Machine Learning Methods for the Optimisation of Urban Mobility

Rui Loureiro
rui.loureiro@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa

Abstract. 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 allow transit providers to collect high volumes of data on their operations and passenger behaviour, enabling better, more informed decisions. 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. This thesis uses Automatic Data Collection System (ADCS) data to infer passengers' origin and destination locations in a multi-modal public transport network in Lisbon, Portugal. With the resulting demand data, we approach the Transit Network Design (TND) problem, using a Genetic Algorithm (GA)-bassed approach to optimize the entire bus network in Lisbon, using real road data. The resulting optimized network achieves a 4 minute decrease in the average travel time of all passengers, and a reduction of the total service duration of 10 hours.
Keywords: Origin-Destination-Interchange Inference, Genetic algorithms, Public Transport Optimization, Transit Network Design, Urban mobility

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 [2, 21].

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 [21], allowing transit providers to collect high volumes of data on their operations and passenger behaviour. These ADC systems include Automatic Fare Collection (AFC) systems, Automatic Vehicle Location (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 [15], 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 [21].

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 [10, 14]. 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 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.

2. Background

2.1 Origin, Destination and Interchange Inference

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) [7]. 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 [7].

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 [16].

Early implementations had several limitations, such as covering only a single mode, or inferring locations, but not times [21, 20]. Others rely on scheduled trip times to determine arrival times [11]. More recent developments have overcome some of these earlier limitations, by taking advantage of AVL systems. [17] 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.

2.2 Transit Network Design

TND is part of the strategical planning, and concerns long term decisions in the public transport planning. 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 [3, 6]. GA-based methods are one of the most used methods used to solve TND in the literature.

[6] 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.

[19] 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.

[12] 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.

[5] develops a GA-based algorithm to optimize two public transport networks in Quebec. They use pedestrian network data and road network data from Open Street Map (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.

3. ODX Inference

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 [8], which has since been used by the (Massachusetts Bay Transportation Authority (MBTA)) to plan service, understand where the network is crowded, estimate passenger volume between stations and assess the system reliability, for example [18].

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.

3.1 Input data

As the input data for the ODX process, the following data sets were used:

- Metro and bus AFC data for 10 days of October 2019
- Metro and bus General Transit Feed Specification (GTFS) feeds
- Additional bus static data information, provided by the bus operator

Pre-processing A comprehensive validation and pre-processing step was applied to the input data. Since the AFC datasets included repeated and redundant data, only the relevant AFC fields were extracted. For example, two fields describing the stop, one for the identifier and another for the stop exist in the AFC data. A separate dataset that maps stop identifiers to stop names was created, and the stop name field was removed from the AFC data. The column names were standardized, and, when possible the fields were converted to more efficient types.

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 [21], where each AFC record is matched to an AVL record, and the AVL location 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 **3.8%** 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.

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 [21, 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 [21, 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 [1], 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 below:

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.

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, our approach was similar to [21]. 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.

Destination Inference Results The results for the destination inference process are shown in Table 1, showing the percentage of stages that failed each test.

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

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.
  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.

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 1. Destination Inference Results

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 [8], 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 [8], 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.

Interchange Inference Results Table 2 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.
– 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.

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** (≈ 7.62%), which is a result of the low destination-inference percentage, discussed in Section 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.

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 2. Interchange Inference results

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 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 [12, 13].

Finally, we present and discuss the obtained results.

4.1 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.

5. Representation

Two representations were developed:

The **road network** representation includes the stop locations and the links that connect these stops, in the underlying road network. This representation is used to compute the paths between stops **in the road network**.

The **route set** representation describes the stops and routes that connect the stops. Each route set as a unique route set representation. This representation is used to compute the paths between stops **in the bus network**.

5.1 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.

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 1 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.

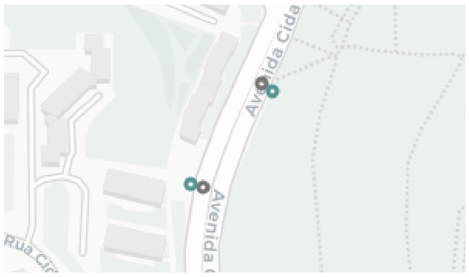

Fig. 1. GTFS location (in green) and actual street point(in gray) for two bus stops

Several research efforts to assign GTFS bus stops to OSM and other road networks have been developed.

[4] 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 software (*pfaedle*) is publicly available[2] and was used in our work, to map each GTFS stop to a location on the OSM road network.

Figure 2 shows the comparison between an original GTFS shape and the corresponding estimated shape computed through *pfaedle*.

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

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

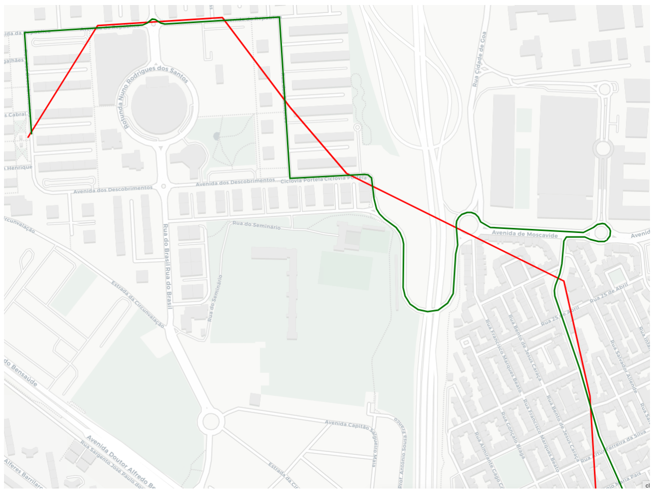

Fig. 2. Comparison between original GTFS shape (in red) and estimated shape (in green)

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 3

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 Open Source Routing Machine (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 illustrates this step.

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.

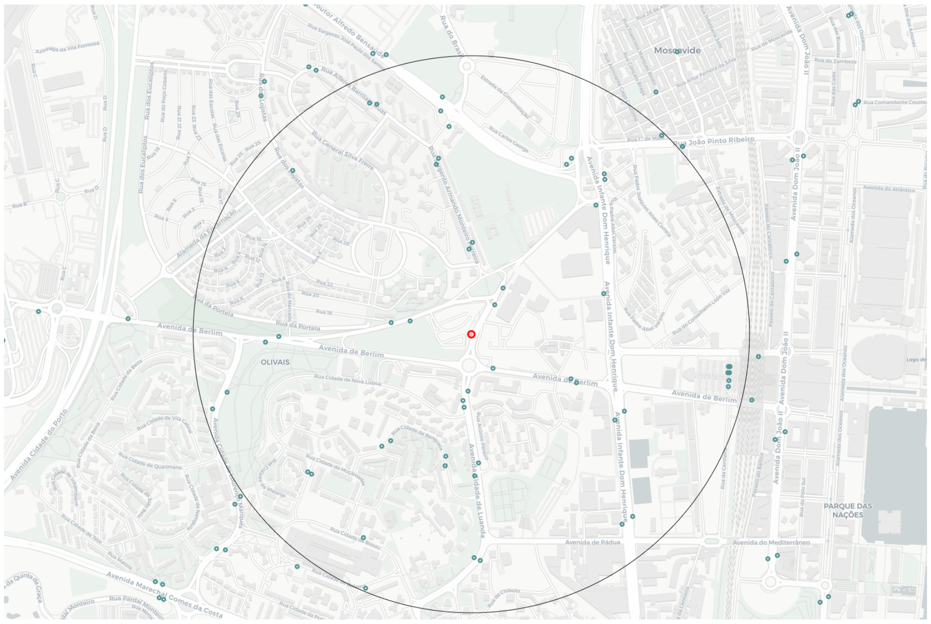

Fig. 3. Circle of candidate neighbor stops (in green) for central stop (in red)

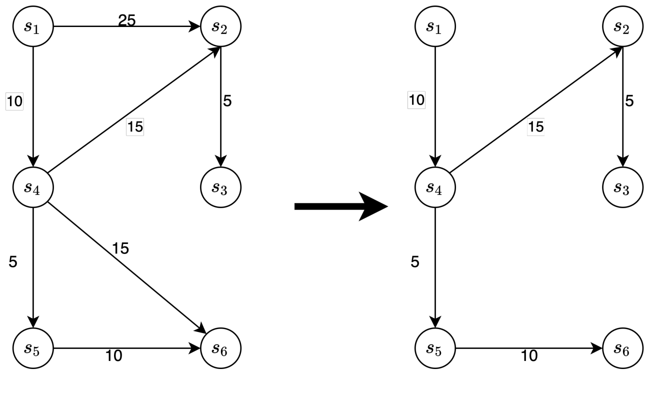

Fig. 4. Example of the road network graph simplification step. The edge weights exemplify the duration of the shortest path between the edge stops

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.[3]

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 5 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*.

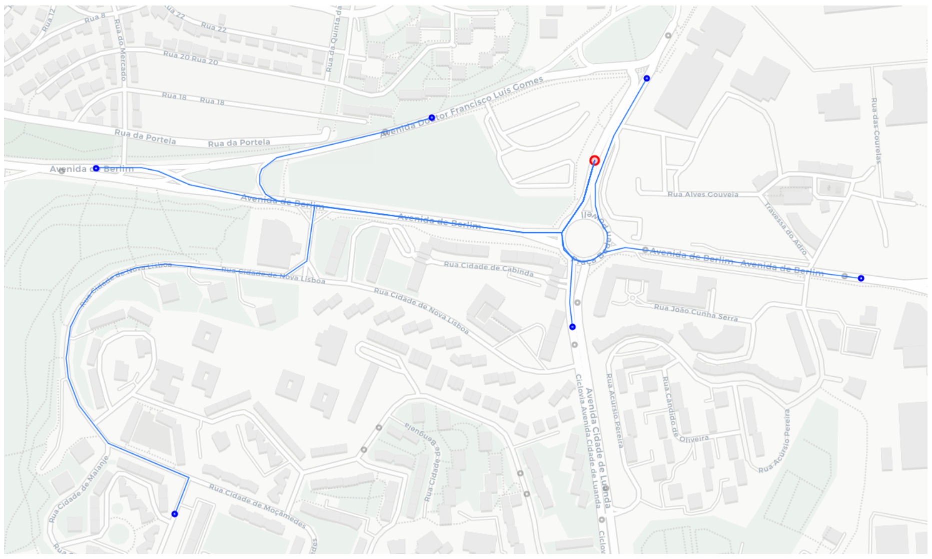

Fig. 5. stop <*7714*> *[Piscina Olivais]* (in red), its neighbours (in blue) and the path to the neighbours. The remaining stops (in grey) are part of C_{7714} but are not neighbors of stop *7714*.

5.2 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 6 illustrates a route set, with 3 routes and 6 stops, where

- ***route 1*** serves stops **1**, **2** and **3**

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

– ***route 2*** serves stops **2**, **4** and **5**
– ***route 3*** serves stops **3**, **5** and **6**

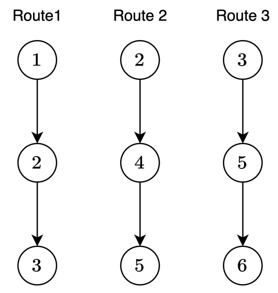

Fig. 6. Simplified view of a route set with 3 routes and 6 stops

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 7

5.3 Pre-processing

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.

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

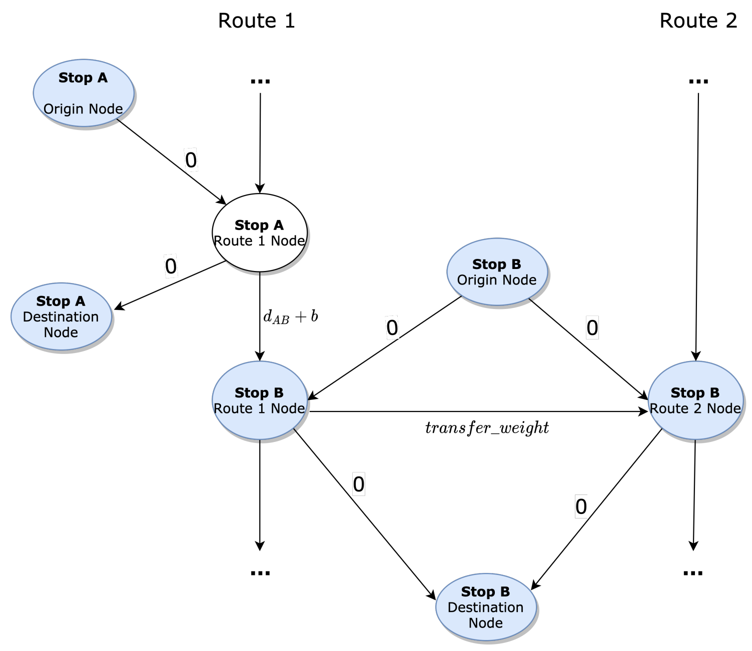

Fig. 7. Route Set graph topology

5.4 Genetic Algorithm

The genetic optimisation algorithm is adapted from the one described in [12].

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 section 3

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 [12, 6, 13]. The methodology for generating the initial solution is the same as [12, 13], 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 $DS = \{ds_{ij} \mid i, j \epsilon [0, 1, 2, \ldots, |N - 1|]\}$
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 [12, 13].

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.

5.5 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. 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}.
     - **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.
   - ***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:

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

     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 [12]. 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.

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.

6. Results

This section presents the results of applying the described optimization method to the bus network in Lisbon.

Two major tests were made. In the first test, the routes were generated from scratch, using the initialization method described in Section 5.4. 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 **Population size** (*pop_size*), **Elite size**

($elite\_size$), **Tournament size** (t), **Probability of a small mutation** (p_{ms}) and **Probability of a delete mutation** (p_{delete}) were used.

To evaluate our results, the following metrics will be used:

– **Objective Function**: The value of the objective function (described in Section **??**) being minimized by the genetic algorithm.
– **Satisfied Origin Destination (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.

7. Optimization from scratch

The following values for the number of routes R were tested: $20, 50, 100, 200, 300, 400$. Table 3 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 4 shows the final metric values, for the best performing experiment of each value of R.

Table 5 shows the parameters that lead to the best result for each of the values of R.

7.1 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 $s_0, \ldots, s_{N-1}$.

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 3. 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 4. 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. Best parameter for each value of R

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

Table 7 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

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 6. 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 7. Description of experiments

Comparison to existing network Experiment 3, having the lowest objective function score, is considered to be the best performing experiment. Table 8 shows the comparison between the final values of Experiment 3 and the existing (unoptimized) network).

Figure 8 shows the distribution of travel time differences between the optimized and existing network.

Figure 8 shows the distribution of the number of transfers difference between the optimized and existing network.

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), while a few trips have an increased

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 8. Initial, final and change in metric values between existing network and optimized Experiment 3

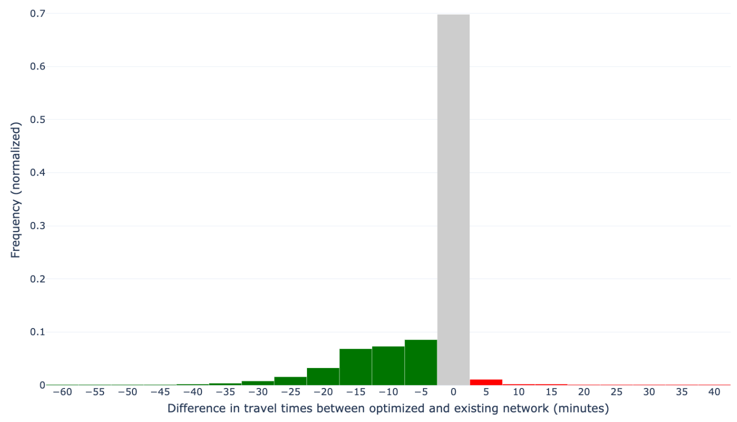

Fig. 8. Difference in travel time between the optimized and existing networks (in minutes). The green bars correspond to a decrease in travel time in the optimized network. The gray bars correspond to a minimal (< 2.5 min) change in travel time and the red bars correspond to an increase in travel time.

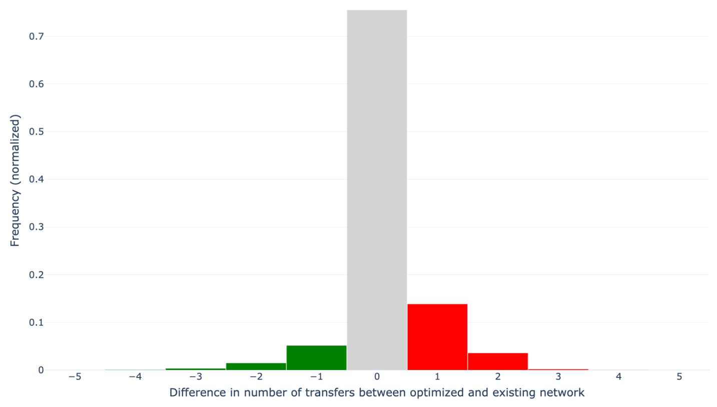

Fig. 9. Difference in the number of transfers, between the optimized and existing networks (in minutes). The green bars correspond to a decrease in the number of transfers the optimized network, respectively. The gray bars correspond to a no change in number of transfers and the red bars correspond to an increase in the number of transfers

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 $\approx$ 110.53 hours. This value decreased to $\approx$ 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.

8. 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.

References

[1] In: *Haversine Formula.*

[2] J. Attanucci, I. Burns, and Nigel Wilson. "Bus Transit Monitoring Manual - Vol. 1: Data Collection Program Design". In: (UMTA-IT-09-9008-81-1 Aug. 1, 1981). Ed. by Inc. Multisystems and ATE Management and Service Company.

[3] M. Hadi Baaj and Hani S. Mahmassani. "An AI-Based Approach for Transit Route System Planning and Design". In: *Journal of Advanced Transportation* 25.2 (Mar. 1991), pp. 187–209.

[4] Hannah Bast and Patrick 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*. SIGSPATIAL '18: 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. Seattle Washington: ACM, Nov. 6, 2018, pp. 480–483.

[5] Pierre-Léo Bourbonnais, Catherine Morency, Martin Trépanier, and Éric Martel-Poliquin. "Transit Network Design Using a Genetic Algorithm with Integrated Road Network and Disaggregated O–D Demand Data". In: *Transportation* (Aug. 23, 2019).

[6] Partha Chakroborty and Tathagat Wivedi. "Optimal Route Network Design for Transit Systems Using Genetic Algorithms". In: *Engineering Optimization* 34.1 (Jan. 2002), pp. 83–100.

[7] Raphaël Dumas. "Analyzing Transit Equity Using Automatically Collected Data". In: ().

[8] Jason B Gordon. "Intermodal Passenger Flows on London's Public Transport Network". In: (), p. 155.

[9] Jason B. Gordon, Harilaos N. Koutsopoulos, Nigel H. M. Wilson, and John P. Attanucci. "Automated Inference of Linked Transit Journeys in London Using Fare-Transaction and Vehicle Location Data". In: *Transportation Research Record: Journal of the Transportation Research Board* 2343.1 (Jan. 2013), pp. 17–24.

[10] O.J. Ibarra-Rojas, F. Delgado, R. Giesen, and J.C. Muñoz. "Planning, Operation, and Control of Bus Transport Systems: A Literature Review". In: *Transportation Research Part B: Methodological* 77 (July 2015), pp. 38–75.

[11] Neema Nassir, Alireza Khani, Sang Gu Lee, Hyunsoo Noh, and Mark Hickman. "Transit Stop-Level Origin–Destination Estimation through Use of Transit Schedule and Automated Data Collection System". In: *Transportation Research Record: Journal of the Transportation Research Board* 2263.1 (Jan. 2011), pp. 140–150.

[12] Muhammad Ali Nayeem, Md. Khaledur Rahman, and M. Sohel Rahman. "Transit Network Design by Genetic Algorithm with Elitism". In: *Transportation Research Part C: Emerging Technologies* 46 (Sept. 2014), pp. 30–45.

[13] Miloš Nikolić and Dušan Teodorović. "Transit Network Design by Bee Colony Optimization". In: *Expert Systems with Applications* 40.15 (Nov. 2013), pp. 5945–5955.

[14] Jan-Dirk Schmöcker, Michael G. H. Bell, and William H. K. Lam. "Special Issue: Importance of Public Transport". In: *Journal of Advanced Transportation* 38.1 (Sept. 2004), pp. 1–4.

[15] Brian L. Smith and Ramkumar Venkatanarayana. "Realizing the Promise of Intelligent Transportation Systems (ITS) Data Archives". In: *Journal of Intelligent Transportation Systems* 9.4 (Oct. 2005), pp. 175–185.

[16] Catherine Vanderwaart. "Planning Transit Networks with Origin, Destination, and Interchange Inference". In: (), p. 175.

[17] Wei Wang, John Attanucci, and Nigel Wilson. "Bus Passenger Origin-Destination Estimation and Related Analyses Using Automated Data Collection Systems". In: *Journal of Public Transportation* 14.4 (Dec. 2011), pp. 131–150.

[18] *Where's Charlie? The Origin-Destination-Transfer (ODX) Model.* URL: `https://www.mbtabackontrack.com/blog/43-odx-model` (visited on 11/12/2020).

[19] Bin Yu, Zhong-Zhen Yang, Peng-Huan Jin, Shan-Hua Wu, and Bao-Zhen Yao. "Transit Route Network Design-Maximizing Direct and Transfer Demand Density". In: *Transportation Research Part C: Emerging Technologies* 22 (June 2012), pp. 58–75.

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

[21] Jinhua Zhao, Adam Rahbee, and Nigel H. M. Wilson. “Estimating a Rail Passenger Trip Origin-Destination Matrix Using Automatic Data Collection Systems”. In: *Computer-Aided Civil and Infrastructure Engineering* 22.5 (July 2007), pp. 376–387.