# REPORT

# Mobility study based on the analysis of Bluetooth data in the city of Brisbane


**Authors:** María Sarrá Paloma and Ana Valero Serratosa

**Supervisor:** Mehmet Yildirimoglu

**Co-supervisor:** Tania Santos


**Academic year:** 2018-2019


**Barcelona School of industrial Engineering**

**Master of industrial Engineering**



# Abstract

In the field of transport planning, many studies have examined the effectiveness of different algorithms related to the problem of transport network design. However, no study has considered the alternative of a problem of transport network modification, which consists of making small alterations to the current network without having to entirely design it.

The focus of this research is on designing an algorithm able to find the optimal route for placing a new public transport line, understanding optimal as capable of minimizing the traffic of private vehicles around the city. As observed, it corresponds to one case of transport network modification.

Data used in this project were collected by the detection of devices through the Bluetooth sensors spread over the city of Brisbane, Australia. The use of real data led to the need of a previous analysis and cleaning of the database, in order to obtain results which are feasible and reliable.

Since it is a non-linear optimization problem, heuristic solution algorithms are used in the design procedure. The algorithm proposed is based on genetic algorithms, and it has been adjusted to this case of study by adapting the functions that compound the algorithm.

The proposed model was tested and the results obtained analysed, concluding that the designed algorithm is effective for generating routes able to minimize the car flow in a transport network.

**Table of contents**

# 1. Abbreviations

BT: Bluetooth

TAZ: Travel analysis zones

CBD: Central business district

LGA: Local government area

OD: Origin destination

GA: Genetic algorithm

SA: Simulated annealing

TS: Tabu search

# 2. Preface

## 2.1. Origin of the Project

The idea of this project appears due to the existence of an extensive database that contains the movements of BT devices through the city of Brisbane. It is observed that many analysis and improvements could be carried out using that database. However, seeing the recent growth of the public transport in Brisbane, it has been decided that a new route would be created in order to bring the maximum number of people from a place to another

## 2.2. Incentive

The main incentive for the realization of this project has been the aim to design an algorithm which is able to find an optimal route given a certain transport network. Although in this specific case of research the model has been tested in the Brisbane network, the motivation behind this project has been not only to design an algorithm for it but one that is also suitable for all other cases (when data is available).

Moreover, another incentive has been the appliance of concepts acknowledged in the university during both the bachelor and master, as well as the aim to continue learning and broaden our knowledge especially in the area of transport management.

## 2.3. Previous knowledge

When carrying out the project it has been very useful the acknowledgements previously acquired in the university, especially in the subjects of transports, quantitative methods of industrial process management, and informatics.

Referring to the different predefined algorithms already been studied in the area of transport network design, it has been necessary to study and deeply understand them, analysing each step in their procedure.

# 3. Introduction

The city of Brisbane faces congestion problems due to an underperforming public transport network. Enlarge the road network, expand the public transport network, improve transfers between public transport, extend bike sharing scheme or broaden the ferry services network are some of the improvements that could be carried out in any city in order to improve congestion problems.

Having analysed the database that contains the movement of the devices through the roads of Brisbane, it has been decided to use it in order to create an algorithm that manages to reduce the congestion not only of Brisbane but also of any city with a similar problem. The aim of the algorithm is to create a bus line that carries the maximum amount of people who did not take public transport previously when moving around the route done by the bus.

## 3.1. Research objectives

The main objective of this research project is to design an algorithm able to find an optimal route, which brings the maximum number of people from their origin to their destination. As previously mentioned, the algorithm has been designed using the data from the transport network in Brisbane. However, the aim of the project is to find an algorithm suitable for many other cases, when data of the transport network is available.

Apart from this main objective, the following secondary objectives have been defined:

- Analysis of the mobility in Brisbane, and the aspects to be improved in the current transport system.
- Analysis of the different algorithms studied in the field of transport network design, studying the pros and cons of each of them and whether they would suit this specific case of research.

- Analysis of the effectiveness of the model designed in this project, testing it with the data obtained from the transport network in Brisbane.

## 3.2. Scope of the research

Regarding the scope of the project, the present case involves the design of an algorithm which, with a given data set of the transport network from a city, and having the data in a particular format, is able to find a potential optimal route in order to reduce the density of cars in the city.

This is achieved by the design of an objective function, which measures the quantity of people that would be brought from their origin to their destination if the route being analysed was the one selected. Once defined the objective function, an algorithm will be designed in order to iteratively improve the current solution until the optimal one is found.

The data used for this case of study is obtained by the detection of Bluetooth devices through a set of sensors distributed over the city of Brisbane (Queensland, Australia). As mentioned, the aim of this research is to find an algorithm suitable for whichever city. However, only cities with a similar database as the one that has been used, with the same structure of data, would be candidates to use the algorithm proposed.

When referring to BT devices, they are understood as vehicles with Bluetooth, capable of connecting with the sensors spread around the city. There is a limitation to this research which is that some BT devices could be not vehicles but mobile phones, laptops, etc. It has been assumed that when circulating around the city, BT is turned off in this kind of devices.

According to Automotive Lease Guide **[1]**, at least 86 percent of all new vehicles in the U.S. offer Bluetooth as a standard feature, which is why it has been assumed that the loss of information due to vehicles with no Bluetooth is negligible. Furthermore, there is no way of knowing how many people travel in each car, which makes it compulsory to suppose there is only one passenger in each vehicle.

## 3.3. Methodology

The first step taken in the thesis has been the analysis of the database. It has been studied the information that is necessary and unnecessary, and the accurate format of the database for working with the information.

Once the database has been analysed, the unnecessary information has been deleted and the essential information has been treated. The results have been shown in a graph with its nodes and edges representing the movement of the vehicles. Seeing the result of the graph, the database has been improved deleting and adding information in order to get a better database, and afterwards the graph has been created again. Iteratively, the database has been improved and graphed until the information shown in the graph and its format is suitable for the algorithm.

Once the database has been analysed and its information has been treated, an algorithm has been designed in order to create a bus line that carries the maximum number of people.

# 4. Brisbane Transport Network and Data collection

When it comes to finding an algorithm to resolve a problem, the structure of the data that is going to be used has an important role. Depending on its structure and its limitations, data will have to be treated in a way or in another, and also the algorithm may be designed differently. In this chapter, the structure of the data that has been used for this project is explained in detail.

However, prior to explaining the aforementioned, it is essential to be on the map and to understand how does transportation in Brisbane work nowadays. For this reason, a brief description of its transport network is found below.

## 4.1. Transport Network in Brisbane

Brisbane is the capital and most populous city of Queensland, and the third most populous city in Australia, after Sidney and Melbourne. It is located in the southeast corner of Queensland and is 960 km north Sidney and 97 km north of the Gold Coast.


*Figure 1 Map of Australia*

Brisbane is a river city (*Figure 1*) and has many bridges that allow citizens to cross from one side to another of the city. It is a bent river that runs through the entire city, and where maritime transport is usually seen. The Central Business District stands in

a peninsula of the Brisbane river, located 15 kilometres from its mouth at Moreton Bay.

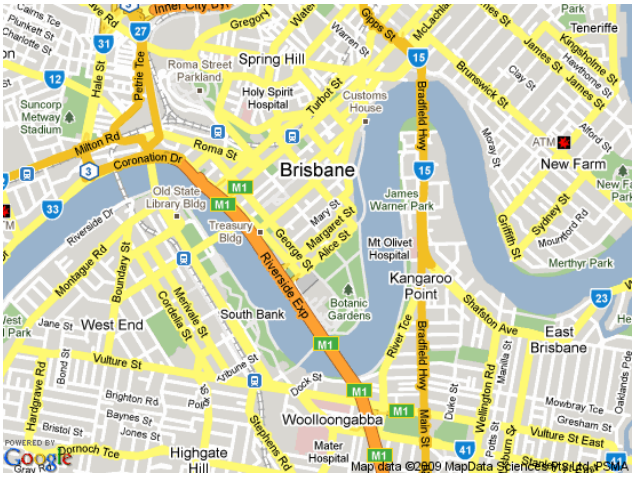



*Figure 2 Zoom of Brisbane CBD*

Transport network in Brisbane is composed of roads, rails, ferries, footpaths, bike paths... The public transport is managed by TransLink, and it includes trains, buses, ferries and taxis. Since 2007 TransLink introduced the go card smartcard based in a ticketing System. Passengers buy one only card and they top it up when there is no money left, which costs significantly less than using a paper ticket.

Referring to buses, they operate both in urban and suburban zones, and move people all around Brisbane. Buses operate from 5 AM until midnight through the week, with some Friday and Saturday night 24-hour services. A busway network has been constructed to provide public transport access to areas of the city beyond the reach of the metropolitan train lines. Busway are roads dedicated to buses only. Brisbane's three busways are the South East Busway, the Northern Busway and the Eastern busway.

The large Queensland Rail City network consists of seven suburban lines covering mostly the southwest, north and outer east sides of the city. It also provides service between the city and Brisbane Airport.

As it was said before, the river has a leading role in the city of Brisbane, and so does its transport. Ferry services are operated on the Brisbane river, where CityCats and CityFerries are usually seen. CityCats are catamarans, and Cityferries are small boats

that freely transport people from one side to another.

Brisbane is constructing a number of cycle routes, off-road bikeways that enable citizens to move easily by bicycle through the city. Citycycle is a service offered in the city of Brisbane with more than 150 stations from the University of Queensland to Teneriffe suburb.

The road system was planned around large spacious suburban areas. There are nowadays several main roads that connect the CBD with these suburbs, some of them being highly populated. As a result, traffic congestion has become a major problem which is being investigated in order to speed up the roads.

## 4.2. Structure of data

There are 1000 Bluetooth sensors distributed around Brisbane City LGA. They were placed with the objective of gathering information of the number of cars that travel through the crossroads where sensors are situated. The information obtained by those sensors is stored into a file and then analysed. A file generated daily is the database used for this project.

The information saved from all the devices that interact with each BT sensor is saved in a file. Each line of that file contains the following information: the ID of the device that is captured, the day and time of the capture, the duration of the detection, the sensor ID, and the latitude and longitude of the BT sensor, and finally the owner of the sensor.

| Device ID | Time | Duration | Sensor ID | Latitude | Longitude | Owner |
|---|---|---|---|---|---|---|
| 4597058 | 1/7/15 0:00 | 1 | B0164 | -27.425818 | 153.014288 | BCC |

*Table 1 Example of a line of the database*

As it was said before, the BT sensors are distributed through Brisbane City area, occupying an area of 1342 km$^2$. The most eastern points are located in a longitude of

153.2514 and the most western points are located in a longitude of 152.922593. Referring to latitude, the most northern points have latitude of -27.285025 and the most southern points of -27.655756. In the *Figure 3* it can be observed the boundaries of the work zone.

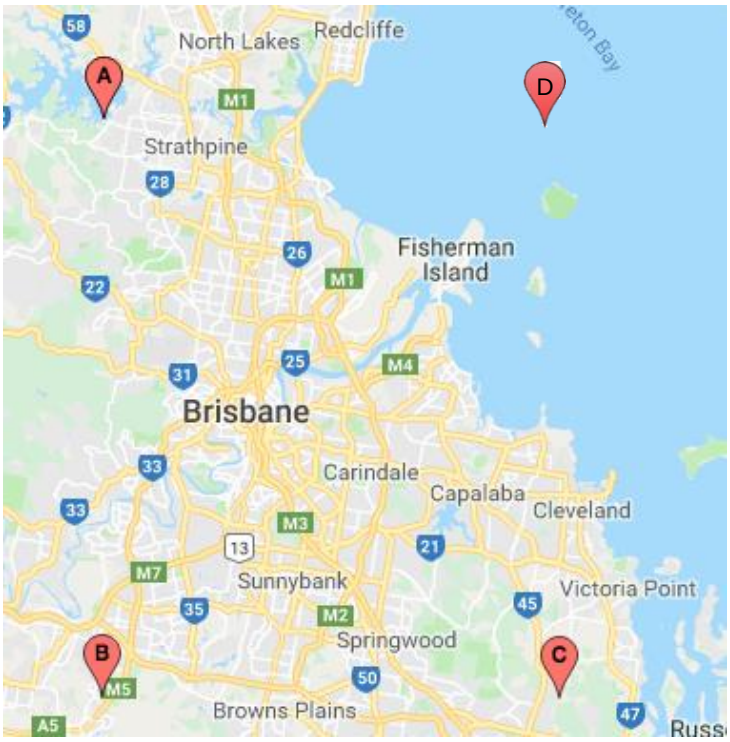



*Figure 3 Boundary of the Bluetooth sensors*

All the information gathered by the Bluetooth sensors in a day is saved in a csv file with a very high volume of data, with more than seven million rows, which is ordered chronologically starting at 00:00 and ending at 23:59. Working with files of such magnitude, makes it compulsory to treat it through programming. In this thesis, Python has been the programming language chosen to work with.

As the thesis has been carried out using real information, it has been necessary to adapt the database in order to be able to work with it. Some of the information that contains the file is unnecessary and has been filtered, some other information of the file is not significant for the research and has not been taken into account, and some other information does not make sense and has been deleted. In the chapter 5.3 can be found the analysis and treatment of the database, which gives room to a database

that is clean, significant and able to be used by the algorithm.

# 5. Modelling

In chapter 4, it is given a first approach of transport in Brisbane. It is also explained how Bluetooth sensors are spread all over the city and its surroundings, storing data every time they detect a Bluetooth device. This data is gathered and saved in a csv file, which is generated every day. As mentioned in section 4.3, due to the high volume of data contained in the file, Python programming language has been used in this research.

In this chapter, it will be discussed the data analysis and treatment. In the first place, the set of sensors needs to be bounded in order to define the area of study.

After that, data obtained from the csv file has to be treated in order to solve its possible limitations and build a suitable database for the model proposed. Moreover, this chapter gives a progressive description of the algorithms used to design the optimization model proposed in this research.

The database used in the model in this case of research mainly consists of, on the one hand, a graph representing all the sensors in their respective locations, as well as the connections between them, and on the other hand the origin destination matrix.

## 5.1. Definition of the area of study

The area to be studied is the section covered by the set of BT sensors. However, there are cases in which the device is first or last detected in one of the sensors in the borderline of the BT sensor area. For this reason, it is needed to define a boundary which delimits the area of study.

The detection of devices in the borderline of the BT sensor area could mean either that the device has started or finished its route in that sensor, or that the route has continued out of the area embraced by the BT sensors. In the second case, keeping the route in the database could lead to wrong conclusions, due to the fact that incomplete routes would be taken into account for the analysis. This is the reason why it has been decided to remove the routes that start or finish in a sensor from the

boundary. In order to do this, a previous definition of the boundary needs to be done.

For the boundary to be defined, the BT sensor region has been divided in different sections according to their latitude. Next, for each section, the sensor with the minimum and maximum longitude has been stored as a boundary sensor.

The number of divisions has been analysed in order to get an accurate limit for the area. As observed in the figures below, with increasing the number of divisions, the number of sensors included in the boundary (represented with a red indicator in the figure) also increases. However, a too high number of divisions can lead to the inclusion of sensors in the boundary that are not really in the borderline. The boundary obtained by setting the number of divisions to 40 has been found to be the most accurate limit for the BT sensors area, due to the fact that with fewer divisions the boundary is not enough defined, whereas, with more divisions, inner nodes are found in the group of boundary sensors.

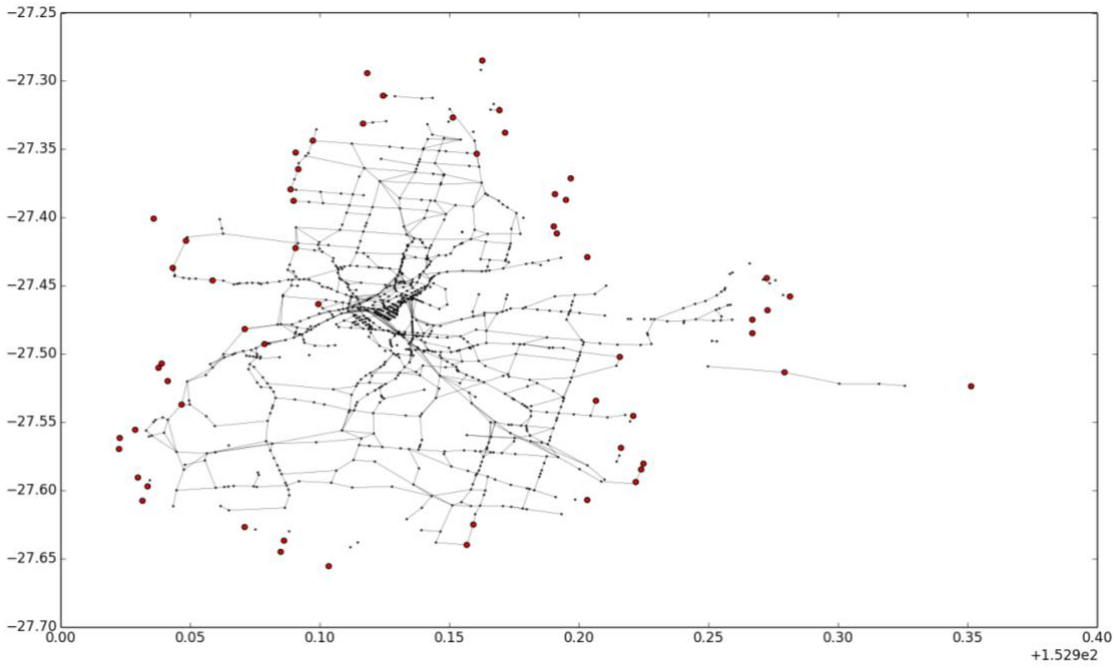



*Figure 4 Representation of the boundary with 30 divisions*

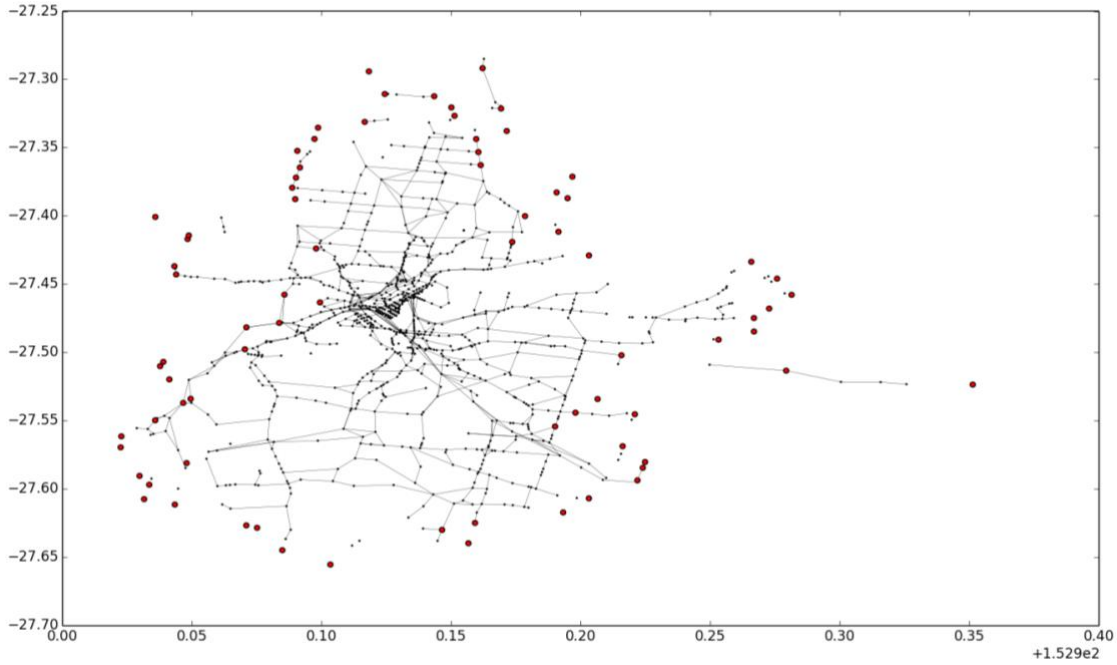



*Figure 5 Representation of the boundary with 40 divisions*

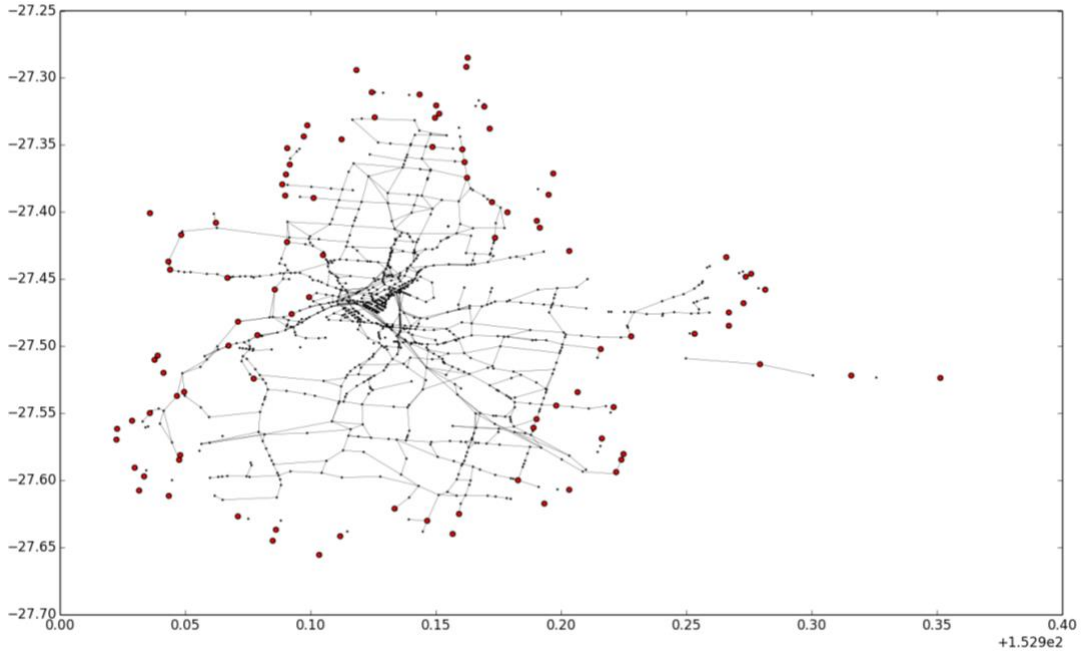



*Figure 6 Representation of the boundary with 40 divisions*

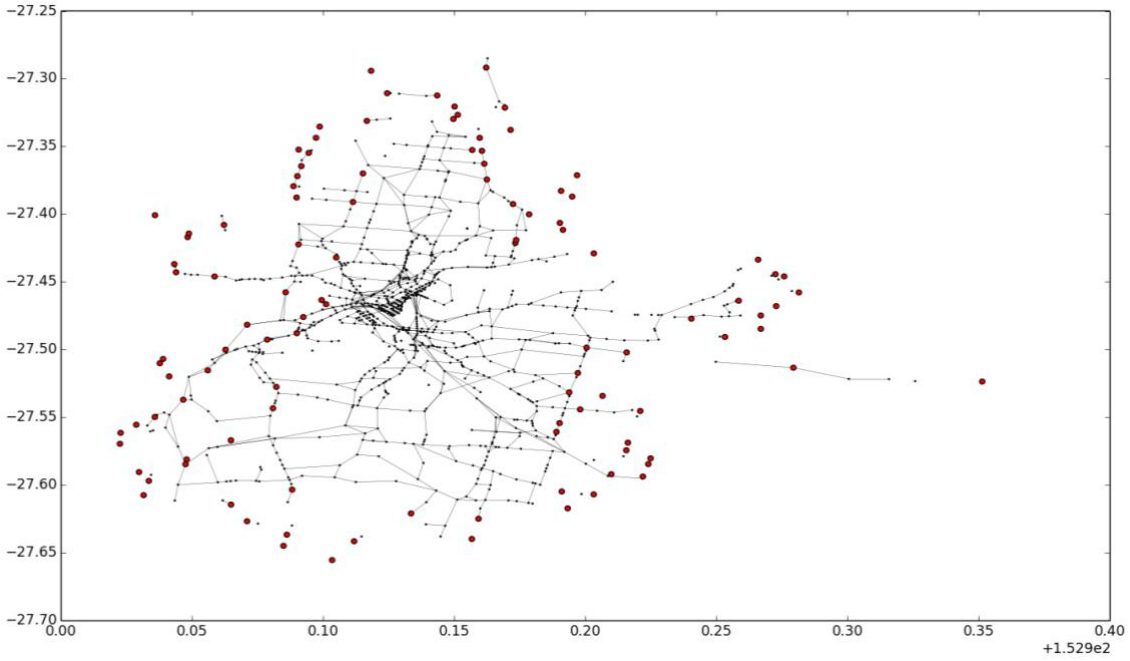



*Figure 7 Representation of the boundary with 60 divisions*

As mentioned in the introduction of this chapter, the origin destination matrix is also needed as input data for the algorithm proposed.

Once defined the boundary of the BT sensor area, the OD matrix is built by the storage of the first and the last time each device appears in the csv, as long as neither the origin nor the destination are one of the sensors contained in the boundary.

In the origin destination matrix it is stored the location in which the device is first and last detected, in the time range considered to analyse.

## 5.2. Network representation

The Bluetooth sensor area and the flow of devices going through them are represented in an undirected weighted graph comprising nodes and edges. Graphs are mathematical structures amounting to a set of entities in which some pairs of the entities are in some sense related.

In this specific case of research, nodes consist of sensors spread around the city and its surroundings, whereas each edge represents a connection between two sensors.

The existence of one edge implies that a movement between the two nodes connected is possible. Hence, knowing connections enables to find whether it is possible to reach a node from another node within the graph. In this case of study, edges are weighted with just one attribute, which corresponds to the length of the road segment.

Although having enough information in the csv file to build a directed graph, it has been considered that for this research an undirected graph is more useful. This is due to the fact that the objective of the algorithm to be designed is to find the optimal route for bringing the maximum number of people from their origin to their destination, but the resulting route could be travelled either one way or the opposite.

A weighted graph has been used in order to assign attributes to the edges in the network, with the length of each link denoting the attribute of that edge.

The networkX library (Python Graphing library), which is a package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks, has been used for the building and analysis of the graph in this case of research.

### 5.2.1. Zoning scheme

From the data in the csv file it can easily be extracted the ID of the different sensors spread all over the city of Brisbane and the outskirts, as well as their location. When displayed in the graph, the X and Y coordinates correspond to the longitude and latitude, respectively, of each sensor.

In most transport planning studies, it is needed to define a zoning scheme into which the study area is sectioned and the corresponding space is discretized **[2]**.Hence, in order to build the graph that will be used as input data in the algorithm proposed in this project, a previous division of the study zone into different travel analysis zones (TAZ) has been carried out. Each section is associated with one only point known as centroid, which gathers all the links going in and out of the nodes located in the corresponding TAZ.

In this specific case of study, the area has been homogeneously sectioned in squares

of 0.5 km side length. This has been decided according to various studies that conclude that the radius of influence of a public transport line stop is between 0.3 and 0.5 km **[3]** which means that people could walk until half a kilometre to access the public transport line.

Below it is showed the distribution of nodes among the network. In the first figure (*Figure 8)* it is observed the node network before having performed the division into travel analysis zones, whereas in the second one (*Figure 9)* is showed the distribution of centroids among the area of study.

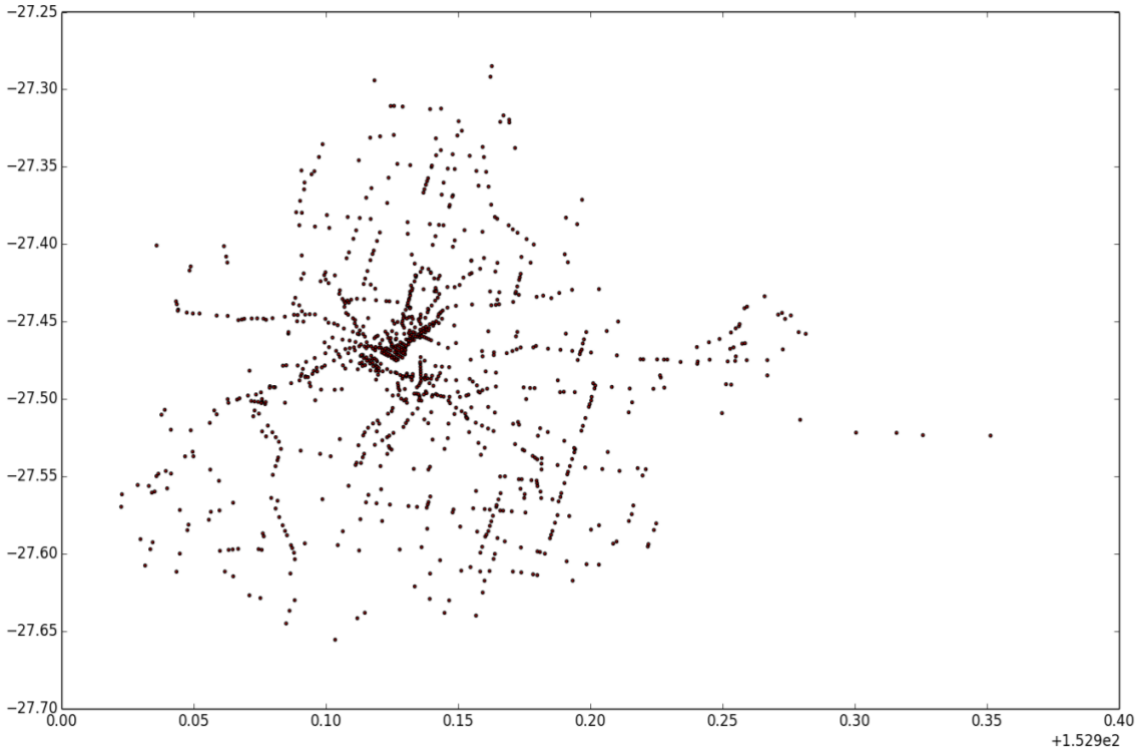



*Figure 8 Node network before performing the division into TAZ*

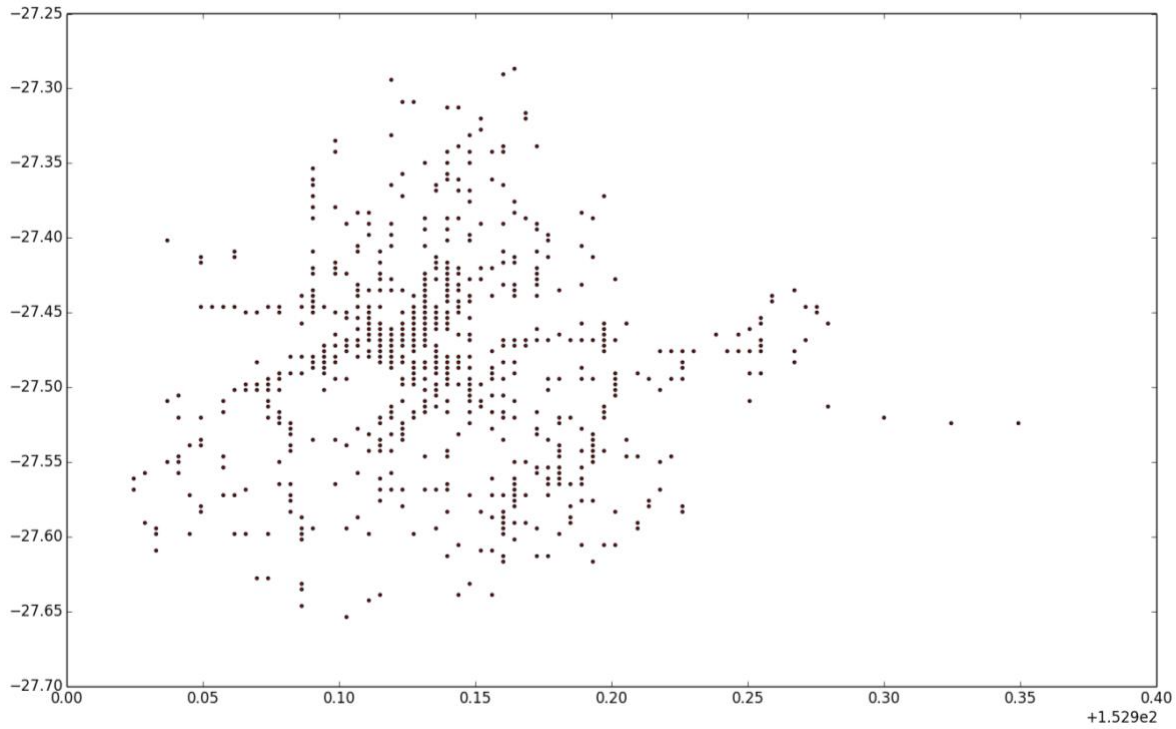



*Figure 9 Node network after performing the division into TAZ*

As mentioned before, there are in total 1000 BT sensors spread over the city of Brisbane. Hence, the graph representing the original network (*Figure 8*) is constituted by 1000 nodes. When the division into TAZ is performed, the resulting graph is the one showed in *Figure 9*, in which the number of nodes is reduced to 674 nodes.

Being the objective of this research to design an algorithm which is able to find a route that brings the maximum amount of people from their origin to their destination, the fact of having the nodes grouped into centroids also introduces the concept of the radius of influence. This ensures that, even if the origin or destination of someone is not a sensor in the route proposed, if it is in the same TAZ, the route proposed will also be suitable for that individual.

### 5.2.2. Data format

The python standard library NetworkX has been used to create a topological representation of the city of Brisbane and its outskirts.

The format in which data in the graph is presented is the following:

- Nodes data which contains the node identifier and X, Y coordinate of the nodes: {Node_label: (X-coordinate, Y-coordinate)}
- Edge data which consists of the start and end nodes of that link, in addition to the link weight: [(Start_node, End_node, length of the edge)]

Regarding the OD matrix, the data corresponding to the travel demand between different origin and destination pairs of nodes in the network is stored in a python dictionary format. The dictionary consists of keys which are the node labels, and values that are other dictionaries containing the number of trips between the key value and each other node. For better understanding, a simple example of how the dictionary is build is showed in Appendix 1.

In the dictionary it is not specified whether the trip is one way or the other. This is because this information is not relevant for the model proposed, as explained in section 6.1.

## 5.3. Analysis and data treatment

### 5.3.1. Filters applied

The data used for building the database in the model of research is extracted from the csv file mentioned in previous sections. Each line of the file contains the ID of the device captured, the day and time of the detection being stored, the duration of the device staying in the location of the sensor capturing it, the sensor ID and the latitude and longitude where the sensor is located, and finally the owner of the sensor ID.

In order to build the database, a loop through the file needs to be performed, storing the data if required. Some of the data are discarded in this loop, due to their non-sense or unfeasibility. Outstanding cases that have been considered to study further, being necessary to perform some treatments on them, are detailed below.

Furthermore, some of the data in the csv don't need to be stored because they are not relevant for this case of study, such as the owner ID and the duration of the detection. The duration of the detection, as it will be explained in the following section, is used to

decide whether the line of the file should be stored, but after checking that, it is not needed in the proposed model.

As previously explained, once the area of study has been defined and the boundary has been determined, the routes starting or ending in the borderline have to be removed. When this restriction is imposed, the number of devices to analyse is reduced from 179.361 devices to 112.608, which corresponds to a 37,21% decrease.

Moreover, for this case of research, it has been chosen to study the data in the file for a specific time range, which corresponds to the range between 6am and 9am. This is because the aim of the algorithm in this project is to come up with a public transport line that reduces traffic, and the period of time mentioned corresponds to the time range with more devices moving among the city.

In the following figure (*Figure 10*) it is shown the number of devices captured by the sensors every hour:

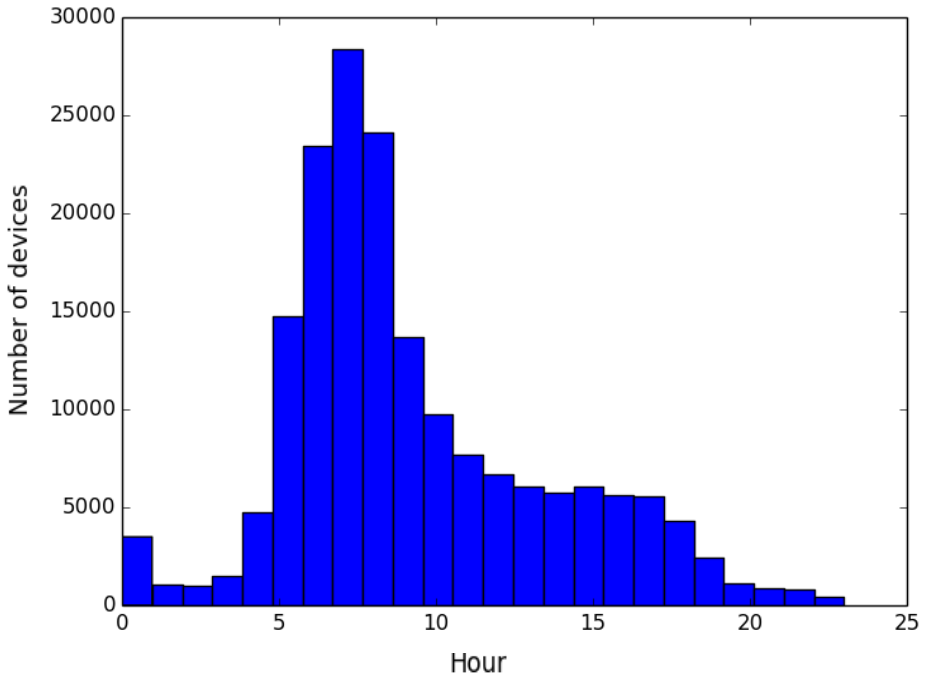



*Figure 10 Histogram that shows the number of devices captured every hour*

As observed in the above figure, in the period between 6am and 9am there is a higher volume of devices moving through the sensor area.

As mentioned, the total number of devices detected in the current case of study (taking into account the boundary and its corresponding restriction) in a whole day period is 112.608 devices. When considering just the devices detected during the period between 6am and 9am, the number of devices studied in this research decreases to 55.385 devices. This corresponds to the 49,18% of the total amount of devices, which is why this period has been considered to be the time range with a higher intensity of traffic, and consequently will be the period studied in this case of research.

### 5.3.2. Data treatment

There are some cases where the sensor stores incorrect data. This could be due to interferences while the measure is being taken or other causes.

Some remarkable misdetections are, on the one hand, extremely long duration detections, and, on the other, devices being detected first in one BT sensor and next in another one just across the city, without having been detected at any point in between, which is, in most cases, impossible. This would lead to having infeasible routes in the database, which is why actions have been taken in order to avoid inconsistent data in the database used for the model. Hence, the lines of the csv containing durations higher than 500 seconds have been removed, assuming that, even when there is a traffic jam, sensors are close enough to not spend more than 8 minutes being detected by the same sensor without moving.

When introducing the condition of the duration, the volume of data analysed is reduced from 55.385 devices to 55.058, which represents only a 0,59% decrease. The low percentage obtained reaffirms that the extremely high durations correspond to anomalies in the set of data, and probably are the consequence of a misdetection. For this reason, it has been decided to remove the lines of the file that do not meet the duration requirements.

It has also been removed the entire routes containing a link between sensors longer

than 4km, taking into account that in the set of sensors the distances between them are much lower (in the city centre each node is surrounded by other nodes with an average distance of half a kilometre). Thus, it has been given a margin of 4km distance between nodes to consider that a higher distance would not be possible in the current network.

The introduction of this requirement implies a decrease of the quantity of data analysed, specifically it is reduced from 55.058 devices to 44.026, which corresponds to a 20,03% decrease.

Moreover, it has been analysed the number of sensors included in each device's route. It has been found that many devices are just detected once or twice in the entire network, which means the device has barely moved or not moved at all among the sensor area. This fact could be a consequence of the BT device being turned off, which precludes the entire route of the device from being studied, and could affect the optimization by taking into account unreal data for getting to a result. For this reason it has been decided for those devices to be excluded from this research.

The consequence of imposing this constraint is the reduction of the number of devices object of study, specifically from 44.026 devices to 29.775. This represents a 32,36% decrease, which is a high percentage of data loss. However, it should be taken into account that the data loss is being compared to the volume of data obtained after the imposition of many other restrictions, which leads to a higher percentage than if it was compared with the initial volume of data. Nevertheless, it has been considered that with studying the routes of almost 30.000 devices it is possible to obtain reliable and consistent results.

In *Figure 11* it is showed a histogram that states the number of sensors visited by the devices analysed. It is observed that, as mentioned, a relevant number of devices are just detected by one or two sensors in the entire network. It is next graphed in a histogram (*Figure 12)* the same data after the removal of those devices.

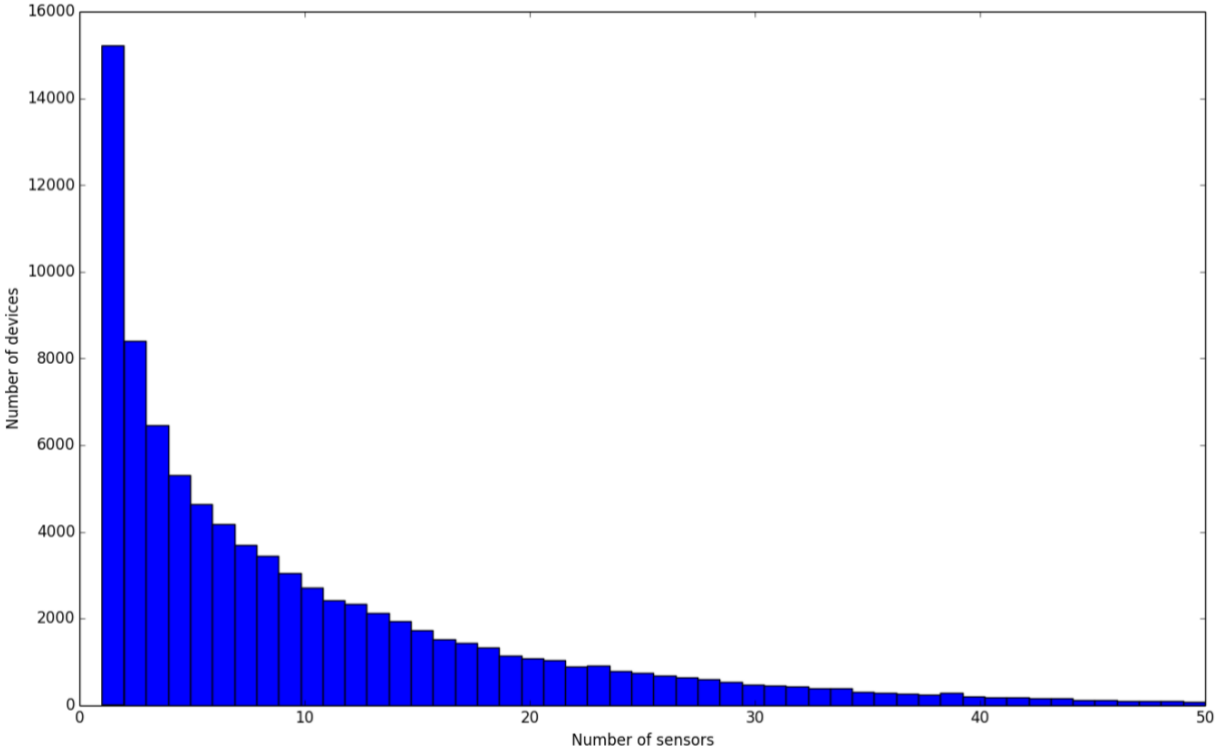



Figure 11 *Histogram that shows the number of sensors visited per device*

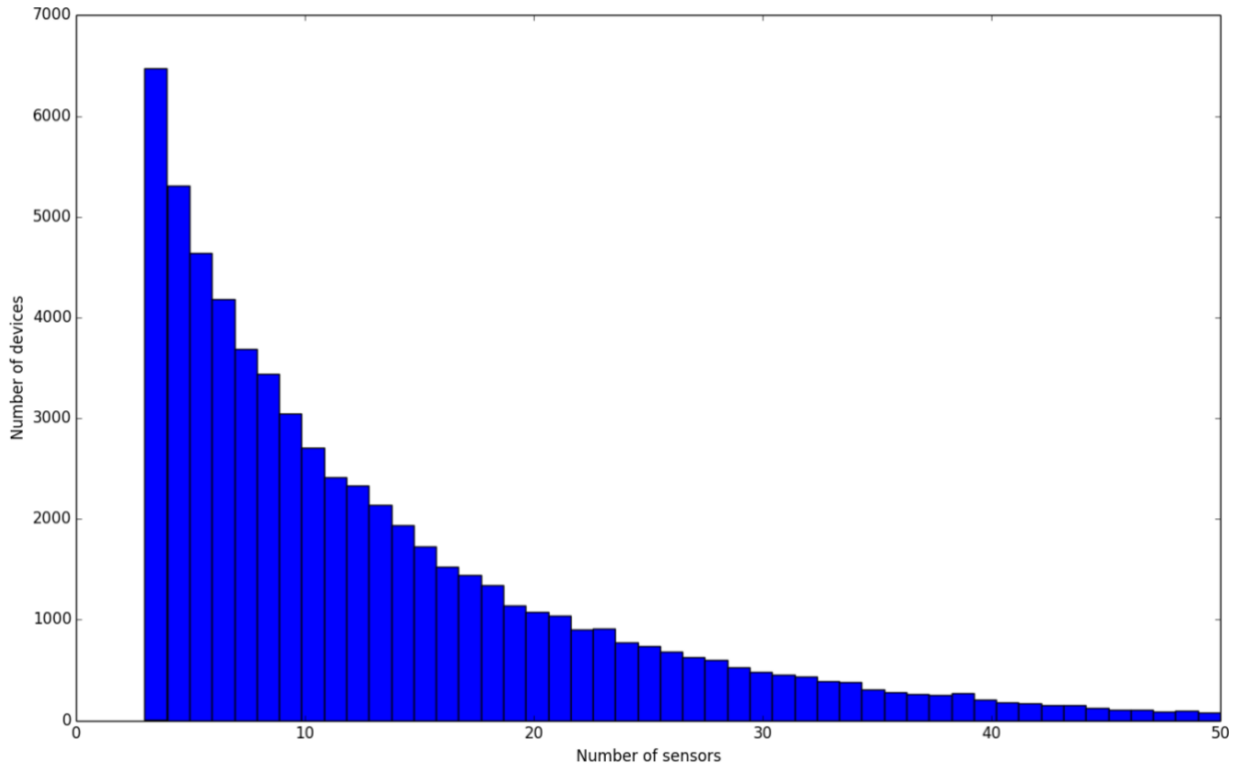



*Figure 12 Histogram that show the number of sensors visited per device after removing*

*less than two sensors*

As observed, the number of devices notably decreases and the distribution of number of detections for each device is more uniform.

Additionally, it has been found in the original file that there is a remarkably high number of links which have too few devices going through them. This is a consequence of devices not being detected in some sensors, which leads to a route containing links between sensors that are barely repeated in other devices' routes. Below it is graphed a histogram (*Figure 13*) that shows the quantity of devices passing through the links between sensors. As observed, a notably high degree of links has a flow of less than 10 devices, which is considered to be too low.

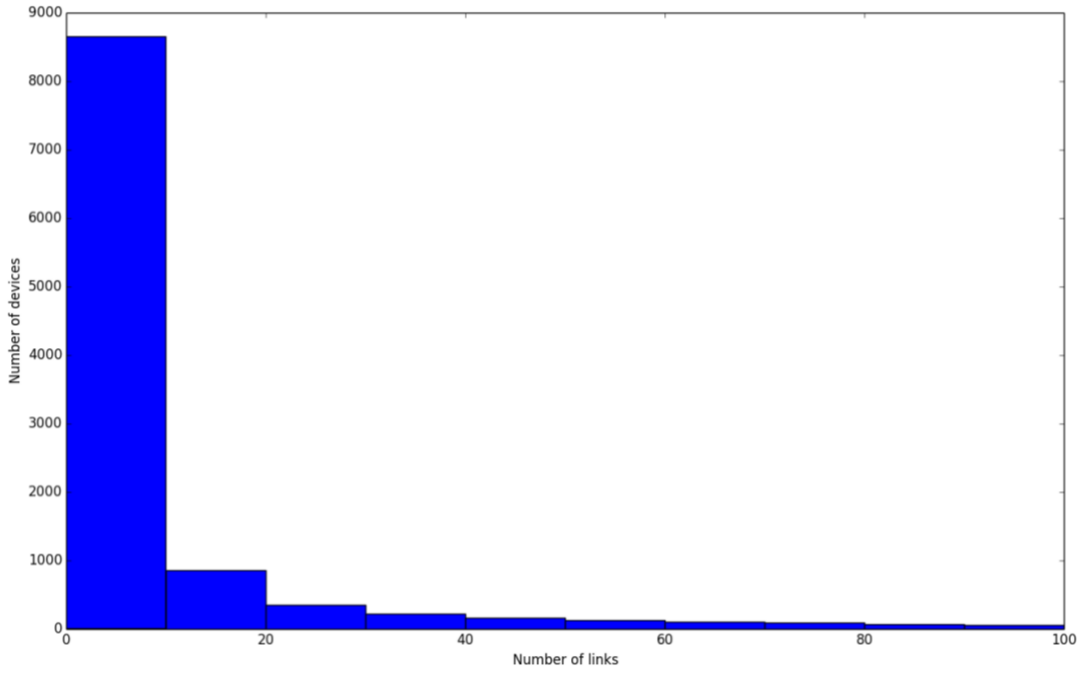



*Figure 13 Histogram that shows the number of devices that go through the sensors*

The result from imposing this condition is a decrease in the number of edges in the graph. The initial number of edges in the network (after setting the requirements mentioned above) is 6.642 edges, whereas after removing the edges with a flow under 10 devices the number of edges turns to 1.810 edges.

Twenty devices passing through an edge linking two sensors, considering the

extension of the population analysed, is still considered to be low, so a study of the resulting graph with the removal of those links has been carried out.

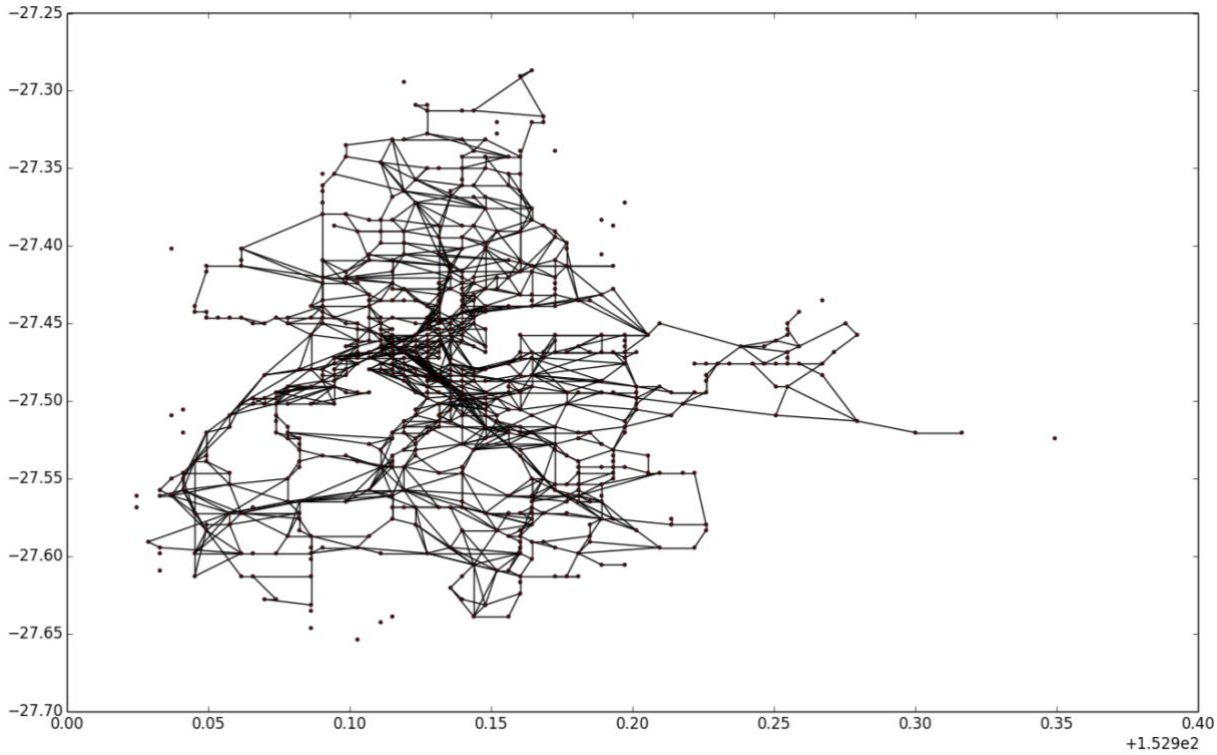



*Figure 14 Graph showing links with a flow of more than 10 devices*

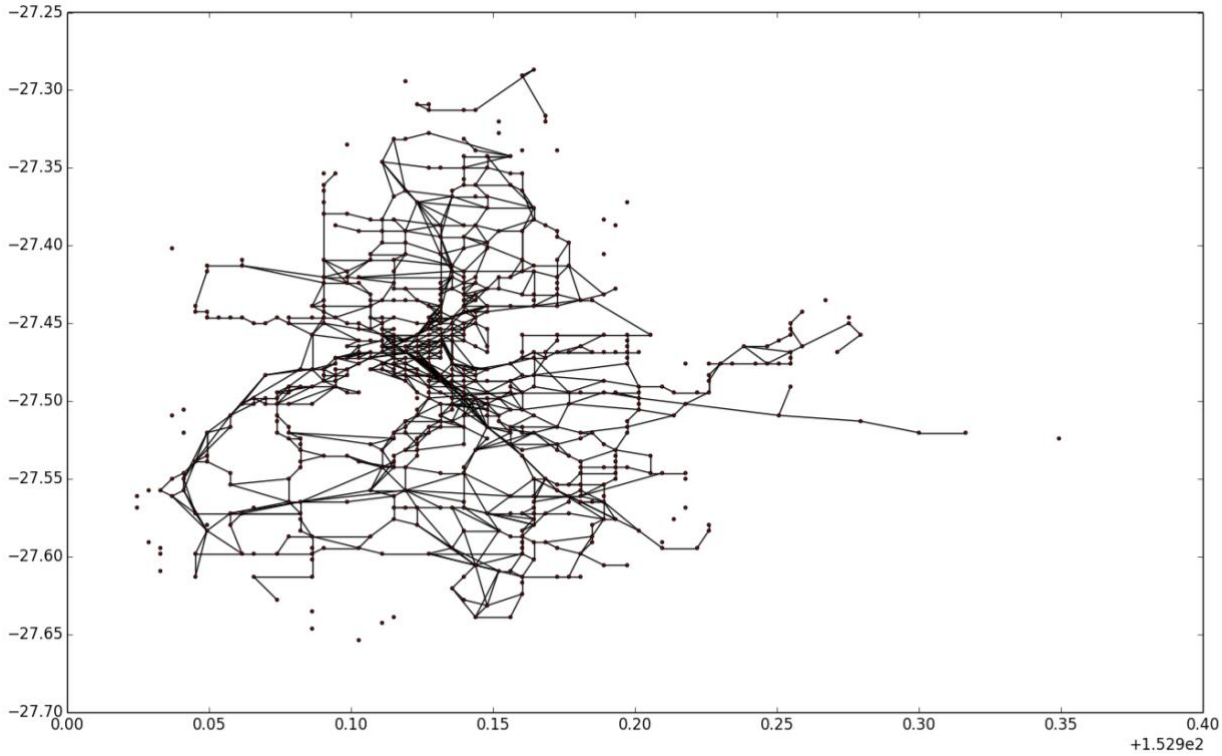



*Figure 15 Graph showing links with a flow of more than 20 devices*

The figures above show the resultant graphs with the removal of edges mentioned. In the first one (*Figure 14*), links with a flow of less than 10 devices have been removed, whereas in the second one (*Figure 15*) it has been the ones with a less than 20 devices flow that have been deleted. It is observed that concerning the graph, this action affects mainly to the edges in the centre of the graph, which correspond to the city centre. This fact makes it more likely to consider that 20 devices flow links should also be removed, due to the fact that the vehicle flow in the city centre is intensive, which reaffirms that the edges with so few devices flow are consequence of a misdetection.

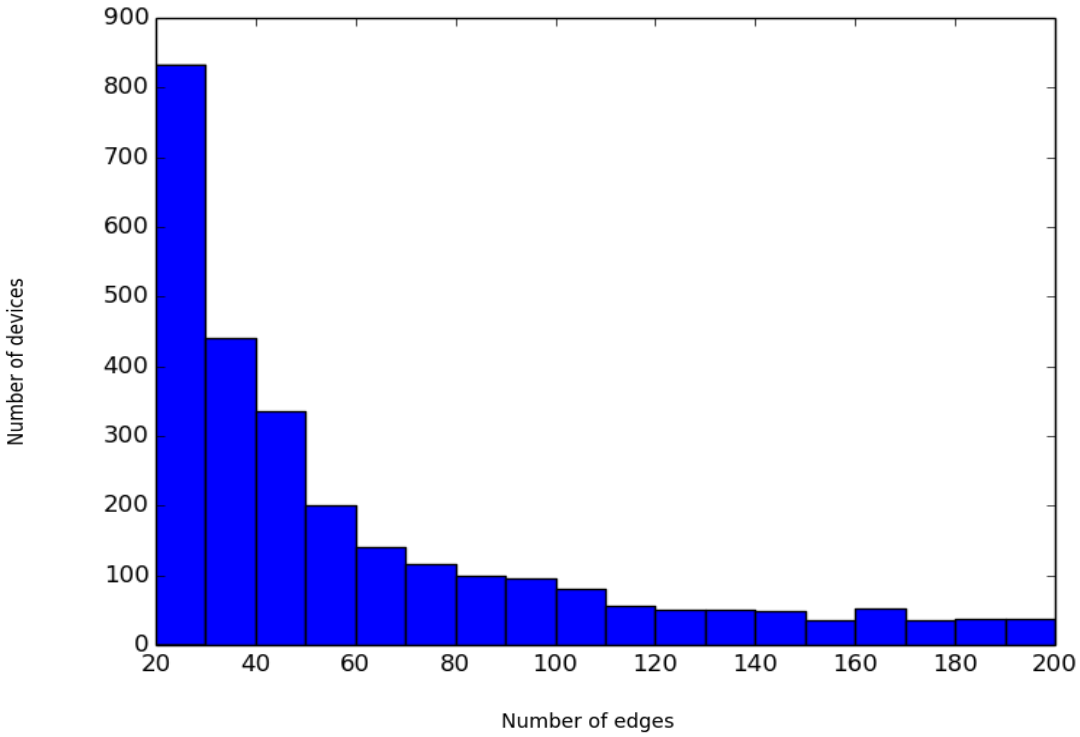



*Figure 16 Histogram that show the distribution of number of devices per edge*

The above histogram (*Figure 16*) shows the distribution of number of devices per edge, after the removal of the links with a flow lower than 20 devices.

# 6. Algorithm selection

There may be several solution approaches to solve this thesis. Conventional and heuristic models are two of them, and they are described below. A development and improvement of heuristic models, called metaheuristic models, are also explained in this chapter. Three of these algorithms namely Genetic Algorithms (GA), Simulated Annealing (SA) and Tabu Search (TS) are introduced and compared. Finally, the choice of genetic algorithm for this research is justified.

## 6.1. Conventional and heuristic models

Conventional models are the ones that use analytical solution techniques to solve a problem. Such models are useful to get one or several parameters such as route length or number of persons going from origin to destiny, but they are incapable of simultaneously defining the network configuration and fixing service parameters. They usually consist of an objective function and several constraints that allow the problem to reach to an optimum solution.

The main limitation that would have solving the current case of research with a conventional model is that, due to its complexity, it would take an extremely long time and would still not guarantee that the solution found would be a global optimum and not a local one.

Furthermore, most analytical optimization techniques are applicable only to idealized problem, rendering them unable to solve realistic problems. This led to the development of heuristic techniques, which are better suited to solve such realistic network problems.

Heuristic models enable the generation of a network and the computation of its service parameters simultaneously, which is not possible in conventional models. They were developed as a result of the several limitations that conventional models supposed. However, heuristic models are problem specific, which means that can be used to solve specific problems in which they are applied. A heuristic algorithm applied to a

problem, will therefore not find a broad-based application in other similar problems.

As a result of the limitations of heuristic models, meta-heuristic algorithms were developed to improve them and enable algorithms to be applied for more than one problem.

## 6.2. Meta-heuristic models

A metaheuristic is a higher-level heuristic which can be applied for a wide variety of problems. Talbi **[4]** asserted that they are problem independent and are capable of being applied to a wide range of optimization problems, by adapting their parameters to the features of a problem to be solved. Moreover, they are able to find the global optimum and are able to avoid getting stuck at a local optimum in the process.

As it will be explained in later chapters, the objective function of this project will be maximized. The situation in this case of study is similar to the one that can be observed in *Figure 17*, which consists of one global maximum and several local maximums. As the algorithm searches for an optimal solution, it encounters a local optimum solution, which is temporarily held until a better one is found. For converging towards the global optimal solution, poorer solutions may have to be accepted, which will prevent the algorithm from getting stuck at a local maximum.

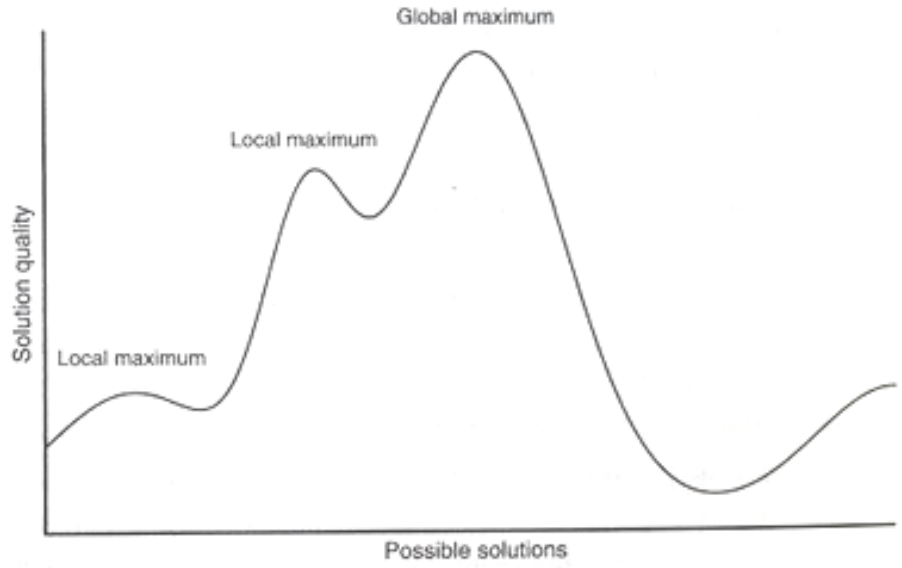



*Figure 17 Representation of the search of a global maximum*

Three metaheuristic algorithms have been studied for this project: Tabu Search, Simulated Annealing and Genetic Algorithm. A detailed explanation and a comparison of the three of them can be found in the following chapters, as well as the decision of the metaheuristic model that has been used in the project.

### 6.2.1. Tabu Search

The word tabu (or taboo) comes from Tongan, a Polynesia's Language, where it means something that cannot be touched because it is sacred. Nowadays, it is used to refer to something that is prohibited or restricted by social or religious custom.

The approach of this algorithm is to avoid entrainment in cycles, by forbidding or penalizing moves that take the solution to points previously visited. Its main objective is to find the best solution, and so it proceeds, according to the supposition that there is no point in accepting a poor solution when there is a better one, unless that one has already been previously analysed. By doing this, it is sure that new regions of the space will be investigated with the goal of avoiding local minimum and finally finding the best and desired solution.

Tabu Search begins by moving to a local optimum. To avoid turning back to solutions that have already been studied, previously selected solutions are marked as forbidden in the Tabu List. Some characteristics of the algorithm include diversification and intensification strategies. Diversification enables the algorithm to maintain the uniqueness of each new solution generated from the previous, while intensification enables the analysis of promising areas of the search space. A flow chart for a basic Tabu Search is shown in *Figure 18*.

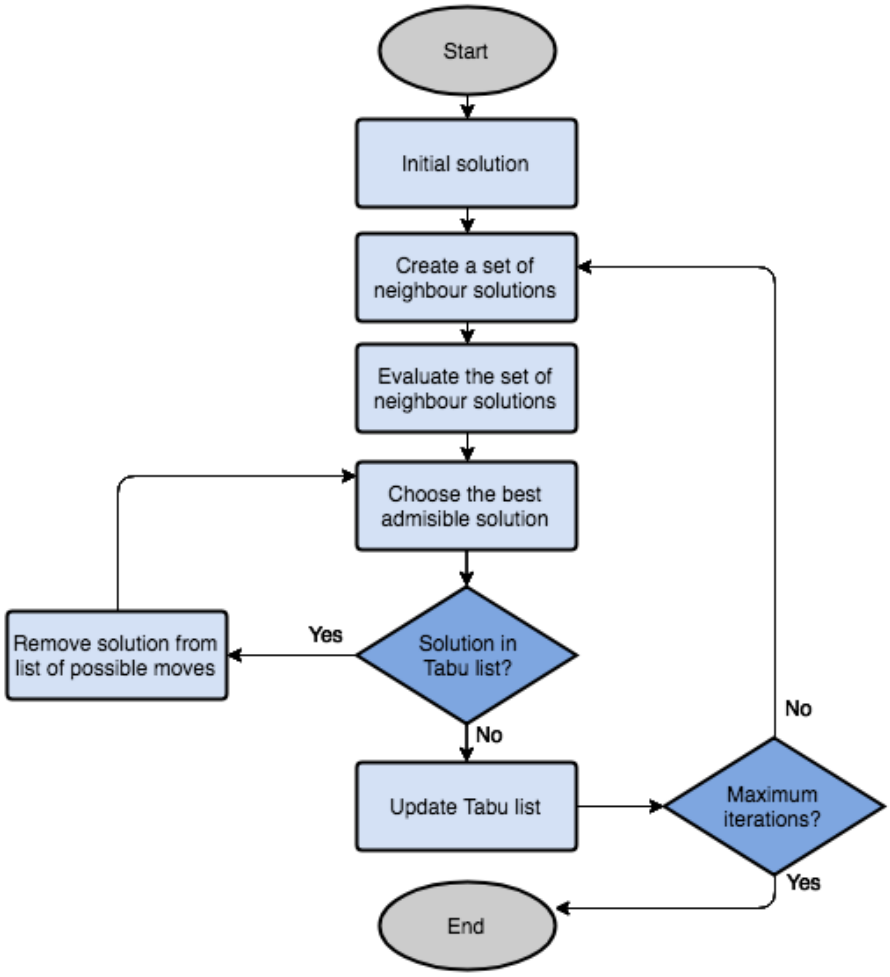



*Figure 18 Flow chart of Tabu search*

If Tabu Search would be the algorithm chosen for this research, random routes would be generated in each iteration, and its best route would be saved in the Tabu List.

### 6.2.2. Simulated Annealing

Simulated annealing is a meta-heuristic algorithm to approximate global optimization in a large space for an optimization problem. It was inspired by the annealing process in metallurgy. Annealing involves heating and cooling material to alter its physical properties due to the change in its internal structure. As the metal cools its new structure becomes fixed, causing the metal to retain its new properties.

The extrapolation to metallurgy process can be observed in the gradual decrease of the probability to accept worse solutions, which corresponds to the temperature of the algorithm's cooling schedule as it iteratively explores the solution search space. Fletterman **[5]**, states, on the one hand, that a first large probability of accepting non-improving solutions enables the algorithm to jump out of any local optimum, and on the other hand, that the slow decrease of this probability allows the generation of a final solution with an objective function value which hopefully will be the global optimum.

In the *Figure 19* it is shown the flow process for a basic simulated annealing, where $T_o$ is the initial temperature, $T_{reduce}$ is the temperature reduction factor, $T_{count}$ refers to the number of iterations done at each temperature and $T_{min}$ is the temperature needed to be attained before finishing the algorithm.

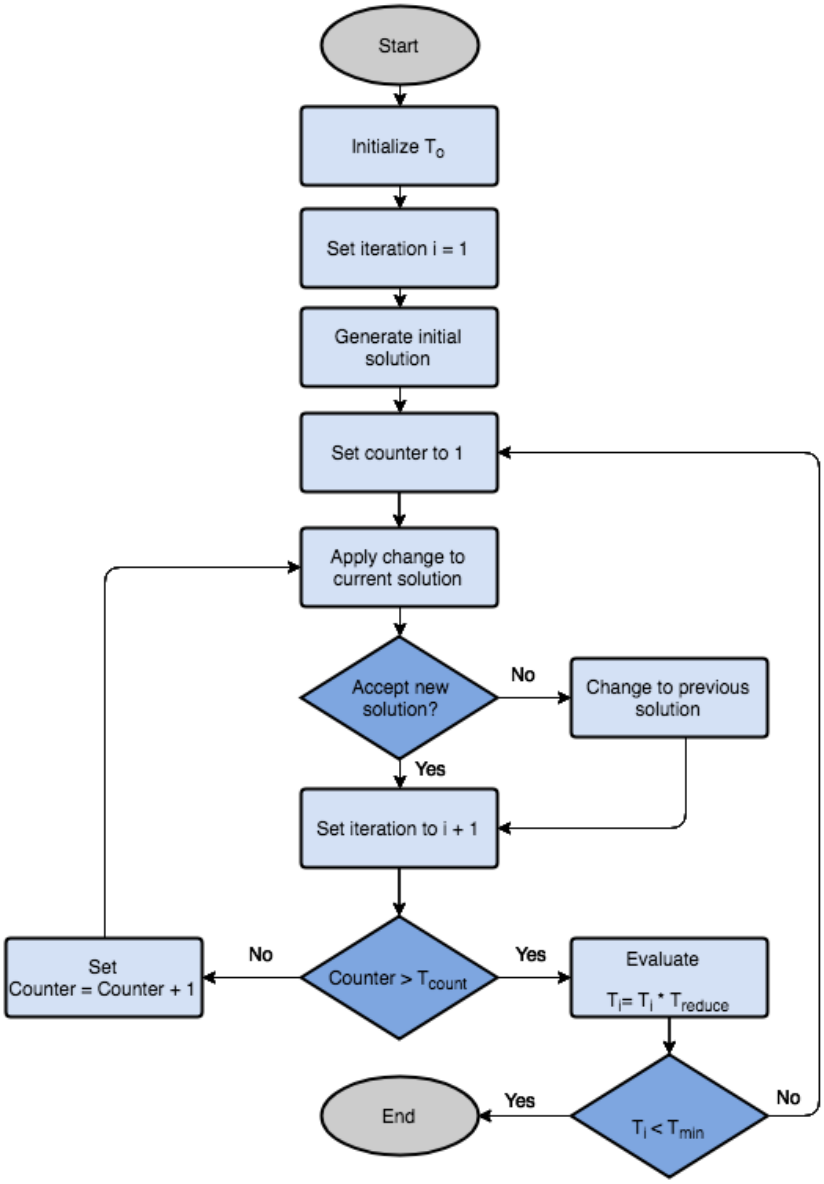



*Figure 19 Flow chart of simulated annealing*

### 6.2.3. Genetic Algorithm

Genetic algorithm is a metaheuristic used for solving both constrained and unconstrained optimization problems, based on Darwin's theory of evolution, which introduces the concept of natural selection. Darwin's theory of evolution states that, as random genetic mutations occur within an organism's genetic code, the beneficial

mutations are preserved because they aid survival **[6]**.

Genetic algorithm is a random-based type of evolutionary algorithms commonly used in optimization and search problems to generate high-quality solutions. The algorithm starts by producing a first population of random possible solutions, taking into account the restrictions imposed. Each potential solution is known as individual, and each individual consists of a chromosome. The chromosome consists of a sequence of genes that define the individual (*Figure 20*).

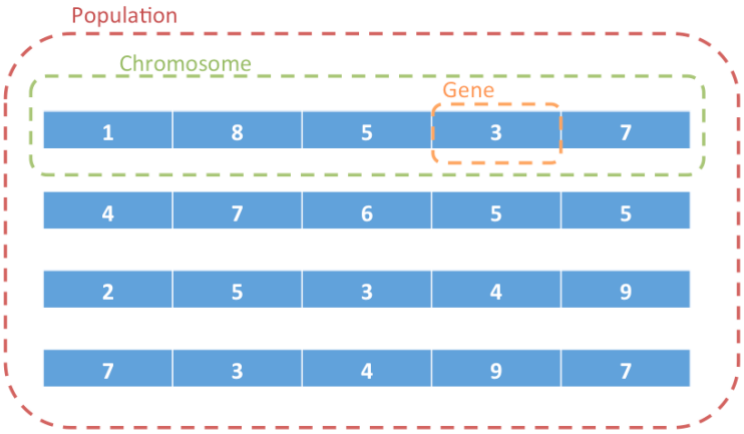



*Figure 20 Structure of genetic algorithm*

Each individual is evaluated with a fitness function, obtaining a fitness value. The fitness value represents the quality of the solution, thus it is taken as decision criteria for the selection of the best individuals.

Once obtained the first population, it is evolved toward better solutions by an iterative process, with the population in each iteration known as a generation. The process for obtaining the population in each generation consists mainly of three phases: selection, crossover and mutation.

In the first stage, part of the existing population is selected to breed a new generation. As mentioned before, this is a fitness-based decision, which means that fitter solutions are more likely to be selected. Individuals selected from the population are known as parents, and the set of them as mating pool.

The next step is the creation of new solutions by the combination of pairs of randomly designated parents from the mating pool. This is the process known as crossover. Every two parents selected from the mating pool will generate two offspring. By the combination of genes from high-quality individuals, it is expected to get a high-quality offspring. In the following figure (*Figure 21*) it is showed an example of one type of crossover.

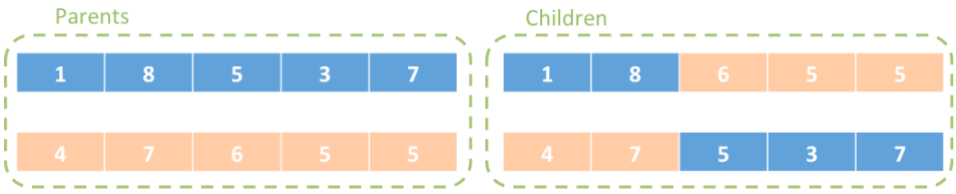



*Figure 21 Example of a single point crossover*

The last stage in the generation of a new population is the mutation. The resultant offspring of the crossover process is only carrying the genes of its parents, there is no new genes added to it, and thus it is needed the mutation in order to maintain genetic diversity from one generation of chromosomes to the next. Mutation alters one or more gene values in a chromosome.

The set of individuals after the mutation results in the population of the new generation. This process is repeated from one generation to the next one until the termination criteria is met, which can be a fixed number of generations reached, the finding of a solution that satisfies minimum conditions, etc.

In the current route optimization problem, each route generated represents an individual. Hence, after the corresponding iterative process for altering each set of individuals, the route with the best attributes represents the global optimal solution.

In *Figure 22* below, it is showed the flow chart for a basic genetic algorithm.

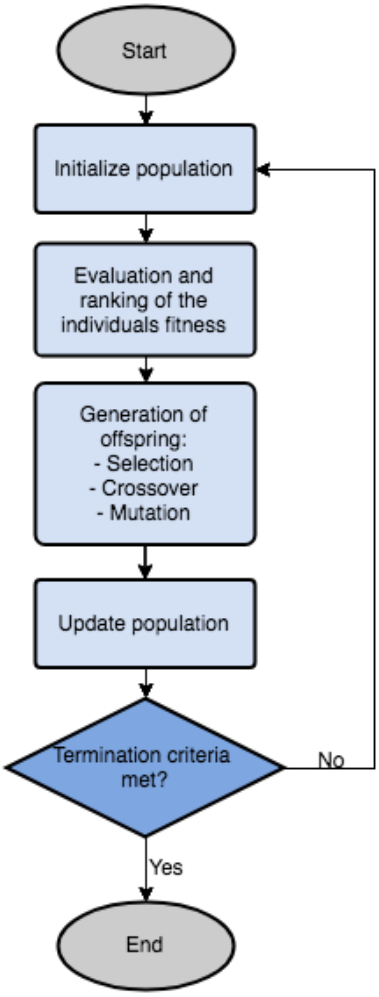



*Figure 22 Flow chart for a basic genetic algorithm*

### 6.2.4. Comparison and decision

The three algorithms mentioned before are optimal meta-heuristic models with a common objective of getting to a global optimum and not getting stuck in a local optimum. However, their behaviour is completely dependent on the nature of the problem to be solved and depending on it there may be one that works more appropriately than the others.

Talbi **[3]** states that Genetic Algorithm is a population-based meta-heuristic which maintains and improves multiple candidate solutions, often using population characteristics to guide the search, while Tabu Search and Simulated Annealing are single-solution meta-heuristics which modify and improve a single candidate solution

at each iteration.

It should also be pointed out that Genetic algorithm and Simulated Annealing are stochastic meta-heuristics algorithms because they both use probabilistic variables, like the probability of acceptance in simulated annealing, and crossover and mutation in genetic algorithm. It does not happen the same with Tabu Search, which is a deterministic algorithm that would produce always the same output given a particular input.

Tabu Search has been the first discarded algorithm in this project, because of its deterministic character. Working with genetic algorithm or simulated annealing ensures that every time the program is run, a different and maybe better solution will be obtained, which does not happen with Tabu Search, that would give the same output for a same input. A stochastic algorithm is necessary in this research because the random variables give reliability to the process, making it less probable to get stacked in a local optimum.

Genetic algorithm improves in each iteration multiple candidate solutions, whereas simulated annealing works with a single candidate solution at each iteration. This is an advantage the Genetic Algorithm has over Simulated Annealing, as it allows Genetic Algorithm to perform more robust and pervasive search across a very large solutions space and in faster time. Furthermore, when it comes to decide between genetic algorithm and simulated annealing, it has to be mentioned that genetic algorithms have been used to solve several Bus Transit Network Design problems, see **([7]**, **[8]** and **[9]**

For all the reasons mentioned above, genetic algorithm is the meta-heuristic model chosen for this research.

# 7. Development of the algorithm

## 7.1. Objective function

Transit network design is a complex combinatorial optimization problem, where the main goal is to achieve an optimized public transport network. In the last years, it has been object of extensive literature, which can be divided according to two different approaches: on the one hand, analytical models supplying a synthetic analysis of network performances, and on the other, optimization processes for solving the network design problem by determining the transit routes and associated frequencies and locating main transit centres **[10]**.

In this specific case of research, the objective is not to design a full transit network but to design an algorithm able to find the optimal location for a public transport line, which would imply modifying the existing transit network and thus would have an impact on it. The approach chosen for the proposed model would correspond to the second group in Fusco et al.'s division.

The objective function comprises one only term, which consists of the number of devices having their origin and destination within the route being evaluated. For getting this value, it is needed to access the OD matrix previously built. The length of the routes is taken into account by imposing a maximum route length constraint, so that although this term is not present itself in the objective function it is also taken into consideration.

Other factors could be considered in the objective function, such as the cost per trip or travel time; however, their inclusion in the objective function would imply making many assumptions due to the lack of data, and those parameters are not considered to be that relevant compared to the function related to the OD matrix.

Therefore, by maximizing the objective function, it can be found an optimal route for placing a public transport line, bringing the maximum number of passengers from their origin to their destination.

## 7.2. Programming genetic algorithm

### 7.2.1. Generation algorithm

The generation is the first stage of the model implemented. It is a heuristic algorithm that generates random routes in the previously defined network, meeting the route length requirements established. The routes generated in this stage of the algorithm constitute the first population of candidates.

#### 7.2.1.1. Generation input data

The graph representing the network, with the proper modifications and considerations described in 6.3, is used as the key input for the generation phase of the algorithm. The user-defined parameters that affect the generation stage are the number of routes generated (population size), and the minimum and maximum length of the routes.

#### 7.2.1.2. Generation procedure

The generation process consists in randomly selecting a pair of nodes from the network and finding the shortest path between them by using Dijkstra's algorithm. If there's no possible route between those nodes, which can happen due to the fact that some links have been removed (as explained in chapter 6.3), the program looks for two new nodes.

Once the route is defined, it is checked that it meets the length requirements. If these conditions are not met, the route is discarded. The restrictions according to the length of the routes in the generation algorithm are eased. This is due to the fact that the length of those routes is the shortest one between the two selected nodes (because Dijkstra's algorithm has been used to find it), so any change in that route (as long as the initial and final nodes stay the same) will make it longer. Therefore, a margin is needed so that later changes (performed by the genetic algorithm) can be made to the initial routes without immediately getting over the length limit.

In this case, the length established for the routes in the generation function is between 6 and 9km.

Once checked that the candidate route complies with length restrictions, it is stored. This process is repeated as many times as specified by the user, who is able to select the number of candidates by defining the parameter of population size.

The set of routes represent the initial candidate solutions, from which the optimized network will derive from.

#### 7.2.1.3. Generation pseudocode

A pseudo code for the generation is provided below, presenting the steps followed to generate the initial set of routes.

1. Read-in network distribution graph.
2. Randomly select two centroid nodes from the study network.
3. Compute Dijkstra's shortest path between the pair of nodes selected.
4. Compute the length of the generated path.
5. Check if the generated route meets the length criteria. If it does, accept the route. If it doesn't, discard it and loop until a feasible route meeting the requirements is found.
6. Store the generated feasible route.
7. Loop until the user-defined number of routes is acquired.
8. Stop procedure.

#### 7.2.1.4. Generation output

The output of the generation stage of the algorithm is a set of generated routes. The set of routes is returned, talking in Python programming language terms, as a list of lists. The list returned contains different lists, each of them representing one route, which consists of a sequence of node labels through which the route passes.

These routes represent the potential solution search space from where the optimal solution will derive. This set of candidates also represents the main input for the next phase of the algorithm, which will be discussed hereunder.

The flow chart for the generation stage is detailed below (*Figure 23*):

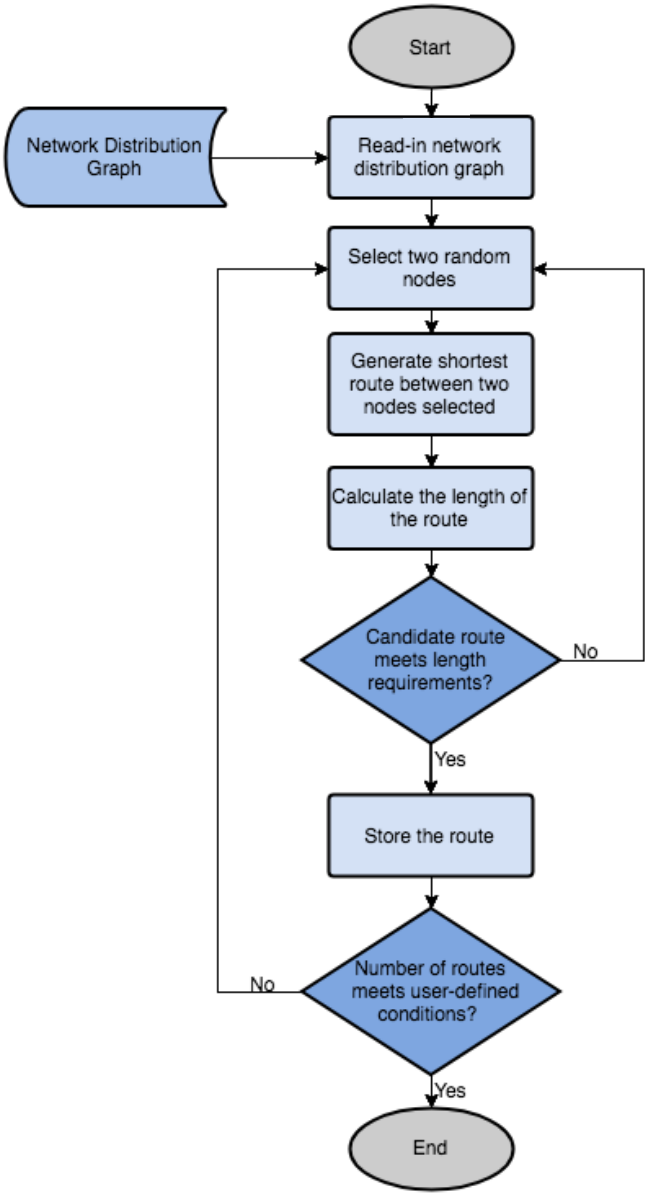



*Figure 23 Flow chart for the generation stage in genetic algorithm*

### 7.2.2. Evaluation algorithm

This phase of the algorithm essentially consists in the evaluation of the objective function for each route selected.

#### 7.2.2.1. Evaluation input data

The main input in this stage of the algorithm is a set of candidate routes. These routes are presented in a format that is readable by the model. Another input is the origin-destination matrix (O-D matrix), which describes people's movement in the defined network, representing the demand of passenger trips between traffic zones in the network.

#### 7.2.2.2. Evaluation procedure

The evaluation process consists in the calculation of the objective function value for each of the candidates.

The aim of this function is to calculate the number of people who have both their origin and destination within the route being analysed. For this to be done, the procedure is to study all the possible combinations of two nodes in the route, and for each combination adding to the fitness the value found in the OD matrix for that pair of nodes. This process is repeated for all the candidate routes.

#### 7.2.2.3. Evaluation pseudocode

The respective steps followed to evaluate the candidate routes can be seen in the following pseudocode:

1. Read-in the candidate routes.
2. Read-in the OD matrix.
3. Evaluate objective function to get fitness values.
4. Loop to evaluate all routes.
5. Stop procedure.

#### 7.2.2.4. Evaluation output

The output of the evaluation stage of the algorithm is the candidate routes fitness values computed from the objective function expression. The fitness values are returned in a Python list format, so essentially the output of this phase is a list of

numbers that correspond to the fitness values, obtained from the evaluation of candidate routes in the objective function.

The flow chart for the evaluation phase is detailed below (*Figure 24*):

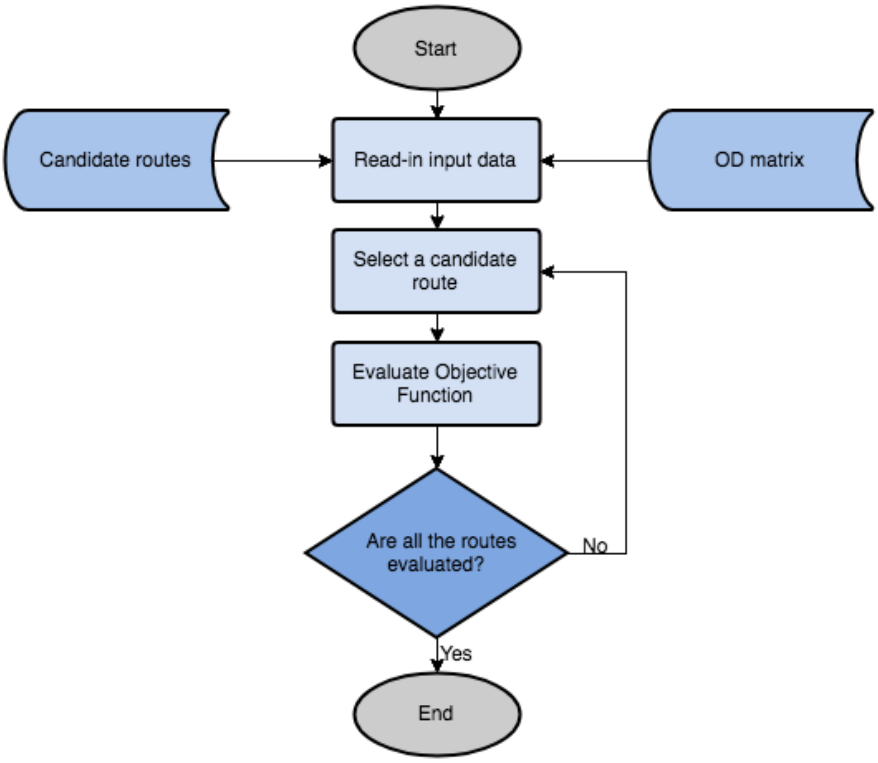



*Figure 24 Flow chart for the evaluation stage in genetic algorithm*

### 7.2.3. Route search algorithm

The algorithm used to search for the best route is based on an Evolutionary Computation strategy of the Genetic Algorithm. The fitness value constitutes the criteria used to rank the candidates.

#### 7.2.3.1. Route search input data

The input data for this phase the network distribution graph. The user parameters to be defined in the route search algorithm are the number of routes to be generated at first, number of candidates selected from each generation, the probability of mutation

and the maximum number of generations.

#### 7.2.3.2. Route search procedure

The route search process starts by calling the two previous algorithms, first the generation and after the evaluation. The first one generates the initial candidate routes, whereas the evaluation calculates their fitness function values.

The generated candidate routes are initialized as the first parent population and each route is assigned its respective fitness value. In each iteration, the algorithm selects a group of routes following a fitness value based criterion. Once selected, the model attempts to improve the fitness values by mutating the corresponding candidates. Each generation contributes to the creation of the following one, known as offspring population. This is achieved by the main operators of the genetic algorithm: selection, crossover and mutation.

A termination criteria is introduced in the algorithm, to ensure it will stop when this criteria is met. In this case, the termination criteria used has been the number of generations, which means that when the algorithm has run as many generations as indicated in the termination criteria it stops.

##### 7.2.3.2.1 Selection

This is the first operator of the genetic algorithm reproduction strategy. Its input is a group of routes and their fitness values, and the number of individuals selected from each generation, which is defined by the user. The objective of this process is to make a selection of a group of candidates on the basis of its fitness value. The better the fitness value, the higher the probability for the route to be chosen.

The selected individuals are referred to in the literature as a mating pool **[11]**. The individuals in the mating pool are called parents. Each generation comes from the evolution of the mating pool selected from the previous one, so it is expected that by selecting high-quality individuals, there will be higher chances to just keep good properties of the individuals and leave out bad ones. Nevertheless, there is a chance that in some cases the optimal solution may come from the evolution of a bad one.

Therefore, the selection operator significantly affects the effectiveness of the search for an optimal route. Fitness proportionate selection, tournament selection and truncation selection are some of the commonly used selection techniques.

In this case, fitness proportionate selection has been used. Also referred to as the roulette wheel selection, it uses the fitness value in order to associate a probability of being selected to each candidate. Being *i* an individual in the population and $f_i$ its fitness value, the selection probability would be the following:

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

Where N is the number of individuals in the population.

While with some other selection operators only the best solutions are kept as parents for the next generations, with fitness proportionate selection there is a chance some weaker solutions may survive the selection process. This is because even though there is a low probability that weak solutions will be selected, it is not zero, which means it is still possible they overcome the selection process.

#### 7.2.3.2.2 Crossover

Overall, the crossover process consists in the combination of two randomly designated parent individuals from the mating pool, in order to generate an offspring population. There are different types of crossover, being the most common the one-point crossover, multi-point crossover and uniform crossover.

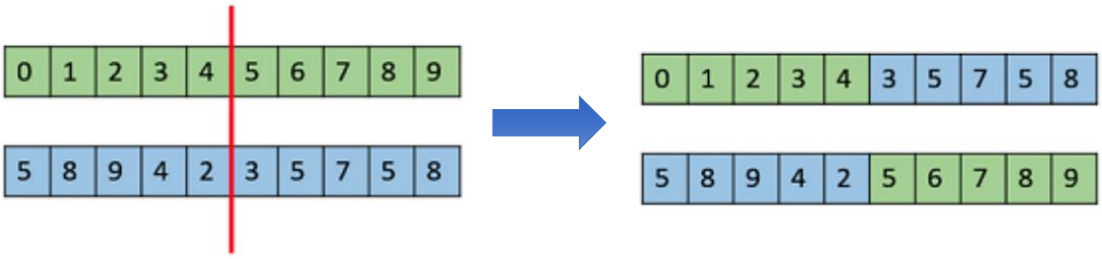



*Figure 25 Example of a single point crossover*

In the previous figure (*Figure 25*), one of the simplest crossover procedures (one-point crossover) is represented. For every two randomly designated parents, crossover takes place by selecting a random point in the candidates and swapping data before and after such point.

The crossover is random-based, and the amount of genes carried from each parent is random. This is the main cause that the performance of crossover in this case of study would imply an extremely long computation time.

In the current case of research, the parents would be two lists of nodes, each list representing a route in the network. The chance of those two routes to be located in the same area of the network is low, but even if it is given the case that they are close to each other, the probability that the combination of the two parts leads to feasible routes is almost non-existent. Therefore, the algorithm would probably get in an infinite loop trying to do crossover but never finding a feasible solution that can be added to the offspring population.

Consequently, the Python Inspyred crossover form used in this research has been the default variation, which simply returns the entire mating pool without any variation. This way the offspring population is composed of feasible routes, allowing the mutation to be carried out afterwards.

#### 7.2.3.2.3 Mutation

Mutation procedure takes place after crossover is performed. It is often used to maintain diversity in the genetic population and is usually applied with a low probability.

Mutation consists in making random changes to the new offspring.

Some commonly used mutation operators are the following: bit flip mutation, swap mutation, inversion mutation. However, mutation depends on encoding and, sometimes, specific mutation made for a specific problem can improve performance of the genetic algorithm. It has been observed that mutation is essential to the convergence of the genetic algorithm while crossover is not. There are many references in Fogel (2006) that support the importance of mutation-based search.

In this particular case, a specific mutation algorithm has been designed for the problem. Its mode of operation is detailed below.

The mutation algorithm gets as input data the population after crossover has been performed. In the current research, as explained in section 6.5.3.2.2, this population is equivalent to the surviving individuals from the selection process. Once this set of individuals has been defined, a percentage of them (depending on the mutation rate) proceed to mutation.

Having a candidate, which comprises a list of sensors making a route in the network, mutation's first step is the random choice of two nodes within the candidate route. The nodes from the route located in between the selected pair of nodes constitute the part of the candidate to mutate. The mutation itself consists in finding in the network distribution graph another possible route between those two nodes. Dijkstra's shortest path algorithm is used to achieve this. However, taking into account that the first population of routes is generated using the same algorithm (Dijkstra), if any changes were made, the result would be the same path as in the current route. To avoid this, a previous removal of the nodes between the pair of nodes selected and their respective edges needs to be performed in the graph.

Thus, the mutation procedure could be summarized in the following steps:

1. Select from the population a candidate route for mutation.
2. From this route, randomly select two nodes.

3. Make a copy(1) of the original network graph.
4. Remove from the new graph the nodes in between the selected ones and their respective edges.
5. Compute, in the new graph, Dijkstra's shortest path algorithm between the selected pair of nodes.
6. Insert the path obtained into the first route, between the two nodes selected in step 2.
7. Check that the new route meets the length constraints and does not already exist in the population. If these two conditions are met, store the new route in the offspring population. Else, discard it and loop until a feasible route meeting the requirements is found.
8. Loop until the number of routes mutated meets the requirement specified by the mutation rate parameter.

Below, it is showed an example of the mutation of a route. As observed in *Figure 26*, the nodes and edges that configure the path in *Figure 27* (referred to as "part of the route to be mutated" in the figure) have been removed, which allows the model to find a new path between the two selected nodes using Dijkstra's shortest path algorithm.

(1) A copy needs to be made in order to not modify the original graph, which would affect the other mutations and the following steps of the genetic algorithm

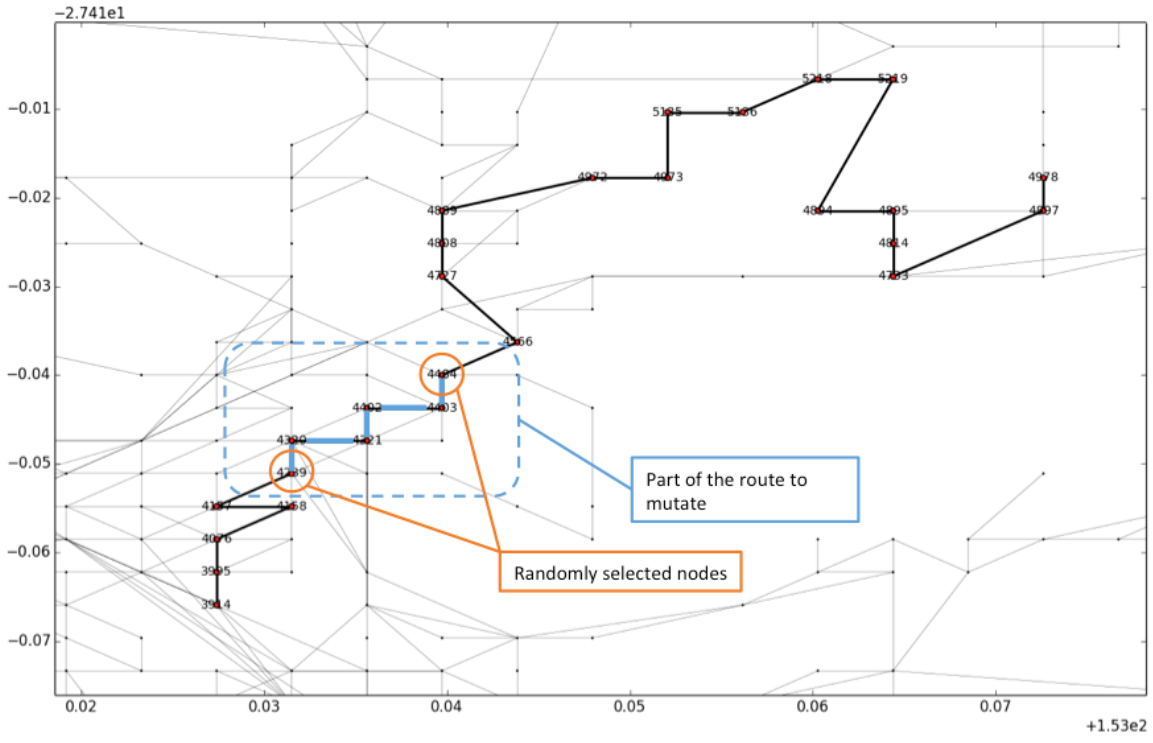



*Figure 26 Step of mutation that removes the nodes between two random nodes in order to create new routes*

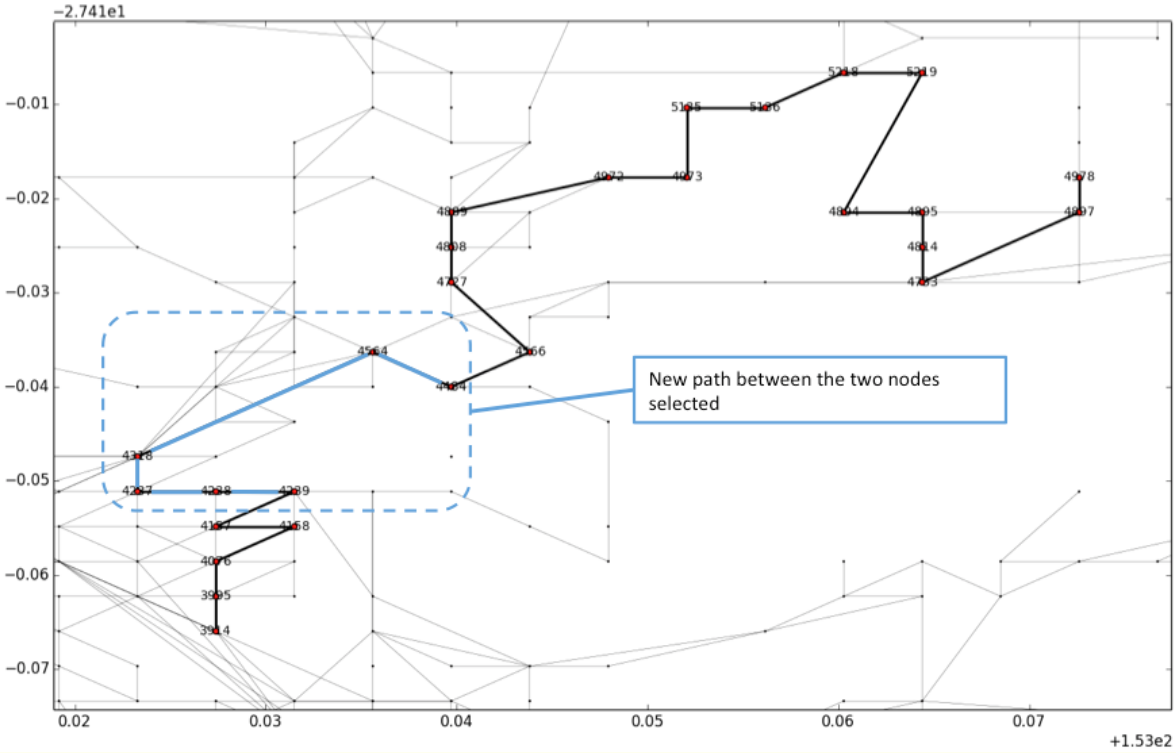



*Figure 27 Result of mutation that creates new and maybe better routes*

#### 7.2.3.3. Route search pseudocode

A pseudo code for the route search is provided below, presenting the steps followed to search for the optimal route.

1. Initialize all performance measures.
2. Set generation to zero.
3. Call the route generation function.
4. Initialize population.
5. Call the evaluation function.
6. Rank routes by their fitness value.
7. Generate population from the next generation:
    - Selection
    - Crossover
    - Mutation
8. If termination criteria is not met, increment generation by 1 and go to step 4.
9. Output the population from the last generation, consisting of the best routes selected from the previous generation plus the new routes obtained by mutating them, sorted in descending order, which means the best candidate (best fitness value) is shown first.

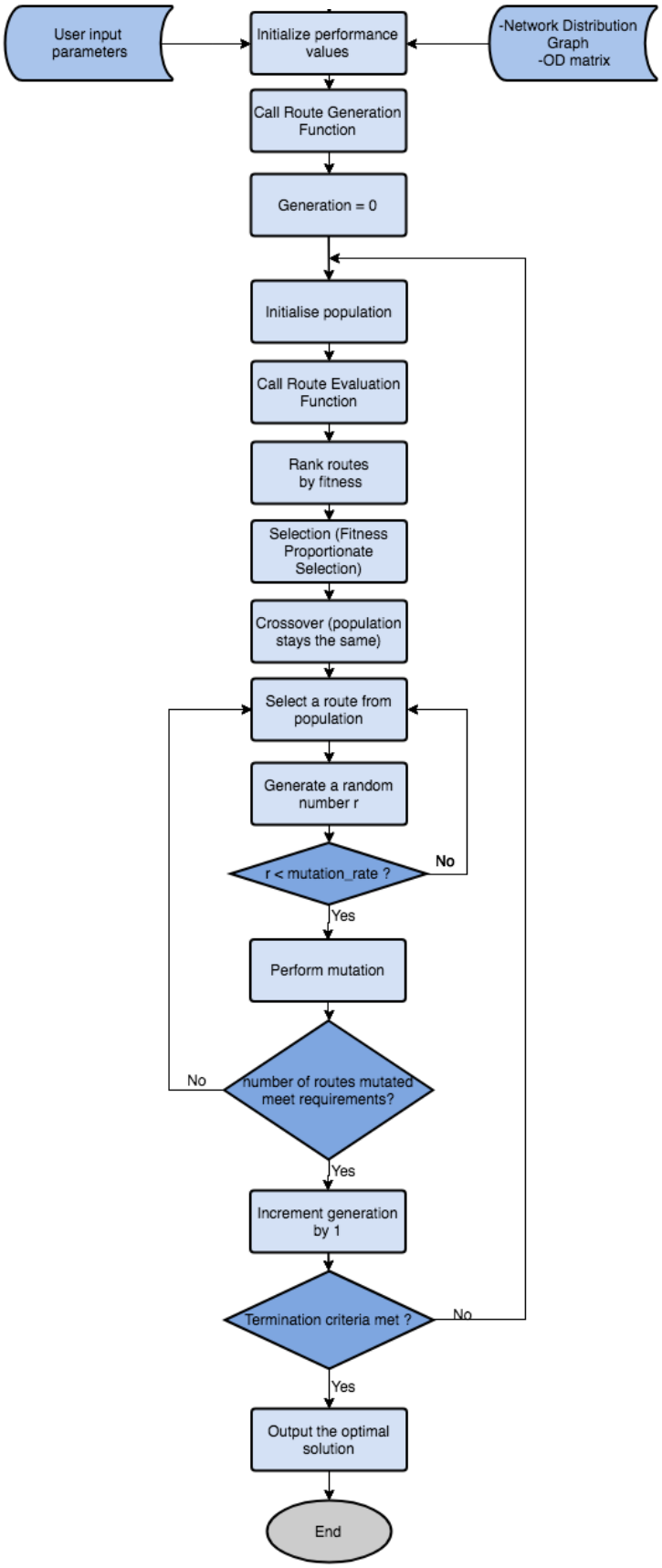
#### 7.2.3.4. Route search output

The output of the route search algorithm is the final optimized set of routes, together with their respective fitness values, sorted according to that value in descending order. Each route is presented in a Python list format, containing the nodes through which the route passes.

The group of routes is sorted in descending order, which means that the first route showed in the output has the highest fitness value achieved. This can be either a local optimum or a global one, depending on the effectiveness of the parameter values chosen.

The flow chart, which depicts the process of the route search, is showed in *Figure 28.*

*Figure 28 Flow chart for the route search stage in genetic algorithm*



# 8. Sensitivity analysis

In this chapter a sensitivity analysis is performed on the main design parameters. Its objective is to understand their impact on the algorithm by running the program with a wide range of values. The parameters that have been tested are: number of generations, population size, mutation rate and number of selected candidates. While a parameter is being tested, its values are varied over a range, whereas the other parameters are kept constant. The standard parameters that have been chosen for the analysis are the following:

| Design parameter | Value |
|---|---|
| **Number of generations** | 20 |
| **Population size** | 1000 |
| **Mutation rate** | 0.6 |
| **Selected candidates** | 50 |

*Table 2 Design parameters used in sensitivity analysis*

The results of each sensitivity analysis are discussed below, showing the effect of the different parameters used in this model.

## 8.1. Effect of the parameters on the fitness value

When a parameter is being tested, its fitness value varies depending on the impact of that parameter in the algorithm. In this section the effect that has each parameter on the fitness value is studied. The results are presented in a graphical form, showing the fluctuation of the fitness value for each parameter tested.

### 8.1.1. Effect of the number of generations

The number of generations is used as stopping criteria in the algorithm, so its effect on the fitness value must be carefully studied. In this case, the sensitivity analysis has been undertaken by running the algorithm with different values for the number of generations, starting with 5 and increasing its value until generation 50.

It is quite more probable not to converge in a local optimum when a higher number of generations is used. This is due to the fact that in every generation a new group of mutated routes is generated, so many other candidates are explored and the analysis gets wider with each generation. However, the compiling time substantially changes when varying the number of generations, which is also something to be taken into account.

As shown in *Figure 29*, for a low number of generations the results obtained are much worse than for a high one. Nevertheless, results get stable and their variation is not remarkable from 40 generations on.

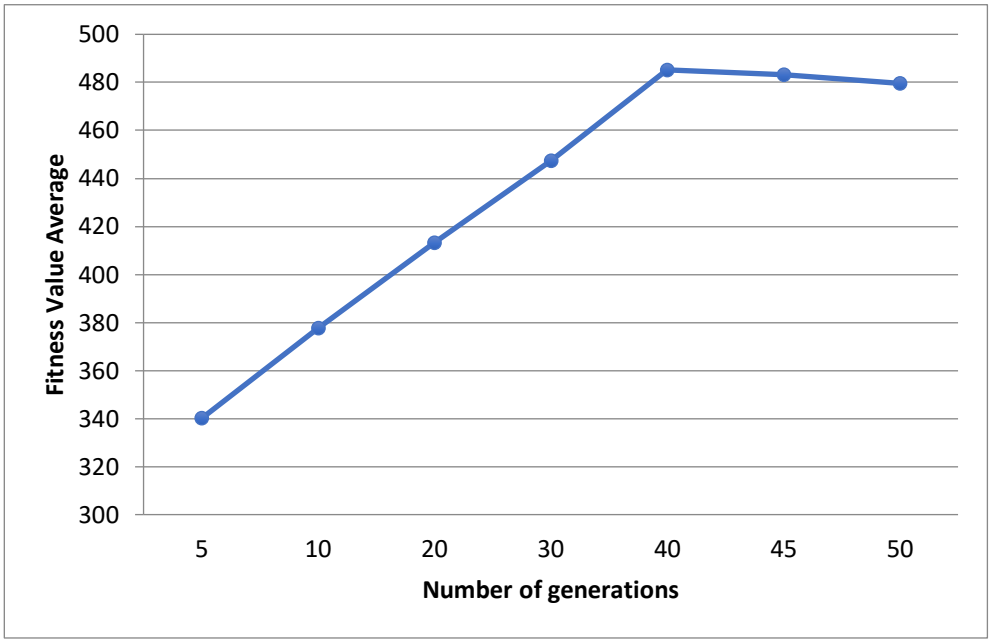



*Figure 29 Effect of the number of generations on the fitness value*

### 8.1.2. Effect of the population size

It has been studied the effect of the population size on the fitness value. The program has been tested with different populations sizes, to value its effect, and many trials have been done with each number in order to have enough results for the analysis.

If the program is run with a small initial population, the probability of getting to a high fitness value is too low. It is due to the fact that, with a small number of initial routes and a very high number of possible solutions, it is very improbable to find a good result. As the population size is increased, the results obtained are better. However, the bigger is the population size, the bigger is the compiling time, which should also be considered.

In *Figure 30* it can be seen that with an initial population lower than 750, the fitness values obtained are not significant. Running the program with an initial population of 1000 individuals, leads the algorithm to a higher result, which keeps growing as the initial population is increased. Best results are obtained with populations of 3000 routes or higher.

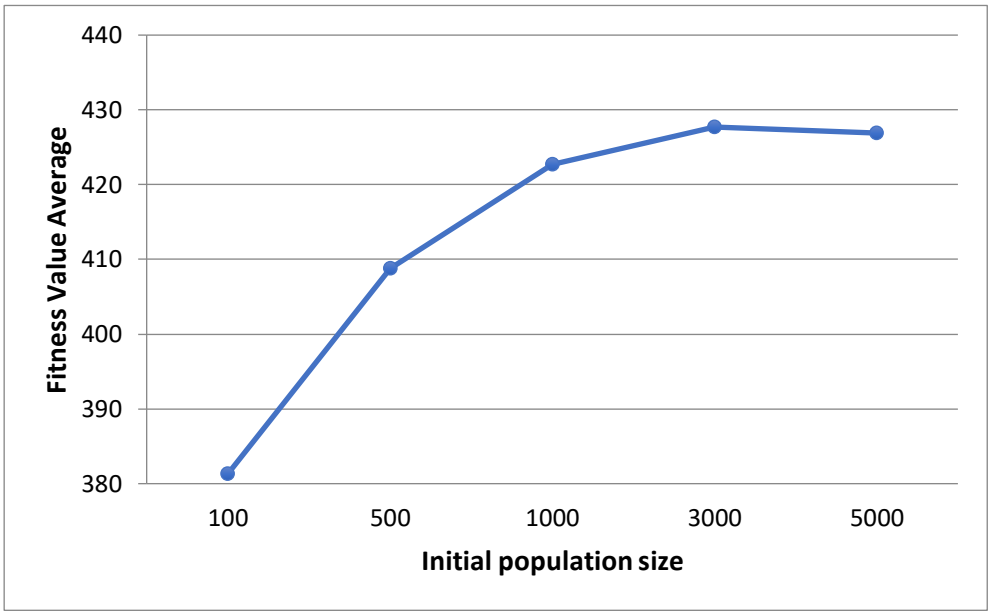



*Figure 30 Effect of the initial population size on the fitness value*

### 8.1.3. Effect of the mutation rate

The effect of the mutation rate has been examined by varying its value between 0.2 and 1. The program has been run many times with each mutation rate, in order to have enough results for the analysis.

As expected, with a low mutation rate the results obtained are worse than with a high one. This is due to the fact that the number of evaluations increases when the mutation rate does, so a higher mutation rate means a higher number of routes is analysed. However, there is usually a mutation rate when results stabilize, and the improvement is not that perceptible.

The computation time is also something to take into account, since it is also increasing when a higher mutation rate is used.

As observed in the *Figure 31*, mutation rate has a significant effect on the fitness value. When lower values are tested, the results obtained are worse and there is a high degree of variation, whereas in the range of 0,6 to 1, the variation of results is barely noticeable.

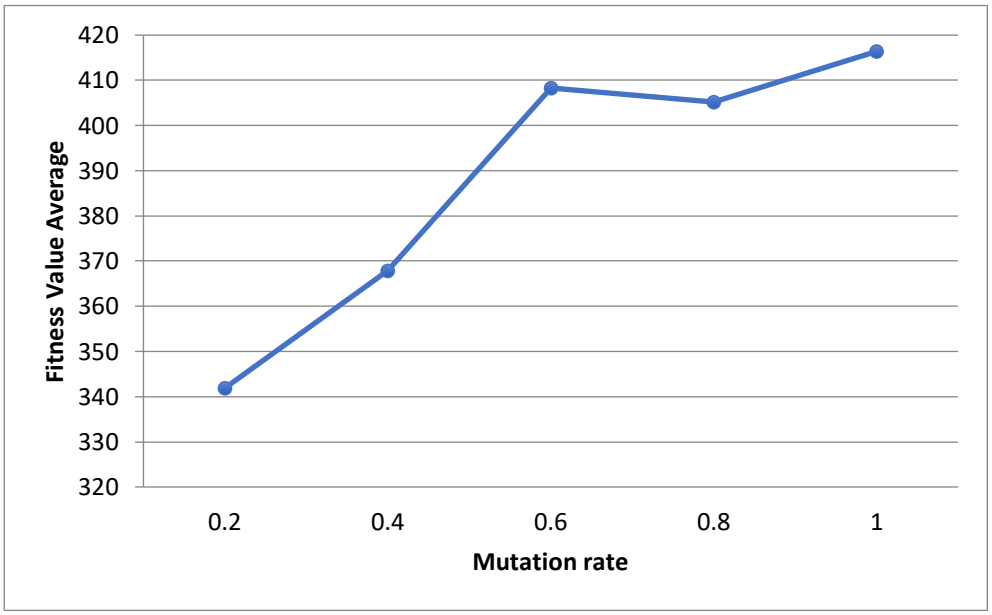



*Figure 31 Effect of the mutation rate on the fitness value*

### 8.1.4. Effect of the number of selected candidates

Every time a new generation is produced, there is a number of candidates selected to be possibly mutated (mutation will depend on mutation rate). By varying the number of selected parents, the repercussion of this parameter into the fitness value has been tested.

Choosing a small number of children at each generation gives fewer possibilities to the algorithm to find new and better routes. As the parameter is increased, fitness values are improved, but at the same time the compilation time gets higher.

In *Figure 32* it is observed that picking only one route at each generation results in low fitness values, whereas as the parameter keeps increasing better results are obtained. It should be reminded that this analysis has been done with an initial population size of 1000, which would be the maximum value that the parameter can take. However, as soon as the parameter reaches to 75, similar results are observed, which shows that there is no need of spending compilation time with a parameter bigger than 125.

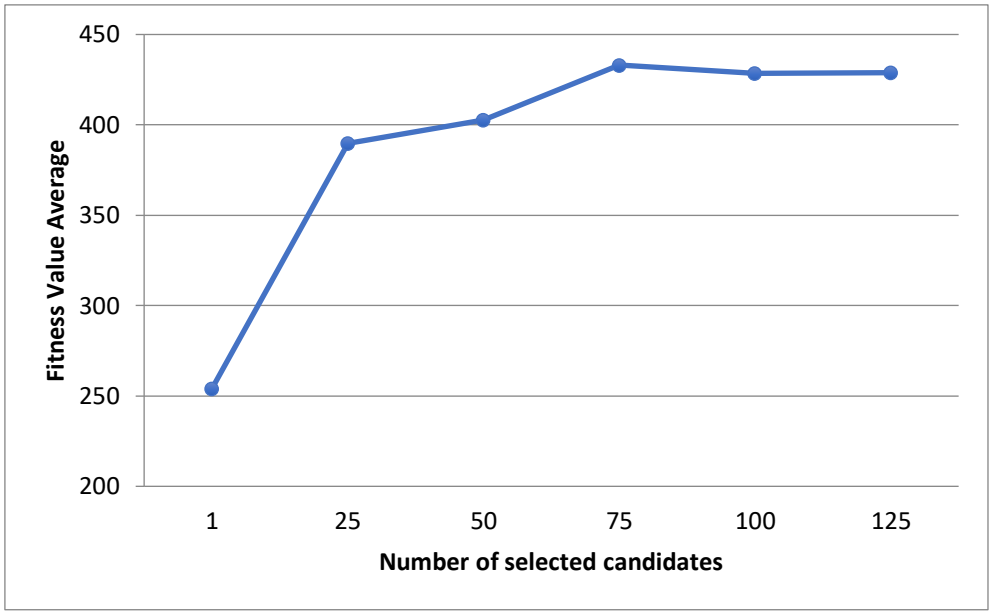



*Figure 32 Effect of the number of selected candidates on the fitness value*

## 8.2. Effect of the parameters on the graph

Not only the fitness value is influenced by the improvement of a parameter, but also the location of the route. In this section, the effect that each parameter has in the location of the route is studied.

The program has been run several times with each combination of parameters. The results are presented with graphs, showing in each graph the location of different routes obtained with different values for each parameter.

### 8.2.1. Effect of the number of generations

When the program is run with one generation, the routes go through one only mutation, which means that the final solutions obtained are almost random. As it can be observed in *Figure 33* results don't converge into any zone, but they are spread throughout the whole area.

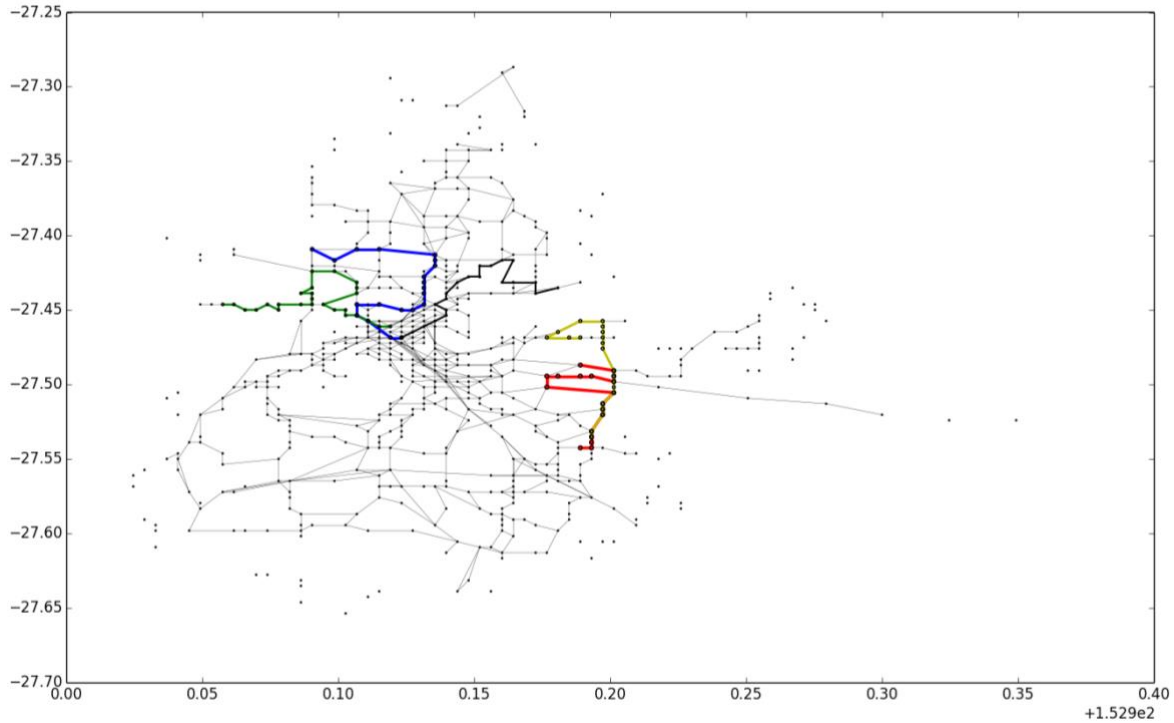



*Figure 33 Effect of the number of generations on the graph with one generation*

When the number of generations is increased to 20, better results are obtained. In the *Figure 34* it is possible to start observing a trend shown by the blue, green, yellow and black routes. However, the red route reveals that all of them still do not converge into a zone, which means that better and similar results could be obtained with greater values of the parameter.

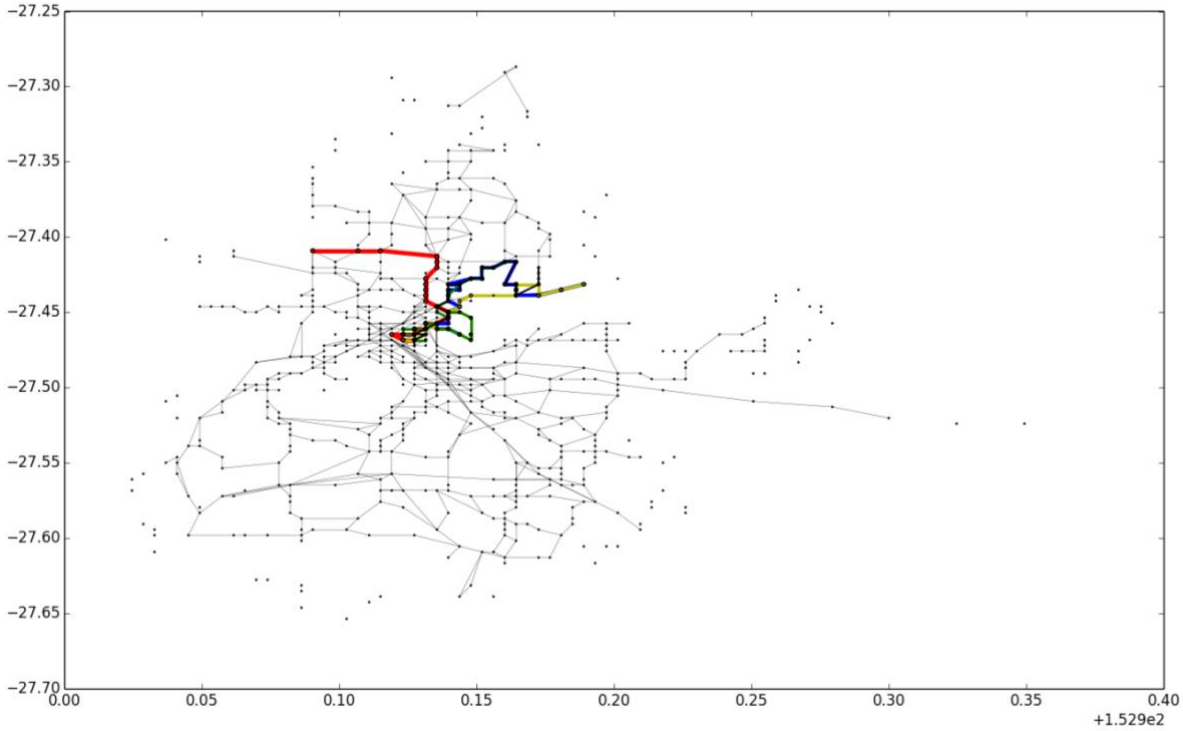



*Figure 34 Effect of the number of generations on the graph with 20 generations*

When the program is run with 45 generations, very similar and nearby results are obtained. In *Figure 35* it can be observed that the best routes are all located in the same area, which ensures that a convergence has been reached.

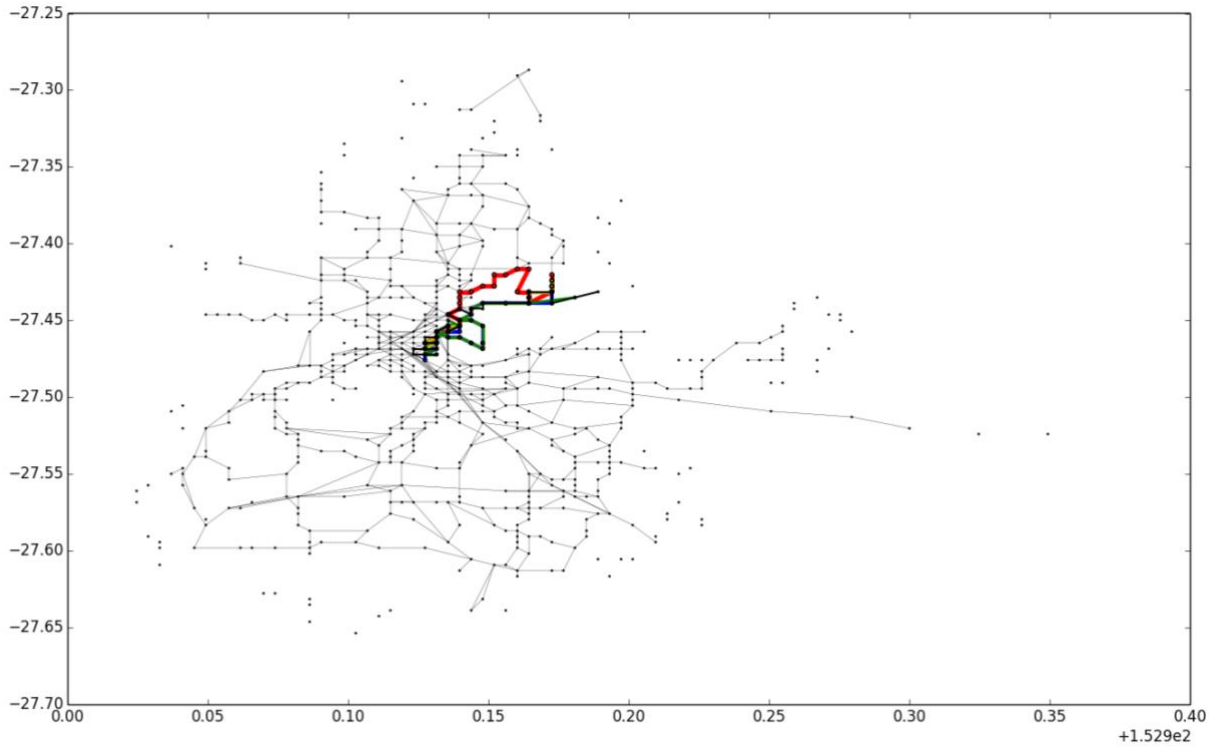



*Figure 35 Effect of the number of generations on the graph with 45 generations*

From 850 edges that has the graph, all the results end up not only in the same area but also with some of the edges common between them. There is no doubt that the optimal bus line would be in the area shown in *Figure 36* and that it would follow the trajectory followed by the orange line.

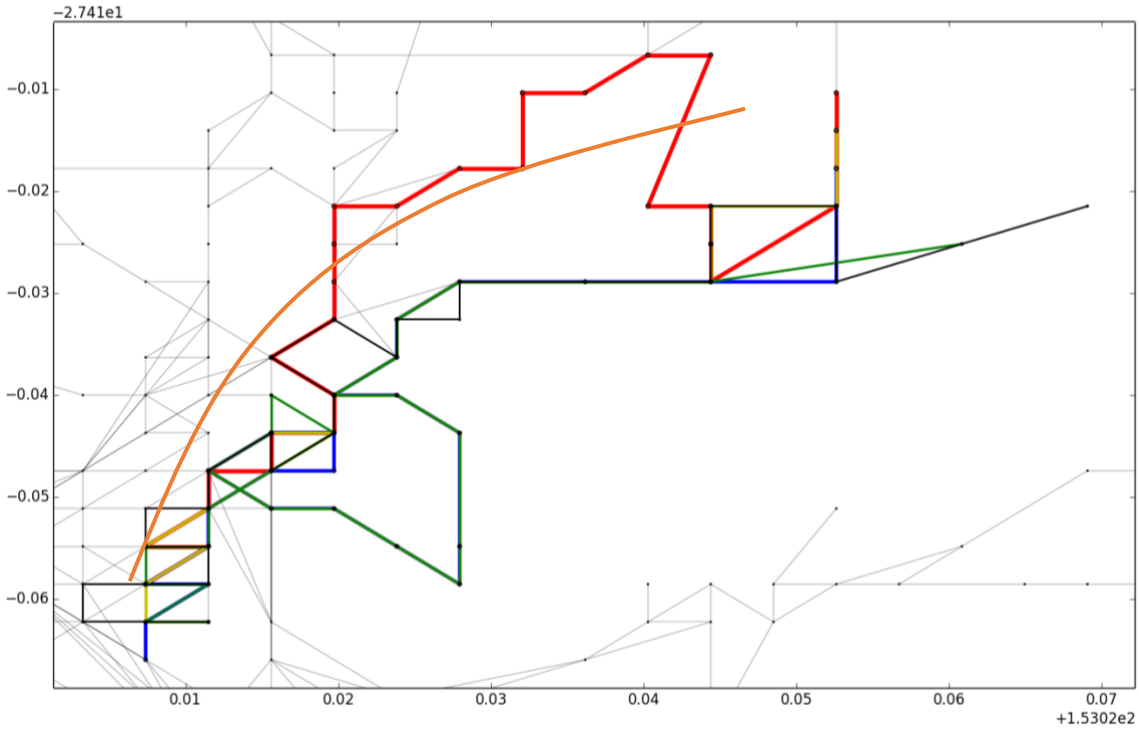



*Figure 36 Trajectory of the routes running genetic algorithm for 45 generations*

### 8.2.2. Effect of the mutation rate, population size and selected candidates

The same analysis explained above has been done for the rest of the design parameters. It has been tested the effect of the mutation rate with values of 0.2, 0.6 and 1, the effect of the population size with values of 100, 1000 and 3000, and the effect of the number of selected candidates with values of 25, 75 and 125.

The results obtained have been very similar to the ones shown in the previous section, finding spread routes when the parameter is too low, starting to see a possible convergence when the parameter improves, and finally observing a clear trend when the parameter is large enough. The complete analysis for these three parameters is shown in appendix 2.

In the section 7.1 it has been observed that all the parameters have a direct effect on the algorithm, and that as each parameter grows, so does the fitness value. The same

effect has been observed in the location of the routes, seeing that when a parameter is high enough it converges with all the results in the same area.

In the following images it can be observed that the analysis of the mutation rate, population size and selected children with their respective higher values lead to similar results. The optimal area shown for the number of generations (section 7.2.1) is also similar as the one seen below for the rest of the parameters. A clear trend displayed by the orange line shows the track that would follow the bus line.

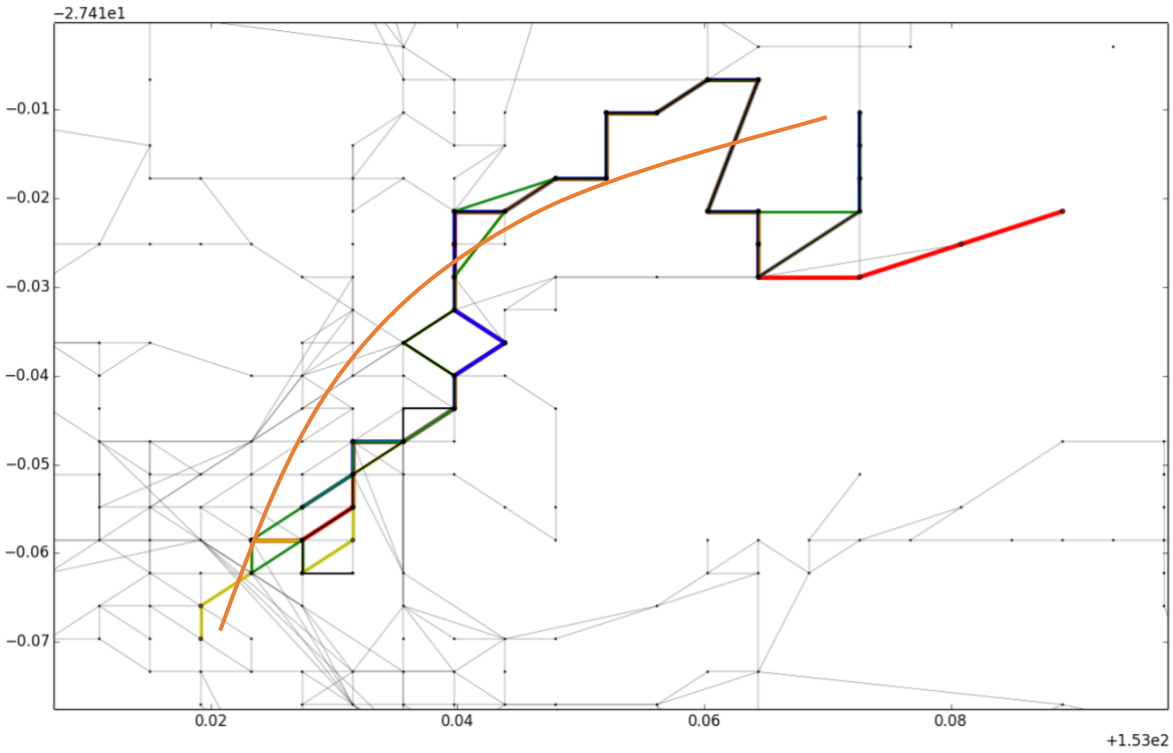



*Figure 37 Trajectory of the routes running genetic algorithm with a mutation rate of 1*

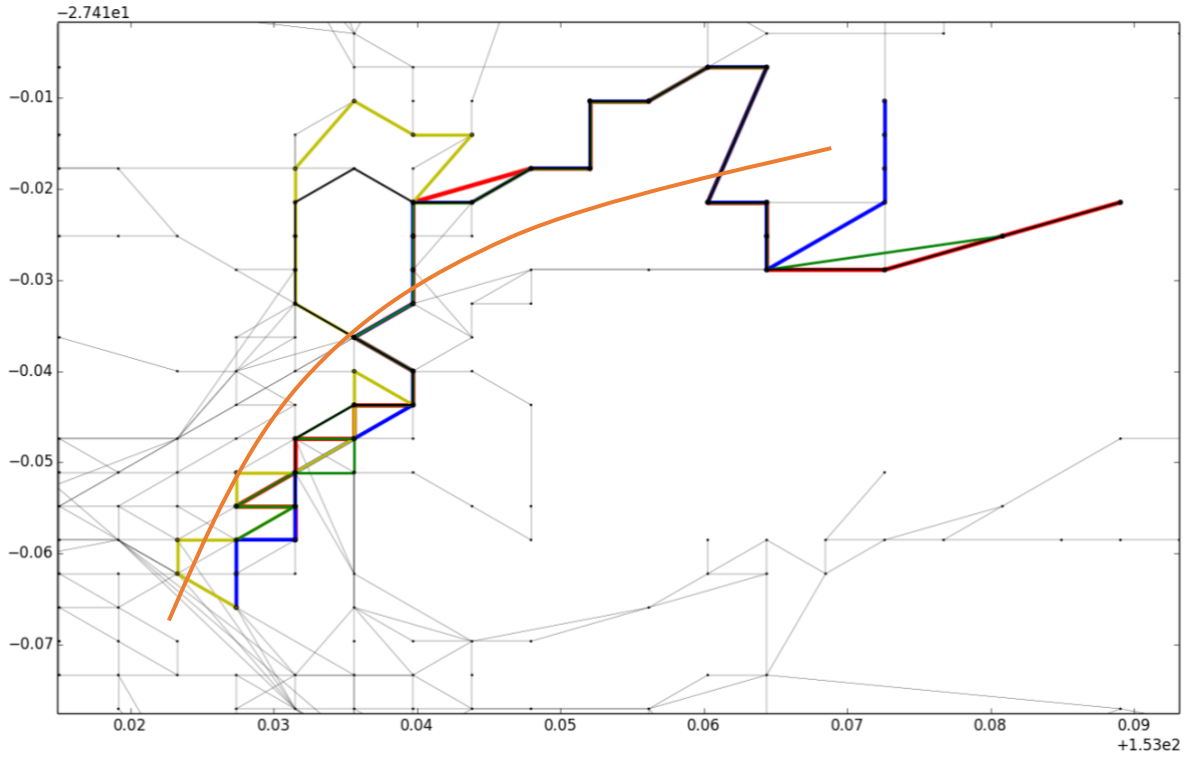



*Figure 38 Trajectory of the routes running genetic algorithm with a population size of 3000*

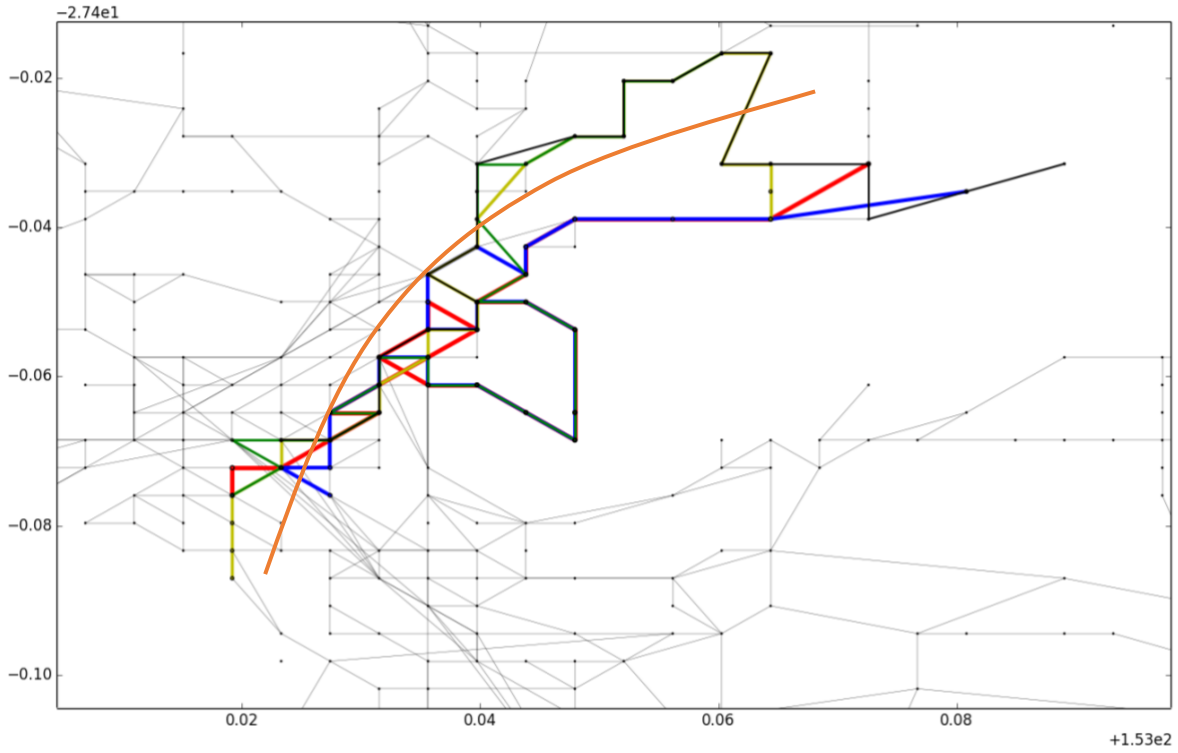



*Figure 39 Trajectory of the routes running genetic algorithm with 125 selected candidates*

# 9. Quality of the solution

In order to evaluate the quality of the solutions obtained with the algorithm proposed in the current case of study, an analysis has been performed.

The first step of the analysis is to find an appropriate value to which the solutions obtained will be compared.

For better understanding of the choice of that value, an explanation of the current fitness value in the algorithm is presented in a simple example below. Please note that the values used in this example are not real.

In the figure below, it is shown an example of how the OD matrix of the whole network object of study would look. The first row and column correspond to the labels of the centroid nodes in the graph, and the numbers represent the number of trips from one node to another. As observed, the matrix is symmetric. This is due to the fact that, as explained in previous chapters in this report, the graph used in this case of research is undirected, so the number of trips that go from A to B is the same as the number of trips going from B to A.

| | S1 | S2 | S3 | ... |
|---|---|---|---|---|
| S1 | 0 | 35 | 20 | ... |
| S2 | 35 | 0 | 46 | ... |
| S3 | 20 | 46 | 0 | ... |
| ... | ... | ... | ... | ... |

*Figure 40 Example of an OD matrix*

Once the algorithm proposed has been run, a final route is obtained. The fitness value of this route is calculated by the sum of just the values under the diagonal in a simplified version of the OD matrix, so that the trips are not summed up twice. The simplified OD matrix only takes into account the nodes of the resulting route.

In *Figure 41* it is showed an example of how a simplified OD matrix would look (being

the route in the example made up of nodes S7, S9, S2, S1, S12) and the values (in orange) which would have to be summed up to obtain the fitness value of that route.

| | S7 | S9 | S2 | S8 | S1 | S12 |
|---|---|---|---|---|---|---|
| S7 | 0 | 35 | 20 | 22 | 21 | 41 |
| S9 | 35 | 0 | 46 | 34 | 53 | 34 |
| S2 | 20 | 46 | 0 | 53 | 33 | 46 |
| S8 | 22 | 34 | 53 | 0 | 27 | 37 |
| S1 | 21 | 53 | 33 | 27 | 0 | 39 |
| S12 | 41 | 34 | 46 | 37 | 39 | 0 |

*Figure 41 Example of a simplified OD matrix for obtaining the fitness value of a route*

This means that, if the route being analysed is made up of *n* nodes, ((n·n – n) / 2) values of the OD matrix will have to be summed up in order to obtain the fitness value of that route.

Therefore, the value proposed to compare the solutions obtained with the algorithm is the sum of the *"((n·n – n) / 2)"* best values in the full network OD matrix.

In the following figure, it is shown an example of how the best values, which would be summed up to get the comparison value, could be distributed in the OD matrix. As observed, there is a high probability that they will be spread over the matrix, which implies that the value obtained is not related to any route.

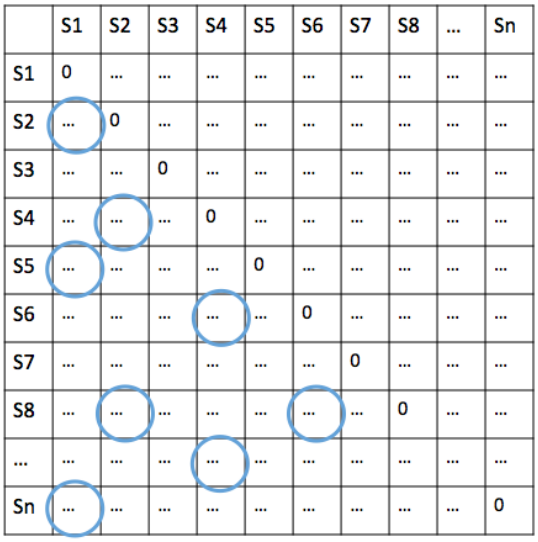



*Figure 42 Example of how the highest value is found from an OD matrix*

The value obtained with this procedure is, therefore, the highest value obtainable from the OD matrix. Clearly, it is expected that it will be much greater than the fitness value of the resulting route, due to the fact that most probably the chosen nodes will not be connected to each other, and instead of getting a route made up by (n-1) edges, the solution will most likely be a group of almost ((n·n – n) / 2) independent edges spread all over the graph, without being connected with each other. The value chosen as a comparing value is then practically impossible to reach in a real route. Hence, the aim of this analysis is not to actually compare the fitness of the routes with this value, but to give an approach of the scope of the problem.

For this analysis, the algorithm has been run 15 times, storing each time the number of nodes "*n"* that are contained in the route and the route's fitness value. After each execution, it has been computed the sum of the best *((n·n – n) / 2)* values in the OD matrix and it has been calculated the corresponding ratio (%). In the *Table 3* shown below, the results of this procedure are listed.

| Fitness Value | Best Values Sum | Ratio (%) |
|---|---|---|
| 508 | 4635 | 10.96 |
| 471 | 4143 | 11.36 |
| 526 | 3177 | 16.55 |
| 563 | 3657 | 15.39 |
| 499 | 3813 | 13.08 |
| 523 | 1938 | 26.98 |
| 505 | 2553 | 19.78 |
| 502 | 3506 | 14.31 |
| 481 | 3813 | 12.61 |
| 495 | 2553 | 19.38 |
| 485 | 2705 | 17.92 |
| 503 | 4480 | 11.22 |
| 500 | 3657 | 13.67 |
| 505 | 4317 | 11.69 |
| 551 | 3506 | 15.71 |

*Table 3 Results obtained in the analysis of the quality of the solution*

# 10. Analysis of the results

At this point, it has already been designed the algorithm that finds the route that brings the maximum number of people from their origin to their destination, which is described in detail in chapter 7. It has also been analysed the effect that the design parameters have in the fitness value and in the location of the routes. A convergence of the results and fitness values has been studied with the sensitivity analysis shown in chapter 8, and the magnitude of the results has been analysed in chapter 9. However, the results obtained have to be evaluated to ensure working with this algorithm is the best way to find an optimal route.

It has already been explained in chapter 6 the reason why genetic algorithm is the most suitable algorithm for this project. Therefore, it is has been assumed that among all algorithms, the best results are obtained with the one chosen.

To ensure genetic algorithm is an appropriate way to solve this case of research, an analysis has been carried out, which has consisted in the comparison of the results of genetic algorithm with random routes. Each random route has been found through the creation of a set of random routes, generated by connecting arbitrary nodes, and taking into account a length constraint. The route with the higher fitness value is the chosen among the set of routes.

With each execution of the genetic algorithm, compilation time and length of the final route have been saved. The set of aleatory routes mentioned above has been created by running an algorithm, which randomly generates routes with a length equal to the one previously saved, with a tolerance of half a kilometre. This algorithm is run for the same amount of time that has spent genetic algorithm finding the final result. Both values have been stored together and compared graphically, as it can be observed in the sections below.

Genetic algorithm has been executed with a variable parameter while the other three have stayed fixed. The standard parameters that have been chosen for the analysis are the following:

| Design parameter | Value |
|---|---|
| **Number of generations** | 20 |
| **Population size** | 1000 |
| **Mutation rate** | 0.6 |
| **Selected candidates** | 50 |

*Figure 43 Design parameters used in sensitivity analysis*

## 10.1. Number of generations

The program has been run leaving mutation rate, initial population size and selected candidates constant, while the number of generations has taken the value of 1, 20 and 45. The results obtained are shown in the *Figure 44*, where it is shown that genetic algorithm results are better than random results. It can also be observed, for each value of the parameter, the pair of fitness values obtained with genetic algorithm and random routes, both executed during the same compilation time.

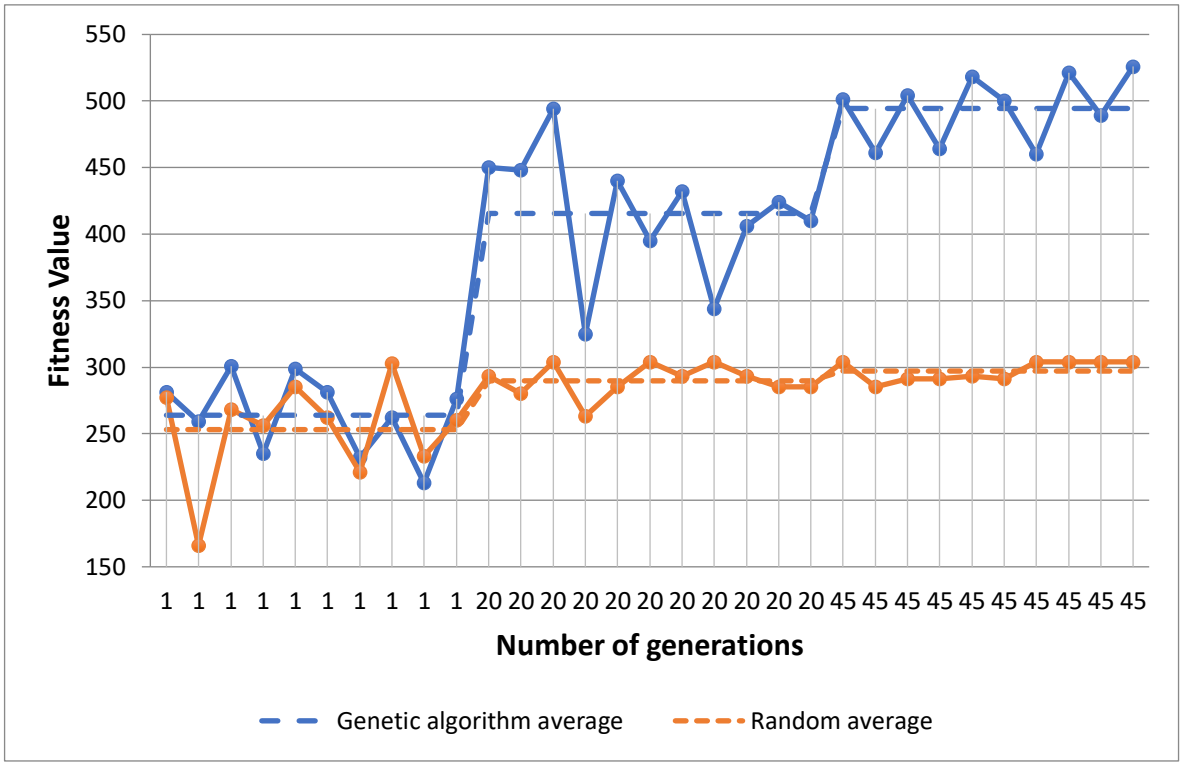



*Figure 44 Comparison of the results obtained with genetic algorithm and randomly while varying the number of generations*

When genetic algorithm is executed with one generation, only one mutation is done to the routes, which does not let the routes improve enough. The routes obtained are then almost random, and that is the reason why there is almost no difference between the execution of genetic algorithm and the random execution.

However, when the algorithm has been run for 20 generations the results obtained are much better. As number of generations increases, so does the compilation time of genetic algorithm, which means much more time for executing random routes. It can be observed in the figure that, when the random average increases by 36 units due to the increase of compilation time, the same happens to genetic algorithm average but by 151 units. The increase from 1 to 20 generations has a very positive effect on the fitness value, but it can still be observed that results obtained fluctuate in a wide range of values, which results in a high standard deviation.

When the analysis is done with 45 generations the compilation time increases

substantially in comparison with 20 generation. That increase almost does not affect to random routes, which have a very similar average to the one obtained with 20 generations. However, working with 45 generations affects considerably to the results of genetic algorithm, which end up with a very high average and with a low standard deviation.

## 10.2. Population size, mutation size and selected candidates

The same analysis explained above has been carried out varying initial population, mutation rate and selected individuals. Very similar results have been obtained and they are explained below and shown in the *Figure 45, 46 and 47.*

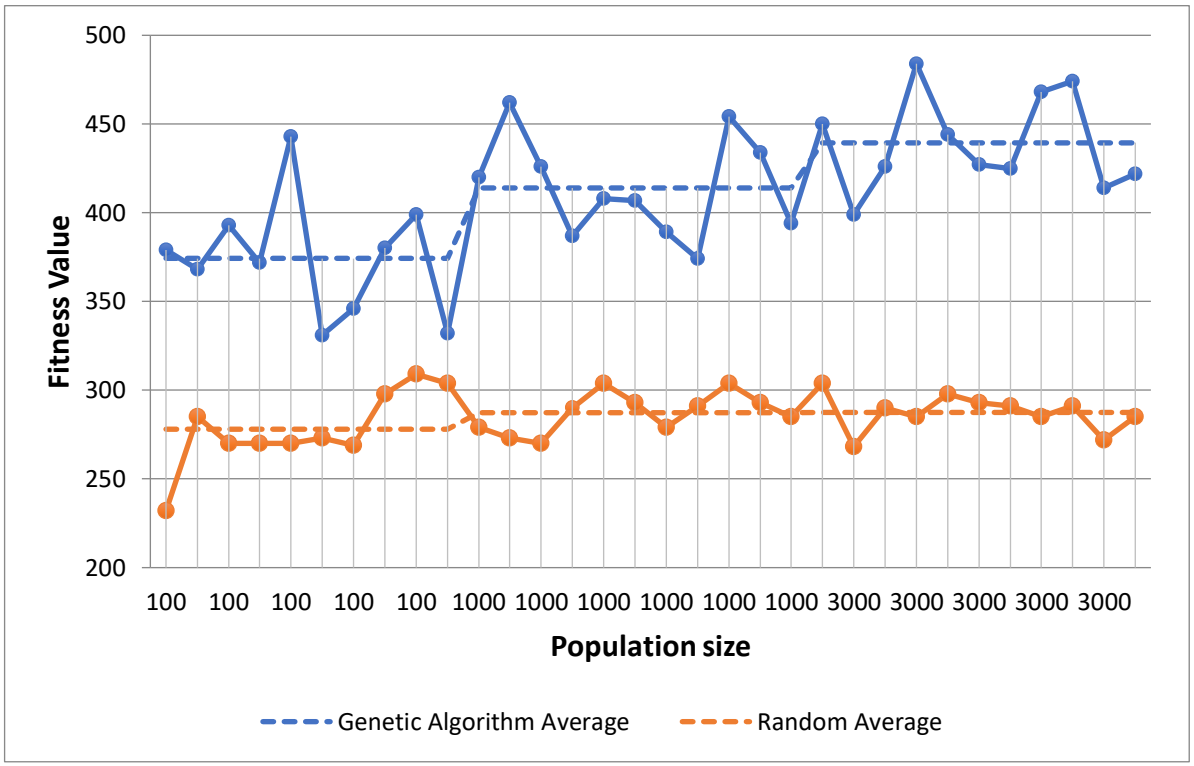



*Figure 45 Comparison of the results obtained with genetic algorithm and randomly while varying the population size*

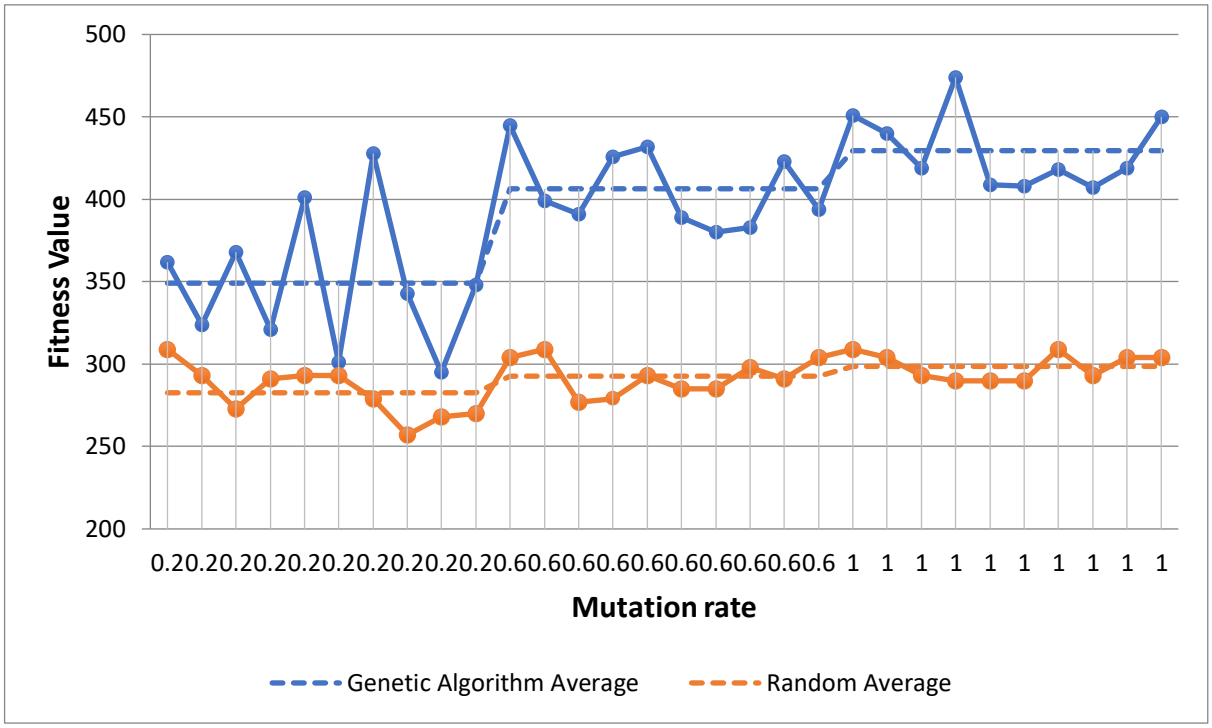



*Figure 46 Comparison of the results obtained with genetic algorithm and randomly while varying the mutation rate*

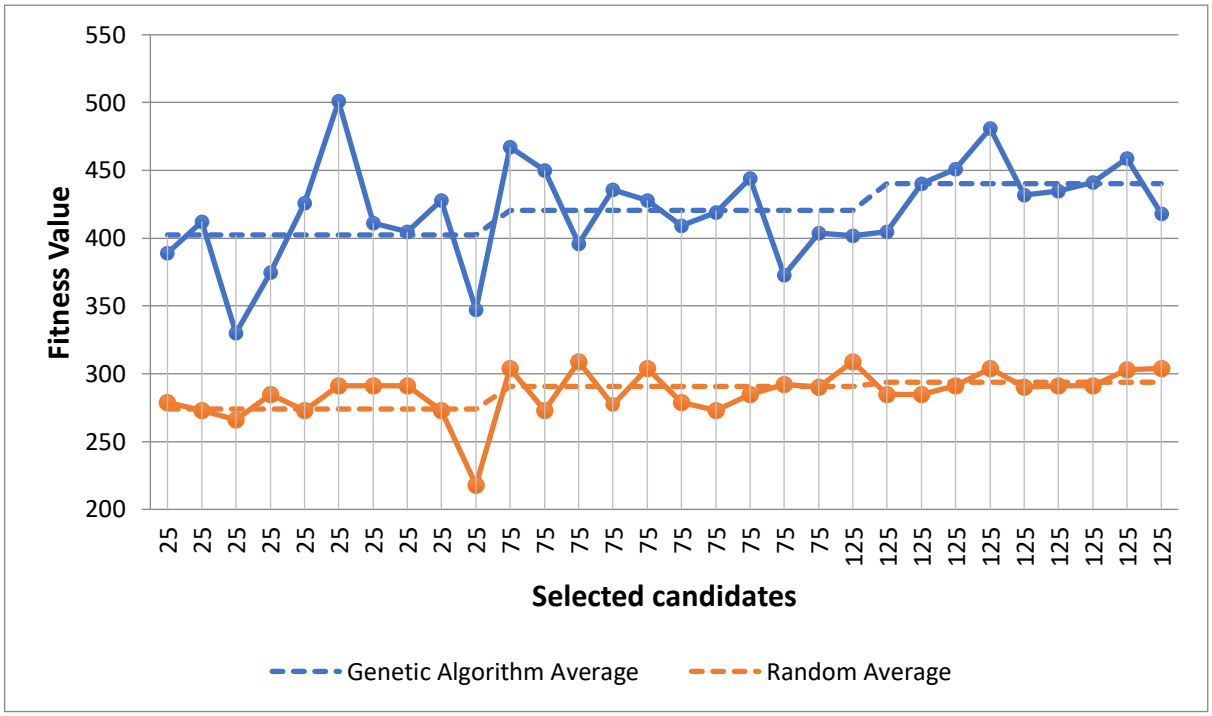



*Figure 47 Comparison of the results obtained with genetic algorithm and randomly while varying the number of selected candidates*

On the one hand, it can be observed that the higher average reached in the analysis of the number of generations is 494, whereas with the rest of the parameters it is between 430-440. This means that the effect of the number of generations in the fitness value is above the rest of the parameters, because as they are all run with 20 generations, does not matter the value of their parameters that they will not reach the results obtained with 45 generations.

On the other hand, standard deviation is another point to be taken into account. It can be observed in the figures that genetic algorithm results fluctuate in a wide range of values no matter the parameter and its value. It is due to the fact that the three analyses are carried out with 20 generations, and, as it has been explained in the previous section, when number of generations takes this value a high standard deviation is observed.

Finally, it has to be discussed and compared the differences between the results obtained with random routes and the ones obtained with genetic algorithm. It can be observed in the figures that no matter what design parameter is analysed and its value, genetic algorithm average is always higher than random average. Therefore, it has been proved that the algorithm designed in this project is a reliable algorithm when it comes to finding an optimal route for the desired bus line.

# Conclusion

Once finished the previously presented analysis, it can be concluded that through the use of the algorithm proposed in this project a public transport line which improves the mobility system in a city can be found.

In the first place, an analysis of the three potential algorithms to solve the transport network design problem has been carried out. From this analysis it is concluded that the most suitable algorithm for this project is genetic algorithm, due to three main reasons: its iterative nature, its stochasticity and the multiplicity of solutions analyzed in each iteration.

Secondly, it has been found that each of the parameters taken into account in the genetic algorithm affect to the solution obtained. Through the sensitivity analysis performed, it has been observed how each of them influences on the results, being the number of generations the parameter with a highest impact.

Moreover, it can be stated that the algorithm is reliable, having observed that results obtained with it are noticeably better than the ones obtained randomly.

Finally, a proposal for future research would be the study of the design parameters in order to find the optimal value of each of them. In this project it has been analysed the impact that each parameter has in the results, finding that they all influence the final solution. Nevertheless, the optimal value of the design parameters could be object of another research project, which depends on the database and thus would be different in each case of study.

It could also be studied further the possibility of placing parking lots in the outskirts of the city. In the present research project it has been observed that many of the routes considered are coming from outside the boundary to the city and conversely. It could be object of study the placement of parking lots in the boundary defined (and a public transport line that brings people from the parking to the city) in order to reduce traffic.

# References

**[1]** PHIL LEBAU, CNBC, *AAA Study: Hands-free connectivity still dangerous.* October, 2014

**[2]** L.M.MARTINEZ, World Conference on Transportation Research , *Zoning Decisions in Transport Planning and their Impact on the Precision of Results*. Berkeley, USA, June, 2007

**[3]** AGENCIA DE ECOLOGIA URBANA DE BARCELONA, *Plan Especial de Indicadores de Sostenibilidad Ambiental de la Actividad Urbanística de Sevilla*. February, 2008.

**[4]** TALBI E., *Metaheuristics: from design to implementation*. 2009.

**[5]** FLETTERMAN, M., *Designing multimodal public transport networks using metaheuristics.* 2008.

**[6]** CHARLES DARWIN, *On the origin of species*. London, 2003.

**[7]** PATTNAIK S., MOHAN S., TOM V., *Urban bus transit route network design using genetic algorithm*. 1998.

**[8]** NGAMCHAI S., LOVELL D.J., *Optimal time transfer in bus transit route network design using a genetic algorithm*. 2003.

**[9]** FAN W., MACHEMEHL R.B., *Optimal transit route design problem: Algorithms, implementations, and numerical results*. 2004.

**[10]** FUSCO G., CIPRIANI E., GORI S., PETRELLI M., *A procedure for the solution of the urban bus network design problem with elastic demand*. 2005.

**[11]** KUMAR, R., *Blending Roulette Wheel Selection & Rank Selection in Genetic Algorithms*. August, 2012.