço* in Lisbon, the distances between the starting point and the center and the end point were measured. Thus, for a congestion $C_k \in \mathbb{R}^2$ with a starting position $C_{ki} \in \mathbb{R}^2$ and ending position $C_{kf} \in \mathbb{R}^2$, and chosen the city center $P \in \mathbb{R}^2$, $C_k$ is towards the city center if $\|C_{ki} - P\| > \|C_{kf} - P\|$. If we filter the congestion into "approaching the center" and "moving away from the center", it is possible to create the maps in figures 3.17 and 3.18, which show the clear predominance of congestion on the routes approaching the city center in the morning.

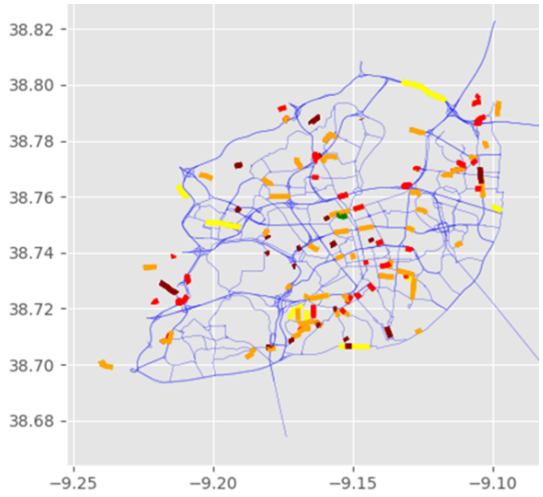



Figure 3.17: Congestion levels moving away from the center 8-8.30 a.m. 2023-01-10

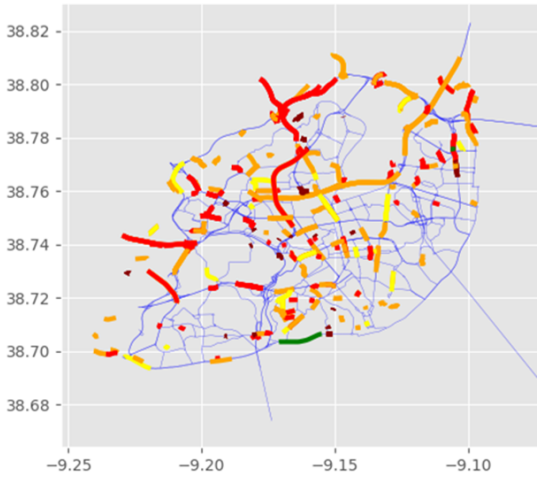



Figure 3.18: Congestion levels approaching the center 8-8.30 a.m. 2023-01-10

# Chapter 4

# Routing Simulation

In this chapter, we describe the process of computing the shortest travel time matrix and the corresponding shortest paths between any pair of grids, using both the OpenStreetMap (OSM) network data and the *Vodafone Quadrículas* dataset. This matrix plays a fundamental role in modeling and predicting the movement of mobile devices across the study area and is central to the implementation described in Chapter 5.

## 4.1 Preliminary Steps

As an initial step—prior to considering realistic routing along the street network—we compute the direct (great-circle) distance between the centroids of each pair of grids. While this approach does not account for road infrastructure or real-world travel constraints, it serves as a useful approximation and is employed in other components of the model where routing precision is not strictly necessary.

To calculate these distances, one might be tempted to use the Euclidean distance in Cartesian coordinates, given by:

$$d_{\text{euclidean}} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{4.1}$$

However, this method is not appropriate for geospatial coordinates on the Earth's surface, due to the planet's curvature. Instead, we utilize the geodesic (or great-circle) distance, which represents the shortest path between two points on a sphere. This is more consistent with the Earth Mover's Distance (EMD) framework used in our mobility modeling.

The geodesic distance between two points defined by latitude and longitude can be approximated using the Haversine formula:

$$d_{\text{haversine}} = 2r \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right) \tag{4.2}$$

where:

- $r$ is the Earth's radius (approximately 6,371 km),

- $\phi_1$ and $\phi_2$ are the latitudes of the two points in radians,
- $\Delta\phi = \phi_2 - \phi_1$ is the difference in latitude,
- $\Delta\lambda = \lambda_2 - \lambda_1$ is the difference in longitude.

One practical use of this distance matrix is that it enables the identification of neighboring grids. By examining the centroid-to-centroid distances, we can determine which grids lie within a certain proximity of each other, which is particularly useful for modeling local interactions and defining neighborhood-based parameters in the inference process.

## 4.2 Shortest Path Computation

This section describes the procedure used to compute the shortest paths and estimated travel times between every pair of spatial grids using OpenStreetMap data and the GraphHopper routing engine [34]. It is implemented in a Java program and systematically calculates the optimal route between grids while avoiding non-vehicular pathways and using the speed profiles defined by the road types in OpenStreetMap to estimate realistic travel times.

### 4.2.1 GraphHopper Routing Engine

GraphHopper is an open-source routing engine written in Java that utilizes OpenStreetMap (OSM) data to compute routes efficiently. It supports various routing algorithms, including Dijkstra's algorithm, A* search, and Contraction Hierarchies (CH), to find the shortest or fastest paths between points on a road network. GraphHopper is designed to be fast and memory-efficient, making it suitable for large-scale routing applications.

The framework employs several algorithms to compute shortest paths efficiently:

- **Dijkstra's Algorithm:** A classic algorithm for finding the shortest path between nodes in a graph. It explores all possible paths from the source node, ensuring the shortest path is found. However, it can be slow for large graphs due to its exhaustive nature.
- **A* Search Algorithm:** An extension of Dijkstra's algorithm that uses heuristics to guide the search, reducing the number of nodes explored and improving performance. It is particularly effective when an admissible heuristic, such as the straight-line distance to the destination, is available.
- **Contraction Hierarchies (CH):** A preprocessing technique that accelerates shortest path queries by creating shortcuts in the graph. During preprocessing, less important nodes are removed, and shortcuts are added to preserve shortest path distances. This results in significantly faster query times, especially for large-scale networks.

Given the size of the problem, the algorithm used is the Contraction Hierarchies. Its use allows for rapid computation at scale, enabling the simulation of dynamic traffic conditions and routing under different constraints if they are requested eventually in the future.

### 4.2.2 Origin-Destination Filtering

To avoid overloading the routing engine with unnecessary computations, destination grids were filtered to include only those located within a great-circle distance of less than 5 kilometers from the origin grid. In the case of Lisbon—a city characterized by a dense urban layout and a scarcity of long, uninterrupted road segments—this threshold is intuitively appropriate. Routing tests using Google Maps for some of the most direct routes in Lisbon suggest that the minimum travel time between two grids approximately 5 kilometers apart is around 5 minutes. This aligns well with the temporal granularity of the Vodafone dataset. For all other destination grids that were filtered out, the travel time values were set to infinity. However, this approach could—and perhaps should—be revisited in future modeling efforts.

To improve the speed of the routing computations, an additional filtering step was performed to exclude airport grids—areas typically restricted from public access and not served by public road infrastructure. Since many of these grids do not intersect with any highway segments, they were first identified using spatial filtering with the OSM edges.

To ensure that the entire airport zone was effectively removed (including contiguous non-highway grids within the same area), a spatial clustering algorithm was applied. Starting from a known airport grid (e.g., grid 2807), the algorithm recursively grouped all neighboring grids whose centroids lie within 200 meters of each other. This was achieved by computing geodesic distances and performing a breadth-first like search to collect all nearby grids into a connected set. The result is shown in the figure 4.1.

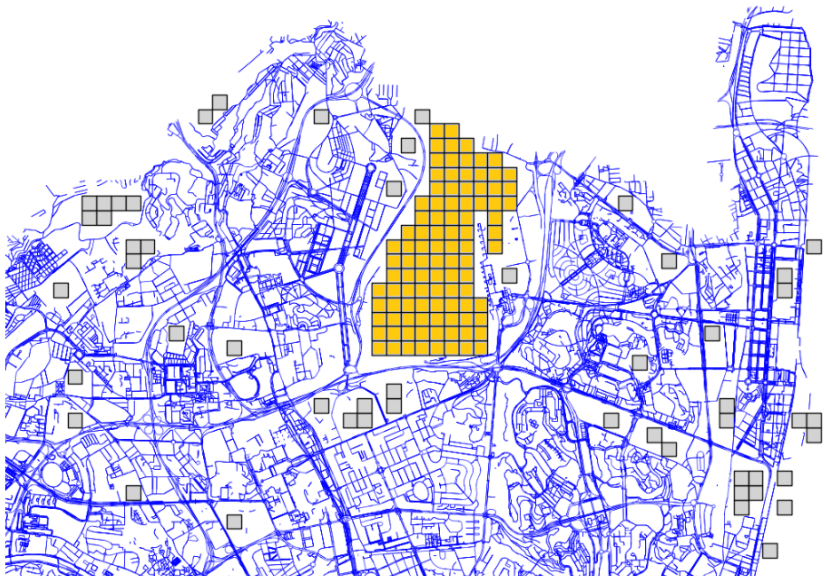


Figure 4.1: Airport Grids excluded from the shortest path routings

### 4.2.3 One Point by Grid Routing

An earlier version of the Java routing program implementation adopted a simplified strategy in which each spatial grid was represented by a single point—the centroid of the grid polygon. This approach significantly reduced computational complexity by limiting the number of origin-destination pairs, as only one route per grid pair was computed. The routing engine, GraphHopper, was then used to calculate the shortest path between the snapped positions of these centroids on the road network.

However, this method proved to be inadequate for realistic travel time estimation. A fundamental limitation arises from the fact that each centroid, when mapped to the nearest navigable road segment (a process known as ”snapping”), can yield highly inconsistent and context-dependent results. In particular, if the snapped road segment is a one-way street oriented in the opposite direction of the intended travel, the routing engine must detour—sometimes extensively—to comply with the road's directionality constraints.

For example, consider the case of routing from Grid 2 to Grid 4. If the centroid of Grid 2 is snapped to a one-way road that only allows travel in the direction opposite to Grid 4, the route generated will be significantly longer and less realistic than if multiple candidate entry points were available. In such cases, the overall travel time becomes heavily conditioned by the arbitrary snapping result, introducing bias and reducing the fidelity of the routing estimations.

## 4.3 Final Routing Model

The final routing model builds upon the limitations identified in the earlier version, introducing a more spatially representative method for assigning points to each grid. The core idea is to define four origin points and four destination points per grid, one in each cardinal direction (North, South, East, and West), to better reflect the actual connectivity of road infrastructure. After that, the graphhopper engine, when computing the shortest paths between each grid, will take into consideration the 16 ($4\times4$) possible combinations between each grid's origin and destination points. This ensures more robust and realistic route generation by minimizing the bias caused by single-point snapping and accounting for directional road access. The following subsections detail the key components of this model, from road-grid assignment to the handling of infrastructure errors and the inclusion of other points on the edges of the city, outside of the Vodafone Quadrículas polygons.

### 4.3.1 Assignment of Roads in Each Grid

Before defining directional starting and ending points, it is necessary to determine which road segments intersect each grid and how they connect spatially. This process began with the loading of a preprocessed OpenStreetMap (OSM) road network, from which invalid or low-importance roads (e.g., unclassified roads) were removed (filter from section 3.13). It's not a major issue to remove these roads as almost all of them are two-way roads, and when a mobile device starts on those roads, it can go both ways depending on the destination.

One issue encountered was that roads were represented as numerous small, disconnected segments, making it difficult to assess their continuity or measure their effective length within each grid. To address this, road segments were grouped and merged based on unique identifiers such as *name*, *ref*, or *osmid*. Their geometries were then unified to form continuous, navigable segments, each carrying metadata including direction, road classification (*highway*), among other attributes.

To ensure that roads assigned to a grid are relevant to its local structure, each grid is first reduced to a smaller square (140 meters wide) centered on its centroid. This reduced area is used to filter and retain only road segments that have a significant spatial presence within the grid. Road segments intersecting these smaller grid squares are checked for validity and a minimum geodesic intersection length of 5 meters, discarding trivial overlaps that may not represent important roads in the grid.

The validated roads are then categorized according to the side of the grid they intersect. For each road segment that intersects a grid square, its geometry is analyzed to determine which boundary—North, South, East, or West—it enters or exits from. This classification is carried out by comparing the geometry of each road segment with the boundaries of the grid cell.

Two distinct classifications are maintained for road segments associated with each grid cell: one for roads serving as **starting points** and another for those used as **ending points**. These are denoted by the prefixes *S_* and *E_*, respectively (e.g., *S_North*, *E_South*). Figure 4.2 illustrates the validated road segments for Grid 1124, including prominent roads such as *Avenida de Ceuta* and *Eixo N-S*. The mapping between edge IDs and their corresponding cardinal directions within the grid is detailed in Table 4.1.

Each edge is identified by a unique ID corresponding to its index in the cleaned OpenStreetMap (OSM) edge dataset. It is important to note that a single edge can serve multiple directional roles, depending on its geometric interaction with the grid cell. For instance, edge *1281* serves both as a starting segment heading north (*S_North*) and as an ending segment arriving from the south (*E_South*) within the same grid.

To classify road segments as starting or ending within a grid, two main geometric scenarios are considered:

- **Scenario 1: Single-Side Intersection**
  In this more common case, a road segment intersects only one side of the grid. The classification is straightforward:
  - For starting roads: if the first point of the segment lies inside the grid and the endpoint lies outside, the segment is assigned to the direction of the intersected boundary.
  - For ending roads: if the first point lies outside and the endpoint is inside the grid, the segment is classified based on the side it intersects upon entry.
- **Scenario 2: Multi-Side Intersection**
  Less frequently, a road segment may intersect two sides of the grid. In such cases, the classification requires additional analysis:

– For starting roads: the side closest to the endpoint (which lies outside the grid) determines the direction. For example, if the endpoint is closest to the north side, the segment is labeled *S_North*.

– For ending roads: the side closest to the starting point (which lies outside the grid) determines the direction. For example, if the starting point is closest to the east side, the segment is labeled *E_East*.

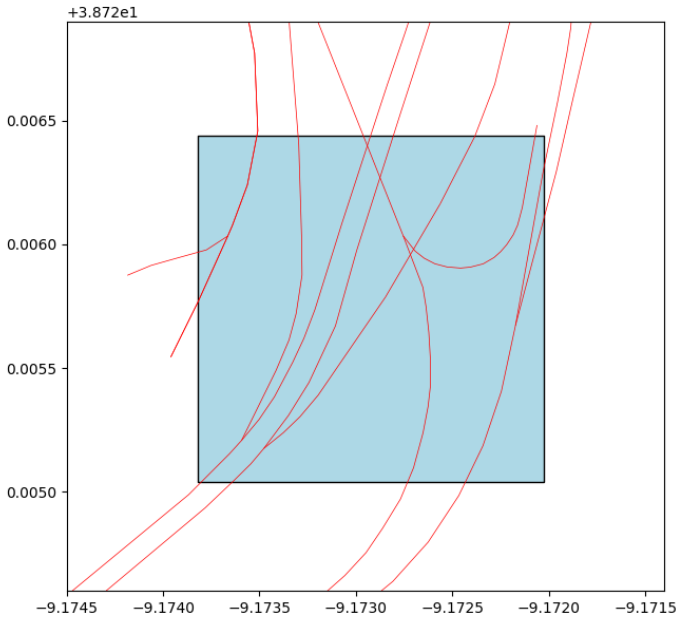



Figure 4.2: Validated road segments of grid 1124.

| Grid_ID | Intersections | S_North | S_South | S_East | S_West |
|---|---|---|---|---|---|
| 1124 | 238, 1011, 1012, 1280, 1281, 3284, 3286, 3288, 3289, 3291, 3292, 3293, 3301, 3302, 4039 | 1281, 3288, 3292, 3302 | 1011, 1280 | 3289, 3291 | 3293, 4039 |
| | | **E_North** | **E_South** | **E_East** | **E_West** |
| | | 238, 1280, 3286, 3293 | 1012, 1281 | [ ] | 3292 |

Table 4.1: Directional edge associations for Grid 1124

This comprehensive labeling of roads allows for precise origin and destination point placement in the next stage of the routing model and ensures that the travel paths respect both directional constraints and the physical configuration of the road network.

### 4.3.2 Main Roads and Origin–Destination Points Attribution

Now that we identified the relevant road segments within each grid, we can define the main road for each cardinal direction (North, South, East, West). To do this, a scoring scheme was applied based on

the hierarchical classification of OSM road types. Each road segment intersecting a grid was assigned a score computed as the product of its geodesic length (within the grid boundary) and a weight derived from its road type. Major roads such as *motorway*, *primary*, or *trunk* received higher weights, reflecting their importance in vehicular movement. This step, however, was done arbitrarily with powers of $6^n$, but the calibration of the weights might not have been the most optimal. The weights assigned to each road type are the following: *motorway* and *trunk* were both weighted at 216, while *primary* was given a weight of 36, *secondary* 6, and *tertiary* 1. For each of these, their corresponding *_link* variants were assigned half the weight.

For each direction, the road segment with the highest score was selected as the main road. Then, a tentative origin or destination point was positioned halfway between the center of the grid and the midpoint of the respective side (e.g., the northern side for *S_North*). If a main road was available in that direction, this tentative point was snapped to the closest location on the main road segment using geometric interpolation.

To ensure the adjusted point remained inside the grid and did not overlap other road segments, further refinement was applied:

- If the point projected onto the road segment fell outside the square or too close to other edges, it was shifted along the main road geometry by a safe offset.

- If no main road was found in a given direction, the fallback strategy used only the side midpoint. The point was then shifted iteratively away from all intersecting road geometries until a minimum safe distance (e.g., 10 meters) was achieved.

- As a last resort, if no valid location could be found after multiple adjustments, the grid's center was used.

This robust strategy produced eight adjusted routing points for each grid—four for entrances and four for exits—stored as (road ID, latitude, longitude) tuples. To visualize how the points can be spread in the grid, the figure 4.3 contains the computed points for grid 2546 as an example (interchange between Eixo N-S and 2ª Circular highways).

**- Adding Vodafone Eixos Points**

To complete the model, additional points based on Vodafone Eixos polygons were incorporated into the grid-based routing structure. The points were manually selected close to the Vodafone Quadrículas edge, carefully examining the road segments corresponding to each axle. Due to the specific nature of these special grids—designed to span a significant portion of a single road and typically connecting to other grids at only one location—it was sufficient to assign just one origin point and one destination point per grid. To integrate these into the routing framework, each axle was assigned a new artificial grid ID.

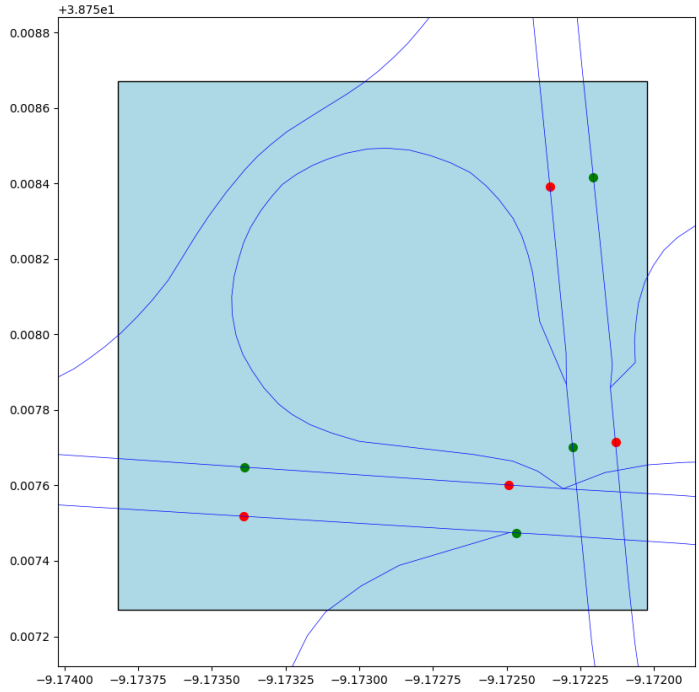



Figure 4.3: Origin (green) and Destination (red) points for grid 2546.

### 4.3.3 Testing and Final Results

The final testing phase of the routing simulation proceeded as anticipated, with improvements introduced through the activation of turn costs in the routing engine. This adjustment enhanced the realism of the travel time estimates by accounting for delays introduced by road geometry and directionality. It is important to note that these travel-time estimates are based on road-class–specific default average speeds (e.g., *motorway*=100 $\mathrm{kmh}^{-1}$). Whenever an *maxspeed* tag is present in *OpenStreetMap*, that limit is enforced at 90% of its legal value. Segment times are therefore calculated from the encoded values of *car_average_speed* rather than the ideal—and often unavailable—Level-of-Service A (LOS A) values; while the resulting averages loosely approximate LOS A conditions, they do not capture the slower operating speeds typical of peak-hour traffic. Other LOS classes are ignored, so every road is evaluated with a single representative operating speed.

A critical aspect of this phase was the scale of the computations involved. Over 50 million shortest path calculations were executed using the GraphHopper routing engine. Thanks to the adoption of parallel computing, the entire process was completed in under three hours. This efficiency was key to making the problem tractable within the few available computational resources.

Another vital optimization involved the handling and storage of the computed shortest paths between grid pairs. Initially, the results were stored in CSV format. However, given the size of the data—spanning several gigabytes—this approach posed challenges in both storage space and loading times. To over-

come this, the files were eventually converted to Python's pickle format, which significantly reduced loading times and improved runtime efficiency during subsequent modeling stages.

Despite these improvements, the massive volume of data necessitated an additional step: splitting the routing results into five separate files. This partitioning made it possible to load only one portion at a time into memory, thus avoiding bottlenecks and enabling a more manageable workflow—particularly on machines with limited RAM.

**- Bridge Snapping Error**

During the testing phase of the routing model, a critical issue was encountered when attempting to compute routes involving the Ponte 25 de Abril and the Ponte Vasco da Gama. Specifically, when defining an origin point on either of the bridges and initiating a routing request through the GraphHopper engine, the snapping mechanism consistently failed. Instead of snapping to the bridge's road segment, the origin was incorrectly snapped to a road segment located on land, often several hundred meters away.

This error significantly distorted the computed travel times and paths, especially for routes entering Lisbon from the south (via Ponte 25 de Abril) or the northeast (via Ponte Vasco da Gama). Despite extensive efforts to resolve the issue—including repeated configurations of the GraphHopper snapping behavior—the problem persisted for both bridges.

Multiple attempts were made to fix the snapping behavior by modifying the configuration file, including setting various *snap_preventions*. Additionally, manual adjustments using the *setSnapPreventions()* method in the *GHRequest* API were explored. Unfortunately, none of these methods produced the desired outcome. In some cases, routing requests returned null results; in others, the snapping mechanism continued to ignore the bridge road segments entirely.

The issue was reported and discussed in the GraphHopper community forums, where it became clear that the snapping mechanism—particularly the *LocationIndexTree* and its internal *EdgeFilter*—relies not only on proximity but also on connectivity within the road graph. After considerable investigation, the root cause of the issue was identified as a flaw in the OSM data extraction process.

Initially, the OpenStreetMap (OSM) road network was cropped directly from the OSM website by selecting a bounding box around the Lisbon metropolitan area. However, this selection sliced both the Ponte 25 de Abril and the Ponte Vasco da Gama near their midpoints, severing their connectivity across the Tagus River. As a result, the road segments representing these bridges became disconnected in the underlying road graph. Because GraphHopper's snapping algorithm considers both distance and graph connectivity, these disconnected bridge segments were not considered valid candidates for snapping—even when they were geographically closer.

This discovery underscored the importance of carefully defining the boundaries when extracting OSM data. To resolve the issue, a larger geographical bounding box was used to ensure full inclusion of both bridges and their connected road segments. Once the bridges were fully represented in the graph, the snapping behavior functioned correctly, and routing from and to these key infrastructure links became possible without distortion.

**- Final Observations and Correction of a Late Routing Error**

One of the most representative outputs of the final routing model is a colored plot showing the travel costs from a given grid to all other grids, present in the figure 4.4. This visualization—based on color gradients rather than absolute values—offers a clear and intuitive understanding of how the model responds to the underlying road infrastructure. In these plots, lower routing costs (depicted in lighter colors) are concentrated along high-speed, well-connected roads, while higher costs (darker colors) are prevalent in areas with fewer or no road connections. This contrast aligns with real-world expectations: areas with major highways should indeed offer lower travel times compared to regions with limited access or poor connectivity.

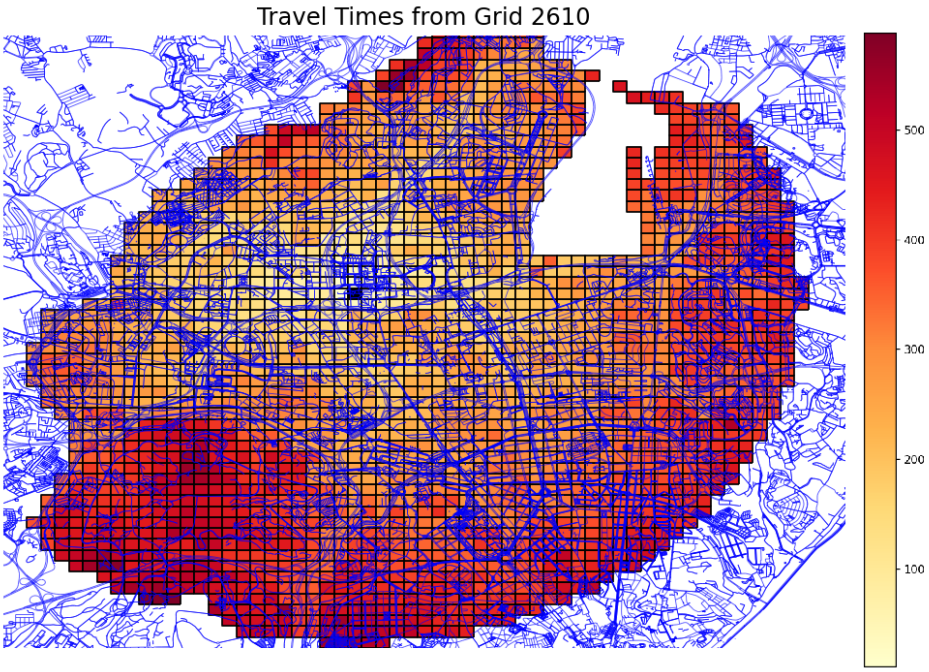



Figure 4.4: Travel times from grid 2610 to all other grids

However, this final visualization was only developed in the later stages of the project. Earlier in the process, the generated plots simply displayed raw cost values embedded within each grid cell—an approach that, while useful for localized or micro-level verification, severely limited the ability to conduct meaningful visual analysis at a broader scale. These initial plots made it difficult to identify spatial inconsistencies or assess the overall realism of the routing outcomes.

With the improved visualization in place, a critical anomaly was identified along the 2ª Circular road segment, after the *Alta de Lisboa* roundabout, in the figure 4.4. Unexpectedly, the cost gradients along 2ª Circular fluctuated between yellow and orange and then returned to yellow, instead of showing a smooth, monotonic decrease. Given that this is a high-speed, linear road, routing costs were expected to decline incrementally and uniformly. The anomaly strongly suggested a fault in the propagation of routing costs.

Closer examination revealed that grid 2746 was responsible for this inconsistency. A thorough check of the origin and destination point allocations for this grid revealed that, in the westbound direction of

2ª Circular, there was no destination point assigned to the corresponding road segment. As a result, vehicles approaching this grid from the west had to detour significantly to reach it, resulting in inflated and unrealistic travel costs in that region.

This issue was traced back to a subtle bug in the logic that assigns directional road access. As discussed in section 4.3.1, when a road segment intersects more than one side of a grid, the correct direction should be inferred based on the proximity to the segment's starting point. However, for destination points, the code mistakenly used the segment's endpoint. Although the logic had been correctly implemented for origin points, the flaw in the destination logic had escaped detection due to the rare nature of such intersection cases.

Once identified, the logic was corrected to consistently use the starting point of each segment to determine its directional assignment. With this adjustment, the destination points were accurately placed, ensuring consistency and correctness, as illustrated in figure 4.5.

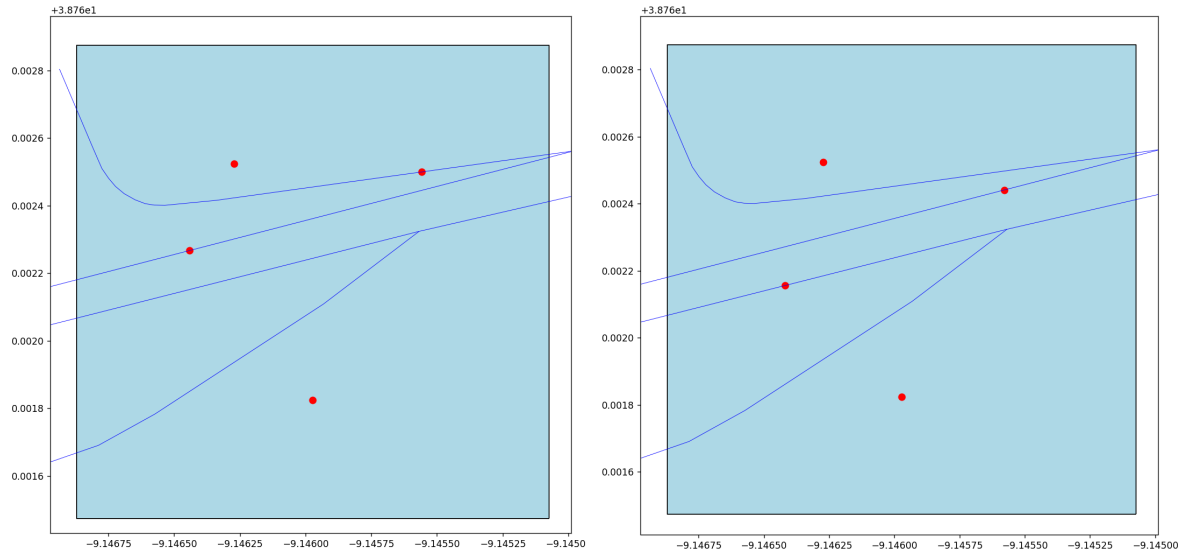



Figure 4.5: Ending points before (left) and after (right) the roads assignment correction

Due to time constraints for this project, it was not possible to rerun the full routing engine with the corrected destination points, and the rest of the project kept this error. Nonetheless, the identification and resolution of this issue represent a clear improvement that can be readily implemented in future iterations, given that the underlying problem has already been diagnosed and fixed. Visualization tools—particularly gradient-based routing plots—prove very valuable in validating and refining complex systems. Far from being merely illustrative, these visualizations function as diagnostic instruments, capable of revealing subtle yet significant errors that may otherwise remain obscured in raw numerical outputs.

## 4.4 Train Routings Travel Times

Given the significant number of commuters in Lisbon who rely on the railway system, it was necessary to incorporate train routing into the cost matrix used for mobility inference. While road-based routing covered a large portion of travel behavior, omitting rail transport would underestimate accessibility and travel time efficiency in areas served by train lines. To integrate train travel times into the model, a complementary cost matrix was constructed based on real-world train routes and schedules, subsequently merged with the vehicular routing matrix.

The implementation followed a three-step approach:

1. **Identification of Train-Crossed Grids:** The OpenStreetMap (OSM) edges dataset for Lisbon was carefully filtered to retain only segments corresponding to national railway lines (e.g., "Linha de Cintura" or "Linha do Norte"). This filtered dataset was then spatially intersected with the Vodafone grid shapefile to identify the grid cells traversed by train routes. Additionally, each major train station was manually associated with the grid cell that either contains it or lies in its immediate vicinity, ensuring accurate placement of station nodes within the grid network.

2. **Assignment of Travel Times Between Stations:** Using official CP schedules[1], direct travel times between key train stations were collected and manually encoded. For instance, the time between Benfica and Sete Rios was set at 180 seconds, and between Braço de Prata and Oriente at 180 seconds, among others. These values represent actual average durations between adjacent stations on Lisbon's urban train network.

3. **Integration into the Cost Matrix:** For each train segment, the path connecting the origin and destination stations was reconstructed using neighboring grid cells that intersect the rail lines. Then, a segment-based travel time was evenly distributed across the path, ensuring consistency of travel times between intermediary grid cells. Each time value was scaled to match the same units as in the vehicular cost matrix (milliseconds), making it possible to merge both sources.

Special attention was given to the generation of paths between grids using a recursive search algorithm with distance-based thresholds, ensuring that only the most plausible rail-path neighbors were considered—aligned with the 5-minute temporal resolution of the Vodafone dataset. The algorithm was adapted to handle exceptions and correct distortions caused by geometric anomalies in the rail network. Furthermore, travel from intermediate grids—rather than only from station endpoints—was modeled to reflect real-world behavior, where mobile phones may be detected mid-route without necessarily disembarking at the next station.

It is important to note that the train line connecting Lisbon to Cascais was not included in this implementation. This decision is justified by the fact that the entire route closely parallels a major road corridor, which is already captured within the car-based routing model. In contrast, the remaining suburban train lines in Lisbon often traverse areas without comparable road infrastructure, making their inclusion significantly more impactful. Additionally, metro lines were excluded from this work, as they typically run

[1] https://www.cp.pt/StaticFiles/horarios/urbanos-lisboa/completo-comboios-urbanos-lisboa.pdf

beneath or adjacent to high-traffic roadways that are already well-represented in the model. Nevertheless, incorporating metro-based routing could be a valuable addition for future research focused on truly multimodal transport modeling.

Overall, the integration of train routing into the cost matrix substantially improves the spatial and temporal fidelity of the mobility model. By capturing travel patterns not accounted for by road infrastructure alone, this addition provides a more nuanced view of Lisbon's transportation dynamics. It also lays the groundwork for future enhancements involving additional transit modes, such as metro systems, in the pursuit of a fully multimodal urban mobility framework.

# Chapter 5

# Wasserstein Distance Model Implementation

Building on the theoretical foundations introduced in Section 2.2.2, this chapter presents the full implementation of the *Wasserstein distance model* that underpins the traffic flow inference framework developed for Lisbon. The purpose of the model is to infer an *origin–destination flow matrix*—expressed as the optimal mass transport plan $x^*_{jk}$—between consecutive five-minute snapshots of aggregated mobile-phone presence data (Vodafone *C3* counts). The underlying optimization problem (Equation 2.3) minimizes the total disutility of moving aggregate "mass" between[1] the $N = 3\,761$ spatial grids while respecting mass–conservation constraints and a rich set of domain-specific rules.

The chapter is structured as follows. Section 5.1 details how raw Vodafone and auxiliary datasets are transformed into the initial ($m_0$) and final ($m_1$) mass vectors. Section 5.2 describes the construction of the spatio-temporal cost matrix $c_{jk}$ leveraging the routing simulation of Chapter 4. Section 5.3 introduces several model variants and constraints applied that improve behavioral realism. Section 5.4 expands on the steps to implement the minimum C5 (entries) on each grid. Solver configuration and performance considerations are summarized in Section 5.5. The final sections explain post-processing of the optimal solution into road network flows (Section 5.6) and describe the evaluation pipeline against ground-truth traffic counts (Section 5.7). All components of the model were implemented in Python, using standard libraries such as *pandas*, *numpy*, *os*, and *matplotlib*, along with geospatial packages like *geopandas* and *shapely*. For the optimization core part, the implementation relies critically on IBM's *CPLEX* solver library.

Figure 5.1 provides a visual overview of the full inference pipeline. Each block usually corresponds to a major processing step covered in the sections above. The entire process is repeated across a user-defined number of consecutive five-minute periods. This temporal granularity is especially important given that the available ground-truth traffic data is aggregated at hourly intervals. By running twelve consecutive five-minute inferences and aggregating the results, the model produces estimates that

[1] Throughout this chapter the terms "mass", "terminals" and "devices" are used interchangeably to denote the number of uniquely detected mobile phones after normalization for Vodafone's market share.

align with the hourly resolution of the reference data, allowing for more meaningful validation. That said, the user may also choose to run a single five-minute period—or any other number of consecutive periods—and validate the output by dividing the hourly ground-truth counts accordingly (e.g., by 12 or by the number of periods used).

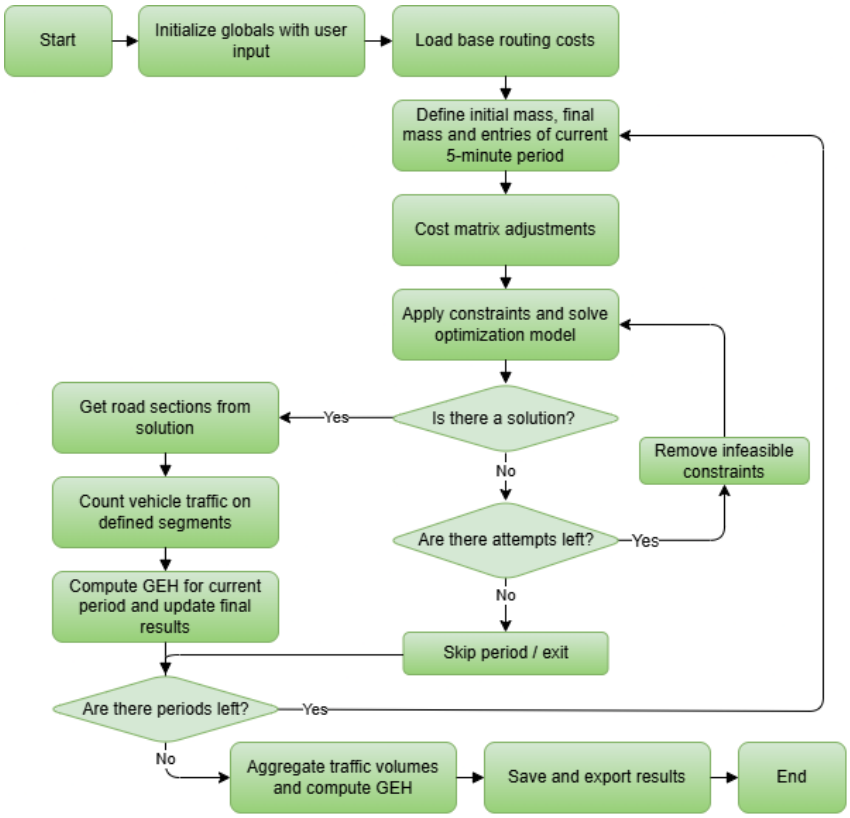



Figure 5.1: Overview of the Wasserstein-based traffic inference pipeline. Each block corresponds to a major stage in the modeling process, from raw data preparation to optimization and validation.

## 5.1 Preparation of Mass Vectors

Prior to running the inference engine, all raw observations of the Vodafone data are cleaned and gap-filled. Outliers (absolute $zscore > 4$) and missing values are replaced by linear interpolation followed by a local–median smoother, yielding fully populated state vectors—an indispensable requirement, as the linear program cannot accommodate undefined masses or costs.

### 5.1.1 Choice of Observables

The backbone of the model is the *C3* indicator—the number of devices observed in each grid at the end of a five-minute interval. Early trials assessed *C5* (entries) and *C6* (exits) to primary masses by treating *C6* as $m_0$ and *C5* as $m_1$. This approach proved unstable because a mobile phone can be detected several times while crossing intermediate cells, generating systematic double-counting. *C5* therefore re-enters the formulation only as a *minimum-flow constraint* to guarantee that a credible share

of observed entries is realized as through-movement (cf. Section 5.4).

### 5.1.2 Vodafone Eixos as Fictitious Grids

Radial inflows along the eleven motorway entrances (the “Eixos”) are captured by mapping each entrance to a dedicated fictitious grid polygon (IDs 3744–3754). A count in *C13* at time $t + \Delta t$ is interpreted as a stay in the final state $m_1$, whereas the corresponding count in *C12* at time $t$ is treated as part of the initial state $m_0$.

Because the IC2 corridor intersects Lisbon’s grid at three distinct locations (see Figure 5.2), the associated mass is proportionally distributed across the three relevant entry–exit pairs. This reflects the likely usage patterns of the actual road network, acknowledging that the true distribution is more complex. Nonetheless, this simplification allows the IC2 axis to serve as a workable, if approximate, proxy for device inflow and outflow.


Figure 5.2: IC2 polygon (red) intersections with the Lisbon grid, including the start points of the same intersections (green)

### 5.1.3 Border Artificial Sinks and Mass Distribution

Because the total number of devices observed at times $t$ and $t + 5\,\text{min}$ rarely matches exactly—largely due to the oscillatory nature of commuting flows in a metropolitan area like Lisbon—the resulting mass discrepancy is rebalanced using five *artificial sink* grids located at the city’s periphery (Grid IDs 3757–3761).

A configurable hyperparameter defines the percentage of this discrepancy that is distributed across these sinks. This portion is then allocated proportionally to the nearby edge grids associated with each

sink region: *Algés*, *Portas de Benfica*, *Pontinha*, *Portela*, and *Moscavide*. The remaining portion of the discrepancy is uniformly spread over the 3743 interior grids. The rationale for selecting these locations lies in their significant human activity and strong connectivity to the Lisbon area, despite being situated outside the administrative boundaries of Lisbon.

This correction ensures mass conservation between the initial and final states:

$$\sum_j m_{0j} = \sum_k m_{1k}$$

This is necessary because the optimal transport plan $x^*_{jk}$ must satisfy the marginal constraints:

$$\sum_k x^*_{jk} = m_{0j} \quad \text{(mass leaving grid } j\text{)}$$

$$\sum_j x^*_{jk} = m_{1k} \quad \text{(mass arriving at grid } k\text{)}$$

These summation conditions are integral to the optimal mass transport formulation and ensure consistency between initial and final mass distributions across the spatial grid.

## 5.2 Construction of the Cost Matrix

The cost matrix $c_{jk}$ represents the generalized travel time for moving one unit of "mass" from grid $j$ to grid $k$ within a single five-minute snapshot. Its construction starts with real-world based routing times and then layers several heuristic adjustments that reflect behavioral priors and measurement quirks.

### 5.2.1 Base Travel Times and Localized Tweaks

Initial prototyping relied on great-circle distances between grid centroids. While computationally convenient, this metric proved inadequate for capturing real-world traffic patterns. Specifically, it failed to account for physical and infrastructural constraints: travel between two grids separated by train lines, major roads, or other environmental barriers was often underestimated, while longer but realistic routes along motorways were undervalued. Consequently, the final model adopts the following routing mechanisms:

- **Road travel times** computed using the GraphHopper engine, as configured in Chapter 4, which incorporates road directionality, speed limits, and turn restrictions;
- **Rail travel times** for origin–destination pairs whose optimal route is a heavy-rail segment, as detailed in Section 4.4. If both a road path and a train path exist between the same origin and destination grids, the path with the lowest cost is selected. If the train path is faster than the car path,the corresponding (start, end) pair is assigned as a train path. This distinction will be important in the next sections;

- **Airport-specific movements**, which consist of direct great-circle distances between the filtered set of Lisbon airport grids (identified in Section 4.2.2) and a small group of hand-selected grids that spatially correspond to Terminals 1 and 2. These links model shuttle or internal airport transfers.

All raw travel time values are initially stored in milliseconds for precision during preprocessing. However, to improve numerical conditioning during the optimization phase, the final cost matrix is downscaled to seconds.

**- Brute-Force Capping of Implausible Moves**

Because Vodafone aggregates mobility data over five-minute intervals ($300$ s), any routed travel time exceeding $375$ seconds is considered implausible under the model assumptions. As such, these routes are discarded by setting their corresponding cost entries to $c_{jk} = +\infty$. In addition, two further hard-coded caps are applied to restrict unrealistic flows:

1. *Eixos/Sinks self-loops:* Movements between motorway axis grids or artificial sink grids (introduced in Section 5.1) can be programmatically toggled to either allow or prohibit such flows. When disabled, any OD pair where both $j$ and $k$ correspond to Eixos/Sink grids (i.e., IDs greater than 3743) is assigned $c_{jk} = +\infty$.
2. *Airport-to-non-airport flows:* All flows from airport-designated grids to any other non-airport grid in the system are prohibited by default. This restriction ensures that the modeled mobility does not unrealistically permit transit from secure airside zones to unrelated urban areas.

These caps substantially reduce the dimensionality of the final linear optimization problem by removing infeasible or undesirable transitions in advance, improving both computational efficiency and model validity.

**- Adjacency Fallback Costs**

If GraphHopper fails to return a route between two nearby cells (e.g. across a pedestrian-only footbridge or due to incomplete road connectivity in OSM), deterministic fallback times are assigned based solely on geometric adjacency. These ensure that even in the absence of explicit routing information, basic spatial continuity is maintained. The assigned costs are summarized in Table 5.1.

| Relative position | Approx. distance (m) | Assigned cost (s) |
|---|---|---|
| Side-adjacent (N, S, E, W) | 156 | 330 |
| Diagonal adjacent | 221 | 365 |
| Two squares apart (orthogonal) | 312 | 400 |
| "Knight" move ($\sqrt{5} \times 156$ m) | 349 | 420 |

Table 5.1: Fallback costs when no routing path exists between close grids.

All four fallback values are deliberately set higher than typical car travel times for equivalent distances, subtly biasing the optimizer toward true GraphHopper-derived paths whenever they exist. Nonetheless, these geometric fallback links serve a critical role in model feasibility. Without them, certain grid

pairs—especially those in areas with incomplete road data, physical access restrictions, or disconnected subgraphs—could result in unreachable nodes, leading to infeasibility in the linear optimization problem.

### 5.2.2 Permanence Cost

With $c_{jj} = 0$ the optimizer would keep most devices motionless. To prevent that, a pragmatic solution could be just assigning the same cost != 0 to every grid, independent of the geographic characteristics of the same. After some experiments, it was concluded that, as expected, the equal assigning is not suited for the context of this problem. Given this, two alternative diagonal models were explored:

**- Model 0 ("Buildings")**

This variation models the cost of remaining in the same grid ($c_{jj}$) using georeferenced data on Lisbon's built environment. It takes into account both the total building footprint area and the length of the road network within each grid cell. The underlying rationale is that grids characterized by dense static infrastructure—such as substantial building coverage and lower-hierarchy residential roads—are more likely to host stationary mobile devices and should therefore incur a lower permanence cost. In contrast, grids dominated by high-speed roads and lacking significant built structures are less likely to support prolonged stationary presence, and thus receive a higher permanence cost.

The OSM filtered road network employed in this model differs from the one presented in subsection 3.3.1. Here, we use a subset that includes all highways tagged as *motorway*, *trunk*, *primary*, *secondary*, *tertiary* (and their corresponding *_link* classes), as well as all *residential* roads. Each road segment is weighted by $2^n$, with $n \in \{0, 1, 2, 3, 4\}$, rather than the calibration described in subsection 4.3.2, because the latter, using $6^n$, disproportionately favors higher-hierarchy roads in this context. Building footprints are sourced from the dataset introduced in subsection 3.3.2.

The cost is computed via a piecewise logistic function (double-sigmoid) applied to a composite variable defined as the difference between scaled road length and building footprint area. The curve is constructed to satisfy four constraints: (i) low cost in highly urbanized areas, (ii) high cost in sparsely developed zones, (iii) continuity at the neutral transition point (zero), and (iv) an average value anchored at a predefined level for balance. These constraints are controlled by three hyperparameters: *min_model0*, *avg_model0*, and *max_model0*, which define the lower bound, average central cost, and upper bound of the sigmoid function respectively. Their tuning is essential to ensure a realistic distribution of diagonal costs that discourages artificial stillness without penalizing plausible stationary behavior in key urban zones.

Two implementation variants are supported. The first uses the raw building footprint within each grid, while the second applies a blurred version—generated via convolution—to account for localization errors in mobile phone detection (typically around 50 meters). This adjustment helps prevent the misclassification of devices as being located in main-road grids when they are actually in adjacent urban areas. A height-based weighting option can also be applied, multiplying the footprint by average building

height (normalized using a user-defined scaling factor), thereby favoring vertical density in the scoring.

The final score vector is computed across all 3743 Vodafone grids. Eighteen artificial grids, which represent motorway axes and sinks, are appended with arbitrarily high permanence costs to prevent stationary behavior in those locations.

Figure 5.3 illustrates an example output of this model, with the parameters $min = 10000$, $avg = 61000$, and $max = 360000$ showing how permanence costs vary spatially in accordance with the built environment.

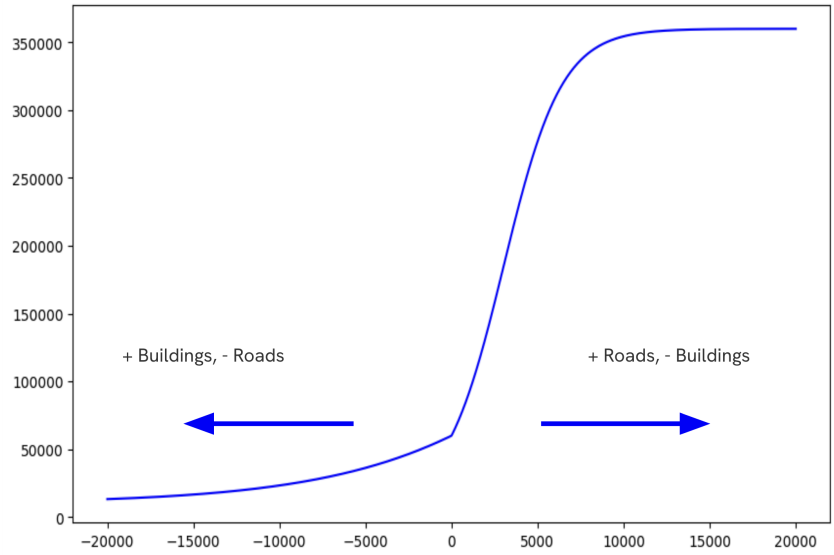



Figure 5.3: Example of permanence cost distribution under Model 0 (“Builds”).

The transition thresholds of the double-sigmoid were set to $x_1 = 8\,000$ and $x_2 = -8\,000$. These values are not the outcome of a formal calibration; they were chosen heuristically after inspecting the empirical range of the composite variable $x = \textit{weighted\_length} - \textit{intersection\_area}$. Most grid cells show $x$ values between $-10^4$ and $10^4$. Placing the 5 % and 95 % points of the sigmoid at $\pm 8\,000$ therefore guarantees that (i) grids at the extremes—dominated either by roads (positive values) or by buildings (negative values)—reach almost the maximum or minimum permanence cost, respectively; and (ii) the bulk of observations falls within the quasi-linear section of the curve, where small changes in $x$ yield informative variations in permanence cost. Although arbitrary, these thresholds respect the data scale and produce a smooth yet discriminative cost surface.

**- Model 1 (“C5/C3”)**

An alternative approach to modeling permanence cost ($c_{jj}$) is based on observed behavioral indicators within the Vodafone dataset. Specifically, Model 1 defines permanence as a linear function of the historical average of the ratio $\overline{C5/C3}$, computed per grid and time interval across several months of data. This ratio compares the number of detected entries (*C5*) to the number of devices remaining in the grid at the end of the five-minute window (*C3*). The intuition behind this model is straightforward: a higher *C5*/*C3* ratio indicates a grid where many devices pass through but few remain, suggesting that

such a location is more likely to be a transit zone (e.g., a highway) rather than a place of stationary activity. Consequently, a higher permanence cost is assigned to discourage the optimizer from keeping devices in such grids.

However, after applying this model to Vodafone data collected between June and December 2023 (restricted to hours from 8:00 to 20:00), a number of inconsistencies emerged. First, the computed average ratios were systematically low across most grids, with maximum values around 4. This appears to be primarily due to the nature of mobile phone detection: the long and irregular ping intervals mean that many devices may enter and exit a grid without being captured in real time. As a result, the *C5* and *C3* values may not accurately reflect the true volume and flow of traffic, leading to artificially low ratios and potentially misleading interpretations.

Additionally, the 50-meter average localization error inherent to the dataset introduces further complications. In some cases, grid cells that are composed entirely of road surface (e.g., motorways) still display unexpectedly low *C5*/*C3* ratios. This can be explained by the fact that mobile phones located in buildings near the edges of adjacent grids may be misattributed to the road-only grid, inflating the *C3* value and deflating the ratio. This effect is especially problematic in densely built areas or near interchanges, where building proximity and road geometry create overlapping detection zones.

To illustrate the model's limitations, Figure 5.4 presents the average *C5*/*C3* ratio for each grid over the observation period. It is immediately apparent that the ratios are lower than expected for a reliable indicator of transit activity. For example, in grid 1115, located on the A5 motorway and containing no buildings, the average *C5*/*C3* ratio is 3.48. Assuming that devices are evenly distributed over time and that cars flow continuously through the grid, this would suggest that only 1 out of every 3.48 entering cars remains in the grid after five minutes. Inverting this, we would expect a car to remain in the grid for approximately $5/3.48 \approx 86.2$ seconds. Given that the length of the grid is about 156 meters, this implies an average speed of 6.52 km/h—an implausibly low figure for a major highway segment.

Further inconsistencies are shown in Figure 5.5, which compares the average *C5*/*C3* values for grids 1115 (A5 highway) and 2546 (the intersection of Eixo N–S and 2ª Circular). Despite both grids being composed exclusively of high-speed roads and lacking buildings, their ratios differ significantly. This discrepancy likely stems from the aforementioned localization errors, which may cause phones in nearby urban areas to be falsely attributed to one grid more than the other, thereby distorting the ratio in the case of grid 2546.

In summary, although Model 1 offers a conceptually appealing, behavior-based approach by leveraging observed flow dynamics within each grid, it was ultimately excluded from the final implementation. The model exhibited structural inconsistencies rooted in the limitations of the underlying data—namely, long ping intervals and geolocation inaccuracies—which significantly distorted the computed *C5*/*C3* ratios, particularly in road-only or boundary grids. Additionally, when tested against Model 0 ("Builds"), the performance of Model 1 yielded inferior results in improving the quality of the traffic inference. For these reasons, and in the interest of robustness, Model 1 was not used in the final formulation of permanence costs.

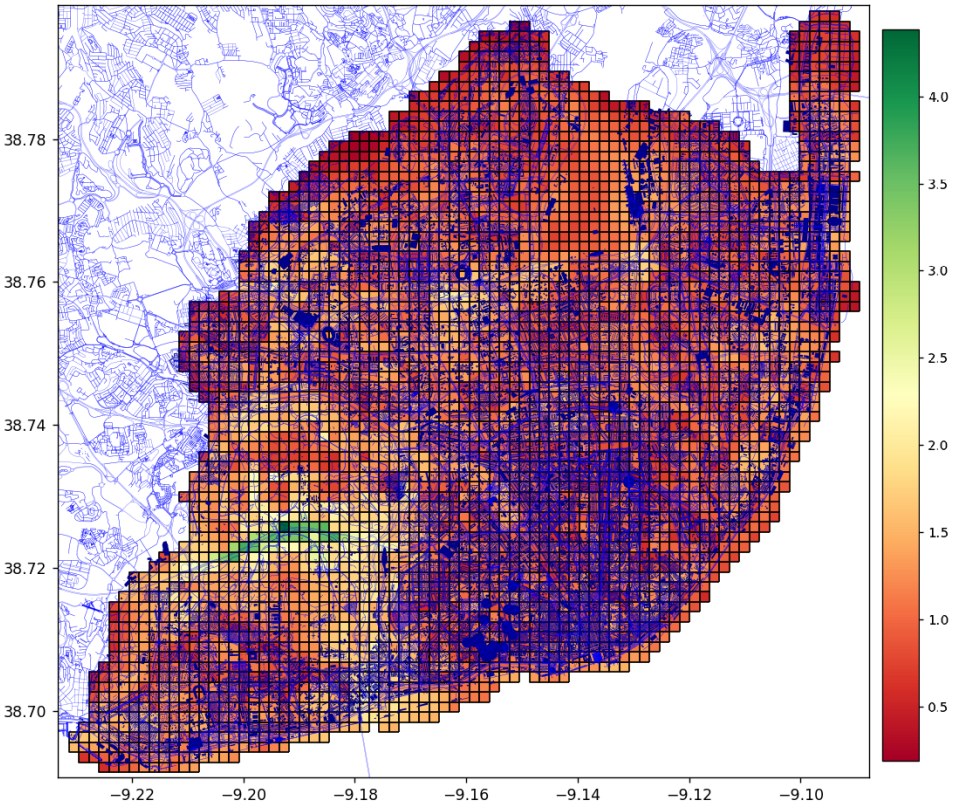



Figure 5.4: C5 over C3 by grid map, with highways and buildings

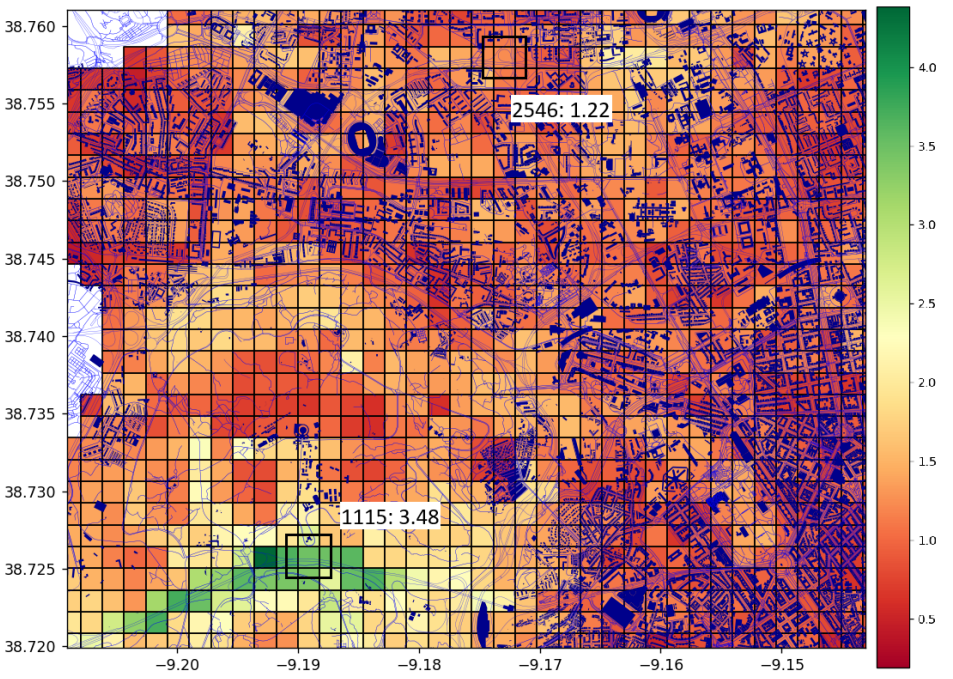



Figure 5.5: C5 over C3 by grid focusing on grids 1115 and 2546

### 5.2.3 Low-Cost Inflation

To prevent short-distance hops from dominating the solution, a cost inflation mechanism was introduced to gently increase the lowest costs while preserving overall structure and monotonicity. This adjustment is achieved through a modified soft-plus function applied to all costs below a chosen threshold $\tau$, producing an inflated cost $\tilde{c}$:

$$\tilde{c} = \frac{1}{\alpha} \log\Big(1 + e^{\alpha(c-\tau)}\Big) + \tau,$$

where $\alpha > 0$ controls the smoothness of the transition and $\tau$ defines the threshold under which costs are inflated. This transformation has some key properties:

- $\tilde{c}$ is strictly increasing with respect to $c$, preserving order.
- For values $c \gg \tau$, the function behaves like the identity, i.e., $\tilde{c} \approx c$, leaving larger costs unaffected.
- For values $c < \tau$, the difference $\tilde{c} - c$ becomes substantial, with a maximum inflation occurring near $c = 0$ and smoothly tapering off as $c$ approaches $\tau$.

The user does not set $\alpha$ directly, but instead defines an input scaling parameter *alpha_in*, which is converted internally as:

$$\alpha = \frac{\textit{alpha\_in}}{\tau/100}.$$

This normalization allows *alpha_in* to remain intuitive and relatively invariant to the scale of the $\tau$ input, making calibration easier.

This mechanism was introduced as part of the model calibration strategy. By increasing the cost of very short-distance movements, the system is nudged toward distributing flow over longer and more realistic paths. This adjustment is particularly useful for better simulating real-world traffic behavior. Figure 5.6 shows an example of this inflation function applied with $\tau = 500$ and *alpha_in* $= 0.04$. The costs in this graph represent tenths of seconds.

### 5.2.4 Expected-Movement and Building-Density Biases

#### - Expected-Movement

Following the introduction of permanence costs (diagonal entries $c_{jj}$), a behavioral shift was observed in the model: mobile phones that were previously static began exhibiting short-range hopping, frequently transitioning only to adjacent grids. To counteract this, an additional mechanism—referred to as the “expected movement” bias—was implemented to encourage longer, more plausible movements, particularly from grids dominated by major road infrastructure.

The central idea is to define, for each origin grid $i$, a minimum plausible movement threshold based on the physical characteristics of that grid. Specifically, an expected movement distance $M_i$ is computed as a function of the total road length $L_i$ within the grid:

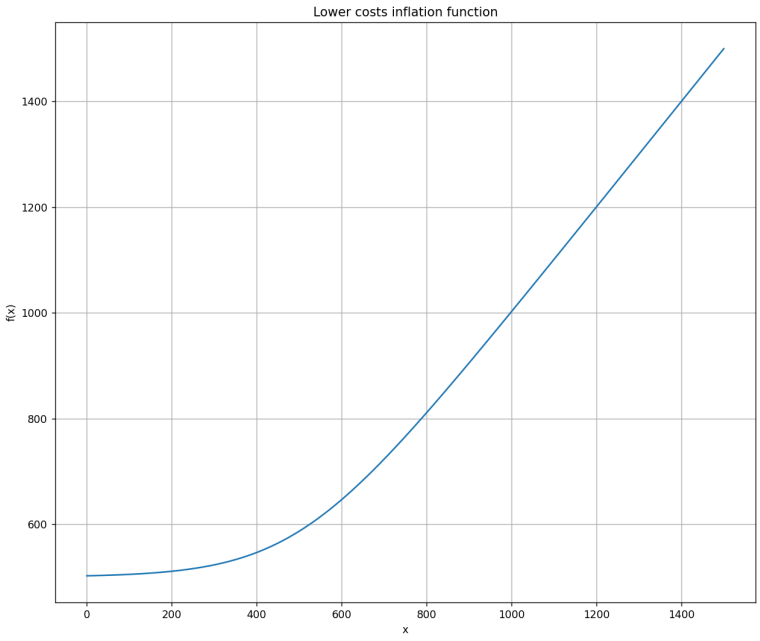



Figure 5.6: Example of low-cost inflation function with $\tau = 500$ and *alpha_in* $= 0.04$.

$$M_i = \frac{L_i}{expec_{div}},$$

where $expec_{div}$ is a scaling parameter that calibrates the strength of the adjustment. In a more refined variant of the model, this expected movement is reduced in grids with substantial building coverage. Let $B_i$ denote the total built-up area within the grid. The adjusted expected movement then becomes:

$$M_i = \max\left(\frac{L_i}{expec_{div}} - \frac{B_i}{div_{builds}},\ 0\right),$$

where $div_{builds}$ is a second divisor controlling the influence of urban density. This building-based correction was introduced in a later phase of the project and still requires further calibration to properly balance its effects in mixed-use environments.

Once $M_i$ is defined, the cost matrix is modified accordingly. For all destinations $j \neq i$, if the original travel cost $c_{ij}$ is less than $M_i$, the cost is inflated to:

$$\tilde{c}_{ij} = M_i + (M_i - c_{ij}^{\text{orig}}),$$

where $c_{ij}^{\text{orig}}$ is the original routing cost. This adjustment ensures that short-distance costs from high-mobility grids are penalized, promoting a shift toward more spatially distributed movement. The inflated cost function is strictly increasing for $c_{ij} < M_i$, and smoothly aligns with the original cost values for $c_{ij} \geq M_i$. For the diagonal entries, the adjustment is applied as:

$$\tilde{c}_{ii} = c_{ii}^{\text{orig}} + M_i.$$

Although this formulation is somewhat simplistic in the treatment of $c_{ii}$, it guarantees that remaining in the grid is always more expensive than traveling the minimum expected distance—thus discouraging artificial stillness in high-flow areas.

Initially, this bias was also applied to the Vodafone Eixos—grids representing city main access points at the periphery. However, these Eixo polygons span large geographic areas, and the start/end points used for routing were very close to the Vodafone grids. Even a short step from such a point might correspond to hundreds of meters of actual road travel. As such, penalizing short costs near these boundaries didn't make sense. Empirical testing confirmed that excluding the Eixos from the expected movement adjustment improved flow patterns and model coherence.

Lastly, to preserve the integrity of rail-based travel, all grid pairs associated with train routing paths were explicitly exempted from this bias. Their original travel costs were restored after the expected movement transformation was applied to the rest of the matrix.

In sum, this mechanism addresses the overconcentration of short-distance transitions in road-dense grids and helps promote more realistic spatial dispersion in travel patterns, consistent with the functional layout of urban infrastructure.

**- Building-Density**

Densely built-up areas—such as residential neighborhoods or commercial zones—tend to favor shorter and slower movements. To account for this, an additive cost adjustment is introduced for every movement that either starts from or ends in a grid with high building density. Specifically, for every grid $i$, and for all destinations $k$, the cost matrix is updated as follows:

$$c_{ik} \mathrel{+}= b_{coef}\, B_i, \qquad c_{ki} \mathrel{+}= b_{coef}\, B_i, \qquad \forall\, k,$$

where $B_i$ represents the building coverage of grid $i$, and $b_{coef}$ is a global scaling factor controlling the weight of the adjustment.

In practice, $B_i$ is computed from the total intersection area between buildings and the grid cell, derived from geospatial data. For each OD pair $(i, j)$, the adjustment is based on the higher of the two building areas between the origin and destination. This asymmetric application helps account for the fact that either entering or exiting a dense area may contribute to slower maneuvering and additional routing complexity.

Diagonal entries ($c_{ii}$) are treated with additional care. Since permanence already carries a separate penalty, the building-based adjustment on $c_{ii}$ is further scaled down to avoid overinflating the cost of staying within densely populated grids. This adjustment uses an extra divisor to reduce its impact when applied to diagonal terms, while still preserving the intended behavioral effect: grids with high built-up density are less likely to encourage movement out of and to the grid.

This mechanism subtly reinforces spatial realism by embedding the influence of urban morphology into the cost structure, encouraging behavior that mirrors real-world movement constraints within complex or congested built environments.

## 5.3 Model Constraints

### 5.3.1 Variables and Filtering

**- Filtering x$_{jk}$ variables**

The decision variables in the model are denoted $x_{jk}$, representing the estimated number of mobile devices moving from grid $j$ to grid $k$ as already mentioned. However, not all potential variables are included: during the construction of the cost matrix $c_{jk}$, any variable assigned an infinite cost was excluded from the optimization problem. This filtering step ensures computational tractability and aligns the model with plausible spatial transitions.

**- Retreat Variables**

To account for transient mobile phone behavior—specifically, when a device exits a grid cell, briefly enters a neighboring cell, and then returns—*retreat variables* $r_{jk}$ were introduced. These variables were critical for satisfying minimum entry constraints, particularly in grids with little connectivity or isolated positions. The cost associated with each $r_{jk}$ reflects the round-trip nature of this movement:

$$\text{Cost}(r_{jk}) = c_{jj} + c_{jk} + c_{kj},$$

where $c_{jj}$ is the cost of remaining in grid $j$, and $c_{jk}$, $c_{kj}$ represent outbound and return movements, respectively.

### 5.3.2 Flow Conservation Constraints

A fundamental requirement of the inference model is mass conservation across the spatial grid system. For each grid cell, the total outgoing mobile phones at the initial time ($m_0[j]$) must match the sum of flows departing from that cell, and the total incoming mobile phones at the final time ($m_1[k]$) must match the flows arriving into it. These flow conservation constraints, referenced in equation 2.4, form the backbone of the optimization problem.

In a standard formulation, the conservation constraints are given as:

$$m_0[j] = \sum_k x_{jk}, \quad \text{and} \quad m_1[k] = \sum_j x_{jk}.$$

However, due to the introduction of retreat variables $r_{jk}$ the standard constraints had to be extended.

**- Outflow Constraint**

The outflow from each grid $j$ must now account not only for transitions $x_{jk}$ to other grids $k$, but also for round-trip movements initiated from $j$ that pass through its direct neighbors and return. Thus, the modified outflow equation becomes:

$$m_0[j] = \sum_k x_{jk} + \sum_{k \in \mathcal{N}(j)} r_{jk},$$

where $\mathcal{N}(j)$ denotes the set of valid direct neighbors of grid $j$ (typically 8-connected).

**- Inflow Constraint**

During the final review of the project, it was discovered that the inflow constraints had mistakenly omitted the retreat contributions. While the model correctly summed the $x_{jk}$ flows arriving at each grid $k$, it failed to include the retreat variables $r_{kj}$ where the round-trip terminates in $k$. As a result, the retreat variables could not ever be different than 0, something that was wrongly not verified at the time.

The corrected inflow constraint for each grid $k$ is:

$$m_1[k] = \sum_j x_{jk} + \sum_{j \in \mathcal{N}(k)} r_{kj}.$$

This ensures that all mass arriving at $k$—whether via standard transitions or returning flows from neighbors—is properly accounted for.

The implementation of retreat variables was initially motivated by recurring infeasibility issues encountered during the mid stages of the project. However, the parallel development of alternative feasibility mechanisms—such as the introduction of slack variables—overshadowed the role of the retreat variables and obscured their effectiveness. While these alternative measures did enable the model to produce feasible solutions, they effectively ”brute-forced” feasibility rather than resolving the root cause of the imbalances.

It was only after correcting the inflow constraint to properly account for the retreat variables that the model began to exhibit the intended behavior. Once this correction was in place, the retreat variables took on meaningful, non-zero values, and the model consistently achieved feasibility without resorting to auxiliary mechanisms such as slack variables. This resolution underscored the retreat variables' critical structural role in maintaining mass balance and ensuring model integrity.

### 5.3.3 Upper Bound Constraints

**- Transition Variables**

Each transition variable $x_{jk}$, representing the estimated number of mobile devices moving from grid $j$ to grid $k$, is assigned an upper bound to prevent unrealistic flow concentration. In real-world urban mobility, it is highly improbable for an entire grid's mobile population to move exclusively to a single

destination grid. Similarly, it is unlikely for a single grid to receive all its mass from only one origin.

The upper bound is defined as:

$$\bar{x}_{jk} = \frac{\min(m_0[j], m_1[k])}{\delta_{jk}},$$

where the divisor $\delta_{jk}$ incorporates two key considerations:

1. **Local spatial structure:** Grids located in the interior of the Lisbon area typically have a larger number of neighboring cells, enabling more diverse and distributed movement patterns. In contrast, border grids are inherently limited in their connectivity within the modeled space. The model determines the number of neighboring grids within a distance of $\sqrt{5} \cdot$ cell_side_length from grid $j$. For interior grids, this number is consistently 20. This count is then divided by a configurable parameter—default value: 5—to compute the divisor. For instance, an interior grid yields $\delta_{jk} = 20/5 = 4$. On the borders, where the number of neighbors may drop to 10 or fewer, the divisor is reduced accordingly (e.g., $\delta_{jk} = 10/5 = 2$), permitting more concentrated flows due to limited spatial options.

2. **Magnitude of mobile population:** When the mass in either the origin or destination grid is small, highly concentrated flows become more plausible. If $\min(m_0[j], m_1[k]) < 10$, the model sets $\delta_{jk} = 1$. Otherwise, the divisor is proportional to the mass: $\delta_{jk} = \min(m_0[j], m_1[k])/10$.

The final divisor $\delta_{jk}$ is chosen as the smaller (i.e., more permissive) of the two, ensuring that the upper bound does not overly restrict feasible solutions while still reflecting spatial and demographic realism. This conservative bounding strategy enables the model to maintain both mathematical solvability and consistency with plausible urban mobility patterns.

**- Permanence Variables**

For permanence variables ($x_{jj}$), which represent the number of devices that remain in the same grid cell $j$ over the inference window, the default upper bound is set to:

$$\bar{x}_{jj} = m_0[j].$$

To improve the realism of this constraint, an enhanced configuration introduces a refined upper bound based on observed entry data (C5) and grid-specific infrastructure characteristics:

$$\bar{x}_{jj} = \min(m_0[j], m_1[j]) - C5_j \cdot (\textit{base\_c5\_factor} + \textit{normed\_c\_diag}[j]),$$

where:

- $C5_j$ is the number of entries into grid $j$ observed during the time interval,
- *base_c5_factor* is a user-defined coefficient that represents the fraction of these entries expected to lead to outflows (i.e., devices entering and then leaving the grid),

- *normed_c_diag*$[j]$ is a normalized permanence factor given by:

$$\textit{normed\_c\_diag}[j] = \frac{c_{jj}}{max_{model0}} \cdot \textit{normed\_mx\_c5},$$

  where $max_{model0}$ is the maximum cost of permanence defined by the user, and *normed_mx_c5* is a scaling hyperparameter that tunes the weight of this term,

- The resulting upper bound is constrained to be non-negative via a $\max(\cdot, 0)$ operation in the implementation.

This refined formula is motivated by the sparse pings issue discussed in previous chapters. In highly active grids with significant observed entries, it is unlikely that all entries correspond to through-traffic. Some proportion of these entries are expected to result in actual permanence, justifying the inclusion of a penalizing term. The *base_c5_factor* captures this intuition by assuming that some fraction of C5 must be balanced by corresponding exits, thereby limiting the allowable permanence.

However, the current formulation does not explicitly account for imbalances between $m_0[j]$ and $m_1[j]$. For instance, if $m_0[j] \neq m_1[j]$, then the assumption that stays imply exits may no longer hold. In such cases, the upper bound may become too strict, undermining feasibility. A more nuanced formulation could consider adjusting $C5_j$ dynamically based on the magnitude of the imbalance, for example:

$$C5_j^{\text{adj}} = \max\left(0, C5_j - |m_0[j] - m_1[j]|\right).$$

Additionally, the *normed_c_diag*$[j]$ term helps differentiate grids by their structural role. Grids with high $c_{jj}$ (e.g., road-dominated areas) imply higher transient behavior—entries are likely to exit again. Conversely, grids with low $c_{jj}$, typically associated with residential or commercial areas, offer little direct evidence regarding $x_{jj}$, and thus will have a less restrictive upper bound.

Together, this configuration introduces a mechanism to modulate the permanence bounds based on both behavioral assumptions and spatial context, while still allowing the user to disable or tune its impact via hyperparameters.

## 5.4 Minimum C5 Constraints

Unlike the conservation and upper-bound rules, the Minimum C5 constraints are driven by the time-varying entry counts in the Vodafone data. They impose a lower bound on the total flow through selected grids, tying the optimization directly to observed movement. Each constraint pools thousands of decision variables, and more than 2 500 such constraints must be enforced in every five-minute interval. This swells the linear program considerably, but the resulting gain in realism fully justifies the added computational cost.

Adding these constraints first requires identifying, for each decision variable $x_{jk}$, the set of grids its route intersects; next, retaining only those grids for which a minimum-C5 bound is meaningful; and

finally translating the resulting grid–variable associations into linear inequalities that are appended to the optimization model.

### 5.4.1 List of Grids Intersected

For every valid grid pair $(j, k)$ obtained in the routing stage (see Chapter 4), the optimal road path is stored as a *LineString*. In an offline pre-processing step this polyline is overlaid with the 3 743 Vodafone grids; wherever the path intersects a grid polygon the corresponding *Grid ID* is recorded. The result is a static mapping

$$(j, k) \longmapsto \{g_1, g_2, \dots\} \subseteq \{1, \dots, 3743\},$$

which is serialized to a compressed *pickle* and reused at every five-minute optimization.

A grid is currently tagged as intersected as soon as any portion of the path touches it. Although pragmatic, this binary rule can over-state cells where the overlap is only a few meters.

Figure 5.7 shows the optimal path from grid 2610 to grid 2169. The highlighted grid, despite being intersected by only a very short segment of the route, is still flagged as “crossed” under the current rule, just like grids with much longer overlaps.

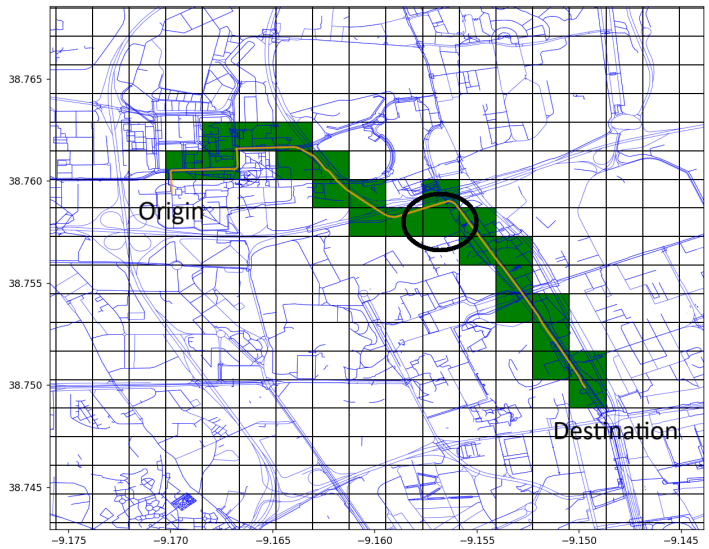



Figure 5.7: Grids intersected in the path between grid 2610 and grid 2169

Given the low sampling rate of localization pings, a mobile phone moving from $j$ to $k$ is unlikely to be registered in such brief crossings, whereas detection is far more probable in grids where the path segment is longer. A natural extension would be to replace the all-or-nothing flag with a weighting that scales each grid’s contribution by the length of road traversed inside it, thereby sharpening the Minimum C5 constraints without incurring prohibitive computational cost. Time constraints prevented its implementation, but it remains a priority for future work.

### 5.4.2 Filtering Grids

Not every grid is a sensible candidate for a Minimum–C5 bound; applying the constraint where through-traffic is implausible would either over-tighten the model or render it infeasible. Accordingly, a four-stage spatial filter is carried out before optimization (Figure 5.8):

1. **Road-coverage test.** Each grid is scored by the weighted length of its road network. Grids whose score falls below a threshold are discarded, as the scarcity or minor hierarchy of their roads makes any minimum-flow requirement dubious.

2. **Road–building balance.** The same dataset reports the building footprint within every grid. After rescaling that footprint, grids where built area clearly dominates road length are removed; these are typically dense residential blocks where most detected phones belong to pedestrians rather than passing vehicles.

3. **C5 plausibility check.** Grids that combine a sparse or low-hierarchy road network with an unusually high entry count (C5) are likely to reflect pedestrian-dominated areas—such as shopping centers, dense residential blocks, or office campuses—or may simply represent data outliers. Since such volumes do not indicate genuine through-traffic, these grids are excluded based on a combined road-length and C5 threshold.

4. **Border exclusion.** Grids on the map edge—identified by having fewer than seven neighbors within 250 m—are omitted. A large share of their entries represents phones entering or leaving the study area, which the mass-balance model cannot accommodate convincingly.

After these automatic stages, a short list of *problematic* grids that repeatedly produced infeasibilities is removed manually. The resulting selection—about 2 730 grids on average—defines the set to which the Minimum–C5 constraints apply.

The present implementation omits grids that are crossed exclusively by the railway lines used in the optimized train paths. As a result, movements confined to these rail-only grids are exempt from the Minimum-C5 requirement and are undervalued in the objective. Extending the filtering pipeline to include such grids—so that train flows can also trigger C5 bounds—remains an important task for future work.

### 5.4.3 Applying the Constraints

Once the set of valid grids is established, the Minimum–C5 constraints are formally introduced into the optimization model. Each constraint ensures that the total predicted flow crossing a given grid $j$ is no less than its observed C5 minimum:

$$\sum_{x_{jk} \in V_j} x_{jk} \;\geq\; \mathrm{C5}_j \,,$$

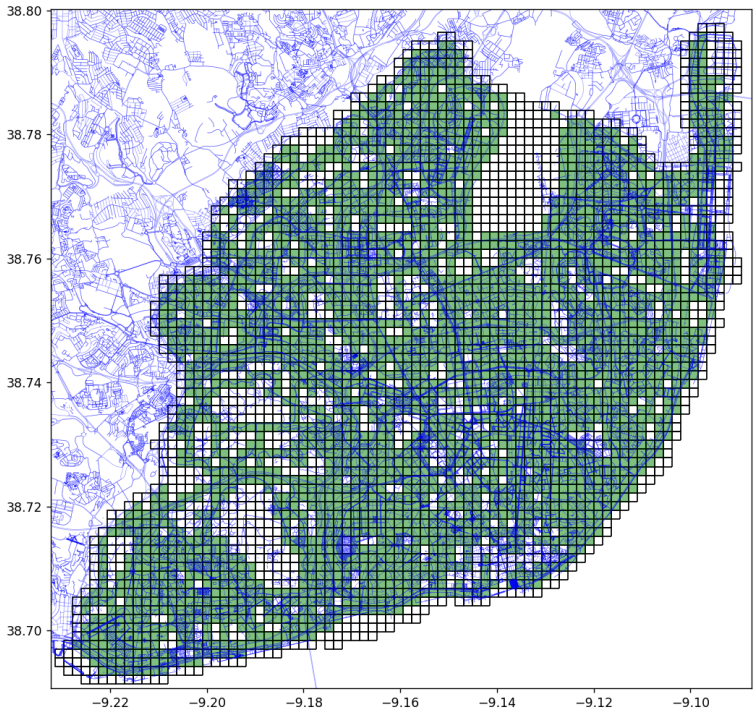



Figure 5.8: Grids retained for the Minimum–C5 constraints after the filtering pipeline.

where $V_j$ is the set of decision variables whose associated routes either intersect or end in grid $j$. These sets are built from the preprocessed routing lists described earlier, where each variable $(j, k)$ is mapped to the list of grids its path traverses. To construct the constraints, the algorithm iterates over every valid variable $x_{jk}$; for each grid intersected by its route, the corresponding variable is added to that grid's list of contributors. In this way, each grid aggregates the full set of optimization variables needed to enforce its Minimum–C5 flow constraint.

To accurately account for the paths of each transport mode, the process excludes train movements from car-only constraints, and vice versa. Train routes are handled separately: their intersected grids are loaded from a dedicated file, and their associated variables are added only to the relevant grid constraints.

Additional care is taken with adjacent movements—short-distance transitions between neighboring grids—that may be underrepresented in the routing data. These are explicitly detected based on grid proximity and incorporated into the corresponding constraints.

Retreat variables, introduced to model looped movements of the form $j \to k \to j$, are also included in the constraints for both grids $j$ and $k$. This ensures that such auxiliary flows are correctly accounted for when enforcing the minimum-entry requirement.

To improve robustness and address potential infeasibilities, all Minimum–C5 constraints are relaxed using slack variables. Each constraint receives a non-negative slack variable, which is penalized in

the objective function with a large cost (typically $10^5$). This allows the solver to bypass specific constraints in rare cases where exact satisfaction is impossible—particularly useful in edge cases involving ambiguous routing or inconsistent data.

The final result is a set of over 2 700 linear lower-bound constraints, one per selected grid, each built from routing-aware variable selection and protected by a high-penalty slack. This mechanism tightly anchors the optimization to observed flows while preserving numerical solvability.

## 5.5 CPLEX Model and Infeasibility Management

The optimization problem is solved with IBM *CPLEX* [35], chosen for its proven performance on large-scale linear programs and its extensive parameter control. CPLEX offers state-of-the-art presolve routines, deterministic parallelism and multiple LP algorithms, all of which are crucial when handling the present instance with $\sim 3.5$ million variables and constraints.

### 5.5.1 Choice of LP Algorithm

During prototyping, no algorithm was preferred *a priori*. Consequently, CPLEX was initially executed in *concurrent* mode, starting the dual and primal simplex, barrier, and sifting solvers in parallel on the available hardware threads.

Although the barrier and dual simplex engines quickly reduced the model, *sifting* was systematically the first and only algorithm to reach optimality. The remaining algorithms tended to stall at an advanced basis, while their memory footprints grew beyond 20 GB because of the search trees they maintain. Based on these empirical observations, the following definitive configuration was adopted:

- **LP method—*Sifting*:** This algorithm alternates barrier and simplex phases, combining the barrier's numerical robustness with the simplex warm-start capability. It performs especially well on our highly sparse, weakly scaled model.
- **Thread count:** A user-controlled parameter; in general, the more CPUs assigned, the faster the solver converges.
- **Deterministic parallelism:** CPLEX is executed in deterministic parallel mode, fixing thread scheduling so that two runs on the same data produce bit-for-bit identical results, regardless of hardware.
- **Scaling:** The solver's internal coefficient-scaling heuristics are disabled, allowing the optimization to proceed with the original magnitudes of the cost coefficients, avoiding rounding artifacts.

### 5.5.2 Automatic Fallback for Infeasible Runs

In early development—before the introduction of adjacent-movement variables, retreat arcs and slack relaxations—the model occasionally became infeasible. To avoid manual intervention, an automated fallback routine was implemented:

1. Run CPLEX and capture its entire log.
2. If the solve terminates with status *infeasible*, parse the log for the most violated $min_{C5}$ constraint.
3. Remove that single constraint and resolve.
4. Repeat for at most five iterations, each time eliminating the next most violated $min_{C5}$ constraint.

With the full set of robustness features now in place, this fallback is rarely triggered; nevertheless, it remains part of the workflow as a safeguard against unexpected edge cases or future model extensions.

If, despite the fallback routine, CPLEX is still unable to produce a feasible solution for a given period, the program's outer loop reacts by simply skipping that period. When the problem is configured for a single period ($n_{per} = 1$), this skip results in immediate termination and no output is produced. For multi-period runs ($n_{per} > 1$), the loop discards only the problematic interval, decrements $n_{per}$ accordingly, and continues with the remaining periods. During the final validation stage, the ground-truth hourly traffic counts are rescaled by the factor $n_{\text{per}}/12$ so that the results remain comparable even when fewer than the original twelve periods are ultimately solved.

## 5.6 Solution Sectioning

After obtaining the solution matrix of all $x_{jk}$, it is needed to project and aggregate all these movements into observable and meaningful traffic volumes. To achieve this, and ever before projection and aggregation, it was needed to differentiate and remove the movements associated with other transport or pedestrian movements.

### 5.6.1 Removing non-vehicular flows

**Filtering short (pedestrian) displacements.** Before translating the solution matrix $\mathbf{X}$ into vehicle flows, all movements whose origin and destination lie in immediately adjacent grids are entirely discarded, as these very short displacements are assumed to represent pedestrian rather than vehicular traffic. For movements covering slightly larger distances—specifically those whose Euclidean distance is greater than the grid side length $s$ (adjacent cells), but still less than or equal to $2s$ or $\sqrt{5}s$ (two cells apart horizontally/vertically or 1 diagonally and 1 horizontally/vertically)—the user may define a coefficient walking_coef $\in [0, 1]$ to proportionally reduce the corresponding values of $\mathbf{X}$. Setting this coefficient to 0 eliminates these movements entirely, while a value of 1 retains them fully as motorized traffic.

**Suppressing rail-related flows.** Movements associated with train movements are next nullified. The set train movements built in the demand model, defined in 5.2.1, lists every pair $(i, j)$ that should be treated as a rail trip; as it was done before, the program simply forces the corresponding $\mathbf{X}_{ij}$ values to zero.

### 5.6.2 Building and counting sections

After applying the filters to exclude non-motorized and rail-associated movements, the remaining positive elements of the solution matrix $\mathbf{X}$ are converted into explicit vehicle flows associated with their optimized paths. For each origin–destination pair $(i, j)$ with a non-zero flow, the corresponding polyline path obtained from the routing optimization (described in Chapter 4) is decomposed into smaller segments (sections). These polylines, stored as *LineString* geometries with multiple vertices, generate $m - 1$ sections from $m$ vertices. While most of these sections repeat across different routes due to the common OpenStreetMap (OSM) network used for all routing optimizations, some sections differ at origins and destinations, which typically do not align exactly with predefined segment breakpoints.

Each section, defined by consecutive pairs of latitude–longitude coordinates, is managed using a dedicated data structure that tracks cumulative vehicle counts, initially set to zero. To enhance computational efficiency, this counting process is parallelized using multiprocessing, with each processor independently handling a subset of grid identifiers and incrementally updating vehicle counts in a local dictionary of road sections. Once all parallel computations are complete, the individual dictionaries from each thread are merged into a single comprehensive dictionary containing aggregated flow counts.

The results are then structured into a geospatial dataset (GeoDataFrame) for subsequent visualization and analysis. Each entry in this dataset corresponds to a unique road segment along with its cumulative vehicle flow, providing the foundation for detailed flow maps. An illustrative example of such a flow map is presented in Figure 5.9[2], demonstrating also that the count for each road segment $s$ is given by the summation of all optimized flows $x_{jk}$ whose paths traverse the segment:

$$\text{Count}_s = \sum_{(j,k)\,:\,s \in \text{path}(j,k)} x_{jk}$$

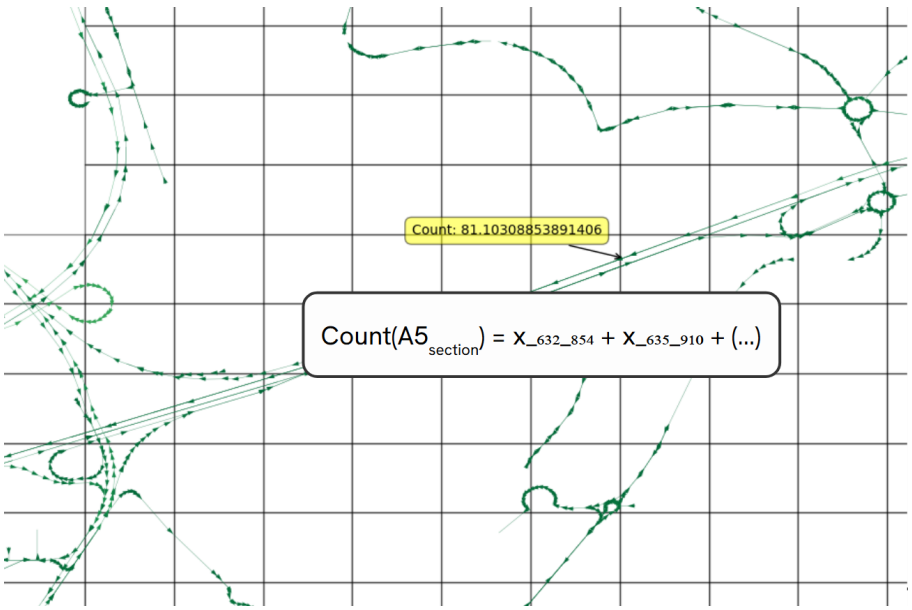



Figure 5.9: Flow map in A5 highway.

[2]The values of $j$ and $k$ used to illustrate $x_{jk}$ in this figure are hypothetical and shown for explanatory purposes only; however, the traffic count itself is based on the actual optimized solution.

## 5.7 Traffic Estimates and Validation

Having computed the total mobile phone flow for each road segment across Lisbon, we now proceed to validate our model's forecasts. This validation involves selecting specific locations, aggregating the bidirectional flows at these points, and comparing these aggregated results against available ground-truth traffic data. The validation process involves several carefully executed steps, detailed in the following subsections.

### 5.7.1 Flow Aggregators or "Radars"

In this subsection, we outline the method used for defining and positioning the flow aggregators, referred to here as "radars" for simplicity. These radars function as aggregators due to the complex geometries of the flows identified previously, which cannot always be straightforwardly combined.

The radar implementation involves creating a flat rectangular area intersecting the road segments to differentiate and sum traffic flows moving in both directions (from top to base and vice versa). Crucially, the placement and orientation of these rectangles must ensure intersection only with the intended road segment, excluding nearby unrelated roads. To achieve precise placement, the radar rectangle's center is carefully positioned equidistantly between the directional edges of the road. Its length and orientation angle are then fine-tuned to guarantee accurate coverage exclusively of the target road segment. The geographical projection employed for this process is WGS84, the same as every other projection of this project.

To effectively validate the model predictions, we first identified locations with available ground-truth data. However, ground-truth data availability is limited to only six locations as listed in Table 3.4.1. Nonetheless, additional radar points were deemed necessary to roughly evaluate the consistency and coherence of model inference across the entire city. Thus, 16 supplementary locations were strategically selected based on several criteria:

- Geographic distribution across Lisbon;
- Preference for locations within grids mainly composed of roads rather than building structures;
- Avoidance of proximity to major public transportation routes (e.g., train or metro);
- Specific scenarios tailored to test and validate particular aspects of the model.

The complete list of the 22 radar locations, including their associated parameters, is presented in Table 5.2, with their geographical distribution across Lisbon illustrated in Figure 5.10.

Two specific scenarios were considered in radar placement to evaluate potential limitations in the model:

1. In grids where parallel roads enter and exit in similar cardinal directions, such as Avenida da Índia and Avenida de Brasília in grid 70, only one road (Avenida de Brasília) was designated for origin and destination points based on intersection length within the grid. Given that this configuration

| Location | Grid_ID | Center Coords. (WGS84) | Angle (deg) | Length (WGS84) |
|---|---|---|---|---|
| Calçada de Carriche | 3574 | (38.783654, -9.166856) | 55º | 0.0004 |
| Av. da Liberdade (Restaur.) | 742 | (38.716510, -9.142417) | 22º | 0.0012 |
| Av. Eng. Duarte Pacheco | 1014 | (38.722968, -9.167577) | 110º | 0.0002 |
| Av. da Républica | 1553 | (38.735931, -9.145660) | 14º | 0.0010 |
| Av. de Berlim | 3058 | (38.769492, -9.122116) | -90º | 0.0005 |
| Av. das Forças Armadas | 2036 | (38.747371, -9.155364) | 100º | 0.0004 |
| 2ª Circular (Eixo N-S) | 2546 | (38.757587, -9.173457) | 90º | 0.0003 |
| Eixo N-S (2ª Circular) | 2546 | (38.758445, -9.172284) | 0º | 0.00025 |
| Eixo N-S (Telheiras) | 3032 | (38.768905, -9.169772) | 120º | 0.00025 |
| Eixo N-S (A2) | 953 | (38.721900, -9.176380) | -30º | 0.00025 |
| A5 (Monsanto) | 1114 | (38.725457, -9.191344) | 90º | 0.0003 |
| Av. das Descobertas | 586 | (38.713144, -9.213735) | 134º | 0.0003 |
| Av. dos EUA | 2175 | (38.749109, -9.138235) | 90º | 0.00025 |
| Pt. Vasco da Gama | 3625 | (38.786619, -9.098836) | 120º | 0.0003 |
| Av. Infante D. Henrique | 930 | (38.719993, -9.115843) | -38º | 0.0002 |
| Av. Brasília | 70 | (38.696422, -9.193307) | 90º | 0.00015 |
| Av. da Índia | 70 | (38.696808, -9.192540) | 90º | 0.00015 |
| Av. Calouste Gulbenkian | 1484 | (38.733614, -9.163949) | 0º | 0.00025 |
| Av. Gen. Correia Barreto | 1888 | (38.743823, -9.193162) | 90º | 0.0002 |
| 2ª Circular (E.S. Comunic.) | 2016 | (38.746310, -9.191312) | 130º | 0.0003 |
| Av. dos Combatentes | 1844 | (38.743074, -9.160466) | 0º | 0.0005 |
| Av. Mar. Gomes da Costa | 2700 | (38.760781, -9.119736) | 90º | 0.0004 |

Table 5.2: All 22 Radars with the respective parameters

recurs frequently along the two roads, radars were specifically placed on both Avenida da Índia and Avenida de Brasília within grid 70 to assess the impact of this choice on estimated traffic volumes.

2. At the interchange between the Eixo N-S and 2ª Circular highways (grid 2546), radars were positioned to independently capture the traffic volumes on each road. This arrangement aimed to evaluate the balance and distribution of traffic flows, given that these highways share traffic mass within the grid, potentially resulting in unbalanced estimations.

The radars' placement for these situations is visualized in Figures 5.11a and 5.11b.

A key consideration in radar placement is its relation to the origin and destination points used for routing simulation. Radars must be positioned "ahead" of origin points and "behind" destination points to accurately capture the flows originating and terminating within the same grid. In grid 2546, radar placement explicitly accounted for this, as depicted in Figure 5.12. Generally, traffic volume decreases in mid-sections of roads due to early destination points and delayed origin points within grids.

A potential improvement for future work is aligning origin and destination points closer to the centers of the grids. This approach could significantly reduce gaps in traffic flow estimations, resulting in a more cohesive and accurate representation of traffic dynamics.

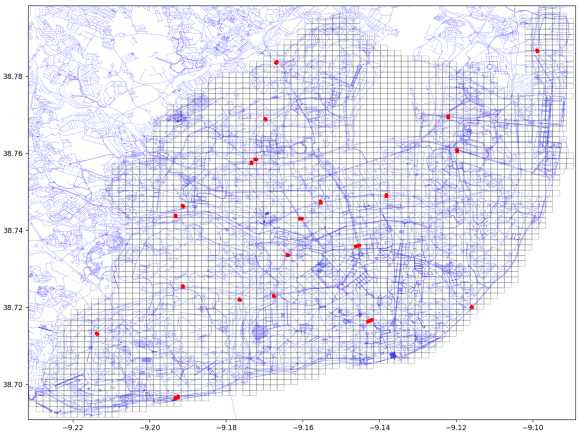
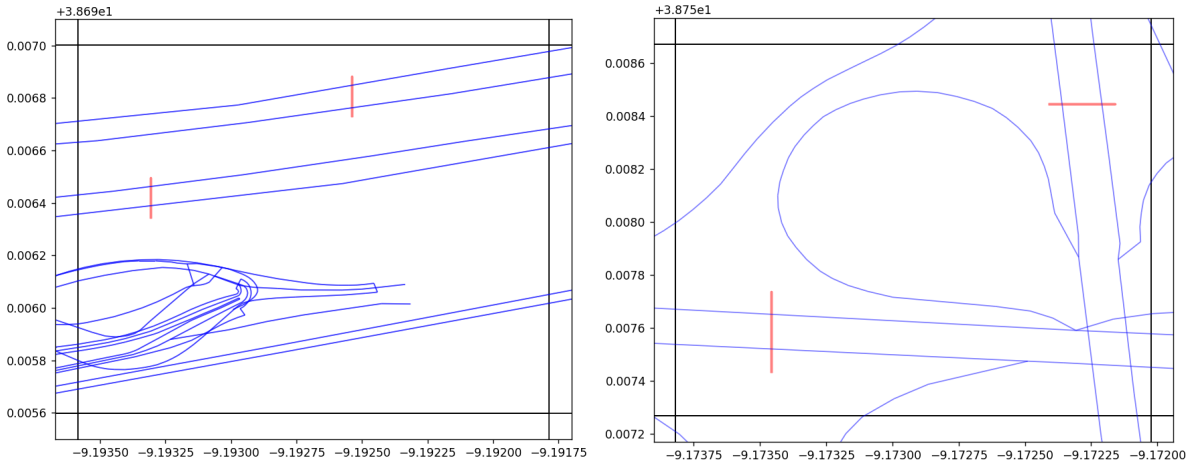



Figure 5.10: Radar Distribution Map Across Lisbon




(a) (b)

Figure 5.11: Radars (flow aggregators/detectors) in grid 70 (a) and grid 2546 (b).

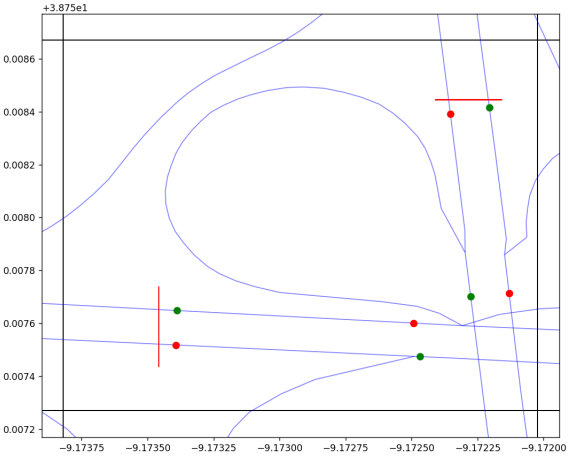



Figure 5.12: Points of origin and destination with the radars in grid 2546

### 5.7.2 Phone–to–Car Conversion

The optimization model yields flows in aggregated *mobile-phone* counts. To compare these volumes with conventional traffic-monitoring data (reported in *veh/h*) we convert, for every radar and for each travel direction, the number of detected phones into an equivalent number of cars. The conversion combines: (i) the raw phone counts provided by the flow aggregations of the previous subsection; (ii) the spatial layout of Lisbon's public-transport network—namely the *Carris* bus routes from the GTFS bundle introduced in Section 3.3.3, and the heavy-rail and Metro alignments extracted from OpenStreetMap (Fig. 3.9); and (iii) a set of global parameters that determine the final phone–to–car ratio.

A constant divisor would overlook the fact that many phones belong to bus, Metro or commuter-train passengers who do not contribute to road congestion. Instead, the procedure starts from a baseline ratio $base = 1.6$, loosely inspired by the "Average car and van occupancy in England" study [36], and *inflates* this divisor whenever public-transport corridors intersect the radar's influence zone. (Later research uncovered the Eurostat "Passenger mobility statistics" for Portugal [37]; these figures were found too late for the present implementation but will be valuable in future work.)

Because bus routes vary across seasons and weekdays, and rail services may be disrupted, the conversion factors are computed on-the-fly for the specific GTFS and rail datasets supplied at runtime. Moreover, to capture public-transport lines that run just outside the original radar rectangle but still parallel to the roadway, an enlarged rectangle—spanning the full Vodafone grid width—is analyzed.

For each public-transport mode, the divisor is adjusted as follows ($bus_{\text{incr}}$, $metro_{\text{incr}}$ and $train_{\text{incr}}$ are tunable parameters of the model):

- **Bus.** Every *Carris* route intersecting the rectangle adds $bus_{\text{incr}}$ (default $+0.1$) to the divisor of the corresponding direction; the increment is modest because buses are relatively small and headways are long.

- **Metro.** If any of the four Metro corridors (Blue, Green, Red, Yellow) cuts the rectangle, both directions receive $metro_{\text{incr}}$ (default $+1.2$), reflecting the large capacity and short headways of underground trains.

- **Heavy rail.** Each distinct main-line (*Linha de Cascais*, *Linha do Norte*, etc.) adds $train_{\text{incr}}$ (default $+1.0$) to both directions.

After all increments, the conversion factors for direction 1 and direction 2 become

$$\alpha_1 = \frac{1}{base + bus_{i1} + metro_{i1} + train_{i1}}, \qquad \alpha_2 = \frac{1}{base + bus_{i2} + metro_{i2} + train_{i2}},$$

where the subscripts indicate the total increments accumulated for each direction. The estimated car volumes per five-minute interval are then

$$V_{\text{cars},1} = \alpha_1 \, V_{\text{phones},1}, \qquad V_{\text{cars},2} = \alpha_2 \, V_{\text{phones},2}.$$

This adaptive procedure prevents corridors that are well served by public transport from being misclassified as road hot-spots, while purely motoring links retain the baseline $1{:}1.6$ ratio. Even so, the implementation is still rather coarse: it draws only on the geometry of the routes of each transport instead of querying the real-time GTFS or at least the static schedules to verify which buses and trains actually traverse each radar at the moment of observation. The car volumes obtained through this conversion provide the benchmark against which the validated traffic counts are compared in the next subsection.

### 5.7.3 Inference Evaluation

A straightforward percentage error is ill-suited to appraise road-traffic forecasts, because it penalizes lightly trafficked sites far more than busy corridors and offers no guidance on what constitutes an *acceptable* discrepancy. Traffic engineers therefore favor the dimensionless **GEH statistic**, defined for an individual location as

$$\mathrm{GEH} \;=\; \sqrt{\frac{2\,(M-C)^2}{M+C}},$$

where $M$ is the modeled flow (veh/h) and $C$ is the corresponding ground-truth count. The GEH blends absolute and relative error: it grows roughly linearly when the two volumes differ by only a few vehicles, yet converges towards a percentage-style error as the denominator increases. Because of this balanced behavior, and thanks to decades of empirical calibration, industry guidelines offer explicit thresholds (Table 5.3) against which to judge simulation quality.

| High | Medium | Low | Useless |
|---|---|---|---|
| 0–5 | 5–10 | 10–15 | >15 |

Table 5.3: Typical interpretation of GEH values for hourly volumes.

In the final evaluation, the modelled count $V_{\text{pred}}$ from the previous subsection is compared with the reference traffic dataset 3.4.1, which provides the ground-truth count $V_{\text{real}}$. Observed volumes are filtered to match the inference time, and total traffic is divided by the number of periods $n_{\text{per}}$ (twelve consecutive five-minute snapshots are aggregated to one hour; otherwise a single five-minute total is used):

$$\mathrm{GEH}_{\mathrm{model}} \;=\; \sqrt{\frac{2\,(V_{pred}-\frac{V_{\mathrm{real}}}{n_{\mathrm{per}}})^2}{V_{pred}+\frac{V_{\mathrm{real}}}{n_{\mathrm{per}}}}},$$

All GEH scores for every radar with ground-truth data are stored along with auxiliary information; the most relevant figures are discussed in Chapter 6.

# Chapter 6

# Testing and Results

In this chapter, we outline the experimental program used to evaluate and validate the Wasserstein-based traffic-inference pipeline developed so far, and we discuss the results it produced. We begin with the *Testing Methodology*, outlining the software framework and the main factors that either facilitated or hindered the testing process. Two complementary calibration phases are then described: an initial exploratory sweep guided by targeted diagnostics, followed by a systematic Bayesian optimization of the model parameters using routines from the *Optuna* library in Python [38].

The second part of the chapter, *Results and Discussion*, contrasts the preliminary and final model outputs, with a focus on interpreting their significance within the context of the project.

GEH statistics are reported for optimizations based on both five-minute and hourly aggregation intervals. The quality of the results is assessed against expectations, and each important traffic counter is individually analyzed. It's also computed and discussed the correlation between the inferred traffic volumes and the validated ground truth.

## 6.1 Testing Methodology

### 6.1.1 Testing Framework

**Obtaining Results.** During the initial tests, merely inspecting the solution matrix produced by the solver proved insufficient, so a plot was generated in which all aggregated road sections were coloured according to the corresponding mobile-phone volumes. To allow closer inspection of any chosen section, an interactive version of this plot was created: hovering the mouse over a section reveals its mobile-phone volume.

After examining several flow maps and comparing them with the raw results obtained directly from the solution variables, it became apparent that some sections were either missing from the map or hidden behind others. This is a consequence of the section-aggregation step, which does not perfectly merge every road segment, as discussed in Section 5.6 of the previous chapter.

To overcome this issue, each section in the plot is drawn with a small random offset—only a few

meters—so that the individual sections remain distinguishable within each grid cell; the zoom tool can be used whenever finer inspection is needed.

This global plot is invaluable for obtaining an overall picture of the inference, albeit at a significant computational cost: some 40 000 road sections must be rendered with distinct colors, placing heavy demands on both RAM and processing time. However, it is not the ideal tool for validation because (i) it depicts handset flows rather than the car flows obtained after phone-to-vehicle conversion, and (ii) the validation effort concentrates on a subset of main highways, some equipped with ground-truth counters, not on the entire network.

Consequently, the validation pipeline detailed in Section 5.7 serves as the primary instrument for analyzing the model's performance. It converts handset volumes into vehicle counts at the selected locations and then quantifies the prediction error against the observed ground truth.

**Testing Throughput.** At the outset, quickly loading data into memory proved to be a major challenge. Converting every large dataset—from the Vodafone records to the routing outputs—from CSV to `pickle` proved essential for both exploratory analysis and the final model run time. Additional speed was gained by precomputing the entire cost matrix $C$—combining car, rail, and airport movements as described in Section 5.2.1—and saving it as a `pickle`; this step cuts roughly 20 s from each run, a meaningful improvement given the large number of experiments conducted.

Early tests ran solely on a personal laptop, limiting throughput to about one optimization every ten minutes. RAM ($\approx$ 15 GB) was the main bottleneck: before the CPLEX model was streamlined to avoid materializing all $3761 \times 3761 > 14$M variables and to rely only on the *sifting* algorithm, memory pressure routinely triggered swap usage, risking disk wear. The code was progressively refactored so that memory-intensive objects are created only when indispensable and freed immediately afterward.

Late in the project, it was granted access to a cloud container at the Institute for Systems and Robotics (Lisbon) equipped with 24 CPU cores and 252 GB of RAM. This environment allowed twelve inference jobs to run in parallel, peaking at about 160 GB of RAM, without any CPU contention—a level of throughput the laptop could not approach.

### 6.1.2 Parameter Calibration

#### - Exploratory Ad-hoc Parameter Calibration

In the earliest stages, parameter tuning relied on visual inspection: the flow plot described above was inspected to understand how the model behaved globally and to stabilize the first working version.

As the project matured, attention shifted to the 22 fixed traffic counters. Each time a new configuration was introduced, its impact on the GEH statistic at these counters was examined, together with a sanity check on roads lacking ground-truth data (to at least ensure volumes were neither zero nor implausibly high). This iterative, trial-and-error process—guided chiefly by GEH but informed by broader qualitative cues—was the main calibration method for a considerable period.

For most of the calibration stage, only four hour periods were used as the reference set—3 Oct 2023

at 17:00, 12 Oct 2023 at 12:00, 20 Nov 2023 at 19:00, and 13 Dec 2023 at 09:00. Despite this limited sample, the optimization process did not lead to major overfitting.

Because the model is highly complex and non-convex, GEH minimization could not be achieved by tuning one or two parameters in isolation; numerous local minima appeared. Consequently, most or all parameters had to be adjusted in concert—an approach taken up formally in the next section.

**- Bayesian Algorithm Parameter Optimization**

Because the model involves more than a dozen inter-dependent parameters and exhibits a highly non-convex response surface, a global search strategy was required. We adopted *Bayesian optimization* as implemented in the *Optuna* [38] framework, which maintains a probabilistic model of the objective function and selects each new trial where improvement is most likely. By default, the *Tree-structured Parzen Estimator* (TPE) sampler was used; in preliminary testing it provided faster convergence than grid or random search, and very similar performance to Covariance-Matrix Adaptation (CMA-ES) while consuming fewer trials.

**Calibrating the phone–to–car converters.** The first Bayesian study focused on the three parameters that translate handset volumes into vehicle counts at the six radar locations equipped with ground-truth counters:

- `PHONE_CAR_RATIO` — the baseline ratio between detected phones and vehicles;
- `BUS_INCREMENT` — the incremental factor applied to radars where buses are present; and
- `METRO_INCREMENT` — the incremental factor applied due to metro presence.

A lightweight Python script evaluates each candidate triplet by (i) converting the predicted handset flows to vehicle flows at every radar and (ii) computing the aggregate GEH across the four validation hours—3 Oct 2023 17:00, 12 Oct 2023 12:00, 20 Nov 2023 19:00, and 13 Dec 2023 09:00. Because the script runs only the phone-to-car conversion, bypassing the full inference pipeline, each Optuna trial takes just a few hundred milliseconds and a few hundred kilobytes of RAM, enabling hundreds of trials to finish in minutes. After 500 trials, the best configuration reduced the total GEH by roughly $2\text{–}8\%$ relative to the manually tuned baseline.

**Joint optimization of all model parameters.** The second Bayesian study tackled a full set of 13 parameters that govern both the mass pre-processing and the Wasserstein cost function (e.g., building-height penalties, walking coefficients, and C5 constraints). Because each trial requires running the full inference pipeline, the objective is far more expensive: one execution takes $\approx$8 minutes on two CPU cores and consumes $\approx$13 GB of RAM, just to run a 5-minute period inference. To keep wall-clock time reasonably finite it was limited the search to 1 000 trials. The amount of trials done

All experiments were executed on the 24-core, 252-GB cloud container described earlier, achieving a sustained throughput of $\approx$12 inferences in parallel; even so, a 1,000-trial study still required two to

three days to finish. Optimization quality depended heavily on the objective design and search space: adding more five-minute windows to the objective diluted the achievable GEH reduction, and the width of the hyperparameter bounds made a clear trade-off between runtime and solution quality—bounds that were too narrow led to premature convergence, while bounds that were too wide slowed progress without proportional gains.

Nevertheless, Bayesian optimization proved instrumental, both for fine-tuning the phone-to-car converters and for locating optimized parameter sets that would have been impractical to discover manually.

## 6.2 Results and Discussion

### 6.2.1 Early Results

Nos primeiros modelos testados, com poucas configurações para modelar a optimização de wasserstein, os movimentos entre quadrículas eram raros e de baixo volume. Mesmo com a adição das configurações do perman

### 6.2.2 Final Model Results

| Dir1_Pred_tel | Dir2_Pred_tel | Dir1_Pred | Dir2_Pred |
|---:|---:|---:|---:|
| 221.99 | 174.63 | 65.29 | 51.36 |
| 441.06 | 404.84 | 210.03 | 192.78 |
| 238.57 | 236.60 | 68.16 | 69.59 |
| 185.33 | 129.80 | 109.02 | 76.36 |
| 203.98 | 188.41 | 119.99 | 110.83 |
| 306.07 | 247.41 | 161.09 | 130.21 |
| 824.04 | 979.44 | 457.80 | 612.15 |
| 493.14 | 552.20 | 290.08 | 324.83 |
| 66.38 | 143.39 | 31.61 | 68.28 |
| 219.90 | 121.84 | 81.44 | 45.12 |
| 306.46 | 336.46 | 145.93 | 160.22 |
| 702.53 | 516.82 | 390.30 | 287.12 |
| 95.17 | 215.42 | 33.99 | 76.94 |
| 69.48 | 29.44 | 23.16 | 9.81 |
| 271.84 | 263.61 | 151.02 | 146.45 |
| 52.12 | 0.00 | 19.30 | 0.00 |
| 65.45 | 39.74 | 40.91 | 23.38 |
| 805.35 | 390.44 | 503.34 | 244.02 |
| 937.97 | 758.27 | 586.23 | 473.92 |
| 647.26 | 696.77 | 380.74 | 409.86 |
| 142.77 | 77.75 | 89.23 | 48.60 |
| 145.98 | 184.53 | 91.23 | 115.33 |

Table 6.1: Valores preditos para Dir1 e Dir2 com e sem `_tel`

| LOCAL | Grid_ID | Axes | Dir1_Pred | Dir2_Pred | Dir1_Real | Dir2_Real | Dir1_GEH | Dir2_GEH |
|---|---|---|---|---|---|---|---|---|
| Restauradores - Avenida da Liberdade | 742 | N-S, S-N | 65.29 | 51.36 | 49.17 | 44.83 | 2.13 | 0.94 |
| Avenida Engenheiro Duarte Pacheco | 1014 | W-E, E-W | 210.03 | 192.78 | 184.17 | 229.58 | 1.84 | 2.53 |
| Avenida da Républica | 1553 | N-S, S-N | 68.16 | 69.59 | 97.08 | 104.92 | 3.18 | 3.78 |
| Avenida de Berlim | 3058 | E-W, W-E | 109.02 | 76.36 | 87.33 | 65.92 | 2.19 | 1.24 |
| Calçada de Carriche | 3574 | W-E, E-W | 119.99 | 110.83 | 128.67 | 166.17 | 0.78 | 4.70 |
| Avenida das Forças Armadas | 2036 | W-E, E-W | 161.09 | 130.21 | – | – | – | – |
| 2ª Circular (E.S. Comunic.) | 2016 | W-E, E-W | 457.80 | 612.15 | – | – | – | – |
| 2ª Circular (Eixo N-S) | 2546 | W-E, E-W | 290.08 | 324.83 | – | – | – | – |
| A5 | 1114 | W-E, E-W | 31.61 | 68.28 | – | – | – | – |
| Av. Brasília | 70 | W-E, E-W | 81.44 | 45.12 | – | – | – | – |
| Av. Calouste Gulbenkian | 1484 | N-S, S-N | 145.93 | 160.22 | – | – | – | – |
| Av. EUA | 2175 | W-E, E-W | 390.30 | 287.12 | – | – | – | – |
| Av. Gen. Correia Barreto | 1888 | W-E, E-W | 33.99 | 76.94 | – | – | – | – |
| Av. Infante D. Henrique | 930 | N-S, S-N | 23.16 | 9.81 | – | – | – | – |
| Av. Mar. Gomes da Costa | 2700 | W-E, E-W | 151.02 | 146.45 | – | – | – | – |
| Av. da Índia | 70 | W-E, E-W | 19.30 | 0.00 | – | – | – | – |
| Av. das Descobertas | 586 | W-E, E-W | 40.91 | 23.38 | – | – | – | – |
| Av. dos Combatentes | 1844 | N-S, S-N | 503.34 | 244.02 | – | – | – | – |
| Eixo N-S (2ª Circular) | 2546 | N-S, S-N | 586.23 | 473.92 | – | – | – | – |
| Eixo N-S (Telheiras) | 3032 | W-E, E-W | 380.74 | 409.86 | – | – | – | – |
| Eixo N-S/A2 | 953 | N-S, S-N | 89.23 | 48.60 | – | – | – | – |
| Pt. Vasco da Gama | 3625 | W-E, E-W | 91.23 | 115.33 | – | – | – | – |

Table 6.2: Comparação entre valores previstos e reais com índice GEH por direção

| LOCAL | Grid_ID | Axes | Dir1_Car | Dir1_Real | Dir1_GEH | Dir2_Car | Dir2_Real | Dir2_GEH |
|---|---|---|---|---|---|---|---|---|
| Restauradores - Avenida da Liberdade | 742 | N-S, S-N | 450.50 | 590.0 | 6.12 | 990.87 | 538.0 | 16.38 |
| Avenida Engenheiro Duarte Pacheco | 1014 | W-E, E-W | 2641.08 | 2210.0 | 8.75 | 1983.26 | 2755.0 | 15.86 |
| Avenida da Républica | 1553 | N-S, S-N | 845.32 | 1165.0 | 10.08 | 1448.96 | 1259.0 | 5.16 |
| Avenida de Berlim | 3058 | E-W, W-E | 1365.33 | 1048.0 | 9.14 | 947.76 | 791.0 | 5.32 |
| Calçada de Carriche | 3574 | W-E, E-W | 1504.39 | 1544.0 | 1.01 | 1369.40 | 1994.0 | 15.23 |
| Avenida das Forças Armadas | 2036 | W-E, E-W | 1970.75 | – | – | 1904.80 | – | – |
| 2ª Circular (E.S. Comunic.) | 2016 | W-E, E-W | 5538.89 | – | – | 6749.29 | – | – |
| 2ª Circular (Eixo N-S) | 2546 | W-E, E-W | 2875.36 | – | – | 4381.45 | – | – |
| A5 | 1114 | W-E, E-W | 384.98 | – | – | 560.98 | – | – |
| Av. Brasília | 70 | W-E, E-W | 832.97 | – | – | 746.85 | – | – |
| Av. Calouste Gulbenkian | 1484 | N-S, S-N | 1708.52 | – | – | 2192.89 | – | – |
| Av. EUA | 2175 | W-E, E-W | 5165.05 | – | – | 4021.65 | – | – |
| Av. Gen. Correia Barreto | 1888 | W-E, E-W | 369.74 | – | – | 1252.27 | – | – |
| Av. Infante D. Henrique | 930 | N-S, S-N | 141.47 | – | – | 217.10 | – | – |
| Av. Mar. Gomes da Costa | 2700 | W-E, E-W | 1659.31 | – | – | 1497.97 | – | – |
| Av. da Índia | 70 | W-E, E-W | 298.51 | – | – | 64.96 | – | – |
| Av. das Descobertas | 586 | W-E, E-W | 681.79 | – | – | 244.34 | – | – |
| Av. dos Combatentes | 1844 | N-S, S-N | 5973.19 | – | – | 3351.15 | – | – |
| Eixo N-S (2ª Circular) | 2546 | N-S, S-N | 5843.67 | – | – | 6078.81 | – | – |
| Eixo N-S (Telheiras) | 3032 | W-E, E-W | 6114.75 | – | – | 4398.63 | – | – |
| Eixo N-S/A2 | 953 | N-S, S-N | 1075.01 | – | – | 700.01 | – | – |
| Pt. Vasco da Gama | 3625 | W-E, E-W | 1055.83 | – | – | 1102.38 | – | – |

Table 6.3: Comparação entre volumes simulados e reais com índice GEH

| Hora | GEH médio |
|---|---|
| 14 | 11,276 |
| 15 | 12,5422 |
| 13 | 12,8002 |
| 10 | 13,4632 |
| 11 | 13,471 |
| 12 | 13,5404 |
| 17 | 14,9832 |
| 16 | 15,1638 |
| 19 | 15,2158 |
| 9 | 15,6452 |
| 18 | 16,0828 |
| 8 | 18,294 |
| 21 | 18,5014 |
| 20 | 18,5966 |

Table 6.4: GEH médio por hora do dia

| Alta | Média | Baixa | Inútil |
|---|---|---|---|
| 0 – 5 | 5 – 10 | 10 – 15 | >15 |

Table 6.5: Classificação da qualidade do modelo segundo o índice GEH

# Chapter 7

# Conclusions and Future Work

# Bibliography


[1] Transportation Research Board. *Highway Capacity Manual 2000*. Transportation Research Board, National Research Council, Washington, D.C., 2000. HCM 2000.

[2] A. D. May. *Traffic Flow Fundamentals*. Prentice Hall, Englewood Cliffs, NJ, 1990. ISBN 978-0139260728.

[3] C. Balzotti, A. Bragagnini, M. Briani, and E. Cristiani. Understanding human mobility flows from aggregated mobile phone data. *IFAC PapersOnLine*, 51(9):25–30, 2018. doi: 10.1016/j.ifacol.2018.07.005. URL https://www.sciencedirect.com/science/article/pii/S2405896318307213.

[4] C. Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008. URL https://cedricvillani.org/sites/dev/files/old_images/2012/08/preprint-1.pdf.

[5] J. P. T. Vaz. Indicador de tráfego: descoberta de padrões na cidade de lisboa. Dissertação de mestrado, Instituto Superior de Engenharia de Lisboa, 2022. URL https://repositorio.ipl.pt/handle/10400.21/16541.

[6] A. Cecaj, M. Lippi, M. Mamei, and F. Zambonelli. Comparing deep learning and statistical methods in forecasting crowd distribution from aggregated mobile phone data. *Applied Sciences*, 10(18):6580, 2020. doi: 10.3390/app10186580. URL https://www.mdpi.com/2076-3417/10/18/6580.

[7] G. Sagl, E. Delmelle, and E. Delmelle. Mapping collective human activity in an urban environment based on mobile phone data. *Cartography and Geographic Information Science*, 41(3):272–285, 2014. doi: 10.1080/15230406.2014.888958. URL https://www.researchgate.net/publication/271764593_Mapping_collective_human_activity_in_an_urban_environment_based_on_mobile_phone_data.

[8] M. Ni, Q. He, and J. Gao. Using social media to predict traffic flow under special event conditions. In *The 93rd Annual Meeting of Transportation Research Board*, 2014. URL https://core.ac.uk/reader/323409409.

[9] D. Zhang and M. R. Kabuka. Combining weather condition data to predict traffic flow: a gru-based deep learning approach. *IET Intelligent Transport Systems*, 12(7):578–585, 2018. doi: 10.1049/iet-its.2017.0313. URL https://digital-library.theiet.org/content/journals/10.1049/iet-its.2017.0313.

[10] Fu et al. Using lstm and gru neural network methods for traffic flow prediction. *Transportation Research Part C: Emerging Technologies*, 67:610–626, 2016. doi: 10.1016/j.trc.2016.02.016.

[11] Villarroya et al. Neural network-based model for traffic prediction in the city of valencia. *Journal of Transportation Engineering, Part A: Systems*, 148(7):04022033, 2022. doi: 10.1061/JTEPBS.0000578.

[12] Tian et al. Lstm-based traffic flow prediction with missing data. *Neurocomputing*, 277:230–242, 2018. doi: 10.1016/j.neucom.2017.01.118.

[13] Min, Wei and Wynter, Laura. Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies*, 19(4):606–616, 2011. doi: 10.1016/j.trc.2010.10.002.

[14] Kumar S. Vasantha and Vanajakshi Lelitha. Short-term traffic flow prediction using seasonal arima model with limited input data. *European Transport Research Review*, 7(3):1–11, 2015. doi: 10.1007/s12544-015-0170-8.

[15] Elragal, Ahmed and Raslan, Hisham. Analysis of trajectory data in support of traffic management: A data mining approach. In *Advances in data mining: applications and theoretical aspects: 14th Industrial Conference, ICDM 2014, St. Petersburg, Russia, July 16-20, 2014. Proceedings*, pages 174–188. Springer, 2014.

[16] Batran M. et al. Urban travel time estimation in greater maputo using mobile phone big data. *Journal of Big Data*, 5(1):1–16, 2018.

[17] Caceres N. et al. Traffic flow estimation models using cellular phone data. *Transportation Research Part C: Emerging Technologies*, 30:161–183, 2012.

[18] J. P. Silva, J. M. Viegas, and E. Macedo. Monitoring multimodal traffic patterns: A spatiotemporal analysis of multimodality indices in lisbon. *European Transport Research Review*, 13(23), 2021. doi: 10.1186/s12544-021-00520-3. URL `https://etrr.springeropen.com/articles/10.1186/s12544-021-00520-3`.

[19] L. Gonçalves and P. Ribeiro. Modelação de tráfego a partir de contagens para avaliar o nível de eficiência da mobilidade. In *II Encontro Nacional sobre Reabilitação Urbana e Construção Sustentável: do edifício para a escala urbana*, pages 289–298, Guimarães, Portugal, 2018. Estudo de caso com PTV Visum e ajuste OD via TFlowFuzzy.

[20] Siemens Mobility. Siemens portugal revoluciona análise de dados de tráfego rodoviário, 2024. URL `https://press.siemens.com/pt/pt/comunicadodeimprensa/siemens-portugal-revoluciona-analise-de-dados-de-trafego-rodoviario`.

[21] Vivacity Labs. Smart traffic monitoring, 2023. URL `https://vivacitylabs.com/north-america/`.

[22] AllRead AI. Real-time monitoring of vehicle and container traffic – port of algeciras, 2022. URL https://www.allread.ai/en/success-cases/port-authority-of-algeciras/.

[23] TomTom Editorial Team. Inside tomtom's traffic products: Origin–destination analysis explained, 2024. URL https://www.tomtom.com/newsroom/product-focus/tomtom-origin-destination-analysis-explained/.

[24] INRIX. Why connected vehicle data is the next big alt dataset, 2024. URL https://inrix.com/resources/on-demand-webinar-turning-connected-vehicle-data-into-alpha/.

[25] University of Žilina. Analysis of the possibility to detect road vehicles via bluetooth technology, 2021. URL https://www.researchgate.net/publication/355850228.

[26] Siemens Mobility. Heimdall vehicle presence radar detector — datasheet, 2019. URL https://assets.new.siemens.com/siemens/assets/api/uuid:3b75b874-5449-4079-a07c-686d0c5b09a6/version:1567517556/heimdall-all.pdf.

[27] MetroCount. Roadpod vt — vehicle tube counter, 2020. URL https://www.metrocount.com/traffic-counters-classifiers/roadpod-vt.

[28] Esri. Arcgis online, 2024. URL https://www.arcgis.com/index.html. Accessed 2024-06-09.

[29] OpenStreetMap contributors. Openstreetmap, 2024. URL https://www.openstreetmap.org/#map=7/39.602/-7.839. Accessed 2024-06-09.

[30] OSM Buildings. Osm buildings data, 2024. URL https://osmbuildings.org/data/.

[31] European Commission, Joint Research Centre. GHS-BUILT-H: Global Human Settlement Layer - Built-up Height, 2023. URL https://data.europa.eu/89h/85005901-3a49-48dd-9d19-6261354f56fe.

[32] Transitland. Transitland: An open data platform for transit networks, 2025. URL https://www.transit.land/. Accessed: 2025-05-17.

[33] C. M. de Lisboa. Dashboards cgiul, n.d. URL https://lisboaaberta.cm-lisboa.pt/index.php/pt/analitica/dashboards-cgiul.

[34] GraphHopper GmbH. Graphhopper routing engine, 2024. URL https://www.graphhopper.com.

[35] IBM. *IBM ILOG CPLEX Optimization Studio*, 2023. URL https://www.ibm.com/products/ilog-cplex-optimization-studio. Version 22.1.

[36] Statista. Average car and van occupancy in england (2002–2022). https://www.statista.com/statistics/314719/average-car-and-van-occupancy-in-england/, 2025. Accessed 25 June 2025.

[37] Eurostat. Passenger mobility statistics, 2024. URL https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Passenger_mobility_statistics. Average car occupancy in the EU; accessed 25 June 2025.

[38] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019. URL https://optuna.org/.