Machine Learning Methods for the Optimisation of Urban Mobility

Rui Pedro Campinhos Loureiro

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Fernando Henrique Côrte-Real Mira da Silva

January 2021

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.

Acknowledgments

First and foremost, I would like to thank my supervisor Fernando Mira Da Silva, for the continuous support, encouragement and pragmatic approach that allowed me to finish this thesis in a timely manner.

Secondly, I must thank my family, in particular my parents and brother, for their encouragement and support.

I would also like to thank Carolina Santos, without whom this work would have been a lot more difficult. Thank you for your continued support and encouragement, and always being available to hear me rambling about the work, and providing valuable feedback.

This work was part of the research project ILU (integrative Learning from Urban data), which unites INESC-ID, LNEC (Laboratório Nacional de Engenharia Civil) and CML (Câmara Múnicipal de Lisboa). I'd like to thank everyone involved for their work towards a more efficient and sustainable urban mobility.

This work would not have been possible without the support of Carris (Companhia Carris de Ferro de Lisboa), especially the planning department, who shared expert insights into several planning strategies and challenges faced, providing stimulating and fruitful discussions that helped shape the direction of this work. A special thank you to Dra.Margarida Nunes, whose continued support regarding the several data sources was indispensable.

Lastly, I would like to thank my colleagues at Jungle and Loka, who made the work-thesis balance a lot easier to manage.

Abstract

For a public transport systems to meet the increasing transport demand in major cities, it must be both reliable and efficient. Procedures for efficient route set and schedule design are crucial in order to develop efficient transit systems that quickly adapt to the dynamic demand of the passengers. Public transport agencies have traditionally been constrained in planning, managing and evaluating their services by having to rely on costly and unreliable manual surveys, heuristic planning and ad-hoc adjustments. Automatic Data Collection (ADC) systems such as Automatic Fare Collection (AFC) and Automatic Vehicle Location (AVL) systems allow transit providers to collect high volumes of data on their operations and passenger behaviour, enabling better, more informed decisions. While these huge data sets have the potential to be synthesized into meaningful information, helping evaluate the system performance and plan service changes in an objective and precise manner, they require extensive data analysis and processing, often limiting their use by operators. This thesis uses AFC data to infer passengers' origin and destination locations in a multi-modal public transport network. With the resulting demand data, we approach the Transit Network Design (TND) problem, using a Genetic Algorithm (GA)-based approach to optimize a bus network, using real road data. We apply our approach to the bus network in Lisbon, Portugal. The resulting optimized network achieves a 4 minute decrease in the average travel time of all passengers and an increase in satisfied demand.

Keywords: Origin-Destination-Interchange Inference, Genetic algorithms, Public Transport Optimization, Transit Network Design, Urban mobility

Resumo

Para que um sistema de transporte público seja capaz de responder à procura crescente nas grandes cidades, deve ser confiável e eficiente. Procedimentos para a geração de rotas e horários eficientes e adaptados às necessidades dos passageiros são cruciais no desenvolvimento da mobilidade urbana. Até recentemente, os operadores de transportes públicos tinham limitações fortes no planeamento, optimização e avaliação dos seus serviços. Em muitos casos, todas as métricas disponíveis dependiam de inquéritos manuais sobre origem, destinos e avaliação subjetiva dos utilizadores que se revelam caros, pouco precisos e pouco representativos. Adicionalmente, as técnicas de planeamento eram em grande parte baseadas em heurísticas e ajustes improvisados, sendo a avaliação do resultado também pouco objectiva. Com a generalização dos sistemas de coleção automática de dados, como sistemas de bilhética e tarifação electrónica, e sistemas de localização de veículos, os operadores de transportes públicos passaram a colecionar um grande volume de dados que descrevem com muito detalhe o serviço e a forma como este é usado pelos passageiros. Embora estes dados permitam uma avaliação de desempenho e planeamento dos serviços de forma mais objetiva e detalhada, requerem a aplicação de técnicas de análise e processamento de dados, sendo frequentemente usados de forma limitada pelos operadores. Esta tese usa os dados de tarifação para inferir, com grande detalhe, origens e destinos de passageiros numa rede de transportes públicos multi-modal. Com a informação de procura resultante, é utilizado um algoritmo genético com o objetivo de otimizar uma rede de autocarros real, usando dados relativos à rede rodoviária. O método desenvolvido é aplicado na rede de autocarros de Lisboa. A rede resultante, otimizada, demonstra uma redução de 4 minutos no tempo médio de viagem de todos os passageiros, bem como um aumento de procura satisfeita.

Palavras-Chave: Matriz Origem-Destino, Algoritmos Genéticos, Otimização de transportes públicos, Otimização de redes de transportes, Mobilidade Urbana

Contents

List of Tables

List of Figures

Glossary

ACO	Ant Colony Optimization.
ADC	Automatic Data Collection.
ADCS	Automatic Data Collection System.
AFC	Automatic Fare Collection.
APC	Automatic Passenger Count.
AVL	Automatic Vehicle Location.
BSD	Bus Static Dataset.
CP	Comboios de Portugal.
GA	Genetic Algorithm.
GTFS	General Transit Feed Specification.
MBTA	Massachusetts Bay Transportation Authority.
OD	Origin Destination.
ODX	Origin Destination Interchange.
OSM	Open Street Map.
OSRM	Open Source Routing Machine.
RFID	Radio Frequency Identification.
TND	Transit Network Design.

Chapter 1

Introduction

With the increase in population and congestion in major cities, reliable and efficient public transport systems are key to meet the urban mobility requirements.

Public transport agencies have traditionally been hampered in planning, managing and evaluating their services by having to rely on costly and unreliable manual surveys, heuristic planning and ad-hoc adjustments [4, 5].

However, over the past several decades, a number of new technologies have been developed aimed at improving the information available to the public transport sector [4], allowing transit providers to collect high volumes of data on their operations and passenger behaviour. These ADC systems include AFC systems, AVL systems and Automatic Passenger Count (APC) systems.

While manual surveys focus on key locations and typical hours, ADC systems provide extensive spatial and temporal coverage, enabling better, more informed decisions.

While ADC systems are often designed to serve very narrow and specific functions, primarily used in real-time applications [6], the enormous amounts of resulting data generated can be used in a wide range of applications. The gap between what ADC systems directly offer and what is needed in practice by transit agencies requires extensive data analysis before the full value of the data source can be fully exploited [4].

Nevertheless, these huge data sets have the potential to be synthesized into meaningful information, not only to evaluate the performance of current transport systems, but also to aid in the prediction of future conditions and provide key inputs for the generation of solutions to planning problems.

For a public transport systems to meet the increasing transport demand in major cities, it must be both reliable and efficient [7, 8]. Procedures for efficient route set and schedule design are crucial in order to develop efficient transit systems that quickly adapt to the dynamic demand of the passengers. This is known as the TND Problem, which is part of the strategical planning, which focuses on long term decisions in the public transport planning. The TND is an NP-hard problem, making the design of computationally efficient methods a promising research area.

This thesis uses Automatic Data Collection System (ADCS) data to infer passengers' origin and destination locations, and to link trip segments into full journeys, by identifying transfers. The algorithm used in [9], called Origin Destination Interchange (ODX), is applied to the data of a multi-modal public transport network in Lisbon, Portugal to estimate an origin-destination matrix that describes network demand. With the resulting demand matrix, and real road data, we approach the Transit Network Design (TND) problem, using a GA-based approach to optimize the entire bus network in Lisbon.

1.1 Objectives

This thesis approaches the TND problem, by developing an efficient methodology capable of optimizing a real, complex bus network, using precise and representative data.

The developed methodology is applied to the bus network of Lisbon, Portugal.

As a key input to the optimization process, this thesis aims at estimating a multi-modal demand matrix, using AFC and auxiliary static data from the two most used public transport operators in Lisbon, Portugal.

The computation of the demand matrix is an important contribution that has the potential to be used in a variety of applications outside the scope of the TND problem.

More concretely, this thesis seeks to fulfill the following objectives:

1.1.1 Infer alighting locations and times for bus stages in Lisbon

In a bus system where passengers' exit information is not collected, alighting locations and times must be inferred. AFC data is necessary for the Origin Destination (OD) inference process. However, the AFC system in Lisbon was not designed for this purpose, so an initial data analysis and processing step is necessary. Static data is also necessary for the OD process, so the compatibility between the AFC data and additional static data sets must be assured.

1.1.2 Infer Interchanges between journey stages

From the inferred destinations, link together bus and metro stages into journeys, adapting heuristic parameters for the specific context of Lisbon. The absence of AVL data also poses an additional challenge for the interchange inference process.

1.1.3 Develop precise representation of the road network, including bus stops

A graph representation of the bus stops and the underlying road network, that connects these stops should be constructed from public road data. This is useful to assess the performance of the existing network and the impact of future network changes. In our case, it's used as an input to the route optimization algorithm.

1.1.4 Apply route optimization algorithm to bus network

Adapt and implement previously developed GA-based optimisation methods for the TND problem and apply it to the entire bus network in Lisbon. Given the size and complexity of the network, the efficiency of the developed methods is one of the key objectives.

1.2 Structure of this document

The methods developed in this thesis can be grouped into two different categories: ODX inference methods, and optimization methods used in the TND problem.

Chapter 2 provides the necessary background for the methods developed in this thesis. The most relevant literature regarding ODX inference and TND applied to public transport is reviewed, along with the necessary data sources and tools. Chapter 3 describes the methodology used for ODX inference, and the results of the concrete application to the city of Lisbon. Chapter 4 describes the problem formulation and optimization methodology of our approach to the TND applied to Lisbon. Chapter 5 describes the results of the TND optimization. Chapter 6 reflects on the study's findings, presents its conclusions and recommendations for future research.

Chapter 2

Background

This chapter starts by reviewing the data formats and tools used throughout our work, followed by a review of the most relevant research related to ODX inference methods and TND methods applied to public transport systems, with a focus on GA-based approaches. Finally a quick introduction to GA theory is given.

2.1 Data Formats and Tools

Several data sources, specifications and tools were used in our work. These can be divided in two categories:

- Public transport data, specifically ADC systems data and General Transit Feed Specification (GTFS) data
- Road network data, specifically OSM data and the OSRM tool.

2.1.1 Automatic Data Collection Systems

Automatic Data Collection (ADC) systems allow public transport agencies to collect high volumes of data on their operations and passenger behaviour, allowing them to monitor their service quality and make better short and long term planning decisions.

ADC Systems are classified as **AVL** (Automatic Vehicle Location) and tracking systems, **AFC** (Automatic Fare Collection) systems and **APC** (Automatic Passenger Count) systems [10, 11].

Zhao et al. [4] summarises the general characteristics of automatically data collection systems, when compared with traditional manual data collection systems:

- **Spatial and temporal coverage**: While manual systems focus on key locations and typical hours, ADC systems provide extensive spatial and temporal coverage.
- **Data processing and accuracy**: With a data processing and analysis framework, ADC data can be processed much faster and is not influenced by hard-to-detect bias and errors typically associated with manual surveys.
- **Cost structure**: ADC systems typically involve a high initial investment but have a low incremental cost when operational and deployed.
- **Inherent disadvantages**: Since ADC systems were not initially designed for data collection and are only used for a limited purpose such as revenue accounting, many limitations arise when they

are looked at as data collection devices, which have limited the use of ADC data: - The data is often fragmented, intermittent and lacks critical information, which may require extensive processing - Is not in an easy-to-use format and lacks of consistency across vendors

Several errors can arise from data collection, which may be caused by software, erroneous data, faulty hardware, etc. In this regard, data validation and preprocessing methodologies are essential to assess the data quality and retrieve valuable information from AFC data [12].

2.1.2 General Transit Specification

The GTFS is a common format for public transport agencies to publish their transit information.

The GTFS specification is split into two components:

- A **static** component, that includes public transport schedules and associated geographic information
- A **real-time** component, that allows transit providers to publish real-time information such as arrival predictions, vehicle localization and service advisories.

Today, the GTFS data format is used by thousands of public transport providers. A GTFS feed is comprised of 6 required and 7 optional csv files. Figure 2.1 shows the GTFS schema, with the optional files in dotted boxes.

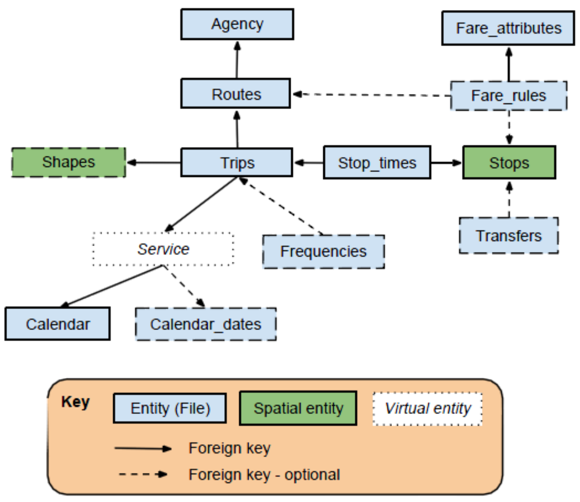


Figure 2.1: GTFS Schema [1]

2.1.3 OpenStreetMap

Geographical data is not free in many parts of the world. Quite often, mapping is developed and maintained by government agencies who in return get to make money by selling the data [13]. On other cases, mapping is performed by private companies with commercial purposes. Google spends 1 billion dollars annually maintaining their maps [14]. Because geographical data is copyrighted and owned by multiple organisations, it is often difficult and expensive to use. The google maps terms of service [15], particularly the 'Map Information' section states that: - "Geocoding data for map content in Google Local is provided under license by Navteq... and/or Tele Atlas... and subject to copyright protection and other intellectual property rights owned by or licensed to NAVTEQ, TANA and/or such other third parties." -

”Also, you may not use Google Maps in a manner which gives you or any other person access to mass downloads or bulk feeds of numerical latitude and longitude coordinates.”

OpenStreetMap is a free, editable map of the whole world that is being built by volunteers largely from scratch and released with an open-content license. More and more major organisations are choosing OSM for their maps, including Wikipedia, Apple, Craigslist and Foursquare [16]. The OSM map data is also used and supported by a range of mature open source projects, that are evolving much faster than any closed source project [16].

Being a community based initiative, there is no central organization that controls the data acquisition and guarantees the quality of the data. There have been several research efforts to determine the quality analysis of OSM data. To describe the usability of the data, assessing its quality is essential [17]. For example, [18] compares OSM data with ordinance survey maps, in London. They conclude that the OSM data is fairly accurate, with an average of 6 meters within the position in the ordinance survey, and an 80% of overlap of motorway objects. [17] assesses the quality of the OSM data in germany, particularly for pedestrian navigation. They compare the shortest path between two arbitrary conditions computed using OSM and using another topographic data set, with the assumption that the shorter the paths are, the better the quality of the underlying network is.

2.1.4 Open Source Routing Machine

OSRM is a routing engine used to compute shortest paths on road, bicycle and pedestrian networks. It’s similar to the Google Maps routing service, but it’s designed to be used with OSM data.

OSRM can be used with a custom profile, taking into account the vehicle characteristics (dimensions and weight, for example) and specified penalties for u-turns or paths that include traffic lights, for example.

Furthermore, unlike Google Maps, OSRM can be ran on a local instance, without query limits.

2.2 Origin, Destination and Interchange Inference

AFC (Automatic Fare Collection) systems have the potential to reveal ridership patterns for the entire service network. However, AFC data is limited in its ability to infer the passenger’s full sequence of trips. This limitations is a consequence of the type of fare collection system, which can closed or open. Closed systems require the passenger to tap its card when entering and when exiting the system, but may not require it for internal transfers. In contrast, open systems only require passengers to tap their card when entering the system, but not when exiting. In the concrete case of Lisbon, the bus AFC system is an open system, while the metro AFC system is closed.

The goal of OD (Origin Destination) inference is to process ADCS data and infer, for every stage, an origin (location and time) and destination (location and time) [19]. A stage is defined as a portion of a passenger’s travel that is associated with a single fare transaction, whether the fare transaction consists of a single boarding tap, as the case with bus travel in Lisbon, or an entry and exit tap, as is the case with metro travel in Lisbon [9]. For networks where the passenger doesn’t have to tap the card for internal transfers, multiple trip segments, each on a line, can make up a stage [19].

By applying several heuristics related to the time the passenger spends between stages, as well as geographical characteristics of these stages, one can infer whether each of the stages of a particular passenger is linked to the next through a transfer. Doing so enables the analysis of full passenger journeys. Inference of origins and destinations has been a subject of research since AFC and AVL data have seen widespread availability [20].

Early implementations had several limitations, such as covering only a single mode, or inferring locations, but not times [4, 21]. Others rely on scheduled trip times to determine arrival times [22]. More recent developments have overcome some of these earlier limitations, by taking advantage of AVL systems. [23] developed methods to infer origins and destinations for the bus network in London. [9] extended these methods to include the rail system, which includes both entry and exit transactions, and infer inter-modal transfers.

In a similar research direction, known origin and destination patterns can be used for modeling passenger behavior and demand for public transportation, allowing the adjustment of optimization models and quantifying the importance of factors like waiting times and transfers [24, 25].

2.3 Transit Network Design

The problem of planning efficient public transport systems is, due to its complexity, commonly divided into a series of sub problems solved at different stages, spanning strategical, tactical and operational decisions [7, 11].

Strategic planning deals with long-term decisions, such defining as the routes layout, the stop spacing and passenger assignment, in order to optimize specific objective functions, such as operator cost or user satisfaction. With access to AFC data, public transit operators have a much better understanding of travel behaviour mechanisms, when compared to manual surveys [26], allowing for more informed and passenger centered long term network changes.

Tactical planning defines the frequency and timetable of routes, as well as the fleet size requirements of such schedule. By exploiting AFC, AVL and APC data over time, operators can extract frequent mobility patterns and assess the reliability of their service [27].

Operational planning deals with short-term planning, such as driver scheduling, driver rostering, and real-time control strategies, aimed at dealing with the uncertainty of travel times, demand patterns and disturbances such as vehicle breakdowns, road blocks and bus bunching [27].

Figure 2.2 summarizes the several decision levels in the public transport planning.

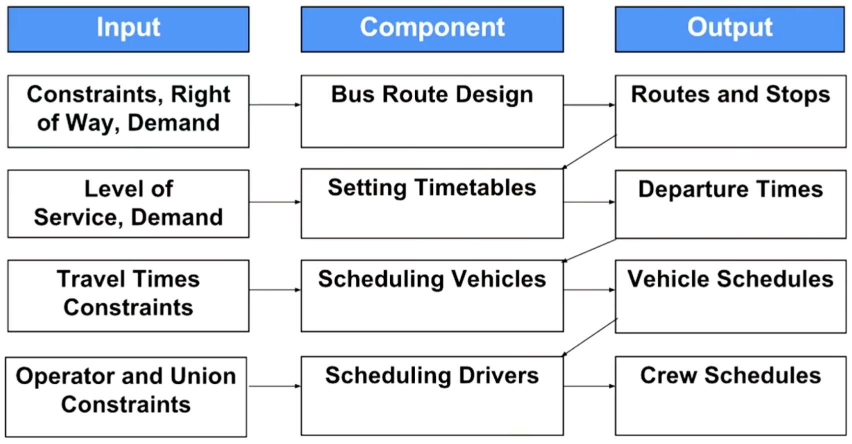


Figure 2.2: Public transport planning stages [2]

TND is part of the strategical planning, and concerns long term decisions in the public transport planning. Designing an operationally and economically efficient bus transit network is vital to satisfy the increasing transport demand in major cities. Up until recently, route designers have relied mainly on historical experience, simple guidelines, local knowledge and ad hoc procedures [28]. However, the widespread use of ADC systems allow for a data-driven and passenger centered transit network design, allowing transport agencies to have a much better understanding of transit demand and travel behavior mechanisms of its passengers.

From the passenger's perspective, a transit network should be highly accessible, covering a large service area, offer a high percentage of direct trips (without the need to transfer), low waiting times, not deviate too much from the shortest paths and globally be able to meet the demand [7, 11]. From the operator's perspective, the transit network should stay within budget, efficiently obtain revenue and be able to satisfy the user's demand. [7]. Hence, one of the main challenges in the transit network design is the vast search space. There are several conflicting factors, mainly due to the fact that both the operator and the passenger's interests must be taken into account [28].

In real life, passengers have different travel behaviors. Each passenger may choose different travel paths in the transit network, to travel from a specific origin to a specific destination, in order to optimize their own criteria. These passengers choices are embedded in an optimization problem called **transit assignment problem** [29]. The route choices taken by passengers depends on the individual weight given to each different factor, such as access, waiting time, travel time and comfort. These weights may not coincide with those used by the network designer. Furthermore, the individual passengers differ between passengers. [7]

The transit network design is an NP-hard problem, with a difficult to calculate objective function, which poses overwhelming difficulties in obtaining a solution through traditional optimization techniques [30, 31]. The Transit Network Design problem can be modeled as a discrete optimization problem, where passenger demand is represented through an origin-destination matrix, or a continuous optimization problem, where passenger demand is represented as a continuous function over a geographical space instead of an origin-destination matrix. The continuous approximation model is based on an idealized representation of the city, with a specific grid structure (rectangular, circular, hub-and-spokes, etc) [7].

[7] divides the discrete approximation model approaches into several categories, that we will now summarize:

- **Sequential approaches and simplified cases**

  This approach decomposes the problem into route generation and transit assignment stages. These include mathematical formulation that, while intractable by exact approaches, are possible to solve by simplifying the problem, such as small instances [32], considering only one line, [33] reducing the size of the network [34], etc.

- **Metaheuristic Algorithms**

  Although the transit network design is still decomposed into several stages, each being solved through heuristic algorithms, it's important to define the feedback between the different stages, namely the decisions in route design with those of the transit assignment. Several works address this topic.

  [31] solves the bus network design problem by first generating an initial route set, where the stops with a higher demand have a higher probability of being selected as the route nodes. Then, a genetic algorithm is used to minimize the average in-vehicle travel time and maximize the number of demand satisfied with 2 transfers or less.

  [35] divides the problem into three stages: skeleton route design, main route design and branch route design and use Ant Colony Optimization (ACO) to solve the model.

  [28] develops two versions of a genetic algorithm with elitism, that minimizes the total travel time, the total number of transfers and the number of unsatisfied passengers. The initial route set is a set of shortest paths between the stops that satisfy the most demand. Numerical results show that their approach outperforms several previous state of the art methods.

[36] develops a GA-based algorithm to optimize two public transport networks in Quebec. They use pedestrian network data and road network data from OSM and demand data from household travel surveys with around 10% household coverage. Their approach is novel by considering the precise pedestrian network data for access, egress and transfer routing.

- **Bi-level approaches** Bi-level optimization is a method where an optimization problem contains another optimization problem as a constraint [37].The outer optimization task is commonly referred to as the upper-level optimization task, and the inner optimization task is commonly referred to as the lower-level optimization task. Several authors addresses the Transit Network Design problem using a bi-level approach, where the upper-level defines the lines, minimizing travel times, waiting times and operation costs, while the lower-level optimization task is concerned with the passenger assignment [38, 39].

- **Robust and stochastic** More robust approaches that the take the demand uncertainty into account have been developed. [40] developed two stochastic models, one that considers spatial equity, which concerned with the benefit distribution among network users and another that considers demand uncertainty.

  [41] developed a GA-based approach to the bus network design problem that takes seasonal transit demand variations into account.

All the previous research

2.4 Genetic Algorithms

GA are a family of computational models inspired by natural selection. These algorithms encode a potential solution to a an optimization or search problem on a chromosome-like data structure, and apply biologically inspired operators such as crossover, mutation and selection [42].

First, the solution to the optimization problem is encoded. Each encoded solution is called a chromosome, or individual. The elements that make up each chromosome are called genes. Then, according to the objective function of the optimization problem, a fitness function is constructed. In each generation, the fitness of each chromosome is computed. Chromosomes in a given generation group in pairs (i.e., mate), with fitter individuals having a higher probability of being selected for pairing. These parent individuals produce offspring individuals by genetic crossover. Random (commonly infrequent) mutations are performed on individual genes within the chromosome. These genetic operations yield two new individuals, which represent two new solutions that have traits from both parents. Thus, a new generation is created. After many iterations (generations), the fitness values of the individuals should increase, because the objective function is more likely to select better individuals to mate and produce offsprings. Commonly, the algorithm terminates after a pre-defined number of iterations or a satisfactory fitness level is reached.

In a GA based optimization process, the following parameters must be specified:

- ***Crossover probability*** The probability that crossover will happen between selected parent chromosomes

- ***Mutation probability*** The probability that mutation will occur on the genes of the offpsring chromosomes

- ***Population size*** The number of individuals in each generation

- ***Crossover operator*** The method used to mate two parent chromosomes, yielding two offpsring chromosomes
- ***Mutation operator*** The method used to mutate the genes of the offspring chromosomes
- ***Selection Scheme*** The method used to select which individuals will be the parents

When compared to traditional optimization methods, GA based approaches have to ability to avoid being trapped in local optima, thanks to the adaptation of the biological evolution model. Furthermore, these algorithms perform well in solving large, discrete, combinatorial optimization problems with difficult-to-calculate objective functions [31], making them a very good candidate for the TND problem.

Chapter 3

ODX Inference

OD matrices are a crucial input for optimization In order to observe passenger journeys using fare transaction data, we must obtain the time and location of both the origin and destination of each journey, where each journey consists of one or more journey stages. As in [9], we define a journey stage as a portion of a passenger's travel that is associated with a single fare transaction, wether the fare transaction consists of a single boarding tap, as is the case with bus travel, or an entry and exit tap, as is often the case with metro travel. When metro system requires the passenger to tap the card when they enter and when they exit a station, time and location information are recorded for both entry and exit stations. The bus system requires passengers to tap the card when they board a bus but not when they alight, so destination time and location must be inferred.

Our implementation is based on the work done by [43]. This algorithm has since been used by the (Massachusetts Bay Transportation Authority (MBTA)) to plan service and understand where the network is crowded . One of the particular applications of the ODX matrices in the MBTA is the estimation of the volume of passengers between stations by time periods on the subway and light rail. [44] Furthermore, using the passenger arrival rates, the MBTA can assess the level of subway reliability, by measuring the percent of passengers who waited less time than the scheduled headway, being a very valuable metric for passenger experience [45]. Although their implementation, like the majority in the literature, uses AVL data, it is straightforward to adapt the algorithm to rely only on AFC data, at the cost of a decrease in the inference percentage.

Usually, the methodology is divided into the following processes: first, the origins for the bus stages are inferred, given that only the boarding times are known. Second, the alighting time and location for the bus stages is inferred. Third, the interchanges between stages are inferred, to reveal passengers' linked transit journeys [9]. Finally, the OD pairs for several time periods are computed. In our case, the bus origin step was already part of the AFC system, as we'll discuss further.

In this chapter, we will describe these processes and how they were applied to ODX inference in Lisbon.

3.1 Input Data

The following sections discuss the several data sources used in the ODX inference process, along with the data validation and preprocessing steps that were applied.

3.1.1 Lisbon AFC

LISBOA VIVA [46] is an AFC card based on Radio Frequency Identification (RFID), and it is the most used fare card in Lisbon. LISBOA VIVA is accepted on multiple public transport modes, including CARRIS' bus network, Comboios de Portugal (CP)'s rail network and the METRO underground rapid transit network. In the scope of this work, only the AFC data relative to the CARRIS bus network and METRO was used.

The AFC data relative to 10 days (from 7 to 16 of October, 2019) was used. The data includes 11 million transactions.

3.1.1.1 Bus

The bus network AFC system is an open system, meaning passengers only use their fare cards when boarding a bus. The fare structure for occasional journeys allows the passenger to travel for 60 minutes, with unlimited number of bus journeys (counting from the first to the last entrance validation) and so no destination information is needed.

A transaction record is generated each time a passengers boards a bus. The relevant information extracted for our work is:

- the **timestamp** associated with the transaction, specifying the date and time, with second precision
- the **card serial number**, which is unique per passenger
- the **stop identifier** identifying the stop where the passenger boarded the bus.
- the **route** being serviced by the boarded bus, which is split into three different fields:
  - the **route_id**, which identifies the route
  - the **route_direction**, which gives the direction of travel and can take the value of `ASC` (ascending), `DESC` (descending) or `CIRC` (circular)
  - **route_variant** which differentiates between the several variations of the route. Some routes may, for example, only serve a subset of stops during non-peak hours.

On most of older the systems reported by the literature, the boarding stop is not recorded [9, 4]. On bus systems for which AFC and AVL data is available, passenger boarding locations are typically inferred through matching each AFC record to an AVL record.

In the AFC system of CARRIS, the boarding stop id is already present in the AFC records. It is assumed that a similar technique was applied, but no details are known.

3.1.1.2 Metro

The metro AFC system is a closed system, meaning passengers have to use their fare cards when entering the first station and when exiting the last station of the journey. The fare structure for occasional journeys is valid for 60 minutes, with a single metro journey (between a boarding station and an exit station), so destination information is needed [47]. It is worth noting that every interchange inside the metro network is made 'behind the gate', i.e, passengers don't have to tap the card when transferring between metro lines. For this reason, the AFC system does not record metro transfers. Thus, it is reasonable to assume that a passenger only exits a station in the end of a journey.

A transaction record is generated each time a passengers enters or exits a station. The relevant information extracted for our work is:

- the **timestamp** associated with the transaction, specifying the date and time, with second precision
- the **card serial number**, which is unique per passenger
- the **station identifier**, identifying the station where the passenger tapped the card.
- the **entry/exit** field specifies if the tap corresponds to an entry or an exit.

3.1.2 AFC Preprocessing

An initial preprocessing step was applied to both bus and metro AFC datasets, with the objective of reducing space, providing better querying performance and establishing a standardized and consistent schema. The steps were performed were:

- **Extraction of relevant fields**

  Both AFC datasets have static data that is repeated across the records, and should thus be stored in another dataset. An example of this is the *stop name*, which is always associated with a single stop identifier. A separate dataset that maps stop identifiers to stop names was created.

- **Type conversion**

  When possible, the column types were converted to types that require less space and/or are more efficient to lookup. An example of this is the conversion of the metro stop_name (`string`) field to an id (`int`) field.

- **Column name standardisation**

  The column names were standardised across the two datasets.

3.1.3 Interface between AFC and static data

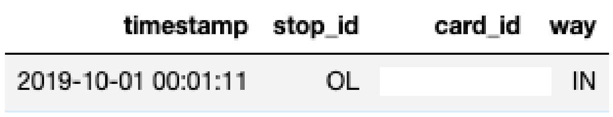
Static data describing bus schedules, bus stop locations and metro station locations is required for the ODX inference process. To match both static data and AFC data, an initial pre-processing step was applied to both metro and bus data sets.

3.1.3.1 Metro

The locations of the metro stations was obtained from its GTFS feed. However, since the stop identifier in the AFC doesn't match the GTFS stop identifier, a preprocessing step had to be implemented in order to match both data sources.

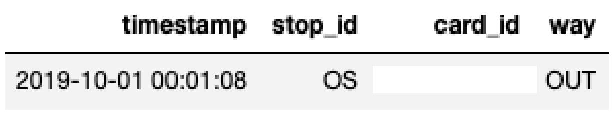
In the AFC side, we have a single field identifying the stop. It has two letters that are an abbreviation of the stop_name. For metro stations that serve two lines (instead of one), there are typically two different physical entrances to the station, each one closer to one of the lines. In these stations, the `stop_id` field in the AFC records has an additional digit, used to differentiate between these two entrances. This digit was not used for the purpose of our work, and was thus ignored. In the GTFS side, each stop is identifiable by a `stop_id` and a stop_name. Since every `stop_name` in the GTFS is unique, our objective was to match each AFC `stop_id` to a GTFS `stop_name`. Even though in most of the cases this can be done using the fact that the AFC stop_id is an abbreviation, of the GTFS stop_name, some cases, like the one shown below, are ambiguous.

To solve the matching problem, the table in [48] was used to convert the abbreviation into a stop name. Then, the stop name was matched to the most similar stop_name in the GTFS, using a fuzzy matching algorithm.



	timestamp	stop_id	card_id	way
	2019-10-01 00:01:11	OL	[redacted]	IN

Figure 3.1: Metro AFC transaction in station *OL*



	timestamp	stop_id	card_id	way
	2019-10-01 00:01:08	OS	[redacted]	OUT

Figure 3.2: Metro AFC transaction in station OS

3.1.3.2 Bus

In order to get a coherent data set, each AFC transaction's route must be mapped to a GTFS route. In the AFC side, we have three fields that identify the route (`route_id`, `route_variant` and `route_direction`). In the GTFS side, we have a single `route_id` that identifies a route. We show an example of an AFC route and the corresponding candidate routes in the GTFS.

From Figures 3.4 and 3.5, it is clear that there are a few issues:

- The AFC `route_id` does not match the GTFS `route_id`. Instead, the AFC `route_id` is part of the GTFS `route_long_name`. Which means the matching would have to be made from AFC `route_id` to GTFS `route_long_name`, which is not a unique identifier.
- The AFC encodes direction as either `ASC` , `DESC` or `CIRC` , corresponding to ascending, descending or circular routes, respectively. This direction field is not present in the GTFS, which instead has several `route_ids` for different directions of the same route (which is against the GTFS best practices) [49].

Furthermore, not all AFC routes were present in the GTFS.

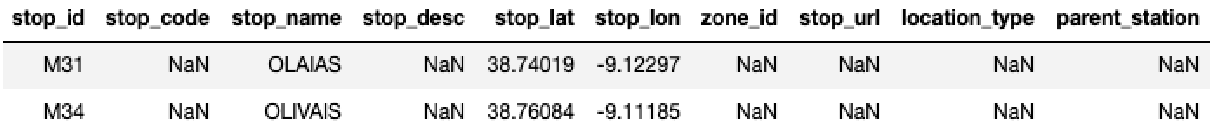
All this made the use of the GTFS dataset very impractical, and, after unsuccessfully trying to solve these inconsistencies through heuristics, carris kindly supplied us **with another data source**.

This additional static data source, which shall be referred to as Bus Static Dataset (BSD), includes the following information:

- **Stop data**
  - Stop identifier (`stop_id`)
  - Stop name (`stop_name`)
  - Stop latitude and stop longitude (`stop_lat` and `stop_lon`)
- **Route data**
  - Route identifier, which includes three separate fields (`route_id`, `route_direction` and `route_variant`)
  - Route name (`route_name`)
  - Sequence of stops served by the route (`route_stop_ids`)

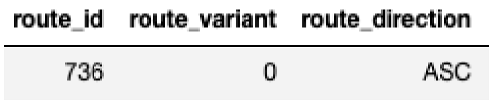
While we cannot directly map BSD routes to GTFS routes, the BSD stops can be mapped to GTFS stops, which will be useful in the following steps.

The `stop_id` in the BSD is identical to the `stop_id` in the GTFS' `stops.txt`, with the difference that every stop_id in the GTFS includes a trailing `1_`, which is not present in the BSD. For example, the



stop_id	stop_code	stop_name	stop_desc	stop_lat	stop_lon	zone_id	stop_url	location_type	parent_station
M31	NaN	OLAIAS	NaN	38.74019	-9.12297	NaN	NaN	NaN	NaN
M34	NaN	OLIVAIS	NaN	38.76084	-9.11185	NaN	NaN	NaN	NaN

Figure 3.3: Metro GTFS stops *OLAIAS* and *OLIVAIS*



route_id	route_variant	route_direction
736	0	ASC

Figure 3.4: Bus AFC transaction in `route_id` 736

`stop_id 12345` in the BSD corresponds to the `stop_id 1_12345` in the GTFS. The BSD has 2193 stops, 6 of which are not present in the GTFS. The GTFS includes 2200 stops, 13 of which are not present in the BSD.

3.2 Origin Inference

As was already stated, the bus AFC system in Lisbon automatically infers the boarding stop of each record. For this reason, and because we did not have access to the AVL data, no origin inference methodology was implemented in this work.

While we have no information about the process used, we can assume a similar technique to [9] and [4], where each AFC record is matched to an AVL record, and the AVL location is used as the passenger boarding location. The method is summarized:

The first step is to select the subset of AVL records that share the same route, vehicle or trip present in the AFC record. Then, an AVL record for which the timestamp closely matches the AFC record is selected. AVL data includes the times at which buses arrived and departed from stops, although the triggers for these timestamps differ between AVL systems. The trigger can happen when the doors open, close, the vehicle approaches or departs from a stop, etc. Finally, the location of the selected AVL record is used as the passenger boarding location.

There are a few errors stemming from the built-in origin inference in the bus AFC system. First, only **88.6%** of the records have an inferred boarding stop. This percentage is considerably lower than the 96% reported in the literature [9], so we believe that improvements can be made to the existing origin inference algorithm.

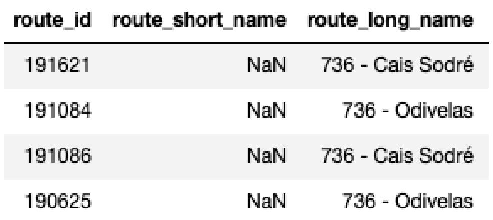
Around **3.65%** of the records have the boarding on the last stop of the route. In some cases, this is likely because the AFC system failed to note a transition from one vehicle trip to the next. An example is the following record, on stop 10309 of route 742 ASC, variant 0.

There are three routes with `route_id` 742:

- 742, ascending direction, variant 0, travelling between stops 83110 and 10309
- 742, descending direction, variants 0 and 4, travelling between stops 10309 and 83110

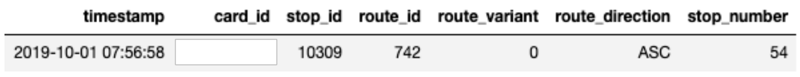
It is likely that the passenger boarded route 742 DESC (descending), instead of 742 ASC (ascending). However, we do not know if he boarded variant 0 and 4.

In other cases, we believe that either the origin inference algorithm or the static data is incorrect. An example is every record that contains the stop ***94904.*** Because this stop is the last stop of every route it appears in, there should be no records with this stop, but **4806** records with this stop exist.



route_id	route_short_name	route_long_name
191621	NaN	736 - Cais Sodré
191084	NaN	736 - Odivelas
191086	NaN	736 - Cais Sodré
190625	NaN	736 - Odivelas

Figure 3.5: Bus GTFS route 736 variants



timestamp	card_id	stop_id	route_id	route_variant	route_direction	stop_number
2019-10-01 07:56:58	[redacted]	10309	742	0	ASC	54

Figure 3.6: Bus AFC transaction on last stop of route

Both the records without stop information and those where the recorded stop are the last of the route were discarded.

3.3 Destination Inference

The destination inference process aims at inferring bus alighting times and locations. The process of destination inference is founded on two key assumptions, commonly described in the literature [4, 9]:

- The best estimate of a passenger's alighting is the stop closest to the next boarding
- The best estimate for a passenger's final day destination is that passenger's first origin of the day.

These assumptions will be referred to as the *closest-stop rule* and the *daily-symmetry rule*, respectively.

There are, however, many cases in which these assumptions are not valid. The closest-stop rule is violated when a passenger alights a bus, walks along a bus route to a shop and then boards a bus in the stop closest to the shop, which may not be the closest stop to the previous alighting stop. The daily-symmetry rule is violated when a passenger's last stage of the day ends somewhere other than his first origin of the day, maybe because, for example, instead of taking the bus home, the passenger decided to take a bike. Despite these shortcomings, these two assumptions have shown to be the best estimates for inferring a passenger's destination [4, 9] To apply these rules, spatial information about the passenger's possible alighting location and subsequent origin location is needed. In the case of Lisbon, two datasets were used to access this information:

- The metro GTFS dataset
- The schedule information dataset, provided by the bus operator

The stop information required are the unique identifiers (`stop_id`) and the stop locations in geographic coordinates (`stop_lat` and `stop_lon`).

The distance between the passenger's next boarding location, referred to as target-location, and each stop served by the current vehicle trip need to be calculated, in order to determine which stop is closest.

The information required to get the sequence of stops served by the current vehicle trip are the `route_id`, `route_variant` and `route_direction` fields present in the AFC record and its correspondent sequence of stops, present in the bus schedule dataset.

The haversine formula, which determines the great-circle distance between two points on a sphere given their longitudes and latitudes [50], was used to calculate the distance between each stop served by the current vehicle trip and the target-location [1] For efficiency purposes, the distance between every pair of stops was calculated before processing the AFC records.

The destination inference algorithm is adapted from [9] and is summarized in Figure 3.7.

If the transaction corresponds to a metro trip, the origin and destination should be known. To form a metro stage, two metro transactions are required. One corresponding to the stage entry, and another corresponding to the station exit. It is possible that the two transactions actually correspond to two different stages, but the exit transaction of the first stage and the entry transaction of the second stage were not registered, due to a system failure. To account for this, a maximum metro stage time is imposed. If the maximum metro stage time is exceeded, the entry transaction is assumed to be part of a metro stage with unknown destination and the exit transaction part of a stage with unknown origin.

If the transaction corresponds to a bus boarding, the stop must be valid and cannot be on the last stop of the route. Otherwise, the origin is assumed unknown and the destination cannot be inferred.

The route field must be valid, otherwise the destination cannot be inferred.

If there are no other records in the card's daily history, the closest-stop rule cannot be applied and, as such, the destination cannot be inferred.

If the transaction is the last transaction of the day, the daily symmetry rule is applied, and the target location is defined as the first origin of the day. Otherwise, the target location is the next stage's origin.

The bus stop that is closest to the target location is selected as the candidate alighting stop. If that stop was served by the route before the boarding stop, the bus is travelling away from the next tap location and, as such, the destination cannot be inferred.

If the distance between the alighting stop candidate and the next tap location is within the pre-defined maximum interchange distance, the destination is inferred.

3.3.1 Bus Alighting time estimation

In addition to inferring alighting locations, the destination inference process also aims at inferring alighting times. More recent methods for bus alighting time estimation in the literature use AVL data. Since we did not have access to AVL data, another approach for estimating the time at which passengers alight the bus had to be used.

Our approach is similar to [4]. We use the GTFS file `stop_times.txt`, which has, for every trip, the scheduled arrival and departure times for every serviced stop. In the case of the bus GTFS, the departure time and arrival time have the same value in every entry. By calculating the difference between the departure time for two consecutive stops, one can obtain an estimate for the time the bus takes between those two stops. However, given that the routes present in the bus AFC are not the same as the routes in the bus GTFS, it is necessary that every pair of consecutive stops in the bus AFC is also a pair of consecutive stops in the bus GTFS.

Out of the 2958 stop pairs present in the bus `routes.json` , 29 are not part of a GTFS trip, so no information regarding bus arrival times is present. For these 29 stop pairs, OSRM was used to compute duration of the shortest path between the two stops. Neither estimation accounts for the extra time the bus takes to stop and wait for passengers to board and alight the vehicle. To account for this time, we add the constant bus_stop_time, with a value of 30 seconds to the estimated time between two consecutive stops. While the accuracy of this value cannot be empirically assessed, it was selected based on the

[1]While the haversine distance is used, given that we are dealing with very small distances, the difference to the euclidean distance is negligible.

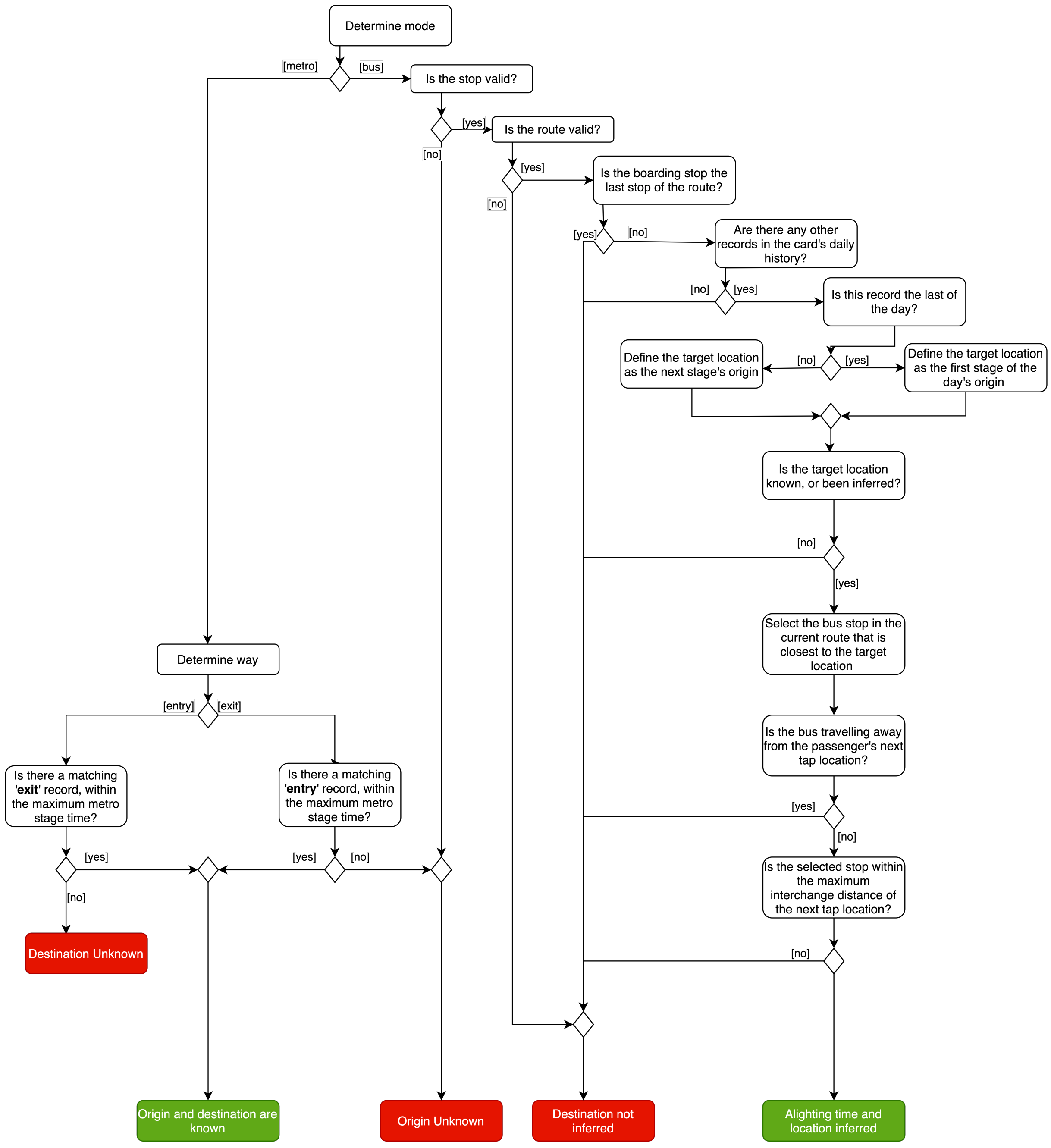

Determine mode
[metro]
[bus]
Is the stop valid?
[yes]
[no]
Is the route valid?
[yes]
[no]
Is the boarding stop the last stop of the route?
[yes]
[no]
Are there any other records in the card's daily history?
[no]
[yes]
Is this record the last of the day?
Define the target location as the next stage's origin
[no]
[yes]
Define the target location as the first stage of the day's origin
Is the target location known, or been inferred?
[no]
[yes]
Select the bus stop in the current route that is closest to the target location
Is the bus travelling away from the passenger's next tap location?
[yes]
[no]
Is the selected stop within the maximum interchange distance of the next tap location?
[no]
Determine way
[entry]
[exit]
Is there a matching 'exit' record, within the maximum metro stage time?
Is there a matching 'entry' record, within the maximum metro stage time?
[yes]
[no]
[yes]
[no]
Destination Unknown
Origin and destination are known
Origin Unknown
Destination not inferred
Alighting time and location inferred

Figure 3.7: Destination Inference Fluxogram

the assumption that it's a good approximation for the average time a bus takes to stop and wait for passengers to board and alight the vehicle.

When calculating the stage time, we subtract the bus_stop_time once, because the passenger does not have to wait on the stop he alights. Since the estimated time for each consecutive stop pair is constant, it is possible that the inferred alighting time for one stage is ahead of the next stage's boarding time. In these cases, the alighting time is given by the boarding time of the next stage, minus the estimated time it took for the passenger to walk to the next stage, given by: $dist_{s1,s2} * walk\_speed$,

where $dist_{s1,s2}$ is the *haversine* distance between the stops, in *km*, and $walk\_speed$ is a parameter describing the passenger walk speed which is set to $4km/h$, which was an observed value for average passenger walking speed, in previous research that uses AVL data [43].

3.3.2 Sensitivity analysis

3.3.2.1 Maximum Interchange Distance

When the alighting location of a bus stage is too far from the passenger's next boarding/entry location, it is reasonable to assume that the closest-stop rule does not apply. A distance of several kilometers suggests that the passenger took another mode of transportation, not recorded by the AFC system.

The distribution of the observed haversine distances is shown in Figure 3.8, and the cumulative frequencies in Table 3.1, where the candidate alighting stop is the closest stop on a passenger's bus stage to his next transaction, regardless of whether or not another stop was eventually chosen by the destination inference process. A value of 1000 meters was chosen as the maximum interchange distance.

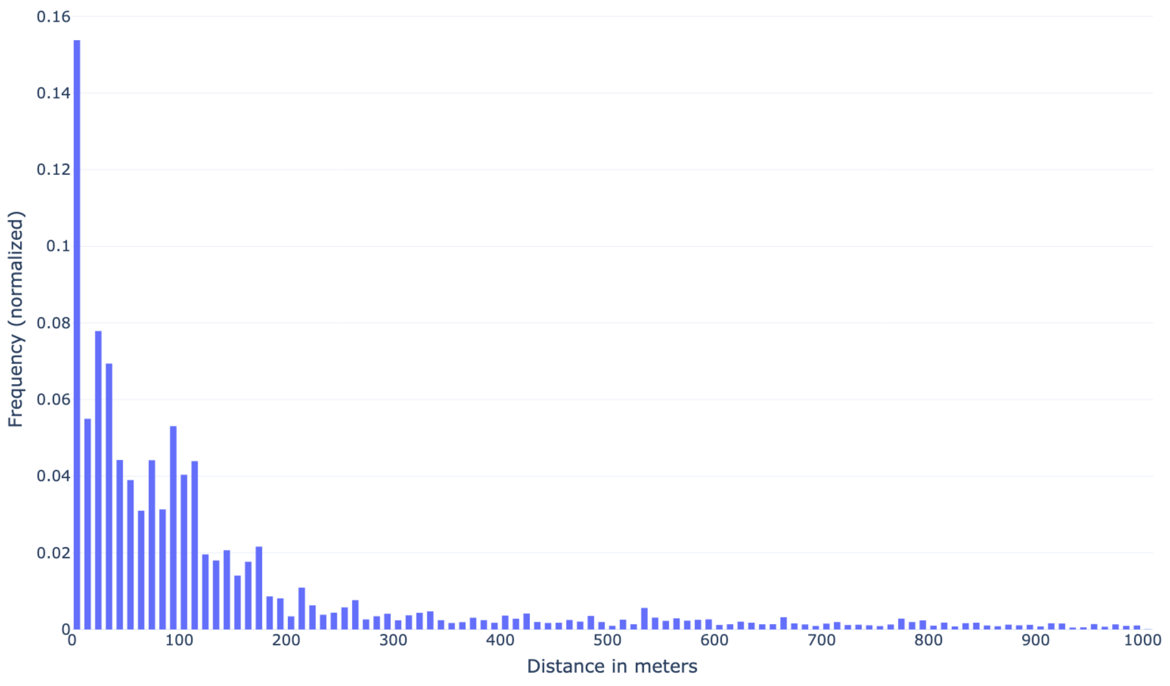


Figure 3.8: Distribution of the distances between each candidate bus alighting stop and next boarding's stop/station

Haversine distance (m)	Percentile
500.00	82.92
750.00	87.39
1000.00	90.29
1500.00	92.95
2000.00	94.65

Table 3.1: Cumulative frequencies of inter-stage distances

3.3.2.2 Maximum metro stage time

When a metro stage time is unusually high, it's likely both entry and exit transactions that make up the stage, actually span two stages, but the missing exit and entry transactions were not registered, due to some system failure. Figure 3.9 shows the distribution of the metro stage times, before any maximum stage time was imposed. Most metro stages span less than 60 minutes, but a more conservative maximum metro stage time of 70 minutes was set.

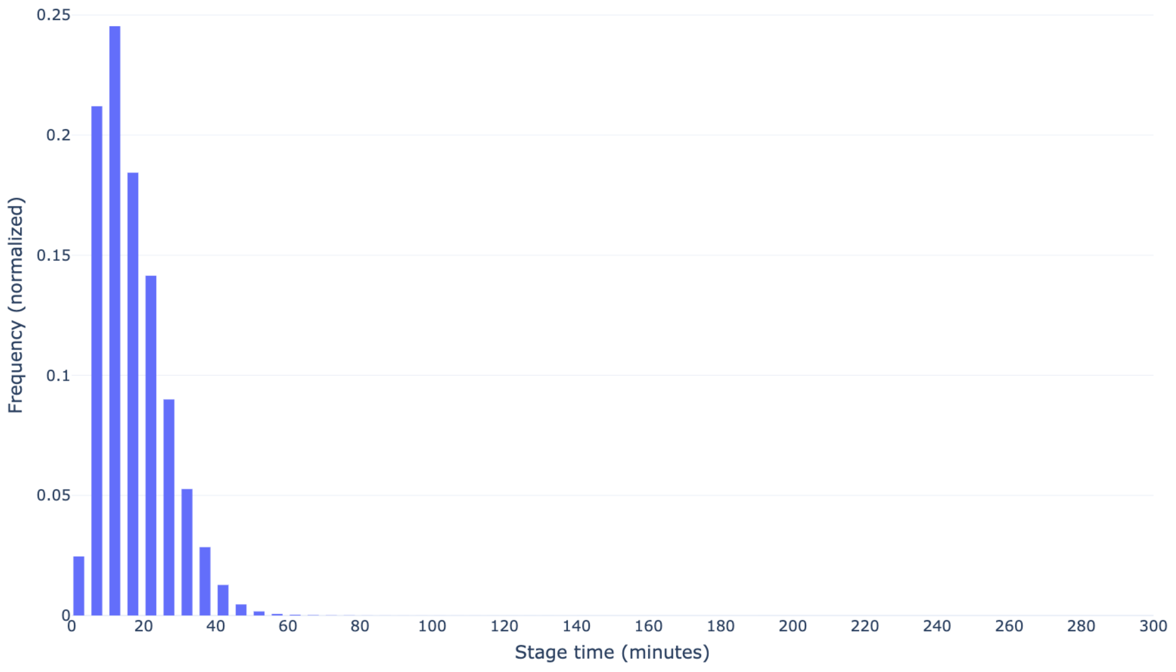


Figure 3.9: Distribution of the metro stage times

3.3.2.3 Destination Inference Results

The results for the destination inference process are shown in Table 3.2, showing the percentage of stages that failed each test. On average, the methods presented in this chapter are shown to infer bus alighting times and locations in ***45 percent*** of cases. Most uninferred destinations are due to:

- A high number of bus stages that are the passenger's only stage of the day ***(16.35%).***

  A portion these cases are caused by one of the limitations of this algorithm, which is that we consider the day to end at some time at which we expect little to no ridership. The time chosen was 4:00am. We don't consider journeys spanning this time.

Result	Count	Percentage
Destination Inferred	854674	45.24
Distance between candidate alighting location and next origin greater than 750 meters	123529	6.54
Next stage has no boarding stop information	62551	3.31
Current stage has no boarding stop information	217081	11.49
Rider traveling away from next origin	112168	5.94
Rider boarded bus on last stop of the route	68946	3.65
Only one stage on card that day	308975	16.35
Last stage of day; distance between candidate alighting location and first origin of day greater than 750 meters	33274	1.76
Last stage of day; rider traveling away from first origin of day	108092	5.72
Total	1889290	100.00

Table 3.2: Destination Inference Results

For example, if someone made a transaction at 3:45am and then at 4:05am, we don't consider the 4:05am transaction as a target or potential transfer from the 3:45am transaction.

- A high number of bus records with no boarding information ***(11.49%)***

  The lack of boarding information is a result of the boarding inference algorithm, which was not developed in this thesis.

  This lack of origin information doubly affects the destination-inference process, since origin locations are required both for the journey stage being processed and for the subsequent stage, if the later corresponds to a bus stage. This is reflected in the ***3.31%*** of stages for which the next stage has no boarding stop information.

3.4 Interchange Inference

The destination inference process discussed in the previous chapter enriches the AFC data, by adding destination time and location to the individual passenger stages. This section describes the methodology used to infer interchanges between stages to reveal passengers linked transit journeys.

The work in [9] defined an interchange as a transition between two consecutive journey stages that does not contain a trip-generating activity.

A trip-generating is an activity (aside from travelling) that caused the passenger to transfer at that particular stop.

There is an important distinction between:

- A passenger that chose to transfer at a particular location to make some purchase
- A passenger who purchases a cup coffee while walking from his alighting bus stop to his subsequent bus stop, who only got the coffee because he happened to be transferring at that location

In the latter case, the passenger's demand for travel lies in the final destination, rather than the bus stop near the coffee house, since the primary purpose of the transfer was to connect her previous origin to the next destination.

As in [43], we infer interchanges by distinguishing them from trip-generating activities. Temporal and spatial data are used as proxies for passenger activity information, based in the assumption that trip-generating activities are longer in duration than convenience activities.

Binary, spatial and temporal tests are applied sequentially to every stage, in order to infer if a transition between two stages consitutes an interchange, i.e, if the two stages are linked and thus part of the same journey.

A failing test means that the stage being tested is unlinked from the subsequent stage.

The tests, summarized below, are very similar to those reported by [43], with small changes due to the lack of AVL data and difference in the metro system fare collection.

Binary Conditions

- ***Final Transaction:*** If the transaction is the cardholder's final stage of the day, it is considered unlinked.
- ***Successful Bus Destination Inference****:* If the stage corresponds to a bus transaction, its alighting time and location are required. The interchange status cannot be inferred for stages without alighting information.
- ***Repeated Service:*** If two consecutive stages of a card used the same bus route, the first is not linked to the next, regardless of the direction of travel.
- ***Consecutive Metro:*** If two consecutive stages of a card are metro stages, the first is not linked to the next. Because all metro line transfers in Lisbon are done behind the gate, if the passenger left the station, it suggests a trip-generating activity was performed.

Temporal Conditions

- ***Maximum interchange time:*** Because trip-generating activities are likely to have longer durations than interchanges, a maximum interchange time is imposed. This limit is calculated as a function of the Haversine distance between the alighting stop of the current stage and the boarding stop of the next stage. A lower limit is set for the maximum interchange time, because interchanges between very close stops would result in unreasonably short maximum interchange times.
- ***Bus Wait Time:*** If the following record represents a bus stage and the maximum interchange time fails, it may mean that the passenger did the interchange in the maximum allowed time, but spent some time for the bus. This test is applied when the interchange time test fails and represents the maximum time someone would wait for a bus.

Spatial Conditions

- ***Maximum Interchange Distance:*** An upper limit is applied to the distance between the current stage's alighting/exit location and the next stage's boarding/entry location.
- ***Circuity***: It is assumed that a multistage journey will not entail an overly circuitous path. If the combined (haversine) distance over two consecutive stages is sufficiently greater than the (haversine) distance directly between the current stage's origin and the next stage's destination, it is assumed that the disutility of the additional travel would outweigh the utility of the interchange. The ratio of these two distances is compared with a circuity ratio parameter.
- ***Full-Journey Length:*** After all other tests are applied and the full journeys are computed, The journey length test is applied to each journey and unlinks all stages in the journey if the journey's origin and destination are closer than the user-defined minimum journey length parameter.

3.4.1 Results and sensitivity analysis

The results of the transfer inference algorithm are influenced by six user-defined parameters. The selection of the parameters allows the user to select a balance between accuracy and completeness of the results. Having more conservative parameters will increase the accuracy of the result, at a cost of the number of linked stages. In order to choose an appropriate value for each of the parameters, a sensitivity analysis was conducted.

3.4.1.1 Minimum walk speed

The first parameter, **minimum walk speed** is used to determine the maximum interchange time between two stages. Figure 3.10 shows the distribution of the avenger passenger speeds between bus inferred alightings/metro exits and subsequent metro entries. Subsequent bus boardings are excluded, to eliminate wait time from the calculation.

Lower speeds likely correspond to activities. Someone who is at their workplace for 9 hours and who arrived and departed on the same route (having inbound and outbound stops in the opposite sides of the street, 20 meters apart), would have an observed speed of 0.002 km/hour [43].

However, we would expect a second cluster of speeds around a normal walking speed (4km/hour), representing actual transfers. The reason why this cluster is much more subtle than expected is a result of the way we estimate alighting times using the scheduled trips. Since the scheduled time is generally optimistic and independent of the time the trip took place, it will be too short during peak-hour trips. Being too short, the perceived transfer time will be too long and, as such, the speed will be lower than the actual speed.

For example, if a passenger boarded a bus at **stop A** at 08:00h, alighted at **stop B** at 08:40h, and entered a metro **station C** at a 200 meter distance, at 8:43h, that would give a speed of $4km/h$. If the scheduled time says the trip from **stop A** to **stop B** takes 30min, the perceived speed would be of $\approx 0.923km/h$, which is very low, and can be mistaken for an activity.

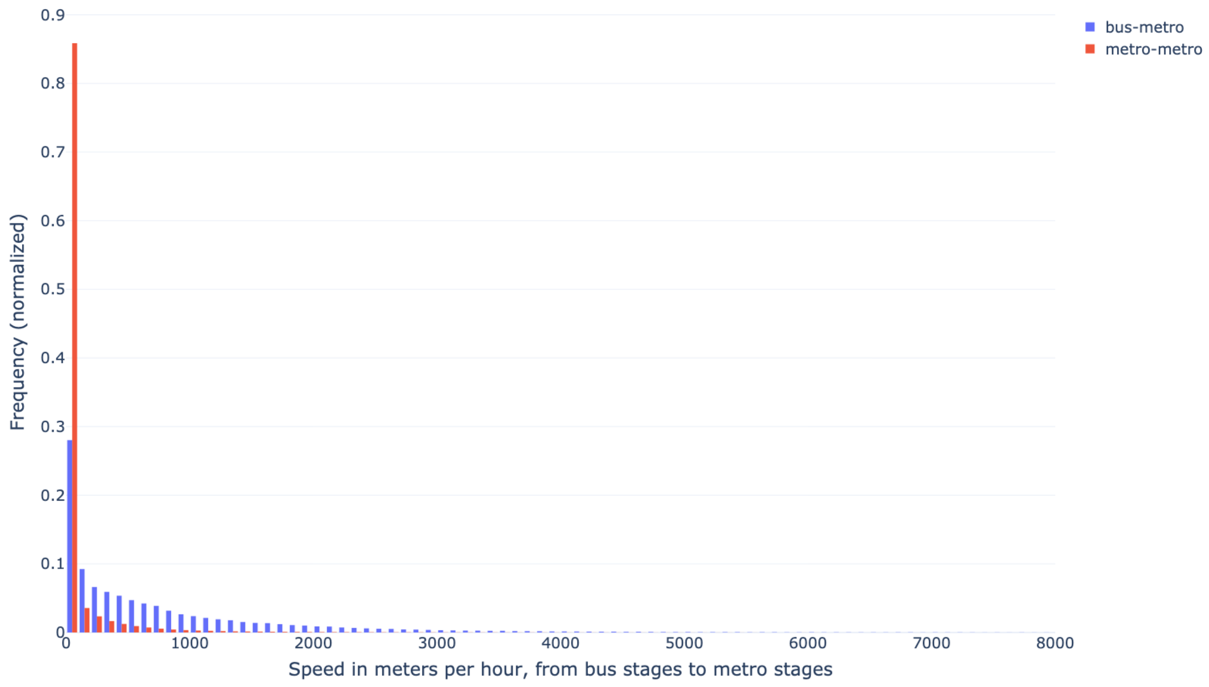


Figure 3.10: Average passenger speeds between bus inferred alightings/metro exits and subsequent metro entries

3.4.1.2 Minimum interchange time allowance

When the distance between the stops in an interchange is very low, the maximum interchange time would be unreasonably low. This parameter provides a floor for the maximum interchange time and should be set in a way that it excludes transitions of excessive duration, while preventing the imposition of any unreasonably short time limit.

Figure 3.11 shows the cumulative frequency of the interchange times between stages, with all the possible mode transitions (bus to metro, metro to bus, bus to bus and metro to metro). Interchange times for metro-metro transitions are much larger because, as specified earlier, the metro transfers are not recorded by the AFC system and, as such, a passenger should only exit a station in the end of its journey.

The interchange times corresponding to bus-metro and bus-bus transfers are shorter than expected. This is consistent with the low average speeds we discussed previously.

Transfers from bus to metro are not affected by the time the passenger spends waiting for the bus and, as such, are shorter.

There is a noticeable rise between 8 and 9.5 hours, particularly for metro-metro and bus-bus, which likely represents the time between to-work and from-work stages. It is higher for same mode transitions because trips tend to be symmetrical, i.e, the mode of the last stage, in the journey from home to work is likely to be the same as the first stage of the journey from work to home.

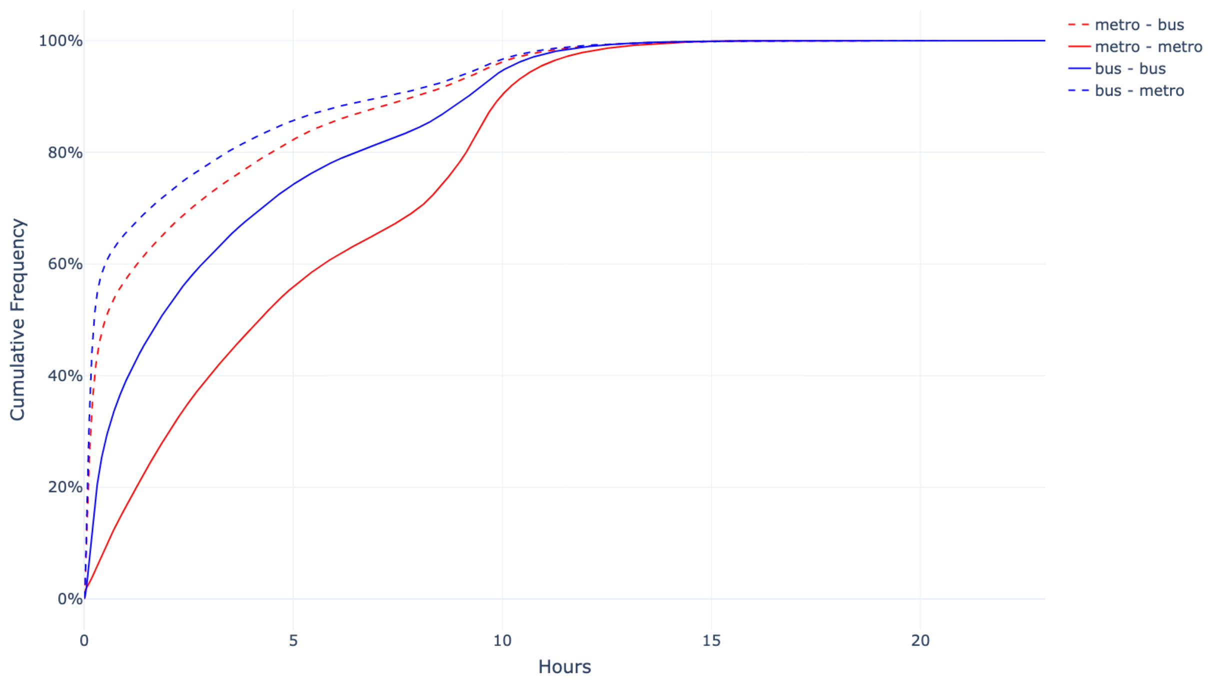


Figure 3.11: Time between passenger's consecutive journey stages

3.4.1.3 Maximum bus wait time

Figure 3.12 combines the metro-to-bus and bus-to-bus interchange times, after subtracting the estimated interchange time required for the passenger to arrive at the next stage entry stop. We would expect a higher concentration of low waiting times. However, because the estimated trip times are too short, as discussed earlier, the implied waiting time will be greater. A value of 60 minutes was chosen for the maximum bus wait time parameter.

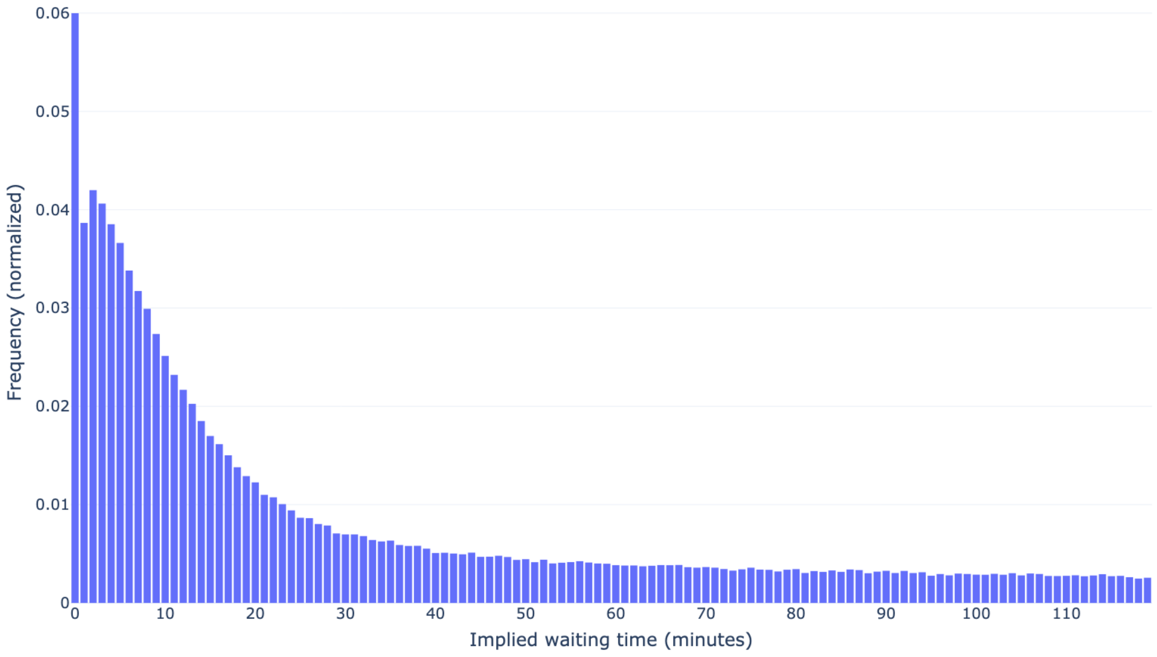


Figure 3.12: Implied waiting time for bus-to-bus and metro-to-bus transfers

3.4.1.4 Maximum circuity ratio

The maximum circuity ratio defines the maximum circuity between two stages for an interchange to be inferred. Figure 3.13 shows the circuity ratio of all potential interchanges that passed all the previous tests, while Table 4.2 shows the cumulative frequencies.

While there is a high number of interchanges with a low circuity ratio, there is still a considerable number of interchanges (17%) with a circuity ratio greater than 2. However, increasing the maximum circuity ratio from 3 to 3.5 will only give a marginal change of $\approx 1.35\%$.

Figures 3.14a and 3.14b show two transfers, with the same origin and destination, but along different routes, with circuity ratios of ≈ 1.3 and ≈ 1.8, respectively. Passengers might choose to take longer, more circuitous routes over shorter ones when the shorter one has a longer wait time, is more crowded, etc. A value of 2.5 was chosen for the maximum circuity ratio parameter. We still link 88.01% of the potential interchanges, while discarding the $\approx 12\%$ of interchanges with very high circuity ratios.

Circuity Ratio	Percentile (%)
1.50	70.31
1.70	80.15
2.00	87.29
2.50	92.46
3.00	94.75
3.50	96.08

Table 3.3: Cumulative frequency of the circuity ratios

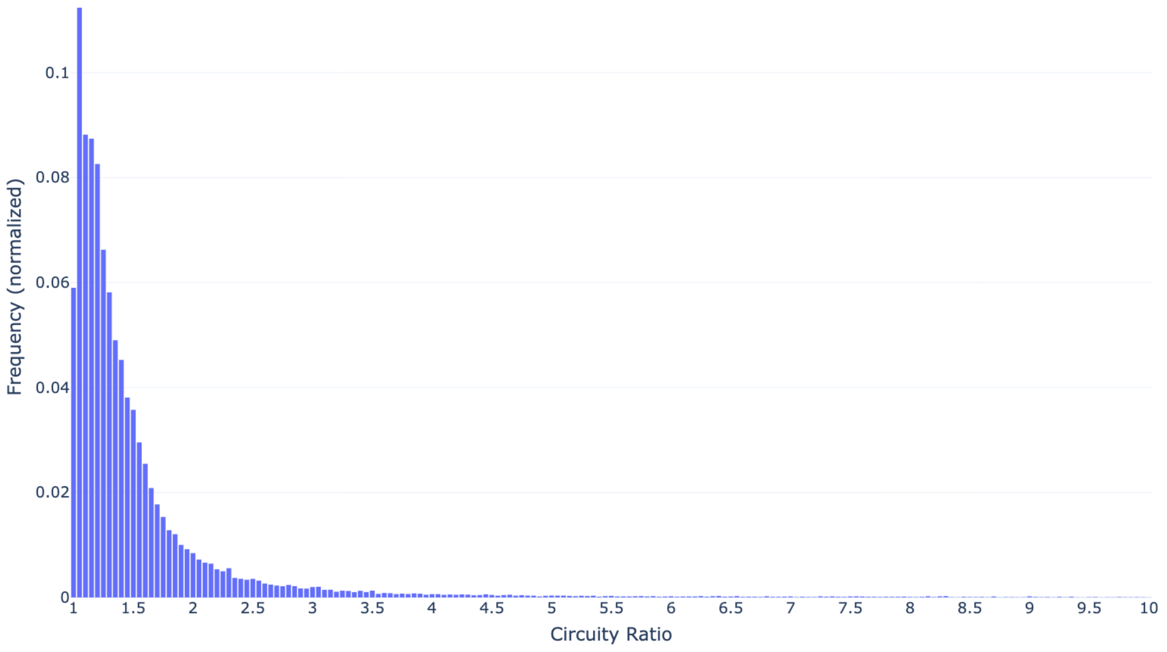
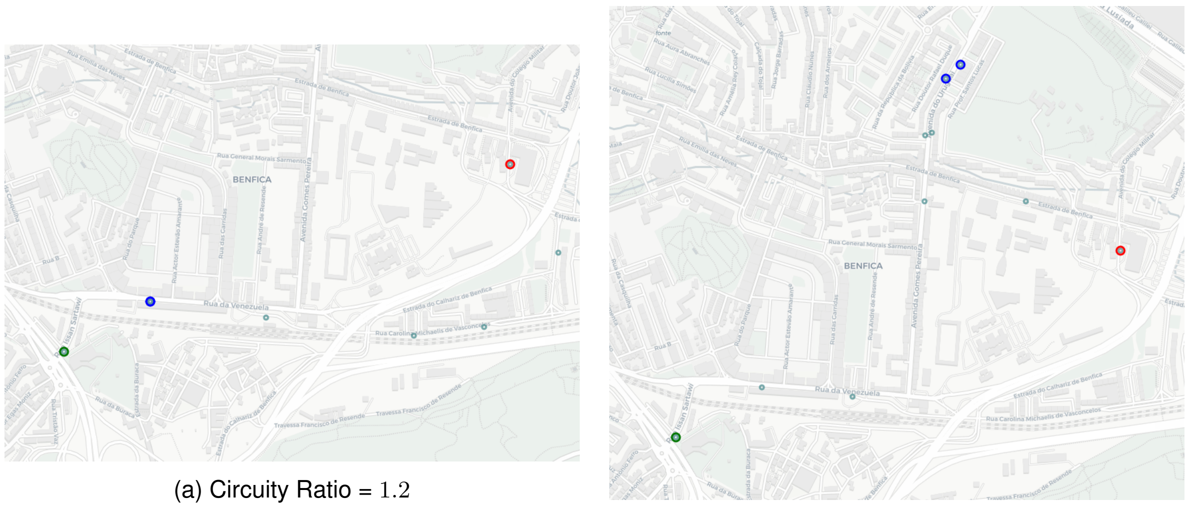


Figure 3.13: Circuity Ratio of potential interchanges



(a) Circuity Ratio = 1.2

(b) Circuity Ratio = 1.8

Figure 3.14: Two trips with same origin and destination stops (in green and red, respectively), following different paths (with different circuity ratios). The interchange stops are shown in blue, and the intermediate route stops in light green

3.4.1.5 Minimum Journey Length

Figure 3.15 shows the distribution of haversine distances between the origin and destination of every potential journey that passed all the previous tests.

The small cluster at the left represents journeys with round trips that weren't discarded by the previous tests. The transition to the right cluster occurs at 200 meters, but a more conservative value of 400 meters was set as the minimum journey length parameter.

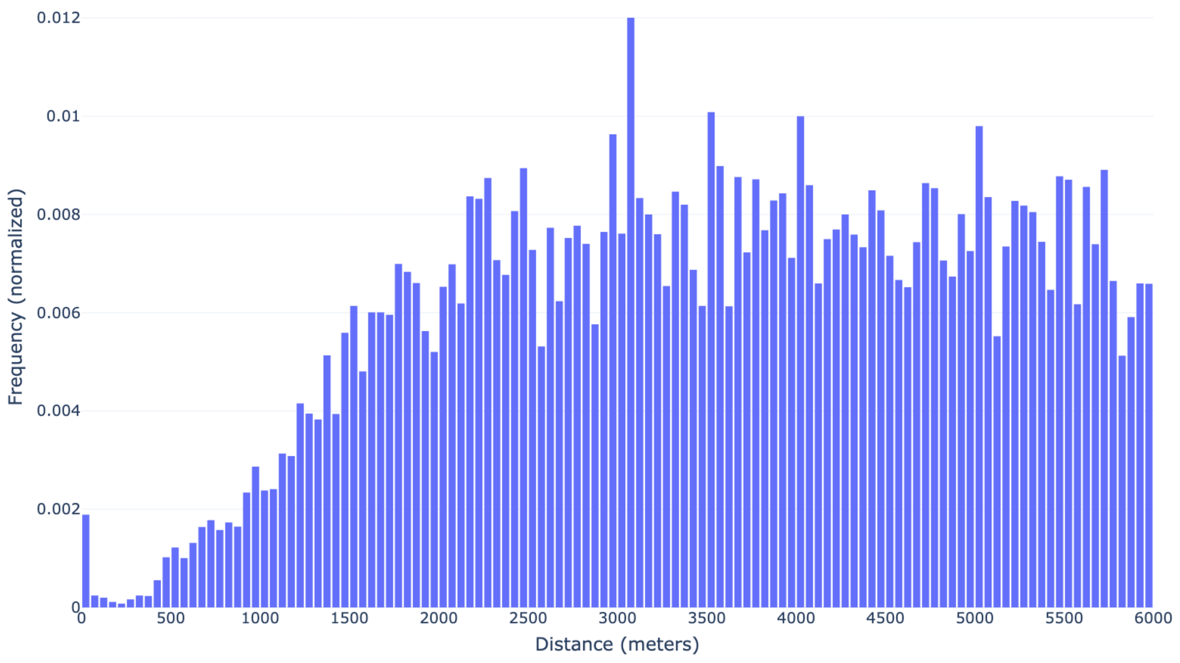


Figure 3.15: Histogram of potential journey distances

3.4.2 Interchange Inference Results

Table 3.4 summarizes the results after applying the interchange inference algorithm to the destination-inferred stages.

The results show that 2.51% of stages are considered linked to the subsequent stage, 86.69% are considered unlinked, and 10.80% of stages without an inferred link status.

The majority of stages considered unlinked are due to:

- A high percentage of transitions (41.97%) between two metro stages. This reflects the high number of passengers that only travel by metro, which is supported by Figure 3.16, which shows the high number of metro stages, in comparison to the number of bus stages.

- A high percentage of stages that are the final stage of the day (35.39%).

- A high percentage of transitions that fail the *Maximum interchange time* test. This reflect the limitations of the method used for bus alighting time estimation, discussed in Section 3.3.1.

There is also a high percentage of stages for which the interchange status could not be inferred. These are mainly due to:

- **Bus stages without an inferred boarding** $(\approx 2\%)$, which is a result of the high number of bus transactions without stop information, discussed in Section 3.2.

- **Bus stages without an inferred alighting** $(\approx 7.62\%)$, which is a result of the low destination-inference percentage, discussed in Section 3.3.

These results reinforce the need to develop a more robust origin inference algorithm, and show the limitations associated with trying to infer linked stages without access to AVL data.

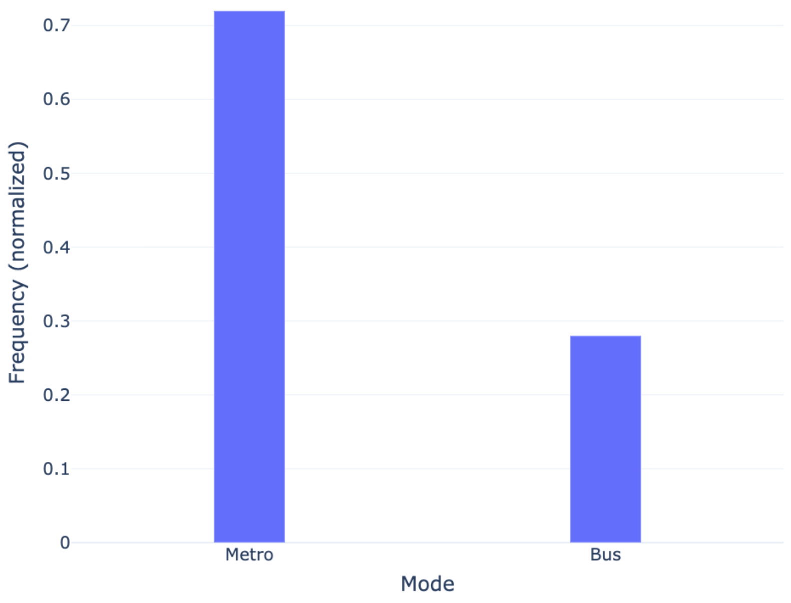


Figure 3.16: Distribution of stage modes

Result	Count	Percentage
INTERCHANGE STATUS NOT INFERRED BECAUSE:		
Current stage is metro without entry station info	9562	0.2%
Current stage is metro without exit station info	14483	0.3%
Next stage is metro without entry station info	12048	0.25%
Next stage is metro without exit station info	20340	0.43%
Current stage is bus without boarding info	59608	1.25%
Current stage is bus without inferred alighting	130014	2.73%
Next stage is bus without boarding info	35839	0.75%
Next stage is bus without inferred alighting	233216	4.89%
NOT LINKED TO NEXT BECAUSE:		
Same bus route	151836	3.18%
Current and next stages are metro	2001733	41.97%
Maximum interchange distance exceeded	8174	0.17%
Maximum interchange time exceeded	259944	5.45%
Current stage is the final of the day	1688070	35.39%
Maximum Circuity Ratio	24112	0.51%
Minimum Journey Length	699	0.01%
LINKED TO NEXT:		
Linked bus to metro	36612	0.77%
Linked metro to bus	51743	1.08%
Linked bus to bus	31529	0.66%
Subtotal, interchange not inferred	515110	10.8
Subtotal, not linked	4134568	86.69%
Subtotal, linked	119884	2.51%

Table 3.4: Interchange Inference results

3.5 Summary

This chapter described the ODX inference methodology used in our work. The methods described were applied to 10 days of AFC data, from the bus and metro systems in Lisbon, Portugal, which includes 11 million transactions. Since no AVL data was available, the methodology described relied solely on AFC data. The boarding stops of the bus transactions were already part of the bus AFC system, so no development regarding origin inference was done in our work.

An efficient methodology was developed, capable of processing the 10 days in under 10 minutes, using an i5 2500k CPU.

Destinations were inferred for $\approx 45.29\%$ of the bus stages. Most uninferred destinations are due bus stages that are the only recorded transaction on that passenger's card, that day, and due to the high number of bus stages without an inferred origin stop.

Interchanges between two stages were inferred to be linked in $\approx 2.51\%$ of the cases. The majority of due to factors unrelated to the algorithm accuracy, such as the high number of metro stages. However, the high number of bus stages without an inferred boarding and alighting account for almost the entirety of the stages with uninferred interchange status.

These results reflect the importance of increasing the robustness of the origin inference algorithm, as well as using AVL data in the interchange inference process.

Chapter 4

Transit Network Design

We approach the TND problem, with the objective of developing efficient bus routes for the existing road network and existing bus stops in Lisbon, Portugal.

While previous research mostly focuses on approaching the TND problem using synthetic data or optimizing only a subset of the network, we considered the entire bus network of Lisbon, PT. The OD matrix computed in Chapter 3 is used as the demand factor.

The TND problem is usually decomposed in two stages: the generation of the set of routes and the determination of the frequency for those routes. In our work, we focus exclusively on the generation of the set of routes.

We start by discussing the necessary inputs, as well as the problem representation used. Several challenges arise from the fact that we are dealing with a real, complex network. An efficient representation was developed, and the impact of data errors was discussed. A pre-processing step is described, where stop clusters are formed, to simplify the problem and thus increase converge speed. Then, we describe the optimization technique, which is based on the GA method described by [28, 51].

Finally, we present and discuss the obtained results.

4.1 Inputs

4.1.1 OD Matrix Pre-processing

We use the OD matrix computed in chapter 3 as a representation of the passenger demand. However, a pre-processing step is applied to the OD matrix, before it can be used as an input to the optimization process.

Since we're only optimizing the bus network, we must decide what to do when journeys include metro stages. In journeys that only include a metro stage, we assume that no bus route can outperform the metro network, and, as such, the OD pairs of those journeys are not considered in the optimization process.

In journeys where we have both metro and bus stages, we only consider the bus stages.

Figure 4.1 illustrates such a journey, where the passenger:

1. travels from stop B_1 to B_2 using bus route R_1
2. transfers from stop B_2 to stop B_3
3. travels from stop B_3 to B_4 using route R_2
4. transfers from stop B_4 to metro station M_1

5. travels from station M_1 to M_2 using metro line L_1

6. transfers from station M_2 to bus stop B_5

7. travels from stop B_5 to B_6 using bus route R_3

In reality, the passenger wants to go from B_1 to B_6. However, by optimizing the bus network to go directly from B_1 to B_6, we would be competing with the metro network, rather than complementing it. Instead, we optimize the network to go from B_1 to B_4 and from B_5 to B_6.

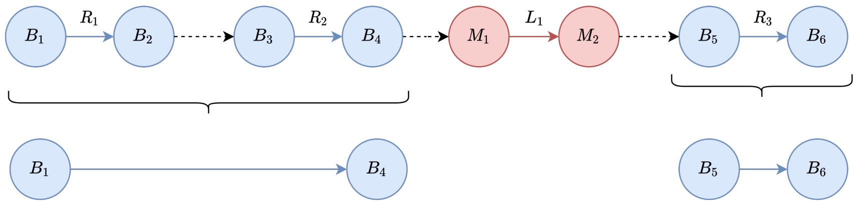


Figure 4.1: OD Matrix preprocessing before the optimization process

4.1.2 Road Network data

OSM data is used to obtain the information related to the underlying road network. OSM provides very detailed information regarding the road links, turn restrictions, maximum speeds, etc, which are crucial for a good representation of the road network.

Aside from the contributions of independent users, the OSM data for the city of Lisbon is gathered from several institutional agencies, which are listed in [8].The OSM data for the entire country of Portugal was downloaded from [4], and later filtered to include only the necessary road links to describe the bus network. Figure 4.2 shows a graph constructed from the OSM data.

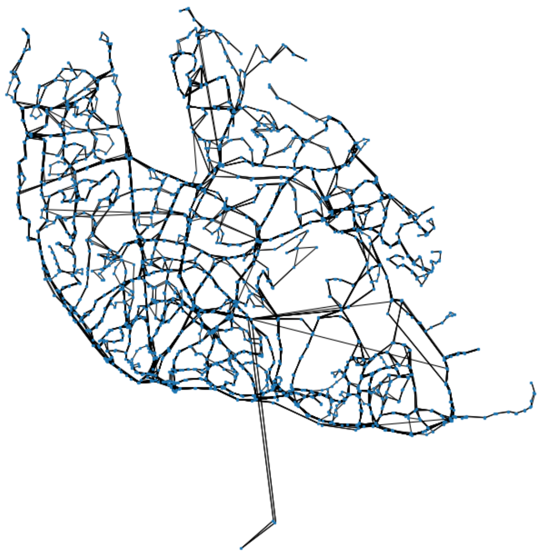


Figure 4.2: Road network graph visualization, built from OSM data

4.2 Representation

Given that we are dealing with a real, complex network, a precise and efficient representation of the road network had to be developed. The **road network** representation includes the stop locations and the road links that connect these stops. This representation is used to compute the paths between stops **in the underlying road network**.

To assess the quality of a bus network solution, we must compute the shortest path that satisfies each OD pair in the ODX matrix. The **route set** representation describes the stops and bus routes that connect the stops. Each route set has a unique route set representation. This representation is used to compute the paths between stops **in the bus network**.

4.2.1 Road Network

Using the road network data from OSM, we compute a graph that includes the bus stop locations and the road links that connect the stops.

There are several challenges associated with this representation, that will be discussed bellow.

4.2.1.1 Bus stop location

The location of each bus stop in the road network is necessary to build the road network graph.

The GTFS (General Transit Feed Specification) includes the location of every stop in the bus schedule. However, these locations are not accurate and typically correspond to the place where the passengers wait for the bus (in the sidewalk, for example). Figure 4.3 shows the GTFS location for two stops in green and the actual street point in gray. It is necessary that we map each bus stop to a point in the OSM network. Simple heuristic algorithms might give the correct result for some stops, but will fail on situations where the street associated with a certain stop is not obvious. Thus, a more robust approach to the stop mapping problem is needed.

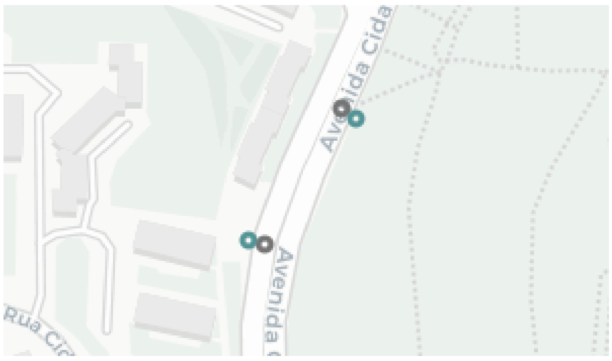


Figure 4.3: GTFS location (in green) and actual street point(in gray) for two bus stops

The geographical course of a bus trip is part of the bus operator GTFS. However, this shape data is not suitable for our needs, since it's simply straight lines connecting the same (possibly inaccurate) stop locations. Several research efforts to assign GTFS bus stops to OSM and other road networks have been developed.

[52] developed an algorithm to map each GTFS bus stop to a point in an OSM network. Their approach finds, for each stop, a set of candidate locations, that are projections of the stop on the OSM road links. Then, an optimization procedure is used to compute the set of projections (one per stop) that minimize the total distance driven to complete all the GTFS trips.

[3] developed an algorithm to solve the problem of finding the most likely geographical course taken by a public transport vehicle, given a sequence of stations taken by that vehicle.

Their algorithm takes as an input the GTFS feed and the OSM data. Their methodology is summarised:

For each stop, the set of candidate nodes in the underlying road network are the nodes that lie in a radius of 100 meters around the stop location. This step is shown in figure 4.4

With the candidates for each stop, they find the most likely node, by using a hidden Markov model, where the stop positions act as observations, and the node candidates as hidden states.

Their software (*pfaedle*) is publicly available[1] and was used in our work, to map each GTFS stop to a location on the OSM road network.

Figure 4.5 shows the comparison between an original GTFS shape and the corresponding estimated shape computed through *pfaedle*.

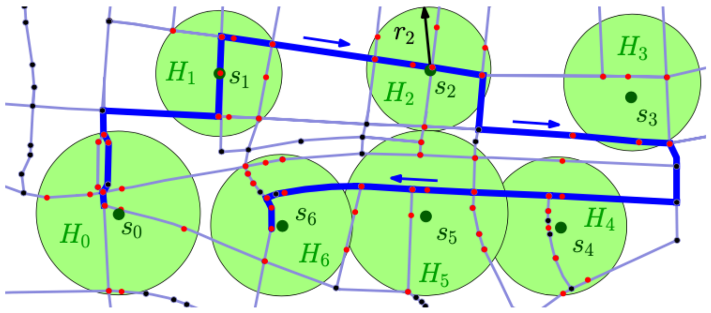


Figure 4.4: Station positions $s_0, ..., s_6$ from a GTFS feed (possibly imprecise) of a bus route through a road network. Red: candidate nodes within a radius r_i around each s_i . Blue: the correct path of the bus. Source: [3]

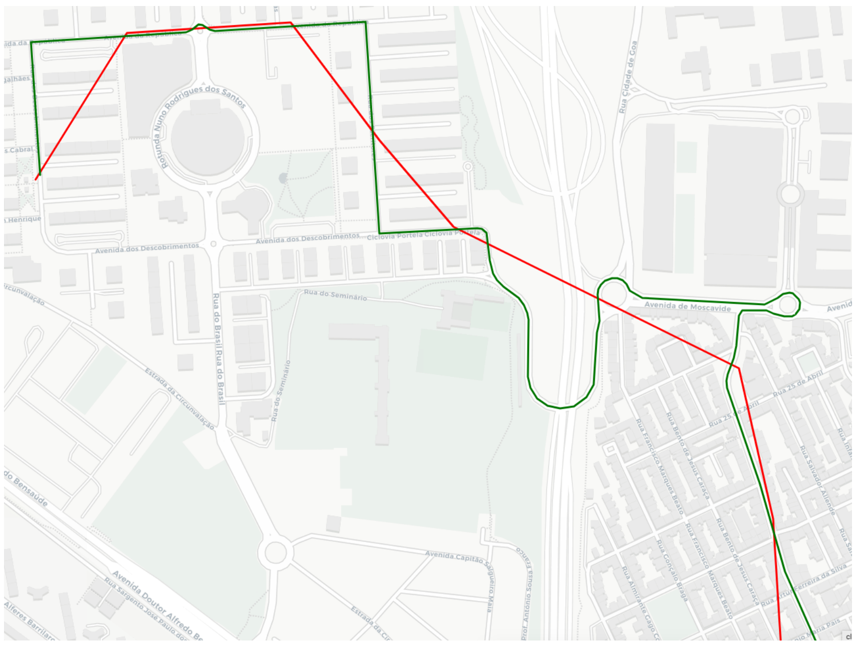


Figure 4.5: Comparison between original GTFS shape (in red) and estimated shape (in green)

[1] https://github.com/ad-freiburg/pfaedle

4.2.1.2 Computation of the bus network graph

Now that we know the location of each stop in the road network, we need to compute the links between the stops.

A directed, weighted graph $G(N, E)$ was used, where the set of nodes N represent the bus stops, and each edge $(i, j) \in E$ is a road link that connects stops i and j. The weight of each edge, $w_{i,j}$, is the duration of the shortest path between stops i and j.

An edge from stop s_1 to stop s_2 exists if s_2 is a neighbour of s_1, i.e, **if the path between s_1 and s_2 doesn't include any other stop**. This leads to a simplified graph that, as we will see, is particularly useful in the optimisation process. The graph is built by sequentially computing the neighbours for each stop $s \in S$, where S is the set of all stops.

To compute the neighbouring stops of stop s, the methodology is as follows:

First, only the stops at a radius d_{max} of s are considered, formally: $C_s = dist(s, x) < d_{max}, \forall x \in S$ where C_s is the set of potential neighbours of s. In the figure below, stop s is marked in red. The remaining stops are marked in green. Only the stops inside the circle are considered as potential neighbours of s and constitute C_s. This is shown in Figure 4.6

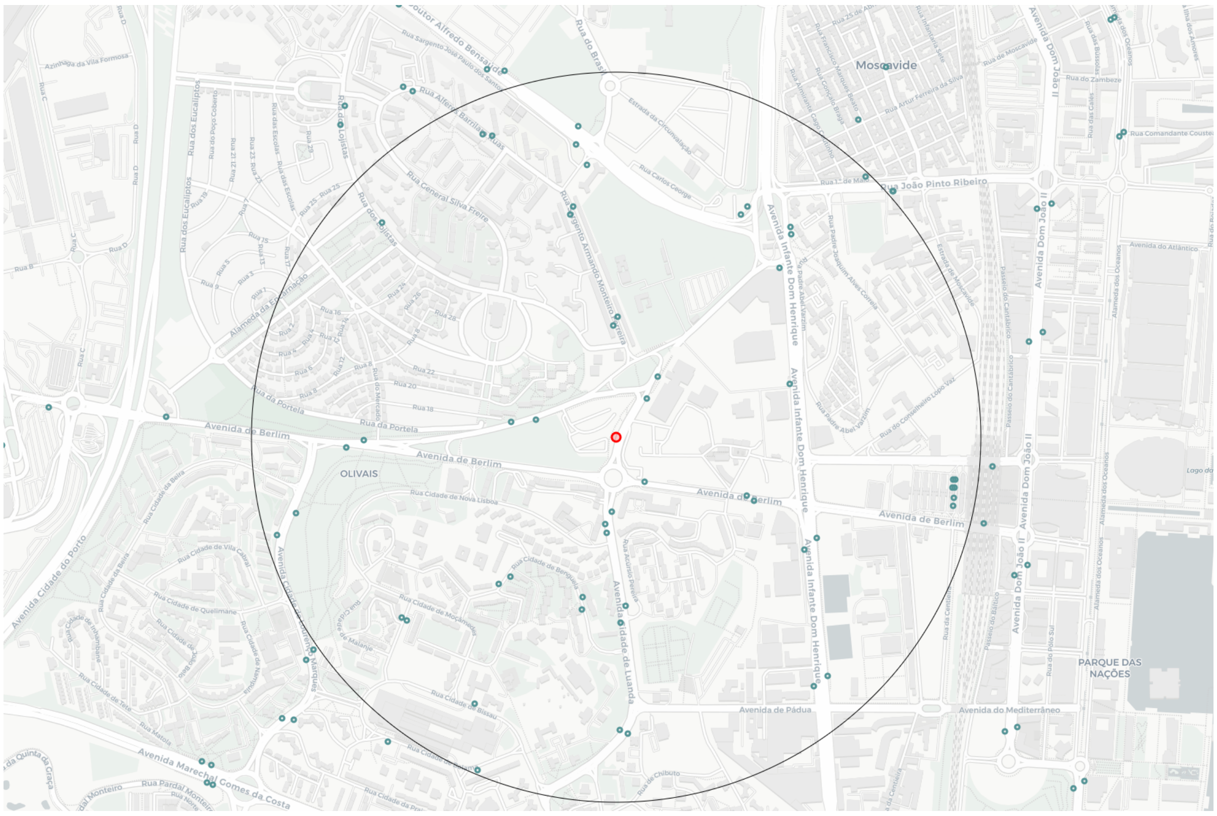


Figure 4.6: Circle of candidate neighbor stops (in green) for central stop (in red)

This upper bound is based on the assumption that two neighbour stops cannot be too far apart, and is used to reduce the computation time of the graph.

Now, with the set of **candidate** neighbours of stop s, we must compute the actual neighbours, i.e, the stops that are directly connected to s, without another stop in between. We find the neighbours of each stop **by relying only on the duration of shortest paths between stops**.

Each stop $z \in C_s$ is a neighbour of s if and only if there is no stop $y \in C_s$ that satisfies the following condition:

$d_{sy} + d_{yz} = d_{sz}, \forall y \in C_s$

where d_{ij} is the duration of the shortest path between i and j, calculated using OSRM.

If no such y exists, the shortest path between s and z doesn't pass through any other stop and, as such, s and z are neighbours. Therefore, an edge between s and z is added.

If such y exists, we know that the shortest path between s and z passes through y and, as such, and z are not neighbours.

Figure 4.7 illustrates this step.

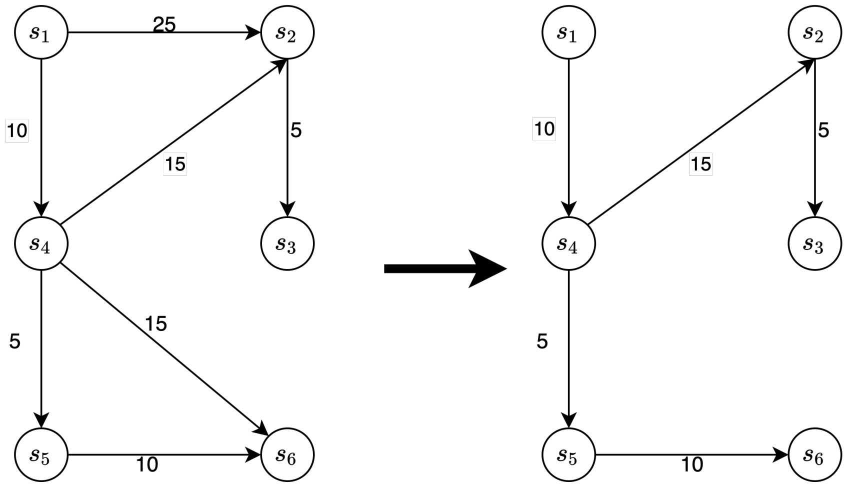


Figure 4.7: Example of the road network graph simplification step. The edge weights exemplify the duration of the shortest path between the edge stops

Since $d_{s_1 s_2} = d_{s_1 s_4} + d_{s_4 s_2}$, the edge from s_1 to s_2 is deleted.

Similarly, $d_{s_4 s_6} = d_{s_4 s_5} + d_{s_5 s_6}$, the edge from s_4 to s_6 is deleted.

While we used the path duration as edge weight, it would also be possible to do the same using the distance.

The reason why the path duration was used instead of the path distance is because at the time of this thesis, OSRM uses a mix of great circle distance and haversine distance to calculate distances in various points throughout the codebase. This lack of consistency leads to weird behavior, such as different path distances for the same origin and destination, depending on the API service used. [2]

Given that we would need to see if two different paths have the exact same distance, the distance calculation would need to be consistent.

Figure 4.8 shows the stop $<$*7714*$>$ *[Piscina Olivais]* (in red), its neighbours (in blue) and the path to the neighbours. The remaining stops (in grey) are part of C_{7714} but are not neighbors of stop *7714.*

4.2.1.3 Assessing correctness of estimated paths and stop locations

It is not straightforward to assess the correctness of the resulting road network graph. We can, however, rely on a few indicating factors that might help reveal errors in the solution.

We start by computing the distributions of the distances between the original GTFS stop locations and the locations computed by pfaedle. A high distance between the original GTFS location for a particular stop and the corresponding pfaedle snapped location reveals inconsistency in the data, which may lead to incorrectly mapped stops. This distribution is shown in Figure 4.9 and the cumulative frequencies in Table 4.1.

We also compare the durations of the paths computed by OSRM and the durations present in the GTFS feed. The bus GTFS feed has information regarding the durations of the path between consecutive

[2]`https://github.com/Project-osrm/osrm-backend/issues/5316`

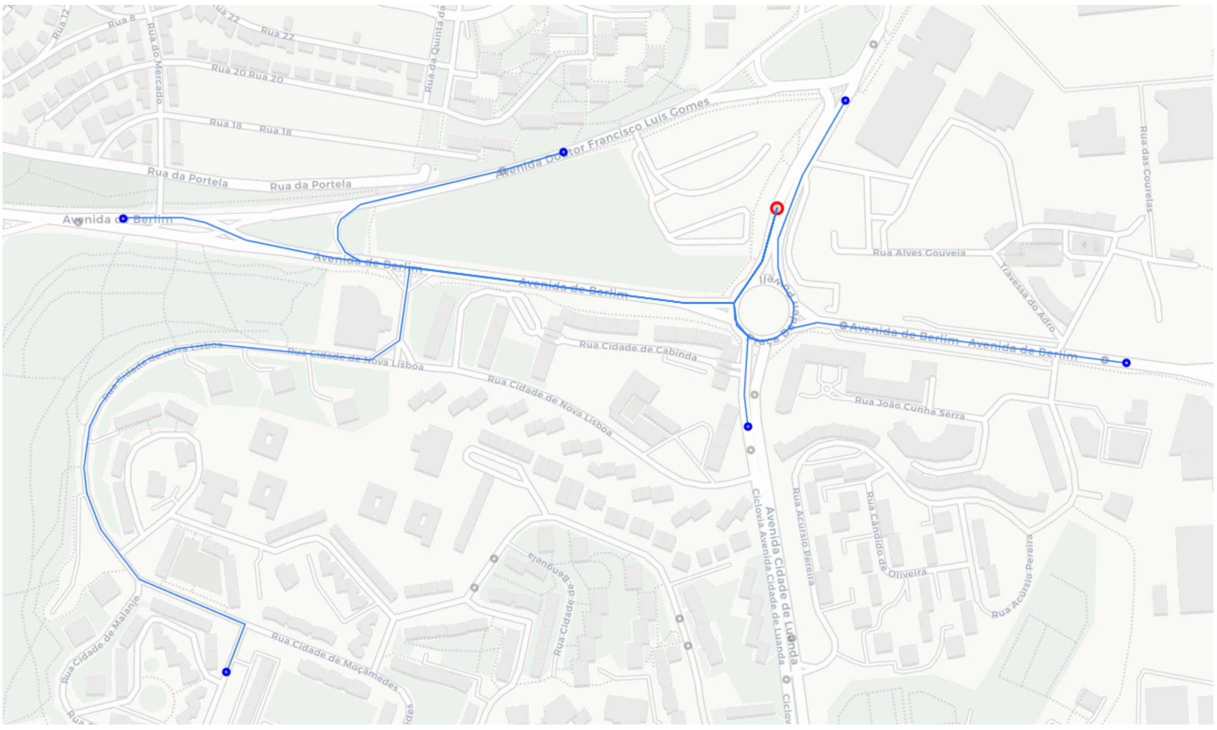


Figure 4.8: stop <*7714*> *[Piscina Olivais]* (in red), its neighbours (in blue) and the path to the neighbours. The remaining stops (in grey) are part of C_{7714} but are not neighbors of stop *7714.*

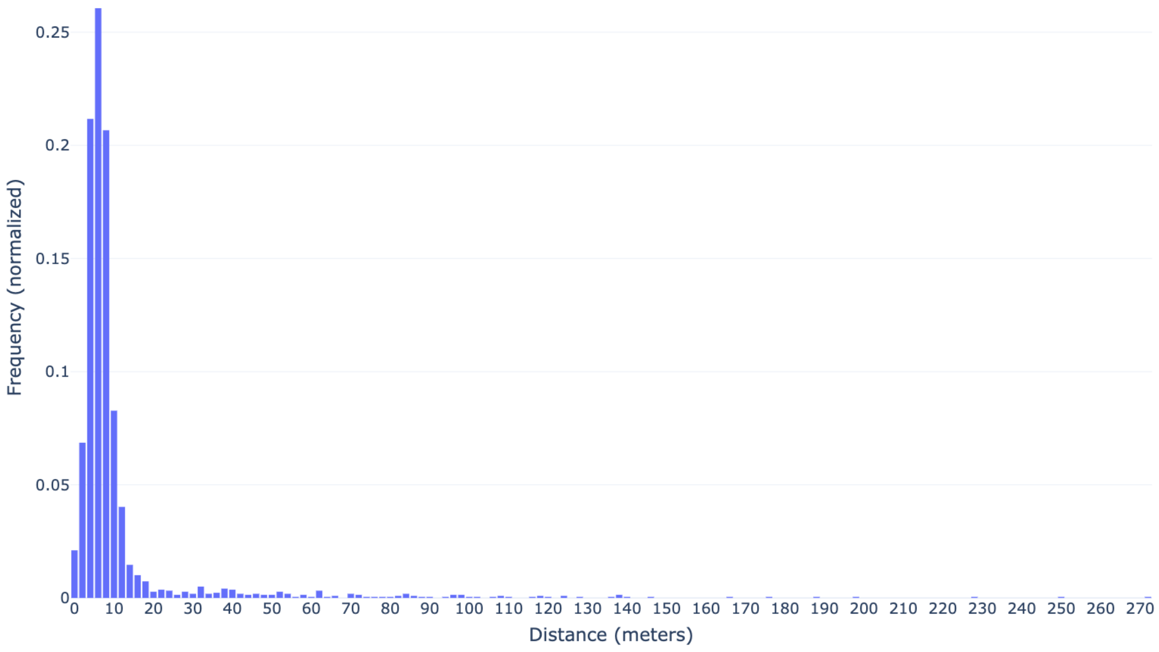


Figure 4.9: Histogram of distances between GTFS and estimated stop locations

Distance (meters)	Percentile
5	29.09%
8	66.07%
10	77.88%
20	88.28%
50	93.43%
100	97.58%

Table 4.1: Cumulative frequencies of distances between GTFS and estimated stop locations

stops in a bus route. For each of these consecutive stops, we compute the OSRM path duration between the street points of these stops. Paths for which the duration in OSRM is considerably bigger than the GTFS duration reveal cases where data errors exist [17].

Figure 4.10 shows such a case, where the path returned by OSRM is clearly incorrect. Figure 4.11 shows the connectivity issues in the graph that give rise to the erroneous path. The pink sections are OSRM detecting 'small components', i.e., parts of the graph that can't be entered or exited. OSRM then tries to snap the origin and destination locations to a component that both have in common. This is caused by a mapping error, shown in figure 4.12. The highlighted node was incorrectly mapped as a *kissing gate*, which breaks the connectivity of OSRM's graph.

Quantity	Mean	Median
GTFS Time	47.49	38.15
OSRM Time	46.90	36.48
GTFS-OSRM absolute difference	29.54	20.20
GTFS-OSRM percentage difference	84.07%	53.19%

Table 4.2: Cumulative frequencies of distances between GTFS and estimated stop street locations

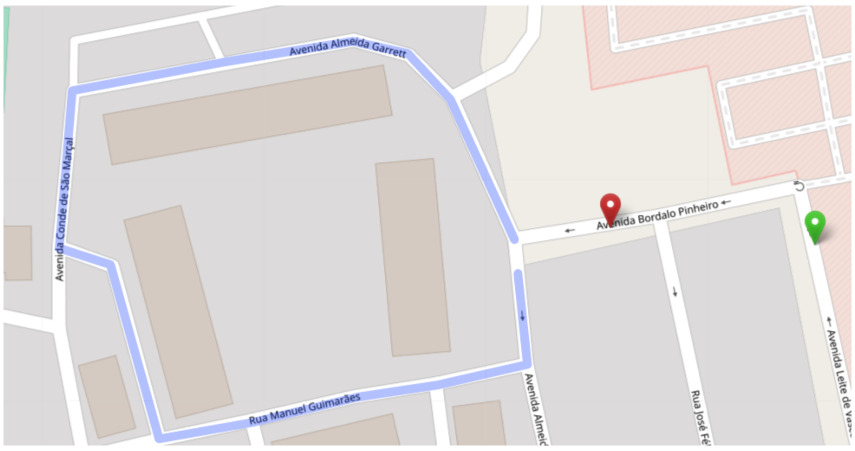


Figure 4.10: Path (in blue) computed by OSRM as shortest path between green and red marker

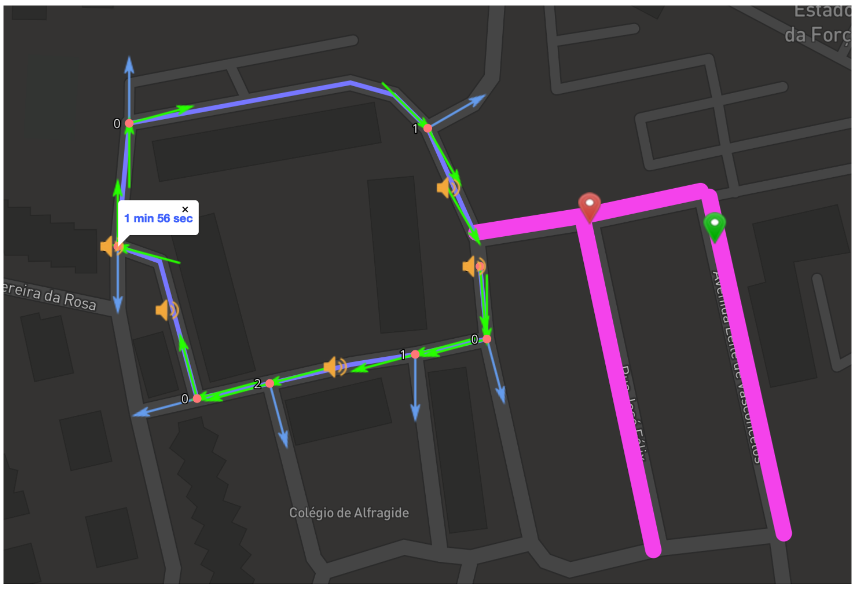


Figure 4.11: Connectivity issues in the OSRM graph. The pink sections represent small components, which cannot be entered or exited.

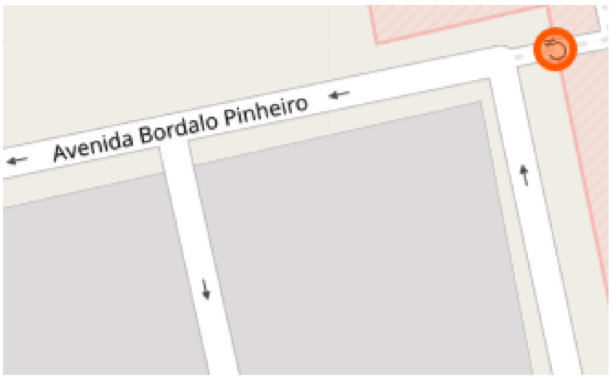


Figure 4.12: Incorrectly mapped gate.

4.2.2 Route Set Graph

Each solution to the TND problem is a route set. A route set is characterised by a list of routes, where each route serves a sequence of stops.

Figure 4.13 illustrates a route set, with 3 routes and 6 stops, where

- ***route 1*** serves stops **1**, **2** and **3**
- ***route 2*** serves stops **2**, **4** and **5**
- ***route 3*** serves stops **3**, **5** and **6**

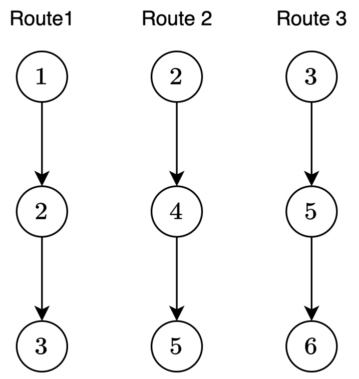


Figure 4.13: Simplified view of a route set with 3 routes and 6 stops

Given the nature of the problem, each route set is described by a weighted, directed graph $G = (N, E)$, where N is the set of nodes representing bus stops and E is the set of edges, representing the connections between stops.

The connection of stops in a route is represented by adding an edge between each consecutive stops served by each route. For example, for route 1 we will create 2 edges, one connecting stop 1 to stop 2 and another connecting stop 2 to stop 3. We will refer to these edges as route edges.

The weights of the route edges represent the duration of the path between the each consecutive stops. Thus, the weight of edge $e_{u,v}$, connecting stop u to stop v, is given by $d_{u,v} + b$ where $d_{u,v}$ is the duration of the shortest path between stops u and v, calculated using ***OSRM***, and b is a constant used to account for the time a bus takes to stop at a stop and wait for the passengers to alight or board the vehicle, and is set to *30 seconds*.

In a directed graph, node v is adjacent to u, if there is an edge leaving v and coming to u.

For each stop u, its adjacent stop v depends on the specific route. For example, in the example above, **stop 2 in route 1** has **stop 3** as the adjacent stop, while **stop 2 in route 1** has **stop 4.**

For this reason, **each stop must have one node per route that serves that stop.** We will refer to these nodes as the ***route nodes***.

In the example above, stop 2 will have 2 *route nodes*, one representing stop 2 in route 1 and another representing stop 2 in route 2.

To allow transfers between routes of the same stop, an edge is added between every *route node* of each stop. We will refer to these edges as *transfer edges.*

The edge weight of transfer edges is given by the parameter $transfer\_weight$ and represents the time a passenger takes to transfer from one route to another, and is set to *5 minutes.* This assumes that every bus in the system has a frequency of about 10 minutes.

When calculating the shortest path between stops, we must specify an origin node. In practice, we don't care about the origin route, but since each stop can potentially have more than one node, (if it is served by more than one route), we need an auxiliary node that serves as the origin for shortest path calculation. By adding one new node per stop, and adding an edge between that node and every route node of that stop with weight 0, we can calculate shortest paths between nodes without having to specify specific origin route. We will refer to this node as the *origin node.*

Since we also need to specify a destination node for shortest path calculation, and, again, we don't care about the route of the destination stop, we need another auxiliary node, that serves as the destination for shortest path calculations. Each route node connects to this destination node, with edge weight 0. We will refer to this node as the destination node.

The reason why we need an origin and a destination node instead of a single base node is: if we had a single base node, it would have to have an edge going to each route node (to serve as the origin) and an edge coming from each route node (to serve as the destination). Since these edge weights would have to be 0, this means that the transfers would be made through this base node, instead of being made through the transfer edge.

To summarise, each stop u will have $R_u + 2$ nodes, where R_u is the number of routes that service stop u and the other two nodes are the *origin* and *destination* nodes.

The described graph topology is shown in Figure 4.14

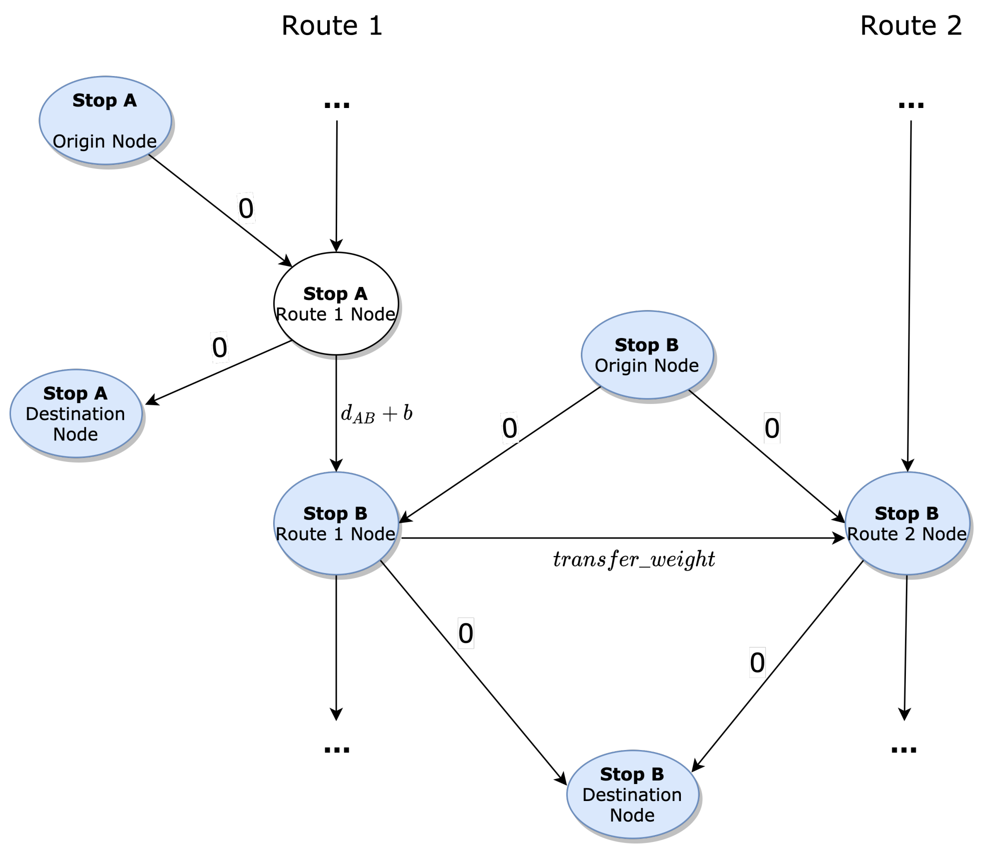


Figure 4.14: Route Set graph topology

4.3 Preprocessing

Before applying the optimization method, a pre-processing step was implemented, with the objective of reducing the number of stops, in order to simplify the problem.

Ignore stops not in OD matrix: Out of the 2193 stops in the GTFS dataset, only 1880 are present in OD matrix. Since they are not present in the AFC dataset, we assume they are inactive stops and, as such, are ignored.

Ignore tram-only stops: In the old part of the city, there are stops only served by tram, which are also operated by the bus operator. These stops and their OD matrix entries are ignored.

Stops clustering: In areas where there is a lot of demand, it is common for several stops, serving the same point of demand, to be very close together, with the purpose of avoiding a high concentration of people in the same physical stop.

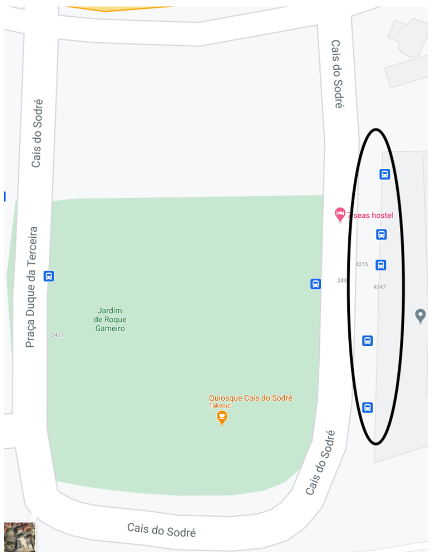


Figure 4.15: Example of a stop cluster

From a network design standpoint, these stops are one single stop, and the distribution of load into several physical stops is more of a logistics concern. By grouping these clusters of stops into a single stop, the number of stops is reduced and the problem is simplified, leading to a lower computation time.

For two stops u, v to be merged into a cluster, the following conditions must be met:

- there must exist an edge from u to v in the road graph
- the path distance between u and v, (computed using OSRM) must be less than 100 meters
- v cannot have any in-going edges, besides the one coming from u.

If the previous criteria are met, a cluster is formed, with the same location as stop u, and the OD matrix entries for stops u and v are merged. Figure 4.16 illustrates a possible cluster.

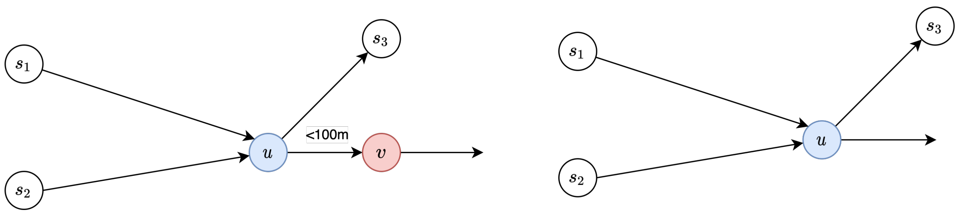


Figure 4.16: Possible stop cluster

As a result of the stop clustering step, 214 stops were merged, with a total of 107 clusters formed.

4.4 Genetic Algorithm

The genetic optimisation algorithm is adapted from the one described in [28].

Each gene is a route, and a set of routes make up an individual solution. The objective function attempts to minimize the average travel time experienced by each passenger, considering that the demand is described by the OD matrix computed in chapter 3.

4.4.1 Initial route set

In the first step of the genetic optimisation process, it is necessary to produce an initial solution (route set). Ideally, an optimisation algorithm should be able to find optimal route sets irrespective of the initial set of routes.

However, given the complexity of the route design problem, a good initialisation is very important for an efficient procedure [28, 31, 51]. The methodology for generating the initial solution is the same as [28, 51], which is based on simple, yet logical guidelines.

In the initial solution, one single route set (consisting of a predefined number of routes, R) is obtained. Each of the R routes is a shortest path (based on travel time) between a selected pair of stops. Using a greedy algorithm, we select the R pairs (i, j) that have the highest values of ds_{ij}, where ds_{ij} is the number of passengers that enjoy direct services along the shortest path between stops i, j. Formally: $ds_{ij} = \sum_{m \in S} \sum_{n \in S} d_{mn}$ where S is the set of stops that are in the shortest path between i and j and d_{mn} is the number of entries for pair (m, n) in the OD matrix.

The greedy algorithm is summarised:

1. Decide the total number of routes, R, in the solution.
2. Initialise matrix DS, where $DS = \{ds_{ij} \mid i, j \epsilon [0, 1, 2, \dots, |N-1|]\}$
3. Find the pair (i, j) in DS with the highest ds_{ij} value.
4. i and j become the terminals of a new route.
5. Add every node in the shortest path between i and j to the route.
6. Remove every node pair (m, n) that are satisfied by the newly added route from DS
7. If the number of routes reaches N, stop. Otherwise go to Step 3

This simple and deterministic approach is shown to outperform previous methods [28, 51].

[28] shows that, despite being contrary to the natural intuition, the use of identical individuals (route sets) in the initial population rather than a diverse random population works better.

4.4.2 Objective Function

The objective function, used to describe the quality of a solution (route set) reflects the total travel time spent by passengers in the transit service and is given by T:

$$T = TT + w_1 TTR + w_2 TU,$$

where TT is the total in-vehicle time of all served passengers, TTR is the total number of transfers in the network and TU is the total number of unsatisfied passengers.

We define unsatisfied passengers as those that, given their OD pair:

- The origin and/or destination stops are not part of the network
- There is no path between the origin and destination stops

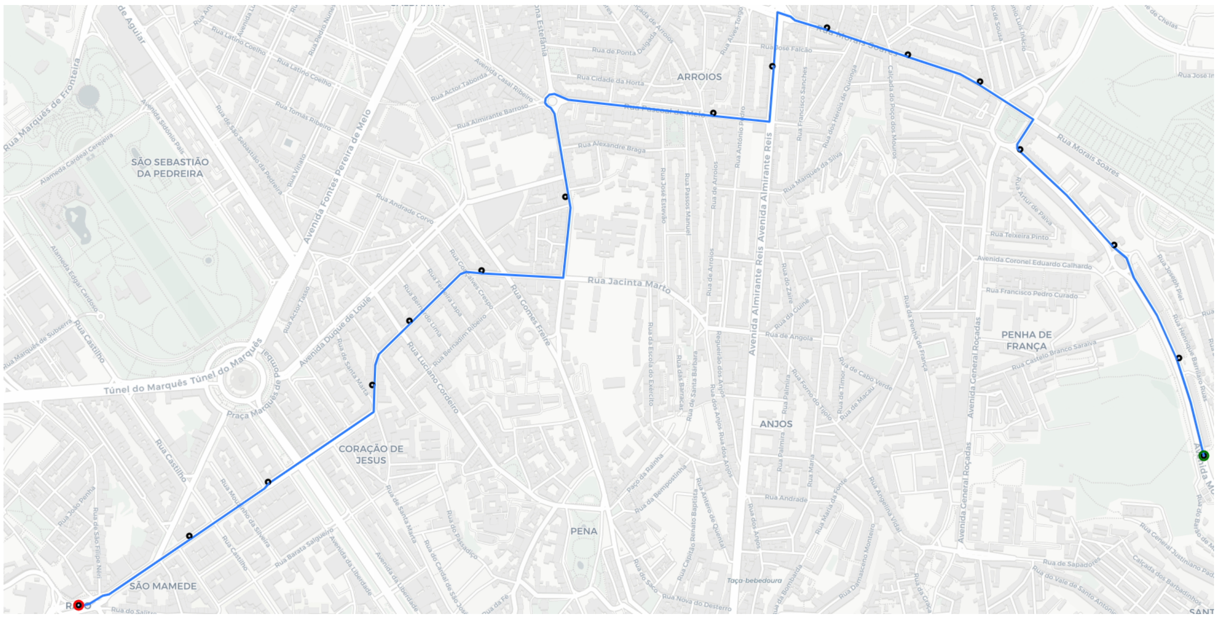


Figure 4.17: Example of a route in the initial route set, with the first stop in green, the last stop in red, and the intermediate stops in black

- There is a path between the origin and destination stops, but it requires more than 2 transfers.

w_1 and w_2 are configurable parameters and correspond to the time penalty for one transfer and the time penalty for one unsatisfied passenger, respectively.

Regarding passenger assignment, we make the simplification that each passenger chooses the path that minimizes travel time and, as such, Dijkstra's algorithm is used to compute the shortest path between each origin and destination, using the duration as the edge weight.

4.4.3 Selection and Crossover

In each generation, individuals are selected from the population to mate and generate offsprings for the next generation. Tournament Selection was used as the selection mechanism. The probability of selecting each individual i is given by

$$p_i = \frac{\Sigma_{j=1}^{N} f_j}{f_i},$$

where N is the population size and f_i is the value of the objective function of individual i.

The crossover operator used is known as uniform crossover, where, at each route index, the route of that position is swapped between the two selected parents with probability $p_{swap} = \frac{1}{R}$, where R is the number of routes. Figure 4.18 illustrates the crossover operator.

4.4.4 Mutation

The mutation operator is applied to the two offsprings that result from the crossover step. The mutation process is designed in a way that will select routes that fulfil less demand more often. Moreover, slight modifications will be made with a higher probability than big modifications, the latter being useful to escape from local optima.

The mutation process is as follows:

1. One route in the individual is selected for mutation with probability p_l.

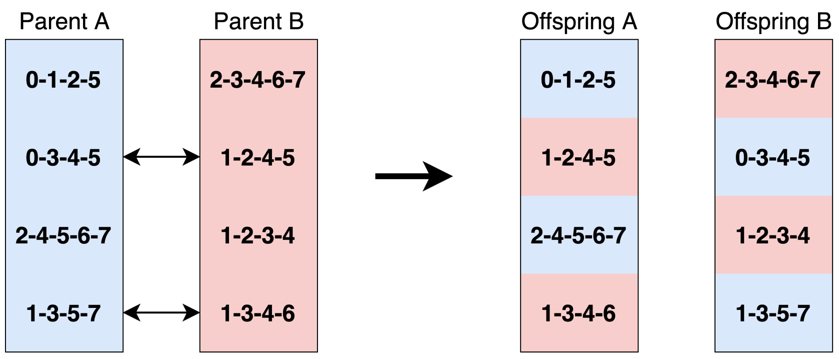


Figure 4.18: Cross operator. The routes at indexes 1 and 3 get swapped between the two parents

The probability of selecting route l is calculated as:

$$p_l = \frac{\frac{1}{ds_{ij}}}{\sum_{q \in L} \frac{1}{ds_{rs}}}$$

where i and j are the terminals of route l, L is the route set, r and s are the terminals of route q, and ds_{ij} is the total number of entries in the OD matrix that are satisfied, without any transfer, by using route l, that connects terminals i and j.

We can see from the equation above that routes that fulfil less demand have a higher probability of being selected for mutation.

2. After selecting the route, we apply a small modification with a high probability p_{ms} and apply a big modification with a low probability $1 - p_{ms}$.

   - ***Small modification:*** selects one of the route terminals with the same (0.5) probability and applies one of two operators:
     - **deletes** the selected route terminal, with probability p_{delete} (Figure 4.19).

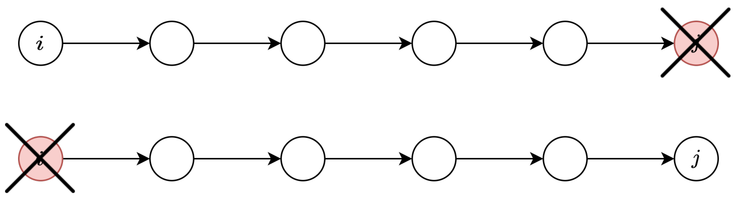


Figure 4.19: Mutation operator - Delete

     - **extends** the selected route, adding a new stop in the selected route terminal, with probability $1 - p_{delete}$
       The new stop is chosen at random from among the adjacent stops to the chosen terminal, in the road network graph.
       If the selected route terminal is the route's first stop, the new stop is prepended to the route. If, on the other hand, the selected terminal is the route's final stop, the new stop is appended to the route. (Figure 4.20)
   - ***Big modification:*** Selects one of the route terminals (say terminal i) with the same (0.5) probability, and then selects a new terminal k, with the following probability:

$$P_k = \frac{ds_{ik}}{\sum_{r \epsilon N} ds_{ir}}$$

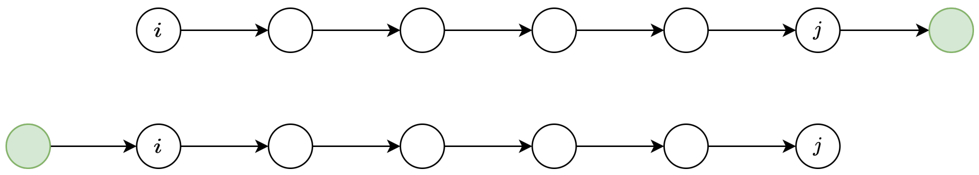


Figure 4.20: Mutation operator - Extend

The selected route will be replaced by a new route that is the shortest path between terminal i and terminal k. A small modification was done to the mutation operator, when compared to [28]. When $\sum_{r \epsilon N} ds_{ir} = 0$ and $\sum_{r \epsilon N} ds_{jr} = 0$, where i, j are the route terminals, both terminals i, j are deleted from the route. (Figure 4.21).

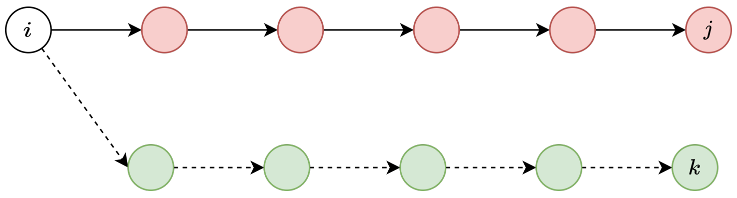


Figure 4.21: Big modification operator, with the old route in red and the new route in green

4.4.5 Elitism

When constructing a new population, the $elite\_size$ best route sets from the current generation carry over to the next, unaltered. This strategy is known as elitist selection and guarantees that the solution quality obtained by the genetic algorithm will not decrease from one generation to the next.

4.5 Summary

This chapter described the problem formulation and GA methods used to optimize the bus network in Lisbon.

Two representations necessary to the the optimization process were developed: a graph representation that describes the stop street locations and the links in the underlying road network that connect the stops, computed using real road data from OSM; a graph representation that describes each route set, capable of efficiently computing the shortest path and duration that satisfies each OD pair, necessary to calculate the objective function value of each solution.

An initial pre-processing step was also implemented, where clusters of stops are computed in order to reduce the complexity of the problem.

For the optimization procedure, a simple genetic algorithm based on the work done by [4] and [28] was implemented.

Chapter 5

Results

This chapter presents the results of applying the described optimization methodology to the bus network in Lisbon. Every experiment was ran on a i5 2500k processor, with 4 tests being run in parallel.

We present the experiments conducted and the metrics evaluated.

5.1 Evaluation and Methodology

Two major tests were conducted. In the first test, the routes were generated from scratch, using the initialization method described in Section 4.4.1. Several values for the number of routes R were tested.

In the second test, the existing (real) bus network was used as a starting point to the optimization process, serving as the initial solution.

Aside from the number of routes R, which is only applicable in the first test, several combinations of values for the following parameters (described in Section 4.4) were used in both tests:

- **Population size** (pop_size)
- **Elite size** ($elite\_size$)
- **Tournament size** (t)
- **Probability of a small mutation** (p_{ms})
- **Probability of a delete mutation** (p_{delete})

To evaluate the results, the following metrics were used:

- **Objective Function**: The value of the objective function (described in Section 4.4.2) being minimized by the genetic algorithm.
- **Satisfied OD pairs**: The percentage of OD pairs (present in the OD matrix) that are satisfied by the network, with 2 transfers or less. All OD pairs have equal weight, regardless of the number of entries in the OD matrix
- **Satisfied Demand**: The percentage of demand satisfied by the network, with 2 transfers or less. Similar to the parameter above, but the weight of each OD pair is proportional to the number of entries in the OD matrix, i.e, OD pairs with a bigger flow of passengers will have a bigger weight.
- **Satisfied stops**: The percentage of stops present in the route set. Only stops in the OD matrix are considered.

- **Average travel time**: The average travel time across all passengers. This includes the in-vehicle time and the transfer time (assumed to be 5 minutes).
- **Average number of transfers**: The average number of transfers needed to complete each passenger trip.

Since the computation time of each test depend on the parameters used, the number of iterations varies between the tests.

5.2 Optimization from scratch

The following values for the number of routes R were tested: $20, 50, 100, 200, 300, 400$. Table 5.1 shows the initial metric values, after generating each initial route set, before doing any further optimization.

For each of the values of R, several tests with different values for the genetic parameters described above were done. For each value of R, the solution with the lower final objective function value was selected.

Table 5.2 shows the final metric values, for the best performing experiment of each value of R.

Table 5.3 shows the parameters that lead to the best result for each of the values of R.

Figures 5.1–5.6 show the metric values for each of these best-performing solutions.

R	Objective function	Satisfied OD (%)	Satisfied demand (%)	Satisfied stops (%)	Average travel time (min)	Average transfers
20	$5.99e+08$	6.87	16.52	17.00	15.83	0.07
50	$6.91e+08$	14.53	30.40	32.07	33.74	0.74
100	$6.53e+08$	24.69	45.25	46.57	39.69	1.00
200	$4.75e+08$	40.17	63.33	61.17	32.79	0.82
300	$4.02e+08$	50.17	72.99	69.30	31.09	0.83
400	$3.46e+08$	58.02	79.01	75.21	28.70	0.74

Table 5.1: Initial metric values for each value of R

R	Objective function	Satisfied OD (%)	Satisfied demand (%)	Satisfied stops (%)	Average travel time (min)	Average transfers
20	$5.25e+08$	6.03	17.43	32.66	8.64	0.05
50	$4.57e+08$	23.28	43.05	57.71	16.26	0.26
100	$3.31e+08$	54.73	74.19	78.23	21.98	0.59
200	$2.17e+08$	82.57	92.88	85.50	22.00	0.57
300	$1.99e+08$	86.93	95.42	88.02	21.74	0.57
400	$1.96e+08$	87.41	95.86	89.55	21.73	0.54

Table 5.2: Final metric values for each value of R

R	pop_size	$elite\_size$	t	p_{ms}	p_{delete}
20	16	4	10	0.7	0.4
50	16	4	10	0.7	0.4
100	16	4	10	0.7	0.4
200	64	16	40	0.7	0.6
300	64	16	40	0.7	0.6
400	64	4	40	0.7	0.6

Table 5.3: Best parameter for each value of R

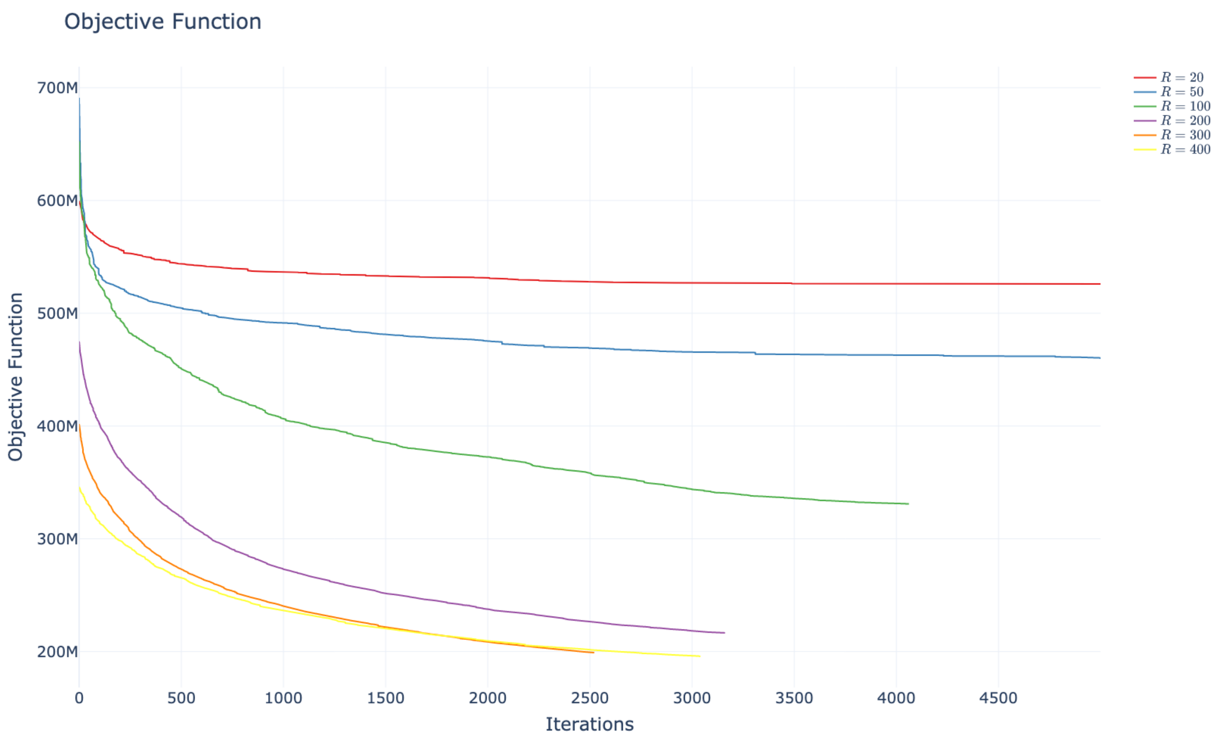


Figure 5.1: Objective function value for each value of R

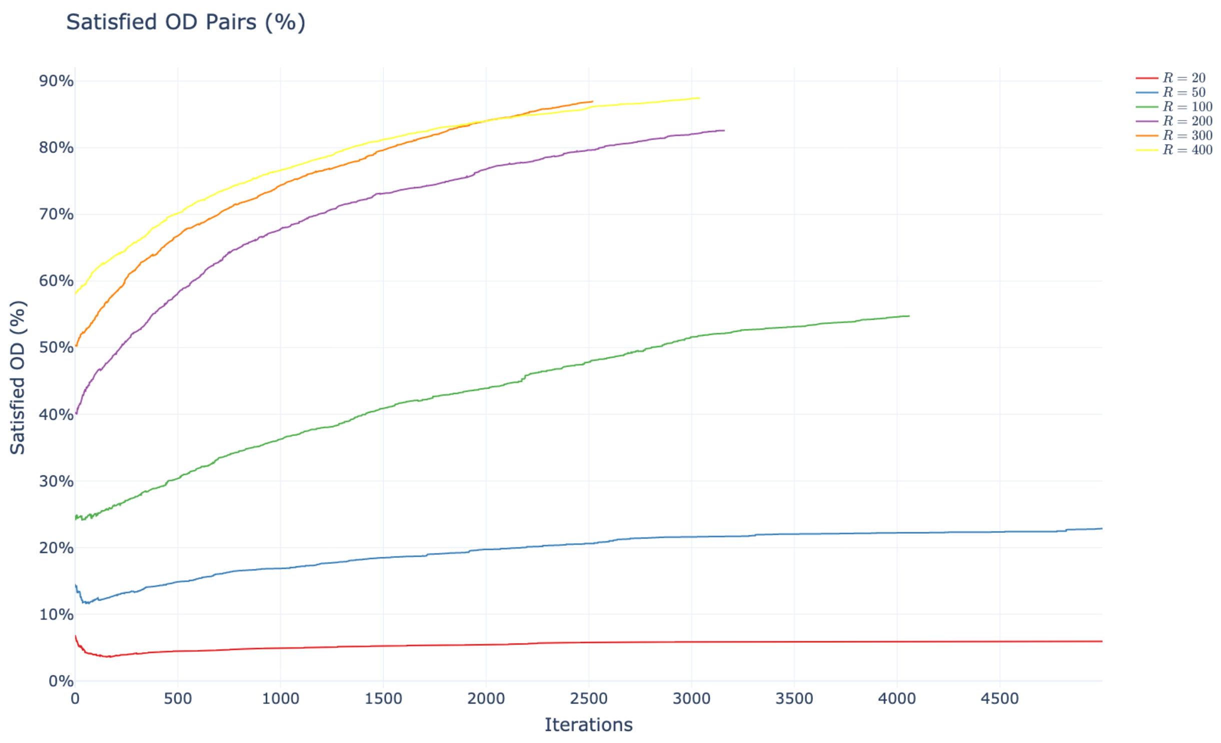


Figure 5.2: Percentage of satisfied OD pairs for each value of R

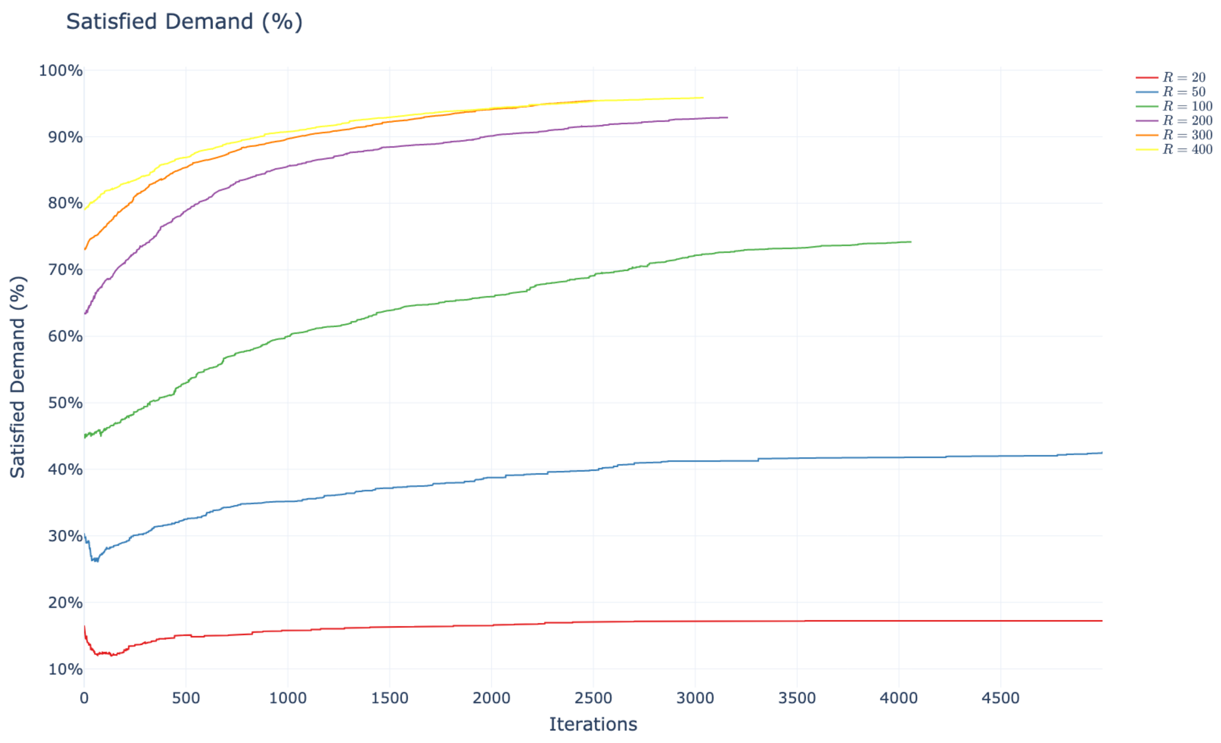


Figure 5.3: Percentage of satisfied demand for each value of R

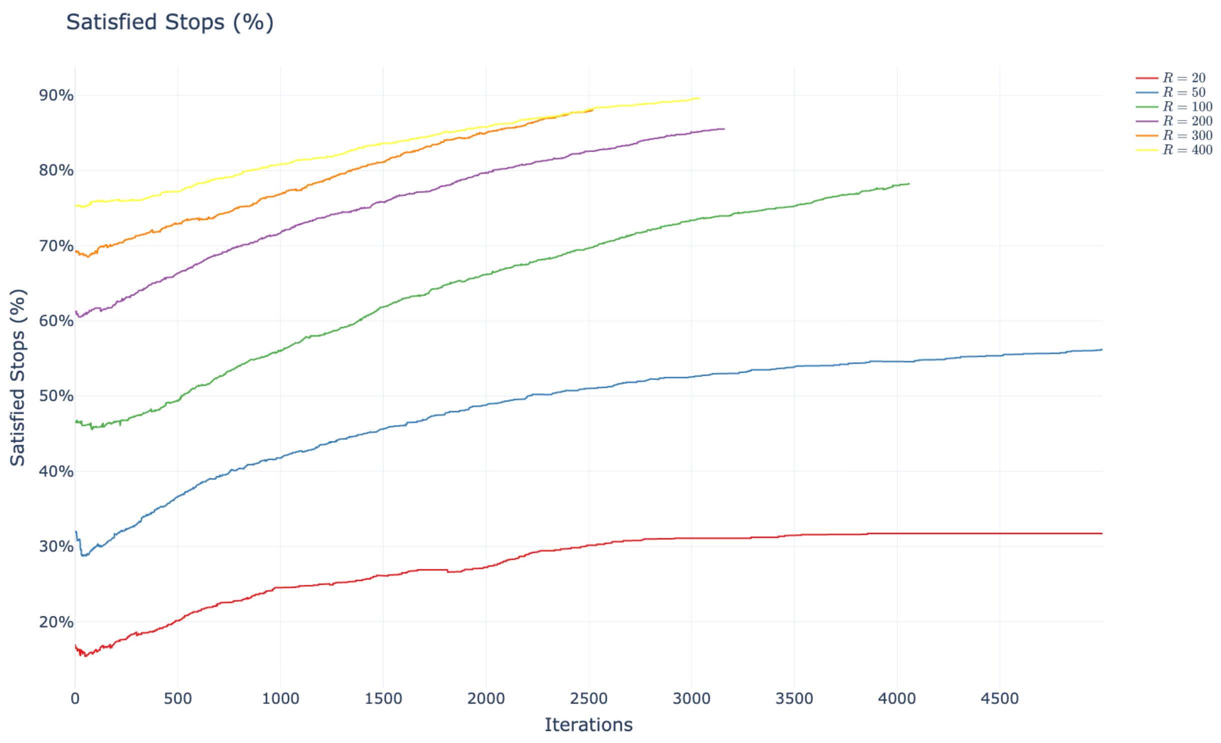


Figure 5.4: Percentage of covered stops for each value of R

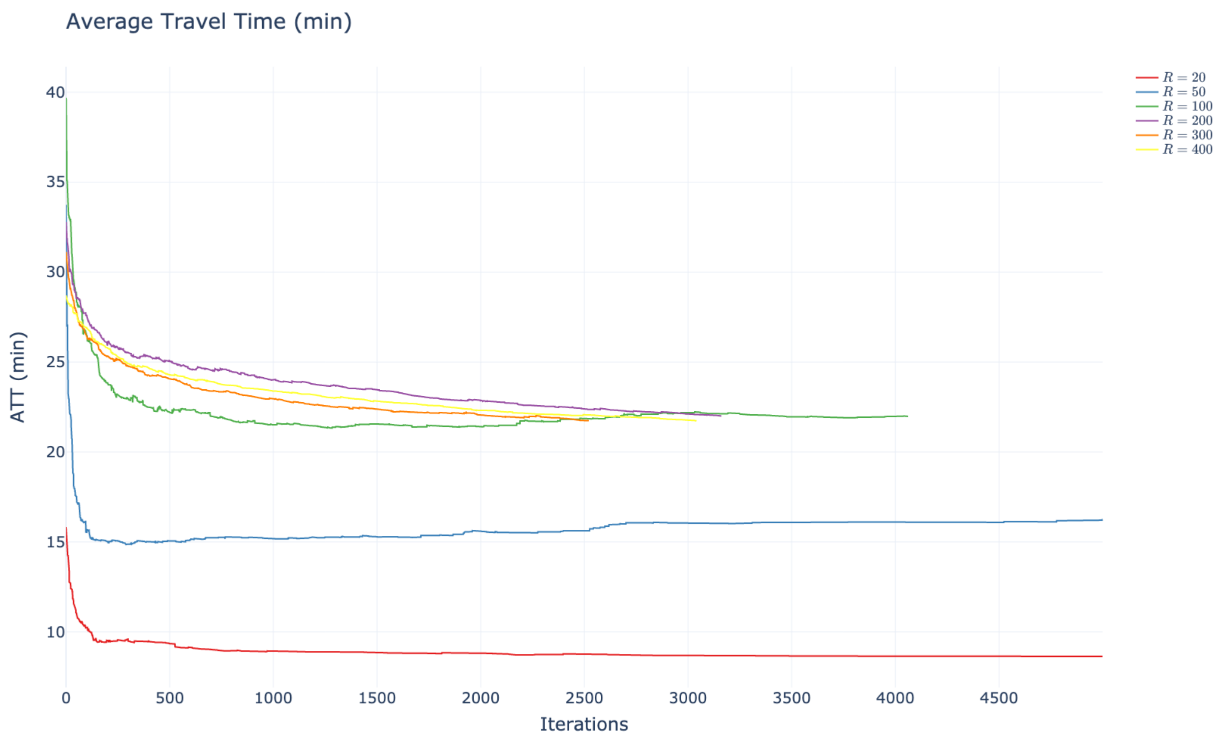


Figure 5.5: Average travel time (in minutes) for each value of R

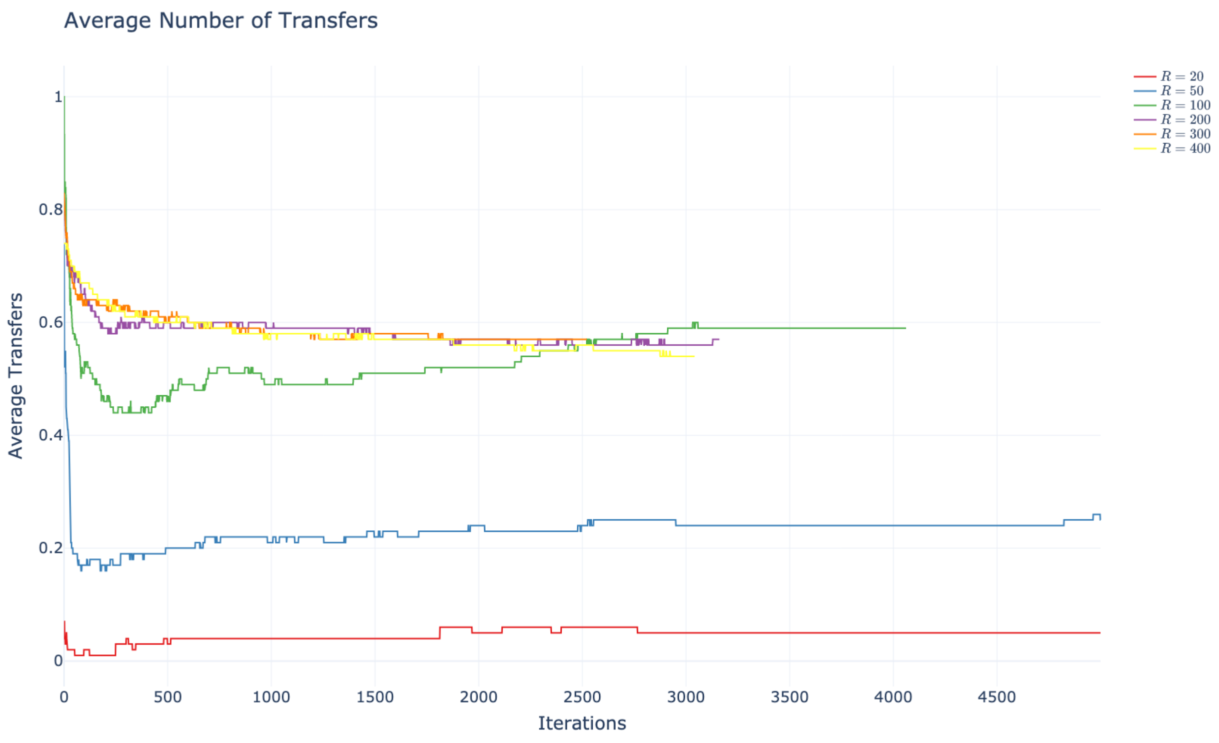


Figure 5.6: Average number of transfers for each value of R

The results show that the increase in each of the evaluated metrics is very small beyond 300 routes. This suggests that the added network complexity and potential operator costs outweigh the benefit of increasing the number of routes beyond 300.

This is consistent with the number of routes of the existing bus network (309).

Further testing should be conducted, with more values of R between 200 and 300. Furthermore, given the high computation time of each experiment and the limited amount of resources, the amount of tested genetic parameter combinations were not extensive enough to fully understand the impact of each parameter and find the optimal parameter combination.

Figures 5.7 and 5.8 show the initial route set and the final, optimized solution, for $R = 300$. We can see that the optimized network is much denser than the initial network. This was expectable, since the optimized network serves almost 20% more stops than the initial network, which requires a lot more connections.

The described experiments achieved a considerable increase in efficiency, when compared to their initial values. However, the final metric values of the best experiment are still below those of the existing (real) bus network.

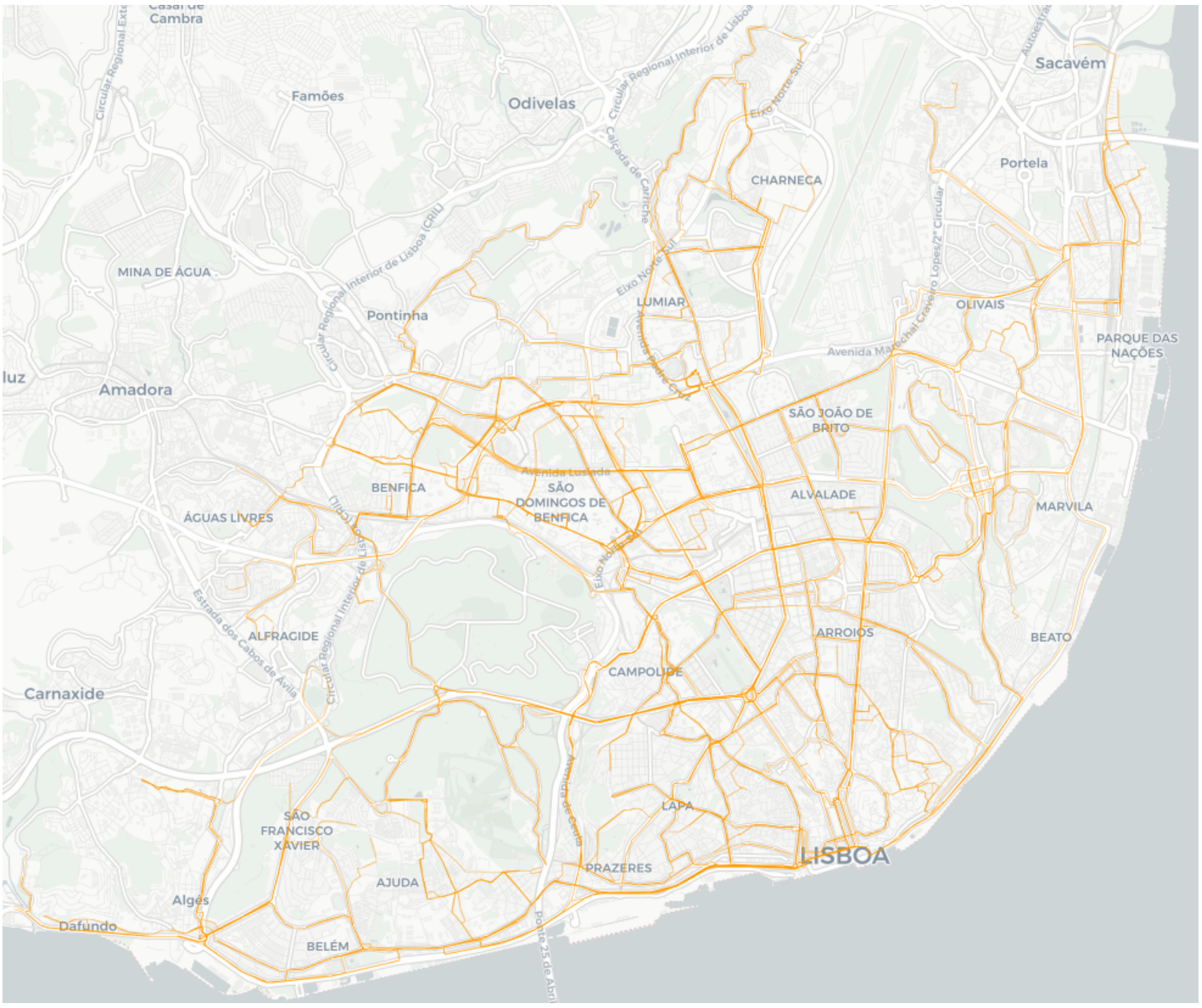


Figure 5.7: Initial route set for $R = 300$

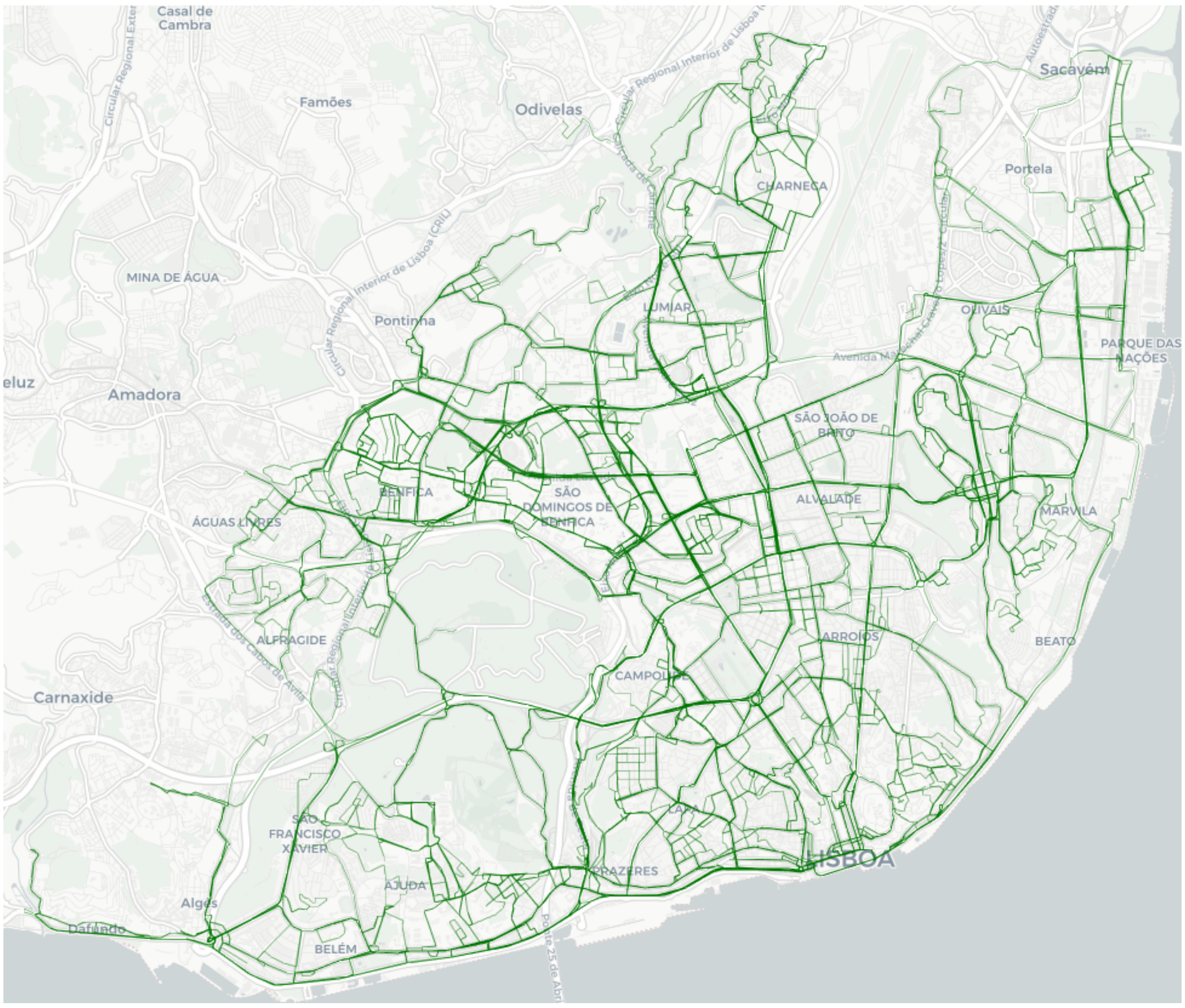


Figure 5.8: Final optimized route set for $R = 300$

5.3 Optimizing the existing network

The optimisation algorithm was ran using the existing bus network that was in service on October 2019, (the same date as the AFC data), as the initial route set solution.

The real route set is comprised of 309 routes, 16 of which are circular routes. A circular route serves stops $s_0, \ldots, s_{N-1}, s_0$, where N is the number of stops in the route. After reaching the last stop s_{N-1}, the route goes to the starting stop s_0.

Since our problem representation does not support circular routes, every circular route was replaced by its linear counterpart. A circular route $s_0, \ldots, s_{N-1}, s_0$ becomes $s_0, \ldots, s_{N-1}$.

It is clear that some pairs that were satisfied by the circular route will not be satisfied by the linear route. For this reason, every OD pair that was satisfied by the circular route but not its linear counterpart, formally $s_i, s_j \forall j < N, i < N, j < i$, was not considered in the results assessment. From this step, 329 OD pairs were discarded (out of 26438).

Table 5.4 shows the initial values of each of the metrics, for the existing network (before any optimization was applied).

Table 5.5 shows the several experiments that were conducted (each with different parameter values). Since the computation time is dependent on the values of the parameters, the number of iterations differ between the several experiments

Figures 5.9–5.14 show the value of each of the metrics over the several generations, for each of the experiments enumerated in Table 5.5.

Objective function	Satisfied OD (%)	Satisfied demand (%)	Satisfied stops (%)	Average travel time (min)	Average transfers
$2.00e+08$	96.25	98.96	100.00	24.91	0.15

Table 5.4: Initial metric values for existing network

Experiment	pop_size	$elite\_size$	t	p_{ms}	p_{delete}
0	64	16	40	0.3	0.6
1	64	16	40	0.3	0.9
2	64	16	40	0.7	0.6
3	64	32	40	0.7	0.6
4	128	16	40	0.7	0.6

Table 5.5: Description of experiments

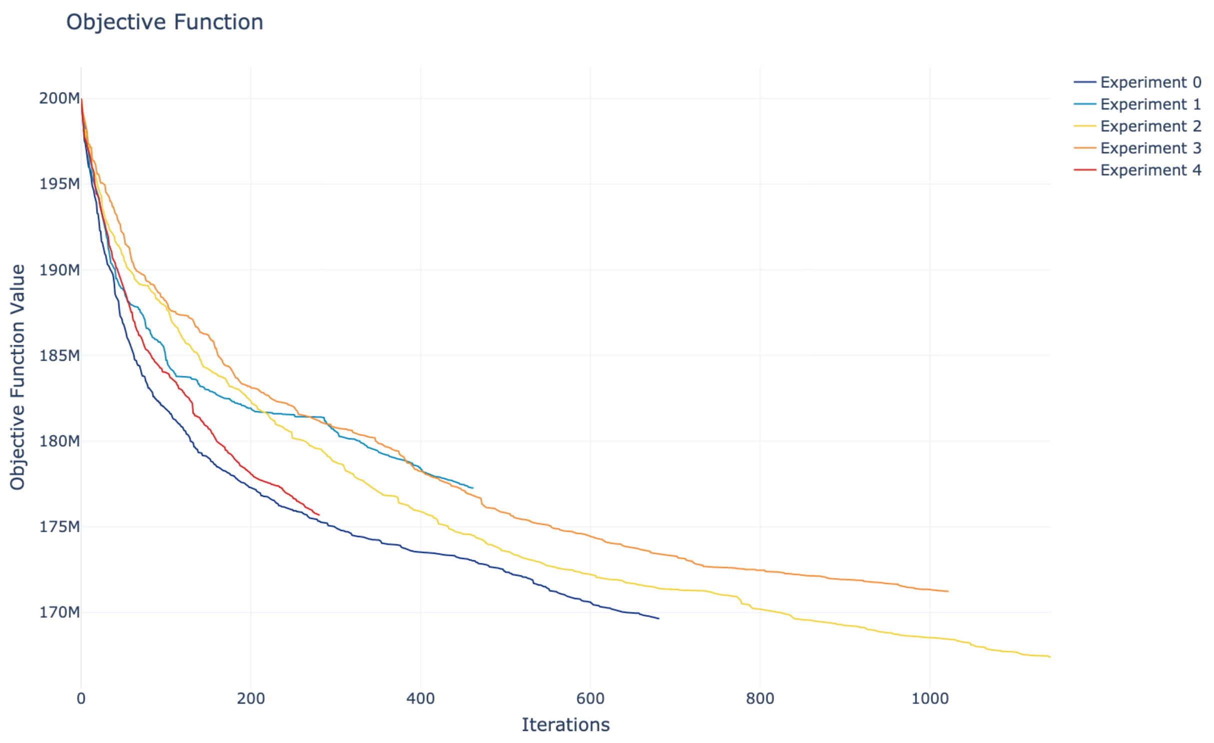


Figure 5.9: Objective function values for the optimization of the existing network

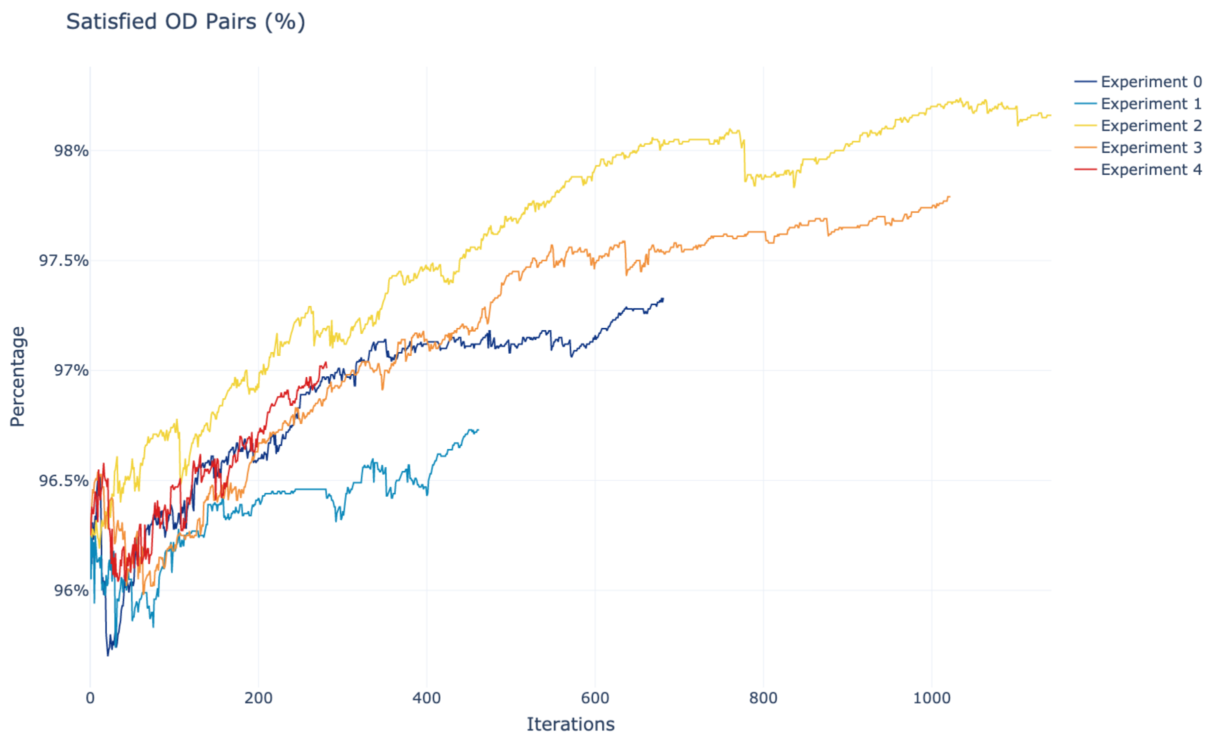


Figure 5.10: Percentage of satisfied OD pairs for the optimization of the existing network

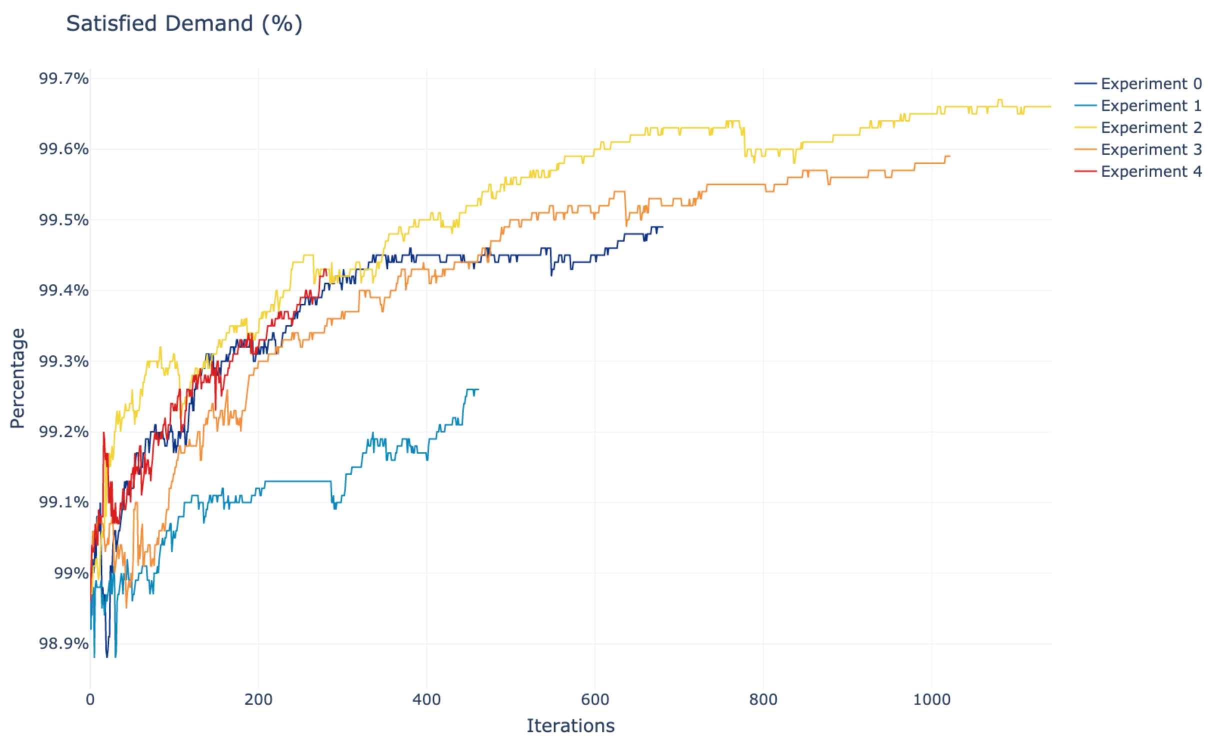


Figure 5.11: Percentage of satisfied demand for the optimization of the existing network

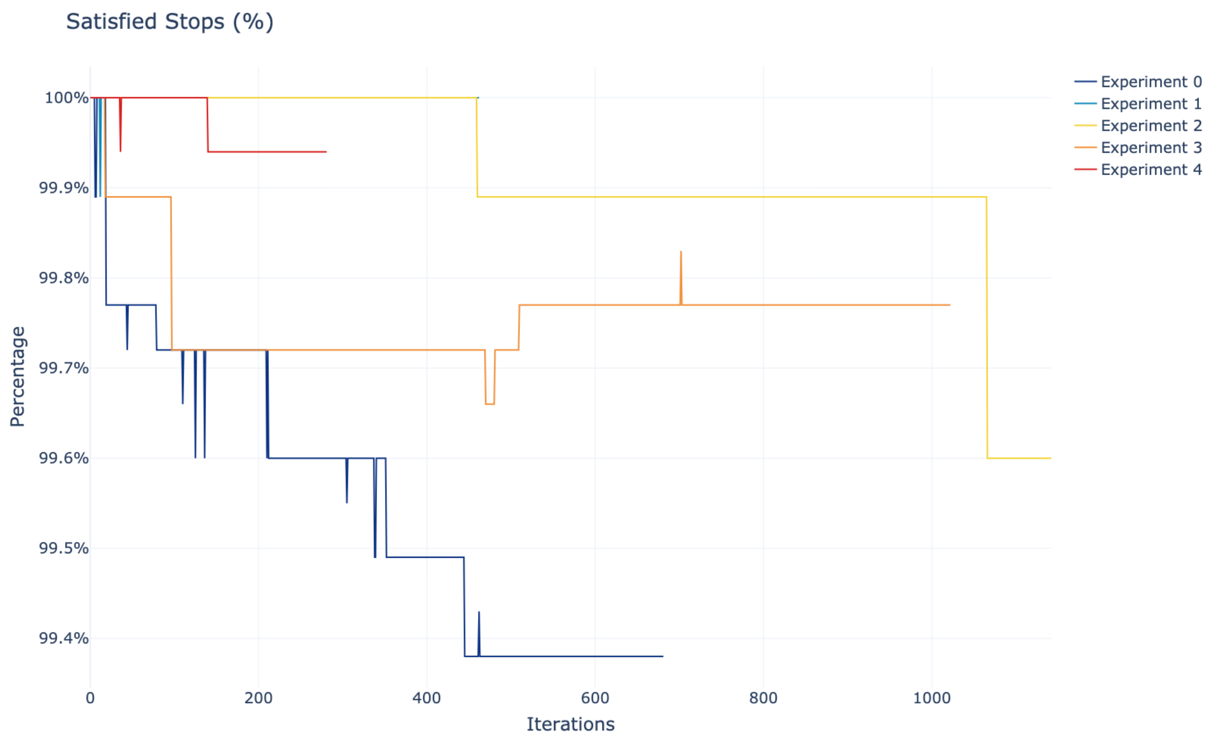


Figure 5.12: Percentage of satisfied stops for the optimization of the existing network

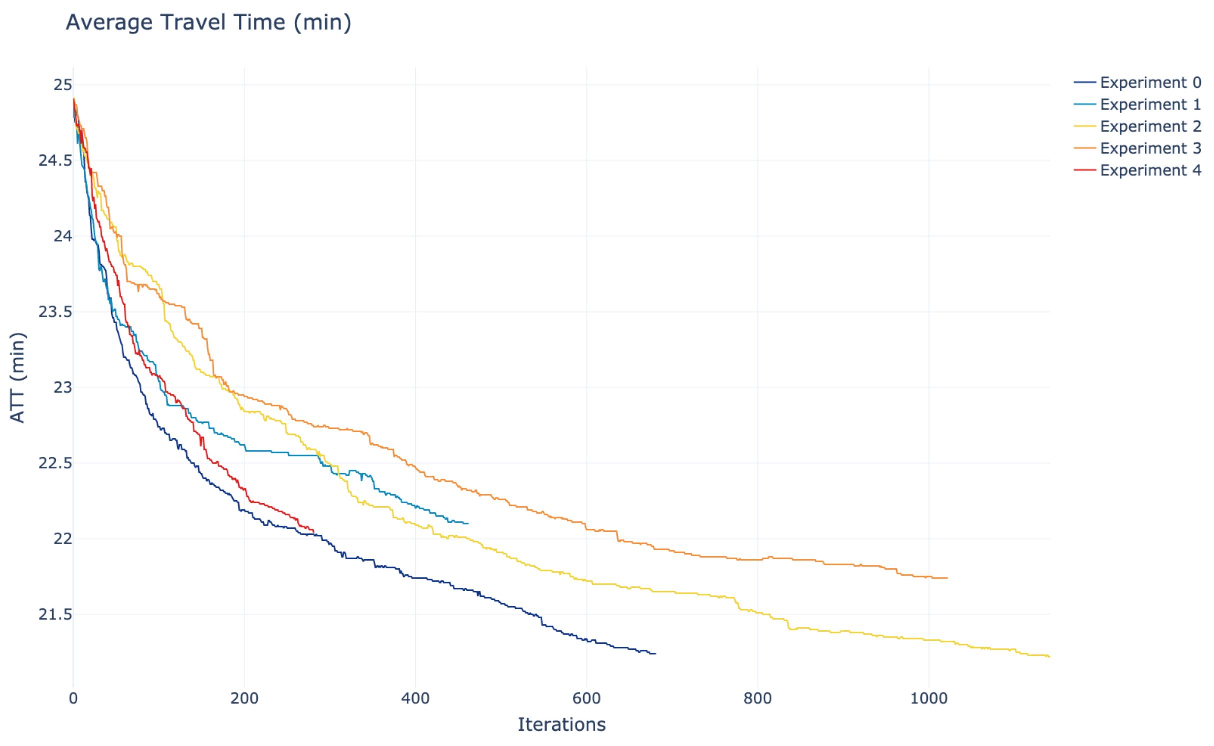


Figure 5.13: Average travel times, in minutes, for the optimization of the existing network

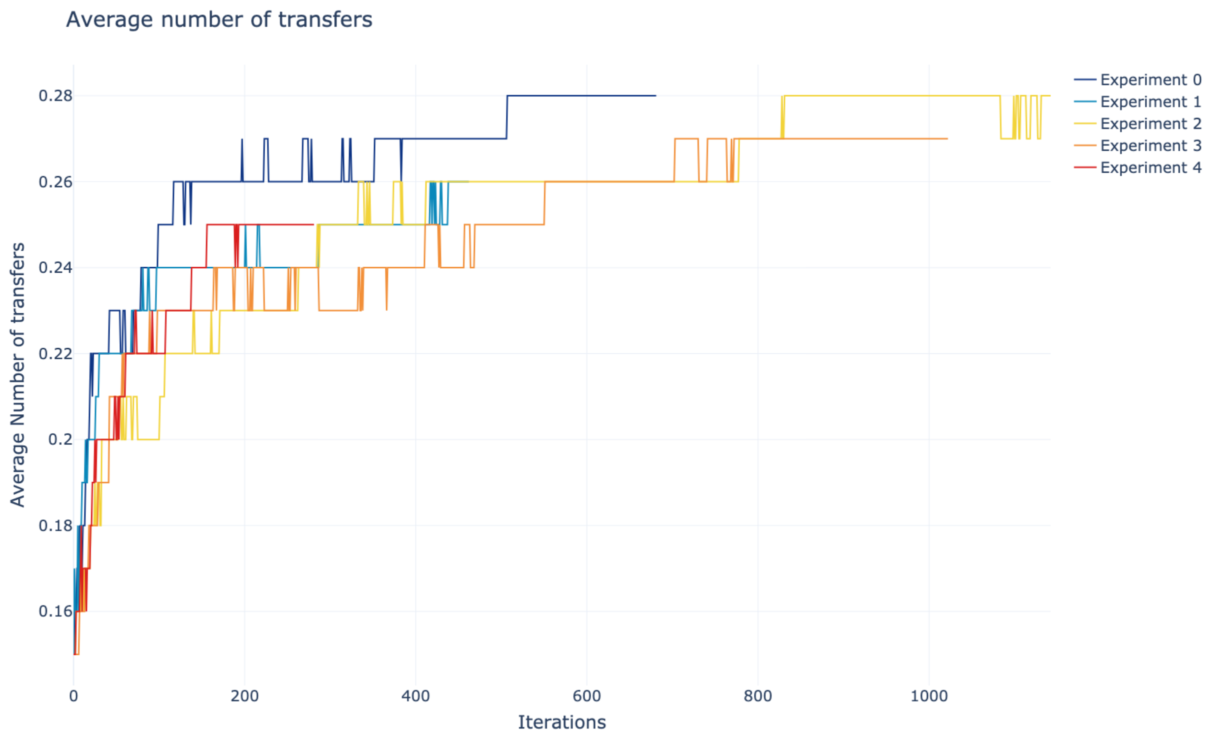


Figure 5.14: Average number of transfers for the optimization of the existing network

5.3.1 Comparison to existing network

Experiment 3, having the lowest objective function score, is considered to be the best performing experiment. Table 5.6 shows the comparison between the final values of Experiment 3 and the existing (unoptimized) network).

Figure 5.15 shows the distribution of travel time differences between the optimized and existing network.

Figure 5.15 shows the distribution of the number of transfers difference between the optimized and existing network.

Measure	Initial Value	Final Value	Difference
Objective function	$2.00e + 08$	$1.66e + 08$	-17.23%
Satisfied OD (%)	96.25	97.84	$+1.59$
Satisfied demand (%)	98.96	99.60	$+0.64$
Satisfied stops (%)	100.00	99.20	-0.80
Average travel time (min)	24.91	20.85	-4.06
Average transfers	0.15	0.29	$+0.14$

Table 5.6: Initial, final and change in metric values between existing network and optimized Experiment 3

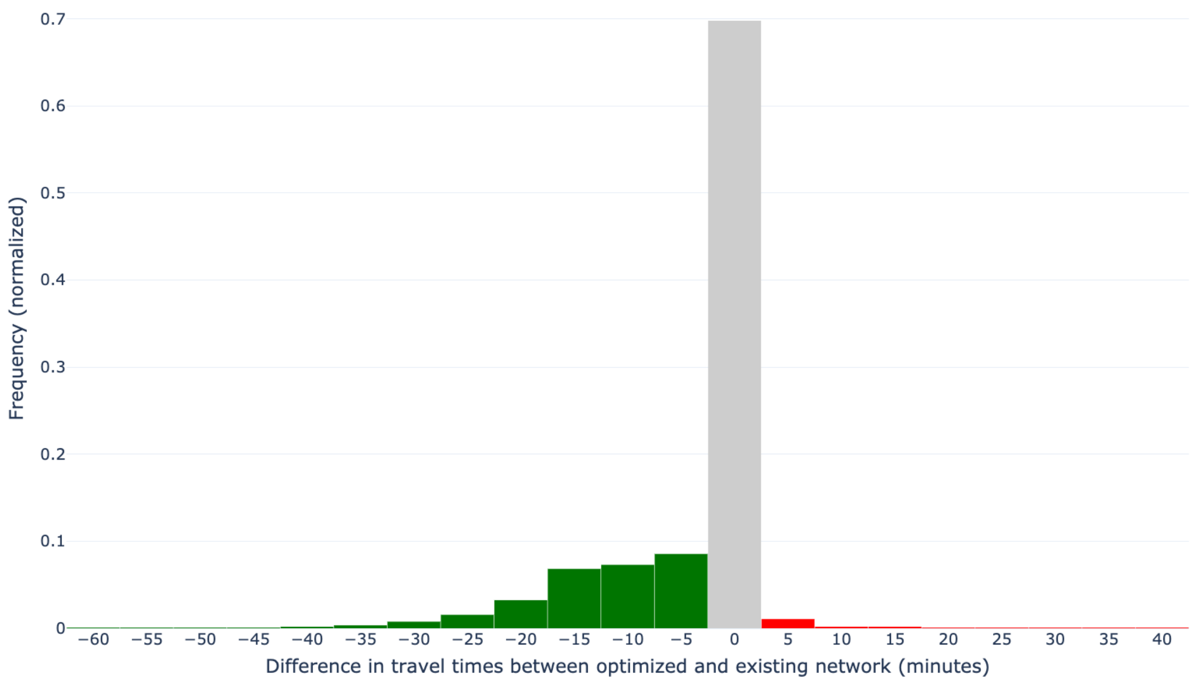


Figure 5.15: Difference in travel time between the optimized and existing networks (in minutes). The green bars correspond to a decrease in travel time in the optimized network. The gray bars correspond to a minimal (< 2.5 min) change in travel time and the red bars correspond to an increase in travel time.

The results show a significant improve in the optimized network, with a 17.23% decrease in the objective function value. Several trips have a decrease in travel time from 5 to 60 minutes (in green),

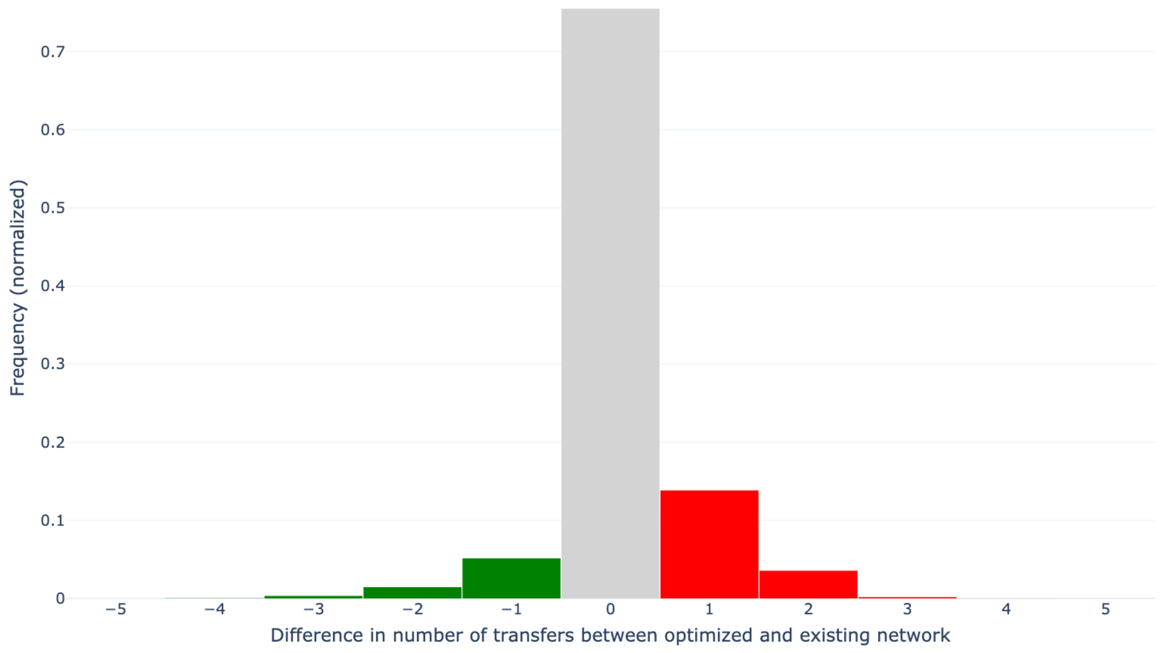


Figure 5.16: Difference in the number of transfers, between the optimized and existing networks (in minutes). The green bars correspond to a decrease in the number of transfers the optimized network, respectively. The gray bars correspond to a no change in number of transfers and the red bars correspond to an increase in the number of transfers

while a few trips have an increased travel time of up to 40 minutes (red). On average, the travel time decreased by 4.06 minutes. The average number of transfers increased by 0.14.

The percentage of OD pairs being satisfied by 2 transfers or less increased by 1.59, which corresponds to an additional 421 OD pairs.

The percentage of demand being satisfied by 2 transfers or less increased by 0.64, which corresponds to an additional 1116 journeys.

The percentage of serviced stops decreased by 0.8, which corresponds to 14 stops no longer being included in the network.

It is also interesting to see the difference in the total bus network length, between the existing and optimized networks. This can be measured by summing the duration of every route in each network.

To complete each route once, the existing network takes ≈ 110.53 hours. This value decreased to ≈ 99.88 hours in the optimized network, which signifies a decrease of $\approx 9.64\%$.

While the genetic optimization process used was designed to be applied to routes generated from scratch, it still showed good results when applied to the existing network.

The careful investigation of each route change is a necessary step before considering implementing any change in the real network.

However, it's hard to assess the benefit of the changes applied to a single route, because we are not accounting for the interactions with the rest of the network.

Figures 5.17 and 5.18 show the existing route set before and after optimization, respectively.

Figures 5.19 shows an example of a route modification, where the final route (in blue) satisfies more demand than the initial route (in yellow). We can see that the final route is longer, with more stops, and goes through an area where the population density is greater.

5.20 shows another example of a route modification, where the final route (in blue), despite serving a very reduced number of stops when compared to the original route (in yellow), goes through a highway, leading to a much shorter trip duration for the demand it serves.

Due to time restrictions, the analysis presented is very preliminary and intuitive. A more extensive analysis is crucial in order to understand and validate the optimization process.

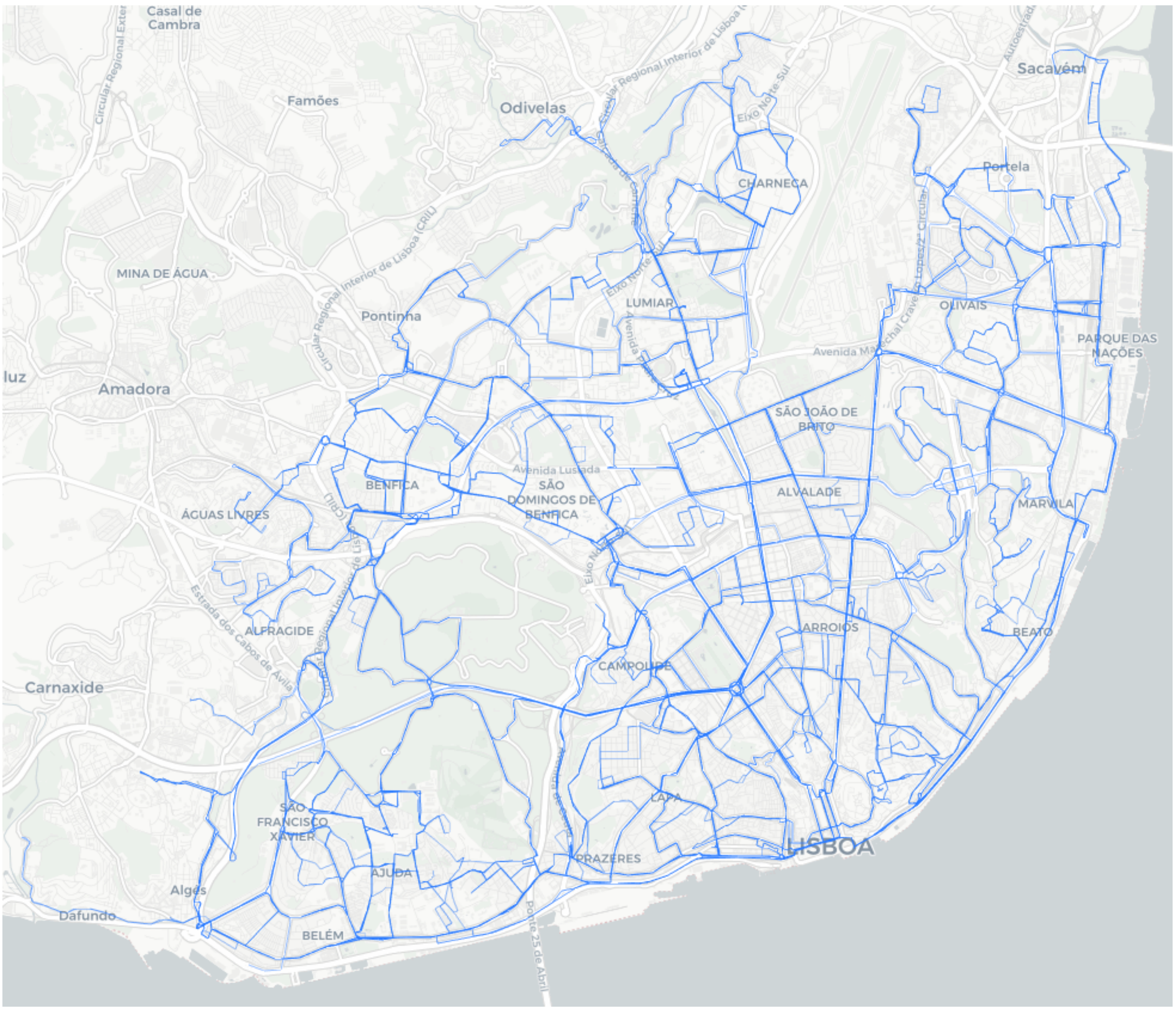


Figure 5.17: Existing route set

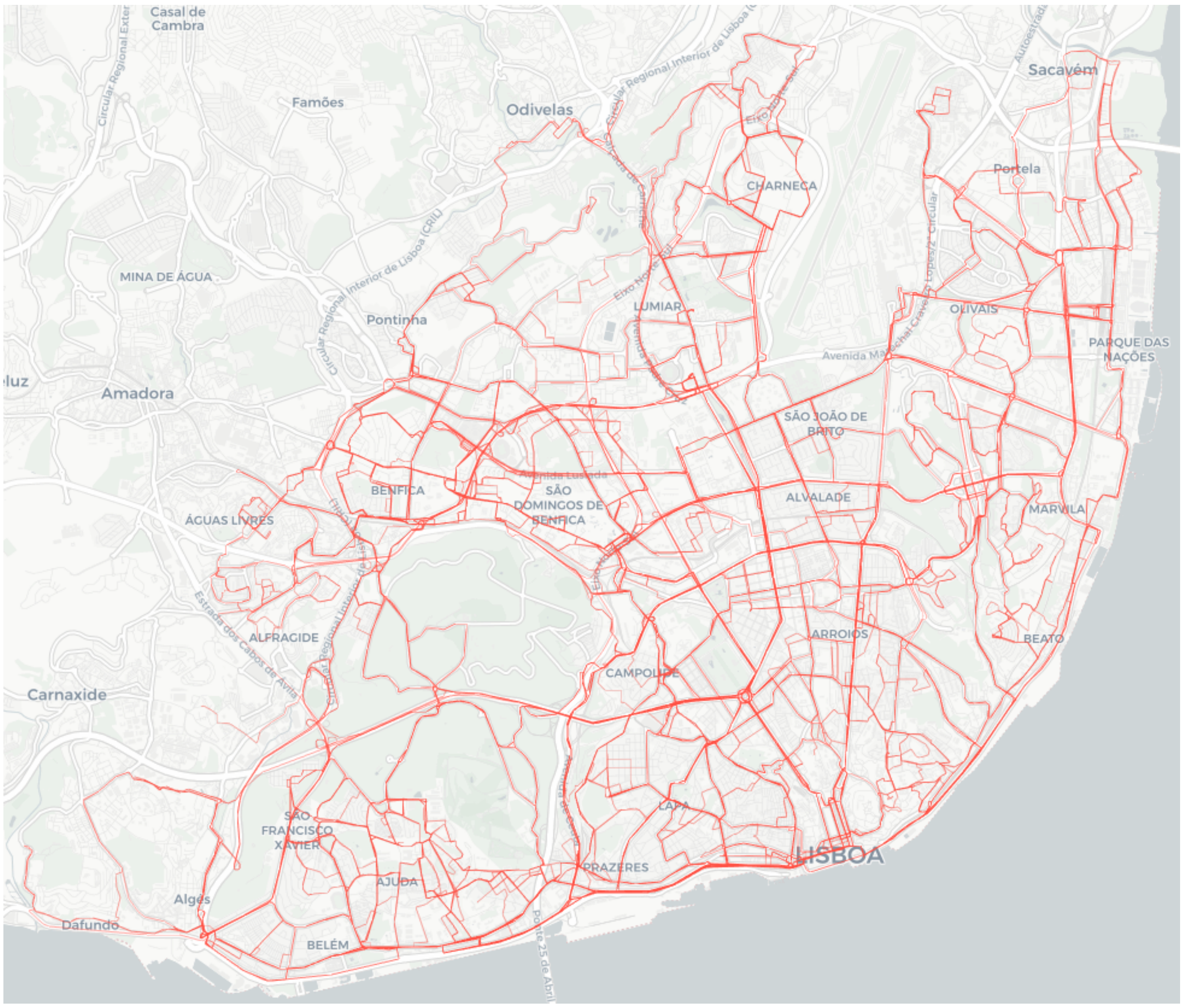


Figure 5.18: Existing route set after optimization

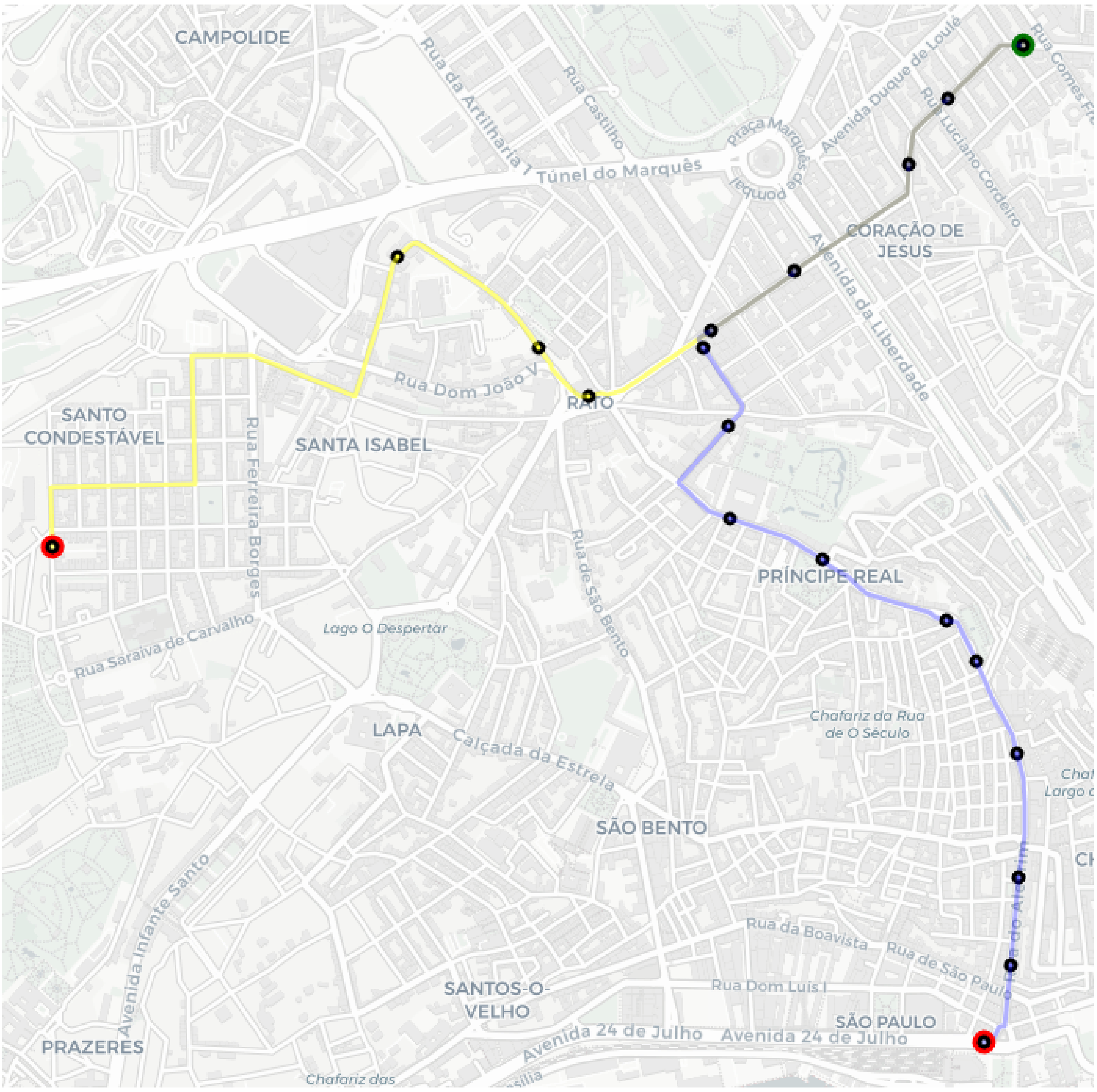


Figure 5.19: Example 1 of a route change, with the same origin (in green) but different destinations (in red). The original route is marked in yellow and the final route in blue.

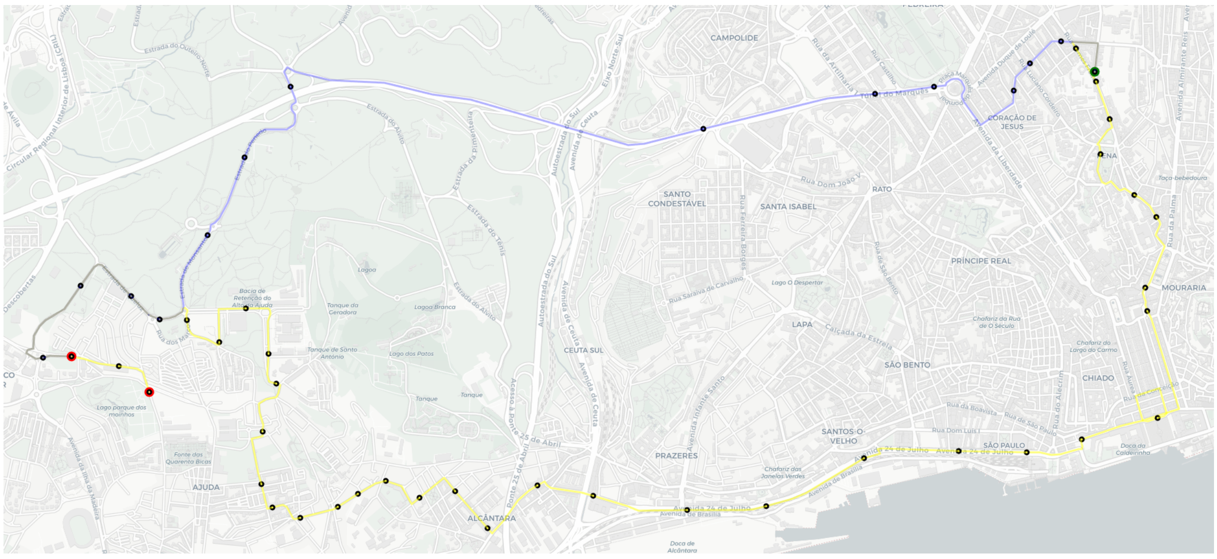


Figure 5.20: Example 2 of a route change, with the same origin (in green) but different destinations (in red). The original route is marked in yellow and the final route in blue.

Chapter 6

Conclusion

This thesis has applied Origin, Destination and Interchange (ODX) inference algorithms to a multi-modal public transport network in Lisbon, Portugal, by using AFC data of both metro and bus systems. We inferred the aligthing locations and times of $\approx 45.3\%$ of bus stages, and linked $\approx 2.56\%$ of stages into full passenger journeys. An extensive data analysis and pre-processing methodology was implemented, in order to allow the interface between the AFC dataset and the static data for each of the used public transport modes.

With the resulting demand matrix, we approached the Transit Network Design (TND) problem, using a Genetic Algorithm GA based approach to optimize the topology of the bus network in Lisbon. The main objective was to develop a simple genetic algorithm and apply it to precise and representative data.

While previous research mostly focuses on approaching the TND problem using synthetic data or optimizing only a subset of the network, we considered the entire bus network of Lisbon, PT.

One of the challenges was the need to compute a precise and efficient representation of the entire road network that connects the bus stops. Furthermore, given the size and complexity of the network, an efficient optimization process had to be designed.

Several experiments were conducted, testing several combinations of parameter values. In addition to optimizing the whole network from scratch with route set sizes ranging from 20 to 400 routes, we ran the optimization algorithm using the existing (real) bus network as a starting point.

The optimized network had a 17% decrease in the value of the objective function, and decrease of ≈ 4 minutes in the average travel time, when compared to the existing bus network.

Designing a more complete optimization problem formulation, taking into account route frequency and additional restrictions (budget, capacity, etc) are interesting future research possibilities.

6.1 Limitations

The limitations in the ODX inference process are mostly related to the lack of AVL data, which results in low destination and interchange inference rates.

The high number of bus transactions without boarding information doubly affects the destination inference rates. The interchange inference rates can be improved greatly by utilizing AVL data to estimate bus alight times, instead of relying on the imprecise schedule trip times.

On the TND side, the limitations are mostly related to the quality and accuracy of the datasets used. The quality and precision of the road data imported from OSM needs to be better assessed, and an effort to correct more problematic errors should be made, in order to guarantee a precise and representative road network representation. Furthermore, a more comprehensive analysis of the parameter values used to estimate the bus stop location should be conducted.

6.2 Future work

The implemented ODX inference methods can easily be extended to incorporate bus AVL data and AFC data from other public transport agencies in Lisbon, which will increase the destination and interchange inference rates, resulting in a more precise and representative demand profile.

More extensive testing should be conducted in order to find the optimal GA parameters. In our case, due to the limited resources and time constraints, only a few parameter variations were tested.

The mutation operators of the GA algorithms can also be further developed, implementing more complex operations.

Our work focuses on optimizing the topology of the network, assuming a constant frequency across all routes. A natural step forward is to also take into account the frequency and schedule of each route. Furthermore, additional constraints such as, for example, the vehicles capacity, budget restrictions and road congestion are important factors that can be considered in future research work.

Our approach only allows same-stop transfers. Further research should aim at developing a more complete route set representation, that, for example, allows transfers between stops in a walkable distance.

Bibliography

[1] "General Transit Feed Specification - TransitWiki."

[2] "Public Transportation Systems — Civil and Environmental Engineering — MIT OpenCourseWare."

[3] H. Bast and P. Brosi, "Sparse map-matching in public transit networks with turn restrictions," in *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 480–483, ACM.

[4] J. Zhao, A. Rahbee, and N. H. M. Wilson, "Estimating a Rail Passenger Trip Origin-Destination Matrix Using Automatic Data Collection Systems," vol. 22, no. 5, pp. 376–387.

[5] J. Attanucci, I. Burns, and N. Wilson, "Bus Transit Monitoring Manual - vol. 1: Data Collection Program Design,"

[6] B. L. Smith and R. Venkatanarayana, "Realizing the Promise of Intelligent Transportation Systems (ITS) Data Archives," vol. 9, no. 4, pp. 175–185.

[7] O. Ibarra-Rojas, F. Delgado, R. Giesen, and J. Muñoz, "Planning, operation, and control of bus transport systems: A literature review," vol. 77, pp. 38–75.

[8] J.-D. Schmöcker, M. G. H. Bell, and W. H. K. Lam, "Special issue: Importance of public transport," vol. 38, no. 1, pp. 1–4.

[9] J. B. Gordon, H. N. Koutsopoulos, N. H. M. Wilson, and J. P. Attanucci, "Automated Inference of Linked Transit Journeys in London Using Fare-Transaction and Vehicle Location Data," vol. 2343, no. 1, pp. 17–24.

[10] M. Hwang, J. Kemp, E. Lerner-Lam, N. Neuerburg, and P. E. Okunieff, "Advanced Public Transportation Systems: The State of the Art Update 2006,"

[11] C. Iliopoulou and K. Kepaptsoglou, "Combining ITS and optimization in public transportation planning: State of the art and future research paths," vol. 11, no. 1, pp. 27, s12544–019–0365–5.

[12] S. Robinson, B. Narayanan, N. Toh, and F. Pereira, "Methods for pre-processing smartcard data to improve data quality," vol. 49, pp. 43–58.

[13] "Why OpenStreetMap? - OpenStreetMap Wiki."

[14] S. Wroclawski, "Why the world needs OpenStreetMap,"

[15] "Google Maps Terms of Service."

[16] A. Buczkowski, "Why would you use OpenStreetMap if there is Google Maps?."

[17] J. Mondzech and M. Sester, "Quality Analysis of OpenStreetMap Data Based on Application Needs," vol. 46, no. 2, pp. 115–125.

[18] M. Haklay, “How Good is Volunteered Geographical Information? A Comparative Study of OpenStreetMap and Ordnance Survey Datasets,” vol. 37, no. 4, pp. 682–703.

[19] R. Dumas, “Analyzing Transit Equity Using Automatically Collected Data,”

[20] C. Vanderwaart, “Planning Transit Networks with Origin, Destination, and Interchange Inference,” p. 175.

[21] J. Zhao, “The Planning and Analysis Implications of Automated Data Collection Systems: Rail Transit OD Matrix Inference and Path Choice Modeling Examples,” p. 124.

[22] N. Nassir, A. Khani, S. G. Lee, H. Noh, and M. Hickman, “Transit Stop-Level Origin–Destination Estimation through Use of Transit Schedule and Automated Data Collection System,” vol. 2263, no. 1, pp. 140–150.

[23] W. Wang, J. Attanucci, and N. Wilson, “Bus Passenger Origin-Destination Estimation and Related Analyses Using Automated Data Collection Systems,” vol. 14, no. 4, pp. 131–150.

[24] T. Kusakabe, T. Iryo, and Y. Asakura, “Estimation method for railway passengers’ train choice behavior with smart card transaction data,” vol. 37, no. 5, pp. 731–749.

[25] E. van der Hurk, L. Kroon, G. Maroti, and P. Vervest, “Deduction of passengers’ route choice from smart card data,” in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pp. 797–802, IEEE.

[26] M.-P. Pelletier, M. Trépanier, and C. Morency, “Smart card data use in public transit: A literature review,” vol. 19, no. 4, pp. 557–568.

[27] L. Moreira-Matias, J. Mendes-Moreira, J. F. de Sousa, and J. Gama, “Improving Mass Transit Operations by Using AVL-Based Systems: A Survey,” vol. 16, no. 4, pp. 1636–1653.

[28] M. A. Nayeem, M. K. Rahman, and M. S. Rahman, “Transit network design by genetic algorithm with elitism,” vol. 46, pp. 30–45.

[29] Q. Fu, R. Liu, and S. Hess, “A Review on Transit Assignment Modelling Approaches to Congested Networks: A New Perspective,” vol. 54, pp. 1145–1155.

[30] M. H. Baaj and H. S. Mahmassani, “An AI-Based approach for transit route system planning and design,” vol. 25, no. 2, pp. 187–209.

[31] P. Chakroborty and T. Wivedi, “Optimal Route Network Design for Transit Systems Using Genetic Algorithms,” vol. 34, no. 1, pp. 83–100.

[32] Q. K. Wan and H. K. Lo, “A Mixed Integer Formulation for Multiple-Route Transit Network Design,” vol. 2, no. 4, pp. 299–308.

[33] J. Guan, H. Yang, and S. Wirasinghe, “Simultaneous optimization of transit line configuration and passenger line assignment,” vol. 40, no. 10, pp. 885–902.

[34] N. E. Lownes and R. B. Machemehl, “Exact and heuristic methods for public transit circulator design,” vol. 44, no. 2, pp. 309–318.

[35] B. Yu, Z.-Z. Yang, P.-H. Jin, S.-H. Wu, and B.-Z. Yao, “Transit route network design-maximizing direct and transfer demand density,” vol. 22, pp. 58–75.

[36] P.-L. Bourbonnais, C. Morency, M. Trépanier, and Martel-Poliquin, “Transit network design using a genetic algorithm with integrated road network and disaggregated O–D demand data,”

[37] A. Sinha, P. Malo, and K. Deb, “A Review on Bilevel Optimization: From Classical to Evolutionary Approaches and Applications.”

[38] Z. Gao, H. Sun, and L. L. Shan, “A continuous equilibrium network design model and algorithm for transit systems,” vol. 38, no. 3, pp. 235–250.

[39] W. Szeto and Y. Wu, “A simultaneous bus route design and frequency setting problem for Tin Shui Wai, Hong Kong,” vol. 209, no. 2, pp. 141–155.

[40] A. Chen and C. Yang, “Stochastic Transportation Network Design Problem with Spatial Equity Constraint,” vol. 1882, no. 1, pp. 97–104.

[41] S. M. Amiripour, A. A. Ceder, and A. S. Mohaymany, “Designing large-scale bus network with seasonal variations of demand,” vol. 48, pp. 322–338.

[42] D. Whitley, “A genetic algorithm tutorial,” vol. 4, no. 2.

[43] J. B. Gordon, “Intermodal Passenger Flows on London’s Public Transport Network,” p. 155.

[44] “Where’s Charlie? The Origin-Destination-Transfer (ODX) model.”

[45] “Explaining Dashboard Metrics: Subway Reliability.”

[46] “VIVA.”

[47] “Comprar - Metropolitano de Lisboa, E.P.E..”

[48] “Lista de estações do metropolitano de lisboa.”

[49] “GTFS Best Practices - Routes.Txt.”

[50] “Haversine formula.”

[51] M. Nikolić and D. Teodorović, “Transit network design by Bee Colony Optimization,” vol. 40, no. 15, pp. 5945–5955.

[52] J. Vuurstaek, G. Cich, L. Knapen, W. Ectors, A.-U.-H. Yasar, T. Bellemans, and D. Janssens, “GTFS bus stop mapping to the OSM network,” vol. 110, pp. 393–406.