PROJECTE O TESINA D'ESPECIALITAT

Títol

Optimization Analysis of the Number and Location of Holding Control Stops to Prevent Bus Bunching

Autor/a

Ferran Mach Rufí

Tutor/a

Francesc Robusté / Haris Koustopoulos

Departament

Infraestructura del Transport i del Territori

Intensificació

Enginyeria del Transport

Data

30 de juny de 2011

Abstract

TITLE: Optimization Analysis of the Number and Location of Holding Control Stops to Prevent Bus Bunching

AUTHOR: Ferran Mach Rufí

TUTORS: Francesc Robusté Antón, Haris Koustopoulos

KEYWORDS: Bus bunching, Holding Point, Optimization algorithm, Myopic algorithm, Genetic algorithm

The growing congestion problems in big cities result in growing need for public transport services. In order to attract new users, public transport operators are looking for methods to improve their performance and level of service. Service reliability is one of the main objectives of public transport operators. Various sources of service uncertainty can cause bus bunching: buses from the same line tend to bunch together due to a positive feedback loop, unless control measures are implemented. The most commonly used strategy for preventing service irregularity is to define holding points along the bus route. The design of the holding strategy involves the determination of the optimal number and location of holding points, as well as the holding criteria. These strategies are classified to schedule- or headway-based. Previous studies showed that headway-based strategies have the potential to improve transit performance from both passengers and operators perspectives.

This thesis analyzes the performance of optimization algorithms when solving the holding problem. The optimization process involves the determination of time point location for a given headway-based strategy. The evaluation of candidate solutions is based on a mesoscopic transit simulation. The input data for the simulation corresponds to the bus line number 1 in Stockholm city.

The objective function is made up of the weighted sum of all time components that passengers experience: in-vehicle riding time, dwell time, waiting time at stop and on-board holding time. The optimization was carried out by greedy and genetic algorithms.

In addition, a multi-objective function that incorporated the performance from the operator perspective was solved using a multi-objective genetic algorithm.

The results demonstrate the potential benefits from optimizing the location of time point stops. The best solution results in an improvement of around 11% in the objective function value. Interestingly, the results indicate that wrongly chosen time point stops can yield transit performance that is worse off than having no holding control.

Resum

TÍTOL: Optimization Analysis of the Number and Location of Holding Control Stops to Prevent Bus Bunching

AUTOR: Ferran Mach Rufí

TUTORS: Francesc Robusté Antón, Haris Koustopoulos

PARAULES CLAU: Bus bunching, Holding Point, Optimization algorithm, Greedy algorithm, Genetic algorithm

Els creixents problemes de congestió a les grans ciutats fan necessaris més serveis de transport públic. Per tal d'atreure a nous usuaris, els operadors de transport públic busquen mètodes per millorar i augmentar el seu nivell de servei. La fiabilitat dels serveis de transport públic és un dels objectius principals dels operadors. El fenomen d'aparellament d'autobusos (*bus bunching* en anglès), pot ésser causat per vàries causes lligades a la incertesa: busos consecutius de la mateixa línia tendeixen a aparellar-se degut a que es tracta d'un sistema intrínsicament inestable de manera retrocativa. Per evitar aquest efecte, s'implementen mesures de control: la més comuna és establir una sèrie de parades de control repartides al llarg de la ruta, en què l'autobús esperarà fins a recuperar l'estabilitat. El disseny d'aquest sistema de punts de control passa per determinar-ne el nombre i la localització òptima, així com el criteri d'espera. Aquestes estratègies es classifiquen en basades en horari (*schedule-based*) i basades en freqüència (*headway-based*). Estudis previs han demostrat que les últimes tenen un gran potencial per millorar el servei des del punt de vista de l'usuari i del de l'operador.

Aquesta tesina analitza la idoneïtat d'una sèrie d'algoritmes d'optimització a l'hora de solucionar aquest problema. El procés d'optimització comprèn la determinació de la localització dels punts de control per a un criteri d'espera predeterminat. L'avaluació de les solucions es fa mitjançant un simulador mesoscòpic de trànsit. Les dades utilitzades per a la simulació corresponen a la línia urbana d'autobús número 1 a la ciutat d'Estocolm.

La funció objectiu a optimitzar està composta per la suma ponderada de totes les components temporals experienciades pel passatger: temps d'espera a la parada, temps en marxa a bord del vehicle, temps per parada i temps d'espera als punts de control (si s'escau). Per tal d'optimitzar la funció objectiu s'han utilitzar dos tipus d'algoritmes: un tipus myopic i un genètic. D'altra banda, a aquest últim se li ha incorporat una segona funció objectiu (en aquest cas des del punt de vista de l'operador), convertint el problema en multi-objectiu.

Els resultats demostren els beneficis potencials d'aquestes optimitzacions, arribant la millor solució aconseguida a millorar fins a un 11% el valor de la funció objectiu. És també remarcalbe la observació que s'extreu de l'anàlisi: una mala elecció de punts d'espera pot portar a un comportament pitjor del sistema que en el cas sense control.

Acknowledgements

This thesis is part of my studies in Civil Engineering in the UPC, Barcelona and has been developed as an exchange student in KTH, Stockholm. More concretely, in the Division of Transport and Logistics.

It is not only necessary but also fair to mention some people that contributed in the development of this work. In the first place, thank my co-advisor, Oded Cats, who has been giving me helpful and wise advice along all the thesis development. His endless patience and willingness to share smart ideas and knowledge have been really inspiring to me. Especially remarkable has been his predisposition to listen to my ideas and shy contributions, an asset difficult to find in someone that is in the position of teaching. I would like to express my gratitude to Professor Haris N. Koustopoulos, for the guidelines and pertinent alternative approaches given in the tutorials. I cannot forget to thank Albania Nissan for her help in monitoring the thesis.

In general, I would like to thank to every single person I have met in Sweden for making my stay so pleasant and exciting from the first day to the last one. Especially to my dormitory mates, for their friendship and all the good moments we shared together.

Finally, I would like to mention my family and friends back home. They all have been encouraging me from the distance and I never had the feeling of being really apart from them. Thank you all.

Contents

List of Figures

List of Tables

I Introduction

I.1 Overview and motivation

Transit reliability is one of the most important characteristics of a successful and attractive public transportation system; hence it is important to describe reliability in a comprehensive way. Many authors have given different approaches to define "reliability" (i.e. Abkowitz et al. 1978, Ceder 2007), but all agree that it is a crucial feature of public transport in order to reach an acceptable level of service and building passengers' loyalty to the mean of transport. Therefore, many traffic agencies have reliability as a main aim of improvement. But big interruptions or even small disruptions or in the traffic can affect the traffic flow and, therefore, bus punctuality. These disruptions can lead to a well-known disruption problem, called bus bunching. Bus bunching occurs due to variability in either travel times or dwell times at stops.

Bus bunching is a classic theory to explain how a bus that runs late tends to get later and later, and the following bus tends to get earlier and earlier. Supposing that passengers arrive randomly at stops (which can be assumed in headway-based lines), a little disruption on a bus way that makes the bus run late translates into more passengers at the next stop. Consequently the dwell time is likely to be increased and the bus will be more delayed. However, the following bus, assuming that has not had any disruption, will be running on time, but is going to find less passengers at each stop, since the previous bus will have taken some of the passengers that were supposed to catch that second bus. The system will not stabilize for itself; like a chaos theory it will lead to the bunching problem. Due to random external conditions such as traffic, stoplights, number of passengers at each stop it is unlikely to predict which buses from an outset will be bunched; therefore it is certainly complicated to develop universal strategies to avoid the problem in every case.

New implemented real-time information systems (e.g. automatic vehicle location (AVL), automatic vehicle identification (AVI), and automatic passenger counting (APC)) have allowed to transportation agencies to collect priceless data, which in many cases is underused. There is still research to be done about the implementation of real-time control systems.

The strategy applied for most transport agencies to avoid bus bunching and headway variability is defining holding points along the bus route. Holding is the process of intentionally delaying a vehicle at a station after passengers have alighted and boarded. Usually, it is unnecessary to hold vehicles at every station. Hence, transit operators must choose which are the most effective holding stations. There are some rules of thumb to determine where these points should be placed (Wirashinge et al. (1995)):

- The first station is the most affective to hold the vehicles, since a vehicle hold at the beginning of the route has impact on the largest number of downstream stations
- Locate holding points when there is enough space available
- Place holding stations regularly for the passengers to be informed about the bus schedule (in case of scheduled routes).

Since the decade of the seventies in the U.S. bus service reliability studies have been carried out. Nevertheless, concerning about holding strategies, only a few have faced the problem of determining the optimal number and location of holding points. Wirasinghe et al. (1995) developed a nonlinear dynamic algorithm to solve the problem that could never be applied and assumed schedule-based operation only. Eberlein (1995) formulated an algorithm to solve the holding problem with a rolling horizon (i.e. that when an operator decides whether or not to hold a vehicle, it considers the impact on a set of consecutive vehicles, called the impact set). Unfortunately the solution could never be contrasted. It is important to point out that both studies were analytical and had limitations in their application to real networks.

The present study deals with the holding problem with a dynamic approach: the holding time is not predefined by the operator but calculated dynamically depending on the headways between buses in real-time. It differs from previous studies also the fact that the holding problem is not solved analytically but via simulation, using a real-world modeled transit line.

I.2 Problem description

The inherent stochastic nature of the urban bus public transport may have big destabilizing effects on bus lines, resulting in buses travelling in pairs instead of being evenly spaced. Even with a small disturbance, the stochastic nature of travel times, dwell times and passenger demand will cause buses to bunch gradually in an escalating process reinforced by the relationship between them, mutually reinforcing each other. A change in the headway will have an effect in the number of passengers waiting at stops, which will make dwell time increase or decrease accordingly and so will alter the scheduled headway. This phenomenon is referred as bus bunching. The explanation for bus bunching is the direct relationship between time spent at stops (dwell time) and number of passengers that need to board at that stop. Since there is a limitation in the space available for boarding and some interaction between the users and the driver or an automatic validation system is common the boarding operation is never instantaneous. The expected number of boarding passengers increases with the time between successive bus arrivals.

If there is a disruption in a point of the route and a vehicle is delayed, the expected number of boarding passengers at the following stop will be increased, which means more dwell time needed and more delay accumulated. Similarly, supposing that the following bus does not suffer any disruption, it will be sped up since less and less boarding passengers are on the stops, because the preceding bus was running with delay. The effect grows over time until the space between buses decreases and they run as a pair. This effect is the bus bunching.

Given a certain bus network or line, is neither clear nor obvious which the best holding strategy is at a guess. Transportation agencies base the holding strategies more due to internal management policies and rules of thumb than on real optimal solutions. However, they cannot be blamed for it, since is not trivial to optimize the location and number of holding points.

Since the analysis will be focused on real lines application, the heuristic character of the data and the results have to be taken into account. For this reason it may not be possible to define a unique function that is valid in all cases, and may not be universal.

With respect to the algorithm, there are different approaches to develop it. Dynamic programming or metaheuristic models are two different options. It is also basic the

definition of good enough first solution (taking the problem constraints into account), and a wise stop criterion. The definition of the algorithm and its parameters will have an effect on the solutions obtained; a relationship that should be studied.

I.3 Objectives and scope

The holding problem is to determinate which vehicles should be held, when they should be held and for how long. The objective of this research is to develop a methodology to optimize the number and location of holding points for a given a bus line. The performance of candidate solutions is assessed using the dynamic traffic and transit simulation model BussMezzo, The evaluation takes into account both passenger and operator perspectives. The optimization procedure uses real-world bus line data from Stockholm.

Following previous studies (Cats et al. 2010a), the holding strategy is based on regulating the headway, with buses held based on the mean headway from the preceding bus and the next bus. This strategy defines which buses to hold and for how long, based on real-time vehicle location data.

I.4 Thesis organization

The current thesis contains seven sections described below:

The first chapter contains general information to understand the motivation and objectives of the work. It introduces the holding problem to the reader, as well as the bus bunching theory to provide him with more insights into the need to optimize the solution of the holding problem.

In the second section a review of some relevant literature about the problem is done. This past literature goes from the definition of the holding problem made by American researchers in the fifties and sixties from the 20th century until the latest optimization methods for dynamic real-time holding strategies.

The third section introduces all theoretical concepts about transit modeling and an explanation about the most important features of the simulator used in this work: BusMezzo. The theoretical basis of the different holding strategies is given afterwards. Finally, a review of the optimization methods that will be used in the study is conducted.

In the fourth section the case study of this work is presented: the blue bus line number 1 in Stockholm city. The main characteristics of the line are described, as well as the model built in BusMezzo to represent the line adjusting the model parameters with the real data provided by the operator.

The three different optimization algorithms that are implemented and assessed in this work are presented in chapter number five: a greedy algorithm, a genetic algorithm and a multiobjective genetic algorithm. In this chapter is also given the definition of the objective functions to optimize, depending on if the problem is approached from the user or operator perspective.

The sixth section presents the most remarkable results obtained with the three optimization techniques and compares them with the current time-point locations and the situation with no holding control. The last section summarizes the main interesting findings of this thesis and proposes some recommendations about future research directions that might arise after reading the current work.

II Literature review

First explained in Newell and Potts (1964), the theory of bus bunching is still nowadays a matter of interest for transportation researchers and agencies, and many different approaches have been made to the problem. However, not many studies have been carried out because of the difficult nature of the problem.

The problem with a single holding point has been widely analyzed, e.g. Osuna and Newell (1972), Barnett (1974), Newell (1974), Hickman (2001) or Zhao et al. (2006) as some examples. Unfortunately, single-point control does not always perform well in long routes with frequent service. In these cases, the whole line has to be seen as a system.

Some approaches to the problem have focused in indentifying the key parameters to apply the holding problem successfully. Lesley (1975) suggested that time points had to be located at bus stops where the coefficient of variance of headway is greater than twice the average over all bus stops. Abkowitz and Engelstein (1984) chose as time points stops where the product of the standard deviation of the bus travel time and the ratio of passengers that would subsequently board the bus along the route to the passengers of the bus was maximized. Later, Abkowitz et al. (1986) proved the convenience of locating the holding points just prior to a group of stops where many passengers board.

Wirasinghe and Liu (1995) developed an analytical model for the determination of number and locations of holding points, as well as the amount of slack time. The model employed dynamic programming to deal with trade-offs among various cost components and incorporates the existing rules of thumb. However, it is a simplified case with a single run with only one bus considered.

A formulation of the holding problem as a deterministic quadratic program was developed by Eberlein et al. (2001). The problem formulation was oriented toward rail transit system. An analytical solution of the problem was found making some assumptions. One of the assumptions consisted in defining an impact set of vehicles that were affected by control strategies instead of considering only the impact that a control strategy has for one vehicle itself. The objective function to minimize was the total passenger waiting times. It was assumed that waiting times are more sensitive to holding policies than in-vehicle travel times. Even for the passenger point of view, studies suggest that users are more sensitive

to waiting time than to in-vehicle riding time (Kemp 1973, Ben-Akiva and Lerman 1985). The properties of the system were analyzed and a solution algorithm developed accordingly. The algorithm was tested using a transit simulator.

Results showed how holding reduces the cost but also reduces dwell times and can result in earlier arrivals, even though it may seem counterintuitive at first glance. This is mainly because holding avoids interstation stops. Another result that is proved is the fact that the first station is always the most effective place to hold a vehicle, since it has an impact in the largest number of stations. It is also interesting the finding that the benefit of a second holding station in that case is less that a 1% reduction.

An approach to solve the bunching problem developed by Pilachowski J.M. (2009) consists in solving the problem using GPS data to counteract the cause of bunching allowing vehicles to cooperate with each other and change their speed based on their relative position. The result is the elimination of bus bunching with a small reduction in the commercial speed of the bus.

Delgado et al. (2009) developed a mathematical model of a bus fleet operating in a corridor with capacity constraints. The objective function to minimize is the total times experienced by passengers, from the waiting time at stop to alighting at the destination. Two different control policies are studied: vehicle holding at any station and boarding limits when passengers entering the bus.

The objective function includes passenger waiting time at stops and in-vehicle waiting (holding time) only. Since the vehicle running times are assumed to be constant, it is not included. Likewise, dwell time is not considered. A particularity of the model is the introduction of the strategy that consists in restricting the number of passengers that can board at each stop. The objective function includes a term that takes into account the waiting time for passengers at stop that have to wait more than one bus because of this capacity constraint.

The simulation was designed in three different high-demand scenarios: no control, only holding and holding plus capacity constraint. The case study developed consists in a one-way loop transit corridor with 24 stops. The results obtained showed that the reduction in the objective function from no control to only holding was about 10%, and from no control to holding combined with capacity constraint more than 22%.

An interesting approach to the problem using a multi-objective optimization was carried out by Cortés et al. (2009). The aim of the study was to minimize a dynamic objective function using two control strategies: holding and station skipping. Station skipping is also referred as expressing problem and consists in speeding up buses by not serving certain stations. It is important to consider the extra waiting time that passengers waiting at skipped stations have to suffer, that is the reason the measure is not popular among passengers. The two dimensions of the problem are the regularization of bus headways on one hand, and the minimization of the impact of the applied strategies on the other. The first objective function reflects the total passenger waiting time at stops (which depends on the predicted headway along with the bus-stop load) and the regularization of bus headways to maintain the actual headway close to the desired one. The second objective function measures the passenger holding time and the extra waiting time for passengers whose stop is skipped.

The problem is solved by a multi-objective genetic algorithm. At each stage of the algorithm, the Pareto set is found using the best individuals from the last iteration. The optimal Pareto set contains all the Pareto optimal solutions at the end of the routine. Since the strategy control is a real-time one, the best stopping criterion corresponds to the number of generations.

A simulation is conducted for a case study, where only some pre-defined stops can be holding points, with only four possible holding time values. Skipping is allowed at every station. Results show how the two objective functions are opposed but there is certain overlapping because both functions improve the level of service regularizing the headways. Despite of the similarities a trade-off can be observed.

Daganzo (2009) analyzed an adaptive control scheme to mitigate bus bunching, which dynamically determined holding times at control points based on real-time information. Seeing the bus line loop as a system with elements (buses) in equilibrium because of attractive and repelling forces it proposes a method to control and compensate the destabilizing forces dynamically (speeding up or retarding buses accordingly).

Another approach to solve the holding problem was developed by Malzoumi et al. (2010) using the ant colony optimization (ACO) to find the optimal solution. The function to optimize includes four terms: waiting time at stops, holding time (only considered for through passengers), a lateness penalty and an operational cost term that is assumed to be

linear with the operation time and includes drivers' wages, fuel, and bus maintenance. The algorithm is applied in a real-world bus route in Melbourne using the micro simulation package VISSIM to calibrate the parameters needed for the ACO. Only 10 bus stops out of the total of 24 that compound the line are considered as possible holding points. Furthermore, only three different slack times are considered: 0, 1 or 2 minutes. An analysis of all feasible solutions with these constraints is done and the results given by the ACO are compared with the real solution to assess the efficiency and accuracy of the algorithm. It is proved how an appropriate set of holding points and slack times can lead to a worse design with higher costs than the case with no control. One of the remarkable conclusions of the work is the suitability of heuristic algorithms to solve the holding problem and an approach to other heuristic models is recommended.

Authors	Problem	Method
Wirasinghe and Liu (1995)	Determine the number and location of the holding points and slack time.	Dynamic programming. One single run with one bus.
Eberlein et al. (2001)	Minimize the total passenger waiting time in a rail transit system.	Definition of an impact set of vehicles affected by the control strategy. Analytical algorithm tested using a simulator.
Delgado et al. (2009)	Minimize the total time experienced by passengers (dwell and riding time not considered).	A function that includes bus capacity constraints is considered. Three scenarios studied: no control, only holding and both holding and skipping. Deterministic travel times and passenger arrival process are assumed.
Cortés et al. (2009)	Minimize a dynamic objective function using two strategies: station skipping and holding. The objective functions represent the regularization of bus headways on one hand, and the minimization of the impact of the applied strategies on the other.	Multi-objective optimization to assess the interaction between holding and skipping, defining a Pareto front.
Malzoumi et al. (2010)	Objective function including four terms: waiting at stops, holding time, a lateness penalty and an operational cost term.	Ant colony optimization (ACO). The algorithm is applied in a real bus route in Melbourne using VISSIM simulator.

Table II.1 Summary table of most relevant literature

III Methodology

III.1 Transit modeling

III.1.1 Transit operations modeling

Models can be classified in analytical, where the solution is obtained from a set of equations using calculus techniques or simulation where the changes of traffic are reproduced by a model.

Simulation models capture the dynamics of the system, which is an advantage to analytical models, since we can get a continuous view of the traffic state over the time. However, until recent time, the high computational cost of a comprehensive simulation was too high and unaffordable. Newer more powerful computers have solved this inconvenience.

For the purpose of this thesis we are going to use a computer simulation. A simulation is a construction of a mathematical model for some process, situation, etc, in order to estimate its characteristics or solve problems about it probabilistically in terms of the model. It is an attempt to model a hypothetical or real situation to be studied for certain purposes, indentify the main variables and make predictions of the possible behavior of the system.

Simulation models can have three levels of detail: from macroscopic via mesoscopic to microscopic. Macroscopic models represent traffic at a high level of aggregation as flow, without considering the small constituent parts (vehicles) and variables (individual speed of the vehicles, for example). Microscopic models describe the small details of the traffic state, from the smallest part (vechiles) and their interactions to the characteristics of the whole set making up the traffic steam. Mesoscopic models are an intermediate state, describing individual vehicles for example, but not their interactions.

III.1.2 Model typologies

Macroscopic models are widely spread nowadays, mainly because traffic measurement systems have been installed in urban areas and highways and the kind of aggregated data that these systems collect is appropriate for macroscopic simulators. The models describe

the evolution of traffic with differential equations that analogue the physical phenomena ºof fluid or gas flows. The solution of the equations can be obtained analytically (used only in segments of the road) or using the simulation (better to describe the whole network). Most of these models split the network as a bunch of cells and applies the law of the conservation of mass for the flows that travel between the different cells. Each cell has certain parameters (density, maximum speed, etc.) that define the behavior of the traffic flow inside the cell and the flows of the adjacent cells.

These models are successfully applied to large scale networks for long time periods, where the shortcomings caused by a low level of detail are negligible.

Microscopic simulation is useful to understand traffic at a more detailed level. In these models traffic is described at the level of individual vehicles, their interactions and the interaction with the road and infrastructure. The information needs is much more detailed (accelerations, decelerations, lane changes). The existing models can be divided in car-following models, lane-changing models and route-choice models. Car-following models describe acceleration and slowing down patterns resulting from the interaction of each vehicle with the vehicle in front and the features of the road. Lane-changing models describe the variables that affect the decision of changing lane when driving. Finally, the route-choice models put attention on the origin and the destiny of the car trip, and how the path choices change along the way depending on the traffic state.

The demand in these models can be described in one of two possible ways. The first one is divide the network to study in several zones with uniform characteristics and set the number of vehicles travelling from each zone to each other in an Origin/Destination matrix. Since the demand patterns vary with the time, a different OD matrix for each time slice is needed. The other method consists on focusing on the turnings (intersections) and analyzing the percentage of vehicles that turn or go ahead at each turning. Speeds, flows and densities are aggregated parameters in this kind of models. The main drawbacks in these kind of models is the large amount of data needed to model the network and the big effort to calibrate all the parameters, which is very time-consuming. Besides, the results from a calibration are usually not transferable to other locations apart from the original zone of study.

Mesoscopic models are becoming more and more popular, since they try to cover the gap between macro and microscopic models. They combine the description of some of the

traffic elements in high and lower level of detail. For example, traffic entities can be described in a high level whereas interactions between vehicles in a lower level. There are different types of mesoscopic models.

- Vehicles are grouped into packets that act as an entity, and in each link a speed-density function is defined. The speeds and densities at the moment of entry are derived from that function. Lane changes and accelerations/decelerations are not modeled. (CONTRAM, (Leonard, et al. 1989))
- Vehicles are grouped into cells that can traverse links and vehicles can leave the cells, but there is no possibility of overtaking between cells.(DYNAMIT (Ben-Akiva, 1996))
- Some other models use a queue-server approach where the road is modeled as a queuing and running part. The vehicles, which are modeled individually, drive through the section of the road at a certain speed according to the speed-density defined for that section until they reach a queue-server downstream that transfers vehicles to the connecting roads. Queue-servers can be either intersections or signal controlled intersections. Representing the vehicles individually allows modeling disaggregated route-choice, which is particularly useful when assessing en-route choices. (DYNASMART Jayakrishnan, et al. 1994), FASTLANE (Gawron, 1998). Mezzo (Burghout, 2004a), the simulator used in this study, belongs also to this third group.

III.1.3 Mezzo simulation model

Mezzo is a mesoscopic traffic simulation model (Burghout, 2004a; Burghout et al., 2006), which models vehicles individually but does not represent lanes explicitly. The traffic network is represented by a joint of nodes and links. Nodes coincide with traffic join or diverge points (intersections, on/off ramps, origins or destinations), while links represent the road between nodes and are unidirectional (therefore a common two-way road is represented by two links, one per direction).

Links are divided into two parts: a running and a queuing part. The queuing part starts at the downstream node and grows towards the upstream node when the incoming flow to

the node is higher than the outgoing flow. The boundaries between the queuing and the running part change dynamically depending on the extent of the queue. The running part contains the vehicles on their way to the downstream node that are not affected by the queue yet. It may happen that sometimes there is no queue or, on the contrary the queue occupies the whole link and there is no running part. Vehicles exit the queue in the same order they entered to it, overtaking maneuvers are not feasible. Earliest exit times of the link are considered a function of the density in the running part only. Travel times on the running part are calculated with the following speed-density function:

$$V(k) = \begin{cases} V_{free} & k < k_{min} \\ V_{min} + (V_{free} - V_{min}) \cdot \left[1 - \left(\frac{k - k_{min}}{k_{max} - k_{min}}\right)^a\right]^b & k \epsilon [k_{min}, k_{max}] \\ V_{min} & k > k_{max} \end{cases} \quad (3.1)$$

where V_{free} and V_{min} are the free flow and the minimum speeds. k is the density in the running part, k_{max} and k_{min} are the maximum and minimum densities thresholds. a and b are parameters. Therefore, if the density in the running part of the link is lower than k_{min} vehicles will move at the free flow speed or if the density exceeds k_{max} all vehicles will move at a constant minimum speed.

A single queue is found at the downstream of the link, where vehicles wait to move out of it. Vehicle processing depends on the queue server, which can be defined for every single node depending on the capacity of each turning movement. The queue server captures lane channeling and connectivity. For each turning, the server looks backwards to find the number of queuing vehicles that intend to use every turning movement and processes them in sequence. The queue look-back for each turning movement represents the relationship between queue length and the blocking process that exists when a too long queue blocks access lanes to other turnings in the same node.

III.1.4 BusMezzo transit operations tool

Mezzo was implemented using the object-oriented programming approach, which allows further enhancements and developments. Each entity (eg. node, queue, vehicle, OD pair) is represented as an object with its functions and variables. This programming mode allowed the development of BusMezzo, the tool used for transit operations simulation (Toledo T. et al. 2008).

BusMezzo uses six different object classes: Bus Type, Bus Vehicle, Bus Line, Bus Route, Bus Trip and Bus Stop.

1. **Bus Type**: Definition of the bus characteristics and attributes: length, number of seats and passenger capacity. It is a static class during the simulation.
2. **Bus Vehicle**: Inherits the attributes of the specific bus type adding the characteristics and functions that are relevant for each vehicle during the simulation. A list of scheduled trips has to be defined for each vehicle and the model represents the trip-chaining including layover and recovery times. During the simulation the occupancy is updated and determines the maximum number of passenger that can board per stop.
3. **Bus Line**: contains the definition of the line (origin and destination terminals and sequence of stops). It keeps track of all trips done.
4. **Bus Trip**: maintains the schedule of expected arrival times at each stop for the specific trip. While running the simulation it calculates the actual departure time from the origin terminal and records arrival times at stops.
5. **Bus Route**: a sub-class of the general Route object defines the route as an ordered sequence of links.
6. **Bus Stop**: keeps the information about each bus stop (i.e. in which link is the stop located, length, type (lane or bay) and availability of traveler information) and holds the list of bus lines that use the stop. The objects holds information about dwell times, bus arrivals and departures and information on passenger waiting times.

Mezzo is an event-based simulation model. Therefore, the simulation progresses from event to event chronologically. The initialization of the model consists in creating the Bus Line, Bus Route and Bus Type objects. When a scheduled trip is started a Bus Trip object is generated, and a Bus Vehicle assigned to this trip with the properties of a certain Bus Type. In case the trip is not the first trip of the vehicle and the recovery time from the previous trip is not completed, the departure is deferred until the vehicle is available.

When a vehicle enters a link on its route, it checks whether or not there are stops to be done for a certain route in the link. If there are no stops the bus crosses the link as any other vehicle, according to the travel time given for a certain traffic conditions. The events of entering and leaving a link are registered. If there are stops within the link, the stopping process is also registered: travel time to reach the stop, arrival time, dwell time (taking into account if holding strategies are implemented) and departure time are tracked.

At the end of a line, the bus waits until the next trip if more trips are scheduled for that vehicle. If the scheduled departure time for a trip has passed, the bus leaves immediately after the recovery time.

The simulation is designed to model the behavior of the vehicle if control strategies are set. For instance, if holding control is in place, the simulation checks for each stop if it is a holding point or not for the trip and for how long should the vehicle be held.

Outputs of the model include stop level statistics (i.e. early and late arrivals, dwell times, boarding and alighting passengers, bus load and travel times between stops). Other aggregation measures like schedule adherence, headway and passenger waiting time distributions are also automatically calculated.

The assumptions made about processes implemented in the model are determinant to understand and assess the results. The main models included in BusMezzo describe passenger demand (arrival and alighting), dwell time and trip chaining.

- **Passenger demand**

Passenger demand is composed of two variables: passenger arrival rates at stops for each line and alighting demand at each stop. The inputs are time-dependent and used as mean values in stochastic processes. Arrival processes are described by a Poisson distribution, tested by authors in other works (Fu and Yang, 2002; Dessouky et al., 2003).

$$B_{ijk} \sim Poisson\ (\lambda_{ijt_k}, h_{ijk}) \tag{3.2}$$

Where B_{ijk} is the number of passengers wishing to board line i et stop j on trip k. λ_{ijtk} is the arrival rate for line i at stop j during the time period t_k. h_{ijk} is the time headway between the preceding bus (on trip k-1) and the bus on trip k.

Passenger alighting process follows a Binomial distribution (Morgan, 2002; Liu and Wirasinghre, 2001). Alighting is modeled as a fraction of the on-board passengers arriving at each stop.

$$A_{ijk} \sim Binomial\ (L_{ijk}, P_{ijt_k}) \tag{3.3}$$

A_{ijk} is the number of alighting passengers from line i at stop j on trip k. L_{ijk} is the passenger load on arrival at stop j on the bus trip k of line i. P_{ijtk} is the probability for a time period t_k that a passenger will get off on line i at stop j.

- **Travel time**

Travelling time consists of running and dwell times. Dwell time is the sum of the time slices consisting in opening doors, boarding and alighting of passengers, closing the doors, and bus getting off the stop. The dwell time is calculated in BusMezzo adopting a model of the Transit Capacity and Quality Service Manual (Kittelson & Associates et al., 2003). Time needed for passengers to board and alight is calculated separately and overall dwell time is the one determined by the door that has the longest service time. It can also distinguish between in-lane stops and bay stops..

The model takes into account possible control strategies at stops. The departure time from a stop is calculated by:

$$ET_{ijk} = \max\ (AT_{ijk} + DT_{ijk}, CT_{ijk}) \tag{3.4}$$

Where ET_{ijk}, AT_{ijk} and CT_{ijk} are the departure time, actual arrival time and the departure time resulting from the control strategy implemented for line i on trip k from stop j, respectively.

- **Trip chaining**

As mentioned before, vehicles are assigned origin and destination stations and a route between them. It is possible to model the effect of a vehicle doing trips in sequence (which is common in real world where each vehicle follows a schedule). The accumulated delay of a vehicle can affect the whole line and it is important to take this feature into account. The actual recovery departure time for a trip is calculated as the later between the scheduled departure time and the earliest time a bus is ready to depart after completing the previous trip and the compulsory recovery time:

$$DPT_{bk} = \max\left(ST_{bk}, AT_{b,k-1} + RT_{min}, \varepsilon_{bk}\right) \quad (3.5)$$

Where DPT_{bk} and ST_{bk} are the actual and scheduled departure time for trip k by bus vehicle b, respectively. $AT_{b,k-1}$ is the arrival time of bus b from the previous trip at the terminal. $RT_{\min}$ is the minimum recovery time required between trips. ε_{bk} is a lognormal error term that captures stochastic departure delays.

The capabilities of Mezzo as a tool for evaluation of transit operations were demonstrated with an application to a real high-demand line in Tel-Aviv (Toledo T. et al. 2009). The case study demonstrates the aptitude of BusMezzo to reproduce the phenomena of bus brunching and propagation of headway variability along the route. It is important to highlight the fact that the model needs a rigorous calibration and validation to be meaningful.

III.2 The holding problem

Holding strategies can be designed in different forms, but there are two major bus control strategies, schedule-based and headway-based. The first one is focused on maintaining vehicles to a predetermined schedule and the second one consists in maintaining a constant headway between successive vehicles.

However, in case of holding a vehicle, the holding time should not be too much. Barnett (1974) established that in frequent services a holding time longer than 60 seconds is not acceptable for passengers.

- **Schedule-based control strategy**

This scheme controls buses toward keeping a preplanned schedule. The location of the previous or following bus is irrelevant; hence, it is an easier control strategy tom implement than headway-based strategies. If all buses keep them attach to the schedule, bus bunching will be reduced.

The formulation of a schedule-based strategy was given by Cats et al. (2010a):

$$ET_{ijk} = \max\left(SET_{ijk} - s_{ij}, AT_{ijk} + DT_{ijk}\right) \tag{3.6}$$

Where $ETijk$ is the exit (departure) time for line i on trip k from stop j, $SETijk$ is the corresponding scheduled exit (departure) time and sij is a non-negative slack size defined for line i at stop j. $ATijk$ is the actual arrival time and $DTijk$ is the dwell time.

- **Headway-based control strategy**

In this case the aim of the scheme is to maintain the headway between consecutive buses within a range, to avoid bus bunching and reduce passenger waiting time. It is assumed that vehicles cannot be sped up; therefore, the only strategy is to hold vehicles at stations. These strategies are more difficult to implement since real-time information systems are required.

If the strategy only takes into account the headway from the preceding vehicle, the holding criteria is defined by a minimal headway requirement (Cats et al., 2010a):

$$ET_{ijk} = \max\left(AT_{ij,k-1} + \alpha H_i^{k-1,k}, AT_{ijk} + DT_{ijk}\right) \quad (3.7)$$

Where $Hik-1$, is the planned headway between trips $k-1$ and k on line i, and α is a threshold ratio parameter. This parameter defines the minimum allowed headway relative to the planned headway. Both analytical and simulation-based studies that searched for the optimal threshold parameter found it to be in the range of 0.6 to 0.8 (Turnquist and Blume 1980, Fu and Yang 2002, Cats et al., 2010a, Rossetti and Turitto 1998) proposed to choose the threshold value dynamically each time that holding strategy is actuated based on the number of passengers on-board.

Headway-based strategies can incorporate into the holding criteria also the headway to the succeeding vehicle. This additional information can be utilized for keeping even headways by applying the following holding criteria:

$$ET_{ijk} = \max\left(AT_{ij,k-1} + \frac{\left(AT_{ij,k-1} - AT_{ij,k-1}\right) + \left(AT_{im,k+1} + SRT_{m,j} - AT_{ij,k}\right)}{2}, AT_{ijk} + DT_{ijk}\right) =$$
$$\max\left(AT_{ij,k-1} + \frac{AT_{im,k+1} + SRT_{m,j} - AT_{ij,k-1}}{2}, AT_{ijk} + DT_{ijk}\right) \quad (3.8)$$

Where m is the last stop that was visited by bus trip $k-1$ and $SRTm$, is the scheduled riding time between stops m and j. This strategy implies that buses are held only if the headway from the preceding bus is shorter than the headway to the succeeding vehicle. Note that this holding strategy is independent of the planned headway (Cats et al., 2010a). Nevertheless, it has been proved analytically by Daganzo (2009) that schedule deviation and planned headway deviation under a similar adaptive control strategy are realistically small. Koutsopoulos and Wang (2007) simulated urban rail operations and found out that headway-based strategy significantly benefits when applied at origin terminals. Cats et al. (2010c) implied that in order to implement this strategy at intermediate stops along the route accessing to real-time AVL data and bus vehicle-control centre communication network is required. Headway-based strategies defined by equations 3.7 and 3.8 can be integrated to form a strategy that keeps even headways while restricting the maximum allowable holding time by the minimum headway requirement (Cats et al., 2010a):

$$ET_{ijk} = \max\left(\min\left(AT_{ij,k-1} + \frac{AT_{im,k+1} + SRT_{m,j} - AT_{ij,k-1}}{2}, AT_{ijk} + \alpha H_i^{k-1,k}\right), AT_{ijk} + DT_{ijk}\right) \quad (3.9)$$

III.3 The optimization problem

III.3.1 Definition and typologies

The optimization of a function or a set of functions according to some criteria belongs to the field of numerical analysis in applied mathematics. The most common form is the minimization of a real-valued function f in a parameter space $\vec{x} \epsilon P$, respecting the constraints in the solution vectors. In many real-life problems, complicated functions with many variables have a lot of local minima and maxima. Finding local optima is relatively straightforward, but finding the global maximum or minimum might be practically impossible in some cases. The approaches to the problem can be deterministic or stochastic. Deterministic problems behave regularly (i.e. given an input always produce the same output) and are usually easier to handle and solve. On the other hand, stochastic problems behave randomly and output values of the objective function are not assumed as exact.

Stochastic problems require stochastic optimization methods that use algorithms which incorporate probabilistic (random) elements, either in the data (the objective function, the constraints, etc.), or in the algorithm itself (random parameter values, random choices, etc.), or in both. In simulation-based optimization and real-time estimation and control operations, random "noise" arises and leads to the use of algorithms which incorporate statistical inference tools to estimate the real values or make statistically optimal decisions about the steps to follow to reach the global minimum or maximum.

III.3.2 Metaheuristic algorithms

An algorithm is a series of steps for solving a problem. Metaheuristic algorithms are computational methods to optimize a problem by improving a candidate solution with regard to a given measure of quality. Since there are few or no initial assumptions about the problem to optimize they can search a large space of candidate solutions. They are used in problems where a no-continuous space is explored, like in the Travelling Salesman Problem (TSP) where the search-space grows more than exponentially with the size. The most popular metaheuristics are simulated annealing (Kirkpatrick et al., 1983), genetic algorithms (Holland et al., 1975), ant colony optimization (Dorigo, 1992) and tabu search (Glover, 1989).

Metaheuristics are not guaranteed to find the optimum or even a near-optimal solution. In fact, depending on the type of problem different metaheuristics can work differently. It is a matter of experience and reviewing the written literature to know which the most appropriate metaheuristics for every problem are.

III.3.3 Greedy algorithms

A greedy algorithm is a sequence of steps in a routine that always chooses the best solution at each step, hoping that this choice will lead to the globally optimal solution. However, choosing the best solution at each step might not yield optimal solutions.

In dynamic programming, a choice is made at each step, but it depends on the solutions to subproblems. Therefore, dynamic-programming problems are solved in a bottom-up manner, from smaller to larger subproblems. On the other hand, in greedy algorithms the best choice is done at the moment and the arising subproblem is solved afterwards. The choice may depend on the choices done so far, but neither on future choices nor solutions to subproblems. Thus, a greedy algorithm progresses in a top-down way, making the decisions one after another and reducing the problem instance to a smaller one.

A general scheme to design greedy algorithms has the following steps:

1. Cast the optimization problem as one in which a choice is made and a subproblem appears at every step
2. Prove that there is always an optimal solution that makes the greedy choice, so that the technique is always safe.
3. Prove that, once each greedy choice is done, a subproblem remains, with the property that if an optimal solution is combined to the greedy choice we have made, it converges to an optimal solution of the original problem.

However, some of the principles cannot be proved when dealing with complicated problems, such as stochastic problems or when exhaustive search is impractical. These principles have to do with two properties that need to be proved if we want to be able to ensure that a greedy algorithm yields a globally optimal solution: the greedy-choice property and the optimal substructure.

The greedy-choice property says that a globally optimal can be arrived at by making a locally optimal greedy choice. A problem has an optimal substructure if an optimal solution contains within it optimal solutions to subproblems.

Both features have to be demonstrated for the problem to be solved to ensure the applicability of greedy algorithms.

III.3.4 Genetic algorithm (GA)

III.3.4.1 Overview and development

According to the natural selection theory stated by Charles Darwin for the first time in 1859, biological organisms evolve over several generations based on the principle of "survival of the fittest". In nature, each individual in a population competes with each other for limited resources. Individuals that perform poorly have fewer chances to survive, and the more adapted to the environment or the circumstances an individual is, the more probability to survive and produce a larger offspring. During the reproduction, the good characteristics of the ancestors can produce more adapted offspring and after a few generations the species evolve spontaneously adapting to the environment. A concept that works in nature could also be reproduced in optimization techniques.

In 1975, John Holland developed this idea and described how to apply those principles to optimization problems, called Genetic Algorithms (GAs). His theory has been further developed and nowadays GAs are considered a useful and powerful tool to solve certain optimization problems. Genetics is the science that deals with the mechanisms responsible for similarities and differences in a species. Genetics helps us to differentiate between heredity and variations. GAs are based on the principle of genetics and evolution.

GAs are an example of mathematical technology transfer; how mathematics can describe different natural phenomena to solve a wide variety of problems. Today, GAs are used to resolve complicated optimization problems like timetabling, job-shop scheduling, games playing.

III.3.4.2 GA as an emulation of nature

John Holland published *Adaptation in Natural and Artificial System* in 1975 with a double aim: to improve understanding of the adaptation process and to design artificial systems with similar properties to natural systems.

The basic idea was simple: the genetic pool of a population potentially contains an improvement to an adaptive problem. However, the solution may not be active because the genetic combination on which relies can be split between several individuals. Through crossover during the reproduction a subject can inherit the desired gene.

Recombination is a key feature for evolution, consisting in taking two genotypes and mixing their genes to produce a new genotype. In biology, crossover is the most common recombination: two chromosomes are cut at one point and the halves are spliced. The child should inherit the best characteristics of the parents and surpass its ancestors.

Mutation is the other technique applied in GAs. It consists in change the value of genes. Mutation is not very frequent in natural evolution and mostly engenders non-viable individuals but it is a good way to explore wider the search space.

III.3.4.3 Properties of GA

GA has a couple of important characteristics. First of all is a stochastic algorithm; therefore, randomness plays an essential role in both reproduction and selection procedures. Another important and advantageous point is that GA considers a population of solutions per iteration, so the algorithm can benefit from assortment recombining the different solutions to get better ones. GA can perform consistently well on a board range of problem types, hence it is considered robust and powerful.

Because of the big success of the algorithm, many other similar algorithms aroused, based on the natural evolution principle. All those are called Evolutionary Algorithms.

GAs are stochastic methods, therefore are not guaranteed to find the global optimum but an acceptably good solution to the problem. The general character of GA makes them very convenient when other techniques have failed or there is no much information about the

search space. They can be also hybridized with other optimization schemes, but must be applied on appropriate problems.

III.3.4.4 Algorithm description

In GA each solution is represented through a chromosome. To code all solutions into chromosomes is the first part. Then a set of reproduction operators has to be determined. Those are applied on the chromosomes to perform mutations or recombinations over solutions. The behavior of the GA is very dependent on these operators but in some cases it may be difficult to find a representation which respects the structure of the search space and the reproduction operators.

The GA starts generating a population of individuals. Generally, the initial population is generated randomly, so that there is enough genetic diversity to ensure that any solution in the search space can be engendered. Afterwards, the GA loops over the following iterative process:

- Selection: the first step is selecting individuals for reproduction. This selection is commonly random depending on the relative fitness of the individuals. The cost function to minimize has to be transformed into a fitness function that evaluates how good candidate solutions are.
- Reproduction: offspring are bred by the selected individuals. Two or more parents' genetic material is combined to create offspring. Recombination and mutation can be used to generate new chromosomes.
- Evaluation: the fitness of the new chromosomes is evaluated.
- Replacement: individuals from the old population are killed and replaced for the new ones.

The algorithm is stopped when the population converges toward the optimal solution.

The basic genetic algorithm is as follows:

- [start] Genetic random population of n individuals (suitable solutions for the problem)

- [Fitness] Evaluate the fitness f(x) of each individual x in the population
- [New population] Create a new population by repeating following steps until the New population is complete
    - — [selection] select two parents from a population according to their fitness (the better fitness, the bigger chance to get selected).
    - — [crossover] With a crossover probability, cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
    - — [Mutation] With a mutation probability, mutate new offspring at each locus (position in chromosome)
    - — [Accepting] Place new offspring in the new population.
- [Replace] Use new generated population for a further sum of the algorithm.
- [Test] If the end condition is satisfied, stop, and return the best solution in current population.
- [Loop] Go to step2 for fitness evaluation.

III.3.4.5 Advantages and limitations of GA

Comparing GA with other optimization methods, one of the main advantages is that the fitness function can be nearly anything; there are not any definite mathematical restrictions to the properties of the function (discrete, multimodal, interger, etc.)

Genetic algorithm differs from conventional optimization techniques in following ways:

1. GAs use coded parameters rather that parameters themselves.
2. GAs does not search from a single point (most optimization strategies do) but from a population of points, which gives them more robustness.
3. GA uses fitness function as a parameter to evaluate rather than derivatives. Hence, it can be applied to any continuous or discrete optimization problem if a proper decoding function is specified.
4. GAs use probabilistic operates rather than deterministic.

Advantatges	Limitations
1. Solution space is wider 2. The fitness landscape is complex 3. Easy to discover global optimum 4. The problem has multi objective function 5. Only uses function evaluations. 6. Easily modified for different problems. 7. Handles noisy functions well. 8. Handles large, poorly understood search spaces easily 9. Good for multi-modal problems Returns a suite of solutions. 10. Very robust to difficulties in the evaluation of the objective function. 11. They require no knowledge or gradient information about the response surface 12. Discontinuities present on the response surface have little effect on overall optimization performance 13. They are resistant to becoming trapped in local optima 14. They perform very well for large-scale optimization problems 15. Can be employed for a wide variety of optimization problems	1. The problem of identifying fitness function 2. Definition of representation for the problem 3. Premature convergence occurs 4. The problem of choosing the various parameters like the size of the population, mutation rate, cross over rate, the selection method and its strength. 5. Cannot use gradients. 6. Cannot easily incorporate problem specific information 7. Not good at identifying local optima 8. No effective terminator. 9. Not effective for smooth unimodal functions 10. Needs to be coupled with a local search technique. 11. Have trouble finding the exact global optimum 12. Require large number of response (fitness) function evaluations 13. Configuration is not straightforward

Figure III.1 GA advantages and limitations

III.3.4.6 Terminologies

GA uses a metaphor and considers feasible solutions as individuals living in an environment. Individuals made of are binary digits, which is convenient to be stored in a computer as a string of bits.

- **Population**

It is a set of individuals currently involved in the search process. It consists of a number of individuals that are being tested at each generation. Two parameters that have to be considered are the population size and the initial population.

The population size should depend on the complexity of the problem. Usually is a random initialization: in binary coding this means that each bit is initialized to a random 0 or 1. However, in some cases when we have known good solutions, preset initial populations can be used to converge faster. In all cases it is essential to make sure that the initial gene pool is large enough and there is diversity, so that all search space can be explored.

There is a trade-off between GA efficiency and computational time to converge that depends on the population size. The larger the population is, the more likely is the GA to find the global optimum instead of local ones (Goldberg, 1987). On the other hand, the larger the population, the more computational cost, memory and time are required. In has been established that the time required by a GA to converge is O (N·logN) function evaluations where N is the population size.

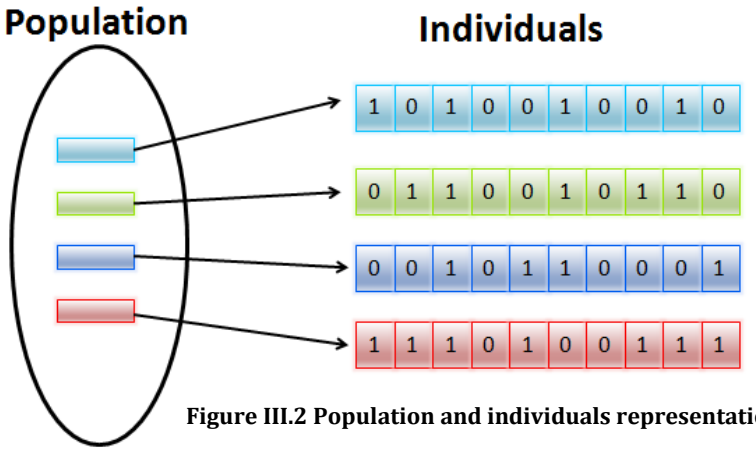

Figure III.2 Population and individuals representation

- **Individual**

It is a single solution. Each individual is represented by a chromosome. A chromosome is a sequence of genes.The morphogenesis function associates each genotype with its phenotype. It is important to ensure that each chromosome defines a unique solution and not several. The function might probably be not bijective but should make sure that all the

candidate solutions of the problem must correspond at least one feasible chromosome, to ensure that the whole search space is explored. If the function is not injective, i.e., different chromosomes encode the same solution, the representation is called degenerated. Small degeneracy is not a problem, but a more serious degeneracy could add confusion in the search because the algorithm might not be able to spot the favorable genes.

Chromosomes are encoded by bit strings:

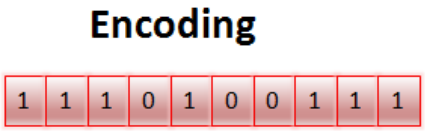

Figure III.3 Chromosome representation

- **Genes**

Genes are the basic units of GAs. A gene is a bit string of arbitrary lengths. It is the GA's representation of a single factor value for a control factor, where control factor must have an upper bound and lower bound.

The structure of each gene is defines in a record of phenotyping parameters: instructions for mapping between genotype and phenotype. This mapping is necessary to convert solution sets from the model into a form that GA can recognize and work with, and for converting individuals from the GA into forms evaluable for the model.

The genes are represented the following way:

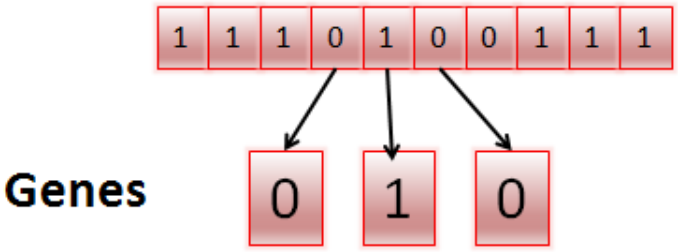

Figure III.4 Genes representation

When dealing with a problem with binary variables, the length of each gene is a unit. Hence, there are as many genes as variables in the problem and each can have a value of 0 or 1. No mapping between genotype and phenotype is necessary since both the problem variables and the chromosomes are binary values.

- **Fitness**

The fitness of an individual is the value of the objective function for its phenotype. The chromosome has to be decoded first and evaluated with the function afterwards. The fitness indicates how good a solution is and how close the chromosome is to the optimal one.

- **Data structure**

The main data structures in GA in binary problems are chromosomes, objective function values and fitness values. Using MATLAB, the chromosome population can be stored in a single array given the number of individuals and the length of their genotype representation.

III.3.4.7 GA Operators

- **Encoding**

Encoding is the process of representing individual genes using bits, numbers, trees, arrays or other objects. Binary is the most common encoding technique and each chromosome encodes a binary bit string. Every bit represents some characteristics in the solution, but it differs from problem to problem. Coded strings with 0s and 1s can represent integers exactly, finite real numbers and obviously binary variables or decisions.

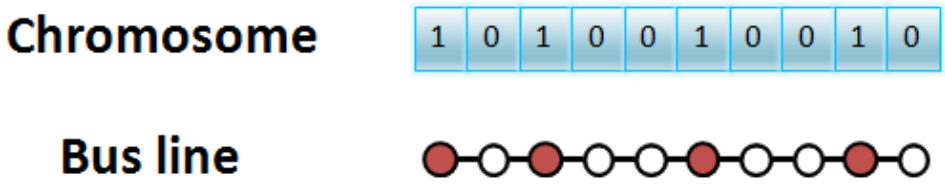

Figure III.5 Binary coded bus line representation

Breeding

Breeding is the key process of GA to create fitter individuals. It consists of three steps:

1. Parents selection
2. Crossing parents to create new offspring
3. Replacing the old individuals with the new ones

- **Selection**

Selection is the process of choosing two parents from the population for crossing. Once the encoding strategy is set, is has to be decided how to choose individuals that will create offspring and how many offspring to create. Emphasizing the fittest individuals, better offspring is expected. According to Darwin's theory the best individuals will be the ones to survive and create offspring.

The selection method picks randomly chromosomes out of the population according to their evaluation function: the higher the value of the fitness function, the more possibilities an individual has to be selected. The selection can be more or less exigent, depending on the degree to which the better individuals are favored. We call this exigency selection pressure. This pressure drives the GA to improve the population fitness over generations. However, if the pressure is too low, the convergence rate will be slow and the GA will take unnecessarily longer time to approach to the solution. On the contrary, if the selection pressure is too high, the GA could converge prematurely to an incorrect (sub-optimal) solution.

Additionally, selection techniques should preserve some population diversity to avoid undesirable premature convergence.

- **Scaling function**

The scaling function converts raw fitness obtained evaluating the individuals with the objective function into values in a range that is suitable for the selection function. Typically there are two types of scaling functions: proportionate functions and ordinal-based functions. Proportionate-based pick out individuals based on their fitness relative to the other individuals. Ordinal-based schemes select individuals upon their rank within the population.

- **Rank**: scales the raw scores of each individual and assigns each its position number in the sorted scores as its rank. The fittest individual has rank 1, the second 2, and so on. This strategy removes the effect of spread raw scores.
- **Proportional**: the probability assigned to each individual is proportional to its raw score. It may be prejudicial if the scores are too spread.
- **Top**: after ranking the individuals, only the best ones are selected for breeding. The size of the group has to be set between 1 and N (N is population size). All the individuals in the group have the same probability of reproducing while the rest have none.
- **Shift linear**: the fitness range of the population is redistributed to adapt the selection pressure.

- **Selection**

Selection has to be balanced with variation form crossover and mutation. Too strong selection means sub optimal highly fit individuals will take over the population, reducing the diversity needed for change and progress; too weak selection will result in too slow evolution. The various selection methods are discussed as follows:

Roulette Wheel Selection

It is one traditional GA technique that consists in selecting from the mating pool with a probability proportional to the fitness of each individual. The principle is a linear search through a roulette wheel, where the sizes of the wheel slots are weighted in proportion to the individual's fitness values. A real-valued interval is determined as either the sum of the individuals' expected selection probabilities or the sum of the raw fitness values over all the individuals in the current population. The operation of the technique is easy to imagine with the parallelism to a roulette wheel: the expected value of an individual is that fitness divided by the actual fitness of the population. Each individual is assigned a slice of the wheel proportional to the individual's fitness. The wheel is spun N times, where N is ne number of individuals of the population, and on each spin the individual under the marker is selected to be in the pool of parents for the next generation.

It is only a moderately strong selection, since fittest individuals are not guaranteed to be selected although they have a greater chance.

Random Selection

This process selects parents randomly from the population. On average is more disruptive than other techniques.

Tournament Selection

Tournament selection puts selective pressure by holding a tournament competition among N_u individuals, a number that has to be specified. Parents are chose randomly in groups of N_u, and the one with best fitness is the winner of the tournament. There are as many tournaments as population needed in the mating pool. At the end the mating pool contains all the winners from all the tournaments and therefore; has a higher average fitness.

Stochastic Universal Sampling

It consists in mapping the individuals to contiguous segments of a line, and as in the roulette wheel selection each individual segment is equal in size to its fitness. Equally spaced pointers are placed over the line, as many as individuals are needed to be selected. Considering *N* pointers, the distance between them is *1/N* and the position of the first one is randomly generated within the range [0 , *1/N*]. This technique provides zero bias and minimum spread.

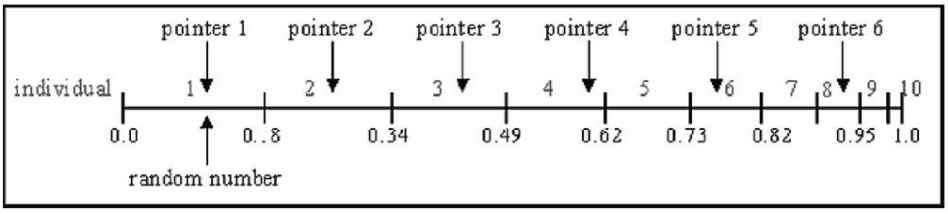

Figure III.6 Example of stochastic universal sampling choosing 6 individuals

- **Elitism**

Apart from reproduction processes, elitism significantly improves GA's performance. It consists in transferring directly the best or the few best chromosomes to the next generation, otherwise they could be lost if they were not selected to reproduce or if crossover or mutation destroyed them.

- **Crossover (recombination)**

Crossover is the process of taking two parents and producing a child from them. It is applied after the selection and should produce fitter individuals by combining the most favorable genes of the parents.

Single Point Crossover

It is the most traditional crossover, where two mating chromosomes are cut once at corresponding points and the sections after the cutting point are exchanged. The crossover point is selected randomly along the length of the individual. If an appropriate point is selected, better children will be obtained.

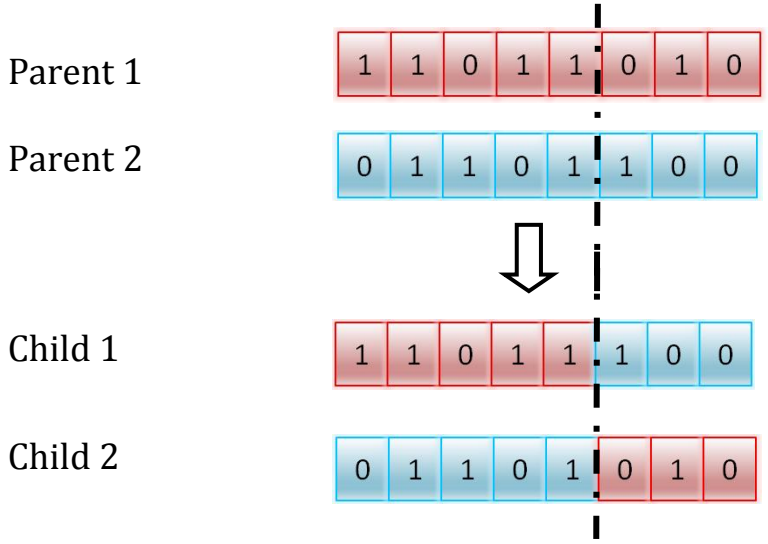

Figure III.7 Single point crossover

Two Point Crossover

The idea of the two point crossover is the same as in the one point crossover, but with another cut point randomly chosen along the chromosome. Two crossover points are chosen and the contents between these points are exchanged between two mated parents.

Having more crossover points blocks are more likely to be disrupted but on the other hand, the problem space might be more thoroughly searched. With one-point crossover both the head and the tail from one individual cannot be passed together to the offspring. However, using the two-point crossover avoids this drawback. In fact, genes that are close to each other have more chances to be passed together and in leads to a correlation between genes next to each other. Uniform crossover avoids this unwanted correlation.

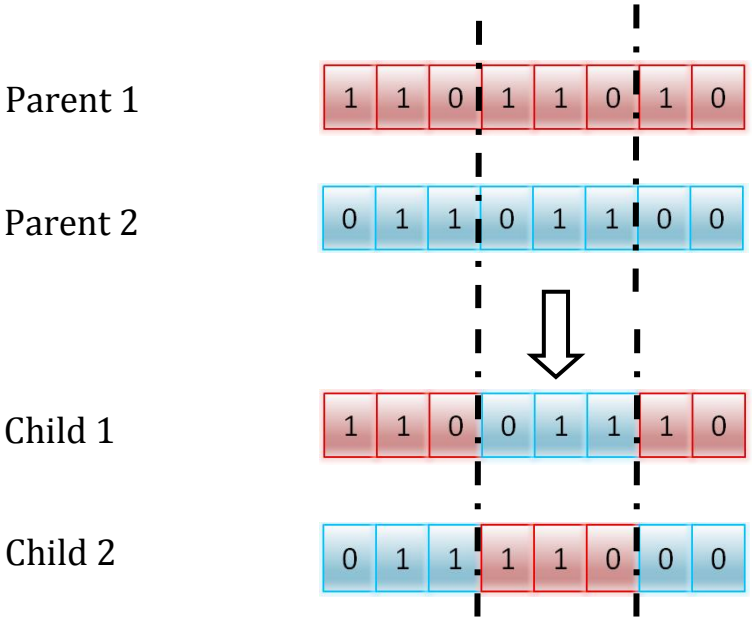

Figure III.8 Two-point crossover

Uniform Crossover

Each gene of a child is taken from one or the other parent randomly, according to a random binary crossover mask of the same length as the chromosomes. If the gene in the mask is a 0, that gene is taken from the first parent, if it is a 1, from the second. The crossover mask is randomly generated with an equal probability to create 0's and 1's.

Other crossover techniques are the multi-point crossover, three parent crossover, crossover with reduced surrogate, shuffle crossover or other more elaborated techniques like Precedence Preservative Crossover (PPX), Partially Matched Crossover (PMX).

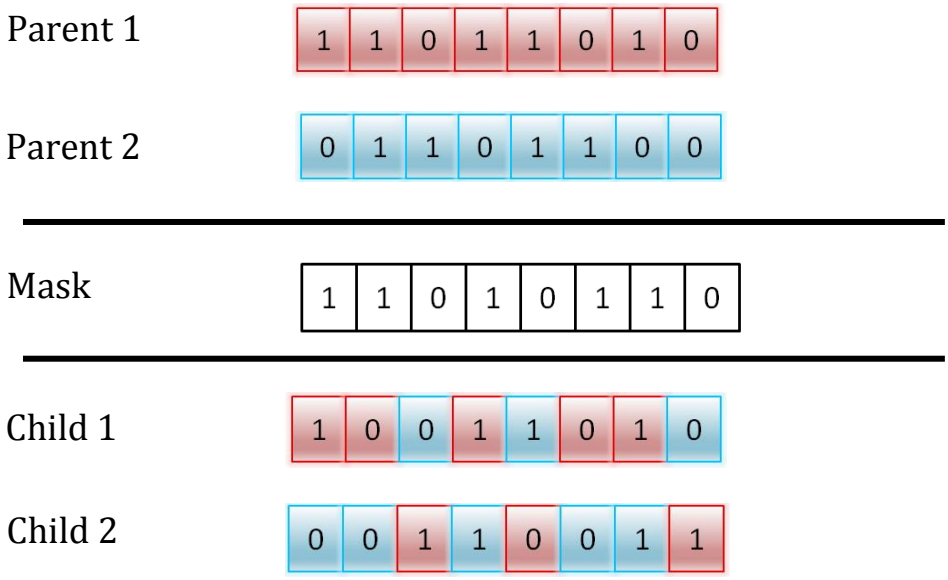

Figure III.9 Uniform crossover

Crossover Probability

A basic parameter in crossover is the crossover probability (Pc). If this probability is 100%, all offspring are made by crossover. However, it is always good to leave some population survive to next generation.

- **Mutation**

Mutation is applied after crossover, as a measure to prevent falling into local minimum and recovering the lost genetic materials. It is a simple operator and whereas the aim of crossover is to exploit the current solution to find better ones, mutation has the aim of exploring the whole search space. Randomly modifying some gens introduces new genetic structures and thus ensuring ergodicity. A search space is ergodic if there is a non-zero probability of generating any solution from any population state.

Flipping

Flipping consists in changing a 0 bit for a 1, or 1 to 0 based on a mutation chromosome generated. The mutation chromosome has the same length as the parent and is randomly generated according to a fixed probability of having ones in the string, which means that the corresponding bit in parent chromosome will be flipped.

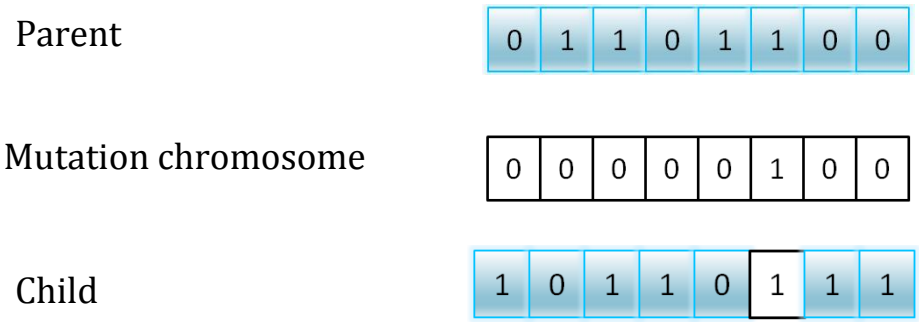

Figure III.10 Mutation flipping

Interchanging

Two random positions of the string are chosen and the bits corresponding to those positions are interchanged.

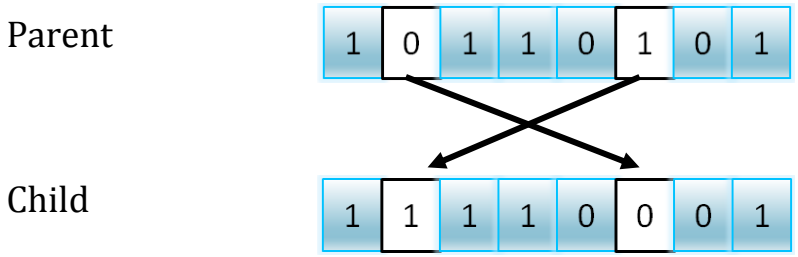

Figure III.11 Interchanging

Reversing

A random position is chosen and the bits next to that position are reversed and child chromosome is produced.

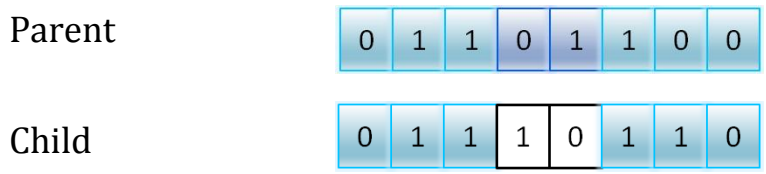

Figure III.12 Reversing

Mutation Probability

Is the parameter that decides how often mutation will happen. Mutation prevents GA from falling into local extremes, but it should not occur too often, otherwise GA would have no effect and the algorithm would be a random search.

- **Replacement**

Replacement is the last step of the breeding process. After parents two by two are drawn from the mating pool and breed two children, it should be decided if the parents should die and be replaced by the children. There are two kinds of methods: generational updates and steady state updates.

Generational updates consist in producing as many children as parents, and replace the complete population of parents by children.

In steady state update the individuals are inserted in the population as soon as they are created and each insertion means that the new individual replaces an old one. The old individual can be the worst one or the oldest one, for example. These techniques put a strong selection pressure.

Some of the most common techniques are:

- Random replacement: the two new children replace two random chosen individuals (the parents are also candidates).
- Weak parent replacement: weaker parents are replaced by stronger children. For example, between two parents and their two children only the fittest two will return to population.
- Both parents: is the case where each individual only breeds once and afterwards both parents are killed and the children survive.

III.3.4.8 GA convergence criteria

The most common convergence criteria for GAs are:

- Maximum number of generations: the algorithm stops when the specified number of generations is reached.
- Elapsed time: the algorithm will stop after when a specified time has elapsed.
- No change in fitness: if there is no change to population's best fitness value for a specified number of generations the process will end.
- Stall generations: stops if there is no improvement in the objective function for a sequence of consecutive generations of a chosen length.
- Stall time limit: the process stops if there is no improvement in the objective function during a time interval with a predefined value.
- Fitness limit: if the best fitness value is less than or equal to the specified value of Fitness limit, the algorithm stops.

➢ **Search refinement**

Parameters like selection, crossover and replacement might be very effective in the early stages of the search, but not necessarily towards the end of the search. In the beginning of the algorithm it is desirable to have spread points through the solution space in order to find at least the start of the various optima. Once the population is close to the optimum it might be better to reduce selection and replacement to cover that stringent region of space.

III.3.5 Multiobjective optimization

It might be the case where problems have more than one objective, since a single objective function is not enough to represent the problem being faced. There is a vector of objectives to be traded off:

$$F(x) = [F_1(x), F_2(x),...,F_m(x)] \tag{3.10}$$

The relative importance of the different objectives is not known until the best capabilities are determined and tradeoffs are fully understood.

Since F(x) is a vector, there is no unique solution to the problem. Instead, the concept of noninferiority (also called Pareto optimality or front) is used to characterize the objectives. Thus by using the Pareto front, a set of solutions can be found that are all optimal compromises between the conflicting objectives. Each solution in the Pareto front is not dominated by any other solution.

To define the concept, consider a feasible region Ω in the parameter space. X is an element of the n-dimensional real numbers that satisfies all the constraints. The performance vector F(x) maps parameter space into objective function space, as represented in two dimensions in the figure.

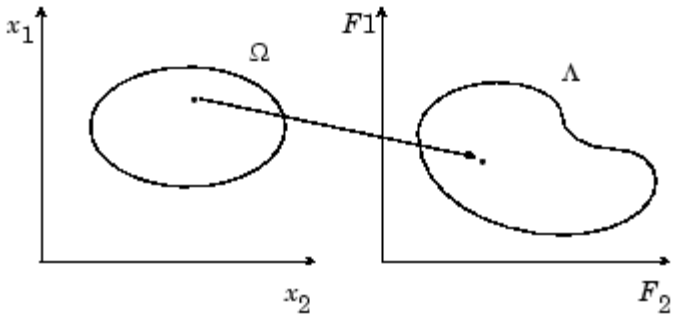

Figure III.13 Parameters and objective function spaces

This allows definition of the corresponding feasible region for the objective function space Λ:

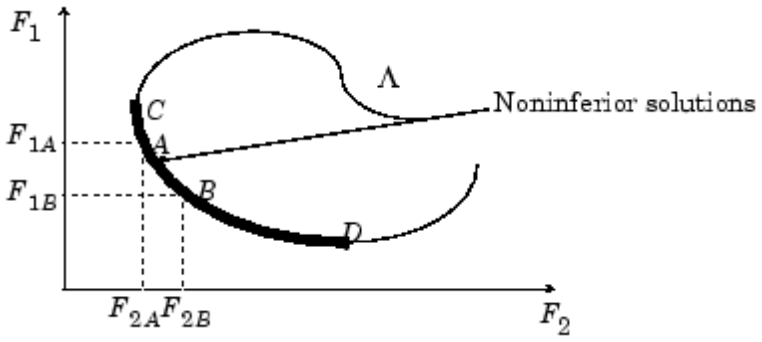

Figure III.14 Pareto front

A and *B* are noninferior solution points because an improvement in one objective, F_1, requires a degradation in the other objective, F_2, i.e., $F_{1B} < F_{1A}$, $F_{2B} > F_{2A}$.

Since any point in Ω that is an inferior point represents a point in which improvement can be attained in all the objectives, it is clear that such a point is of no value. Multiobjective optimization is, therefore, concerned with the generation and selection of noninferior solution points.

Noninferior solutions are also called *Pareto optima*. A general goal in multiobjective optimization is constructing the Pareto optima.

GAs have been applied to solve multi-objective optimization problems in the recent years. The multiple directionality and global search features of the GA make it very appropriate to solve multi-objective optimization problems. The main issues are how to determinate the fitness value of individuals according to all multiple objective functions.

The genetic multiobjective function uses controlled elitist genetic algorithm, which favors not only individuals with better fitness (rank) but also individuals that can help increase the diversity even if they have a lower fitness value. This is done by controlling the elite members of the population as the algorithm progresses. Using a Pareto fraction function, the number of individuals on the Pareto front is limited (elite members) and using a distance function, diversity on the front is maintained.

IV Description of the Case Study

IV.1 Description of the real line

IV. 1.1 Public transport network in Stockholm

The city of Stockholm has a multimodal public transport network that consists of bus, metro, regional/suburban rail, light rail, tram and archipelago boat. The operator of bus and rail services is Storstockholms Lokaltrafik (literally: Great Stockholm Public Transport), SL, owned by the Stockholm Country Council.

The operation and maintenance of the several bus lines is delegated to the contractors, while the management of the whole network is done from a control center.

Stockholm's bus lines are organized into a three-level hierarchy depending on their characteristics:

- Inner-city blue bus lines: these trunk lines travel through the main areas in the city center, with high frequency service and articulated buses. There are a total of four blue lines in the city center, all of them with high demand profiles.

- Suburban blue bus lines: a variant of the inner-city group that act as important feeder lines between the suburbs and public transports hubs in Stockholm, or providing crossway connections between suburbs.

- Service bus lines: the regular buses of the city, with less frequency and less maximum occupancy. They are painted in red, in contrast to the other types that are blue.

IV.1.2 APTS/ITS

World largest digital trunked radio is currently operating in Stockholm, with more than 2000 buses equipped and 28 depots integrated. Advanced Public Transportation Systems (APTS) are those Intelligent Transportation Systems (ITS) applied to public transit to improve operational efficiency, cost savings, safety and quality of service or other transit measures of performance. These new technologies make possible improvements in managing and controlling the bus transit operations, providing benefits in terms of speed, security and convenience directly to the customer. These APTS have the potential of changing in the near future the way operators provide their transit services and the way customers use the service. Automated Vehicle Location (AVL), Automatic Passenger Counting (APC), Automatic Vehicle Identification (AVI) and Electronic Fare Payment (EFP) are already implemented in the city buses and are being increasingly studied by researchers as valuable information to improve transit lines operations.

IV.1.3 Bus Line 1 characteristics

The present study is based in the blue bus line number 1 in Stockholm. The line connects the city west to east from Stora Essingen, an island located at the west of the city, passing through Cityterminalen (main bus terminal) and Hötorget (city commercial center) heading to the north-eastern port of Frihamnen over 33 stops. The way back consists of 31 bus stops and slightly different paths.

- **Bus stops**

Table IV.1 contains the stops in each direction and consecutive distances between them.

For a matter of convenience, the route with 33 stops heading to Frihamnen will be called EF33 and the inbound direction heading to Essingentorget will be called FE31.

The type of bus used in this line is an articulated one with 12 meters length, 55 seats and a maximum capacity of 110 passengers.

Index	Stop name	Distance to Next Stop (m)	Index	Stop name	Distance to Next Stop (m)
1	ESSINGETORGET	340	**1**	FRIHAMNEN	330
2	FLOTTBROVÄGEN	460	**2**	FRIHAMNSPORTEN	260
3	BROPARKEN	620	**3**	SEHLSTEDTSGATAN	210
4	PRIMUSGATAN	380	**4**	ÖSTHAMMARSGATAN	300
5	LILLA ESSINGEN	240	**5**	RÖKUBBSGATAN	180
6	WIVALLIUSGATAN	240	**6**	SANDSHAMNPLAN	500
7	FYRVERKARBACKEN	250	**7**	GÄRDET	540
8	VÄSTERBROPLAN	460	**8**	KAMPEMENTSBACK.	230
9	MARIEBERGSGATAN	290	**9**	STORSKÄRSGATAN	400
10	**FRIDHEMSPLAN**	390	**10**	**VÄRTAVÄGEN**	490
11	S:T ERIKSGATAN	320	**11**	JUNGFRUGATAN	380
12	S:T ERIKS SJH	310	**12**	NYBROGATAN	350
13	SCHEELEGATAN	240	**13**	HUMLEGÅRDEN	390
14	KUNGSBROPLAN	430	**14**	STUREPLAN	190
15	CITYTERMINALEN	180	**15**	NORRLANDSGATAN	310
16	VASAGATAN	340	**16**	SVEAVÄGEN	210
17	**HÖTORGET**	450	**17**	**HÖTORGET**	360
18	NORRLANDSGATAN	210	**18**	VASAGATAN	440
19	STUREPLAN	250	**19**	KUNGSBROPLAN	260
20	LINNÉGATAN	260	**20**	SCHEELEGATAN	400
21	HUMLEGÅRDEN	230	**21**	S:T ERIKS SJH	360
22	NYBROGATAN	490	**22**	S:T ERIKSGATAN	290
23	JUNGFRUGATAN	490	**23**	**FRIDHEMSPLAN**	850
24	**VÄRTAVÄGEN**	390	**24**	VÄSTERBROPLAN	210
25	STORSKÄRSGATAN	320	**25**	FYRVERKARBACKEN	300
26	KAMPEMENTSBACK.	450	**26**	WIVALLIUSGATAN	320
27	GÄRDET	470	**27**	LILLA ESSINGEN	340
28	SANDSHAMNPLAN	320	**28**	PRIMUSGATAN	730
29	RÖKUBBSGATAN	160	**29**	BROPARKEN	280
30	ÖSTHAMMARSGATAN	150	**30**	FLOTTBROVÄGEN	240
31	SEHLSTEDTSGATAN	270	**31**	ESSINGETORGET	0
32	FRIHAMNSPORTEN	470			
33	FRIHAMNEN	0			

Table IV.1 Blue bus line number 1 stops

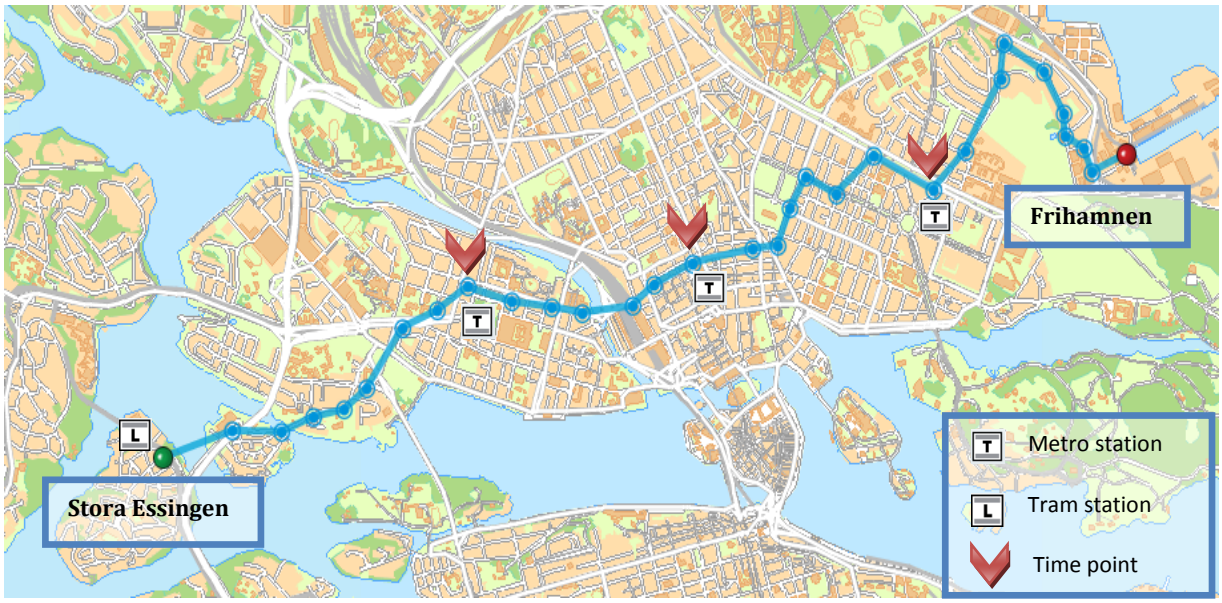

Figure IV.1 Blue bus line number 1 path

In Figure IV.1 the path of line 1 is represented with the connections between the line and the metro and light rail networks. Nowadays the route for line 1 is operated under schedule-based holding control with three holding points in Fridhemsplan, Hötorget and Värtavägen, explicitly indicated in the figure. It has to be pointed out that all three time points coincide with transfer stations. The driver relief point is planned to be at Fridhemsplan in which bus drivers change their working shifts with each other.

- **Timetable**

Since the current operational strategy of the line is schedule-based, a timetable is published for line 1. The frequency for this line changes occasionally from 5 to 4 minutes which makes a frequency of approximately 13 buses per hour.

- **Demand profile**

To simplify the model only one demand profile will be used in the simulation, which corresponds to the peak hour profile. Apart from calculating the number of passengers that board and alight at each stop it is also important to know the through passengers, which neither board nor alight at a certain stop. The attribute is calculated by the formula:

$$TP_i = L_{i-1} - AF_{i-1} * L_{i-1} \quad (4.1)$$

Where TP_i is sum of through passengers at given stop *i*. The bus load at stop *i-1* is presented by L_{i-1} and AF_{i-1} is alighting fraction, which means the probability of a passenger to alighting at the given stop.

Bus load represents the total passenger load after the alighting and boarding process at a stop and is calculated by the formula:

$$L_i = OBP_i + BP_i - AF_i * OBP_{i-1} \quad (4.2)$$

Where BP_i stands for number of passengers who are boarding at stop *i* and OBP_i represents the number of passengers who are already on-board at the same stop.

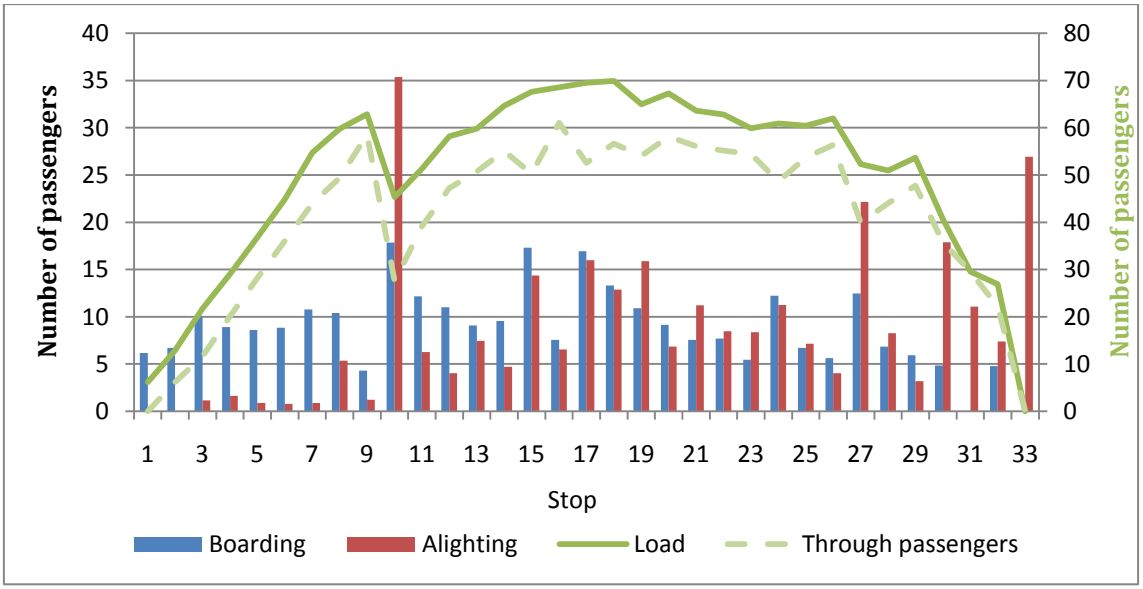

Figure IV.2 Load profile EF33

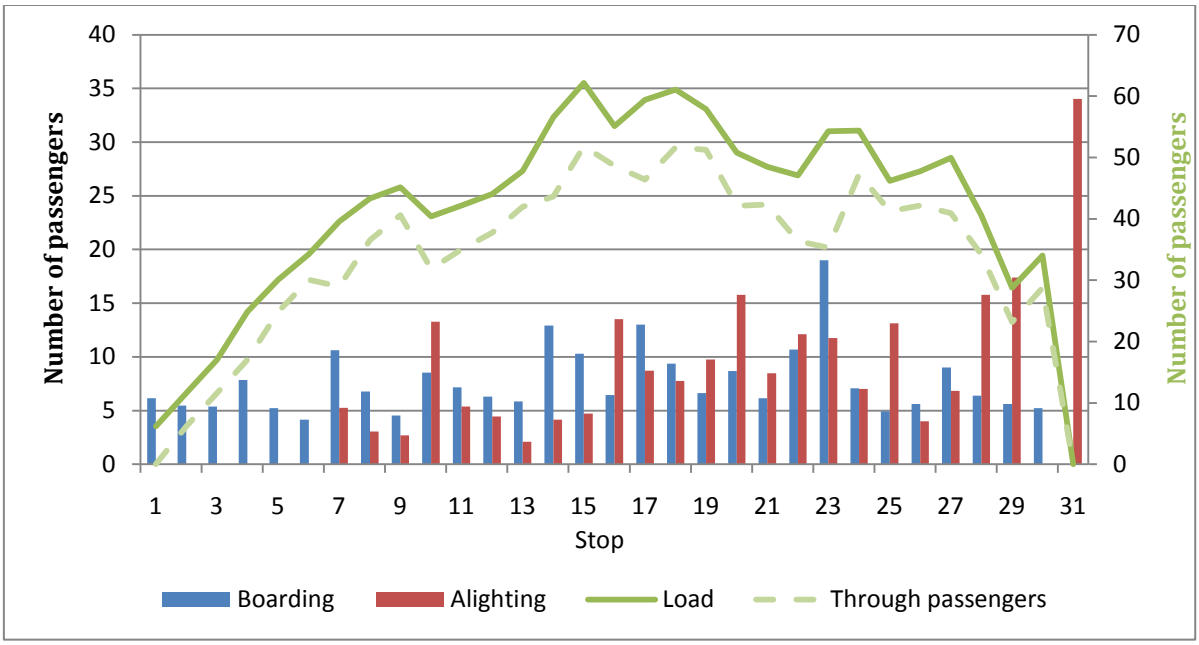

Figure IV.3 Load profile FE31

IV.2 Model of the line in BusMezzo

IV.2.1 Model implementation

Using data provided by SL, the Stockholm public transport operator the bus line number 1 will be modeled in BusMezzo. The data provided by SL contains observed arrival and departure time of the line at each stop, dwell time at each stop, aggregated number of passengers boarding and alighting per stop, distance between stops and other parameters of the line.

To the purpose of studying and assessing the effects of the different optimization algorithms, the blue line number 1 has been adapted and implemented in BusMezzo. The first step is to define the whole network as a joint of nodes and links.

The designed line has two directions, outbound (EF33) and inbound (FE31), with 33 and 31 stops respectively. Each node acts as a queue server, assigning a capacity to every turning movement. Therefore, a node is placed before every stop to reproduce the stochastic effect of traffic conditions. Since in BusMezzo origin and destination nodes of each route have to be defined, four more nodes are needed: two origins and two destinations, for the outbound and inbound routes respectively. Hence, a total of 36 nodes are required, numbered from 0 to 35.

Once the physical network is defined, the BusMezzo objects to model the bus transit lines have to be implemented, adjusting the variables and inputs of the model to the real data:

- **Bus Route**: the designed line consists of two routes: outbound (EF33) and inbound (FE31). Outbound route runs from node 1 to 31, passing through nodes in between them and the links that are defined between them in that direction (Figure IV.4). Inbound runs from 35 to 0, passing again through all the nodes but in the backwards direction. Hence, different links have to be used (Figure IV.5).

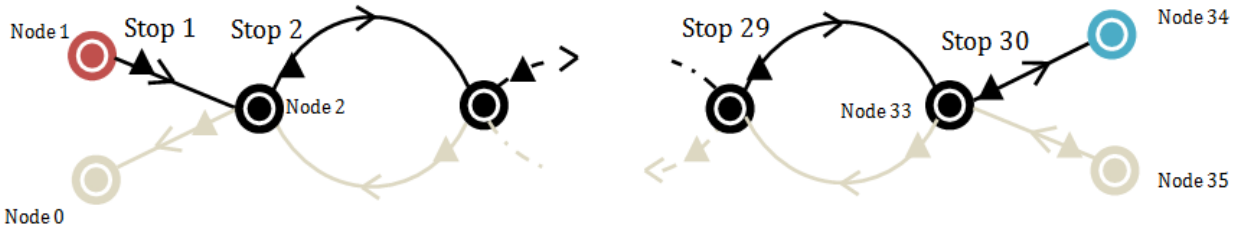

Figure IV.4 BusMezzo line implementation EF33

Figure IV.5 BusMezzo line implementation FE31

- **Bus Stop**: there are a total of 64 stops, each situated within every link, 20 meters from the upstream node. The minimal dwell time per stop is one second.
- **Bus Line**: the outbound and inbound directions are defined as two different lines, lines EF33 and FE31 with 33 and 31 stops respectively. The initial occupancy at the head of the line is considered to be zero.
- **Bus Trip**: in concordance to the total travelling time per direction and the trip chaining, 42 trips are done in each direction within the two hours of the simulation. The scheduled time between stops is calculated from the empirical data and depends on the distance between stops. The scheduled headway is variable changing from 5 to 4 minutes at certain time slices.
- **Bus Type**: the fleet is composed by standard buses of 12 meters, with 55 seats and a maximum capacity of 110 passengers.
- **Bus Vehicle**: According to the designed schedule, 42 vehicles are needed to cover the service in both directions, considering that the total travelling time for each

direction is 40 minutes and 5 minutes at the end of the line are minimum recovery time. According to this, the trip chaining for vehicles can be modeled.

IV.2.2 Replication estimation

To obtain consistent results from the simulation it is necessary to define the number of replications to be done for each algorithm step.

A measure of reliability is the standard deviation of the headway. According to Burghout W., 2004b, Cats et al. 2010b, the following equation can be used to limit the error depending on the number of replications:

$$N(m) = \left[\frac{S(m) * t_{m-1,1-\alpha/2}}{\bar{X}(m) * \varepsilon}\right]^2 \quad (4.3)$$

In which N(m) represents the number of replications while given m initial simulation runs, $\bar{X}(m)$ and S(m) are mean and the standard deviation of the objective function value. In this thesis, an error of 5% was chosen, which means a total of 50 replications per run. The results shown in all the document correspond to the averages of the 50 replications.

V Optimization algorithms

V.1 Optimization objective function

V.1.1 Literature review

The first step to define an optimization problem is to create an appropriate function or set of functions to be optimized.

Referring to the literature on the topic, several authors have used different objective functions to optimize the holding problem. Eberlein et al. (2001) only considered the total passenger waiting times, justifying that waiting times are more sensitive to control policies than in-vehicle travel times. However, no consideration was made for passenger waiting or holding times.
Delgado et al. (2009) used an objective function to minimize is the total times experienced by passengers, from the waiting time at stop to alighting at the destination. The objective function includes passenger waiting time at stops and in-vehicle waiting (holding time) only. Since the vehicle running times are assumed to be constant, it is not included. Likewise, dwell time is not considered.

The multi-objective function used by Cortés et al. (2009) considers total passenger waiting time at stops and on the other hand passenger holding time and passenger extra holding time when a station is skipped.

In the problem developed by Malzoumi et al. (2010) using the ant colony optimization, four terms define the objective function: waiting time at stops, holding time, a lateness penalty and an operational cost term.

As it can be observed from the past literature, there is no agreement yet in which is the best objective function to minimize when solving the holding problem. There is a clear trend to analyze the passenger waiting time at stops and holding time, but not much regard is given to in-vehicle riding time or dwell time. However, the holding problem and its consequences can be analyzed from different perspectives. Hence, several objective functions can be defined, taking into account the appropriate variables that are relevant from the perspective the problem is treated.

V.1.2 Passenger perspective

Since BusMezzo records all the events consecutively an objective function that takes into account all the different time components of a passenger bus trip can be defined. If we look at the different parts that comprise a bus trip from the traveler standpoint, we can classify them in basically three main types (four if we consider the holding strategy):

- *WT*: passenger waiting time at the origin stop.
- *DT*: total dwell time, which is the sum of all dwell times experienced by a passenger along his route.
- *RT*: total in-vehicle riding time; the sum of all the time slices in which the bus is travelling between stops, experienced by a passenger along his route.
- *HT*: total holding time is the sum of all time slices in which the bus is held at a stop because of the control strategy.

To take into account the overall line behavior, the different time components have to be aggregated for all passengers. Some studies (Kemp 1973, Ben-Akiva and Lerman 1985) suggest that users are more sensitive to waiting time than to in-vehicle riding time because of the uncertainty generated for the waiting. Therefore, different weights have to be assigned to each time components depending on the level of unreliability that the traveler perceives. The objective function is defined as:

$$f_1(\bar{x}) = \gamma_w \cdot WT(\bar{x}) + \gamma_d \cdot DT(\bar{x}) + \gamma_r \cdot RT(\bar{x}) + \gamma_h \cdot HT(\bar{x}) \quad (5.1)$$

Where $(\bar{x})$ is the number and layout of holding points along the line and γ_x the different weights for the different time components. The vector of variables $\bar{x}$ contains as many variables as stops and each position in the vector represents the corresponding stop according to their sequence in the line. Using binary system every stop can be either set as a time point assigning the value 1 to that position of the vector or a non-holding station assigning a 0 in the corresponding position.

V.1.3 Operator perspective

An important issue from the operator perspective is to optimize the fleet size and ensure a reliable service once that size is set. Wrong or non-optimal decisions made at this point can represent an increase in operational costs such as labor wage, depots cost and depreciation. The variable used by SL to assign fleet size is the 90th percentile of the total travel time (*TTT*).

Percentiles are specified using percentages, from 0 to 100. For an n-element vector X, percentiles are calculated as follows:

- The sorted values in X are taken to be the 100(0.5/n), 100(1.5/n), ..., 100([n-0.5]/n) percentiles.
- Linear interpolation is used to compute percentiles for percent values between 100(0.5/n) and 100([n-0.5]/n).
- The minimum or maximum values in X are assigned to percentiles for percent values outside that range.

Since only the outbound direction EF33 is studied, the total travel time considered will be the time from the departure from the first stop in that direction in Essingetorget to the arrival at Frihamnen at the end of the line. Therefore, the function to optimize from the operator perspective will be as follows:

$$f_2(\bar{x}) = p_{90}TTT(\bar{x}) \tag{5.2}$$

V.2 Algorithms definition

V.2.1 Greedy algorithm

The first approach to solve the holding problem for the given modeled case will be using a greedy algorithm. This algorithm will start from the base case (no holding points), and by adding one more holding point per loop will pursuit an improvement of the objective function $f_1(\bar{x})$. A flowchart of the algorithm running scheme is showed in Figure V.1.

As shown in the flowchart the algorithm starts initializing the model parameters, which is setting the number of time points in zero to run the base case (no time points). Once simulated the base case with BusMezzo the output data is read and stored. Immediately afterwards a loop is initialized that adds one holding point to the current case. This time point is tried iteratively in every feasible position until all combinations are tried. An internal subprocess changes the input for BusMezzo, runs the application and extracts the output data for every case within the loop. Once all the possible *N* layouts are simulated, their output is compared to choose the case with the lowest objective function value as the optimal one. Afterwards, the stopping criteria is checked: if the current best objective function value is lower than the objective function value obtained in the last loop (i.e. if there has been an improvement in the objective function by adding one more time point), the algorithm goes to the next loop. Before entering to the next loop and adding a new time point, the best time point layout from the current iteration is fixed. Hence, the new time point in the next iteration will have *N-1* feasible positions.

On the contrary, if the new objective function value is higher than the best of the last loop (i.e. there is no improvement by adding a new time point), the algorithm stops and takes as optimal value and optimal time points layout the one obtained in the last loop.

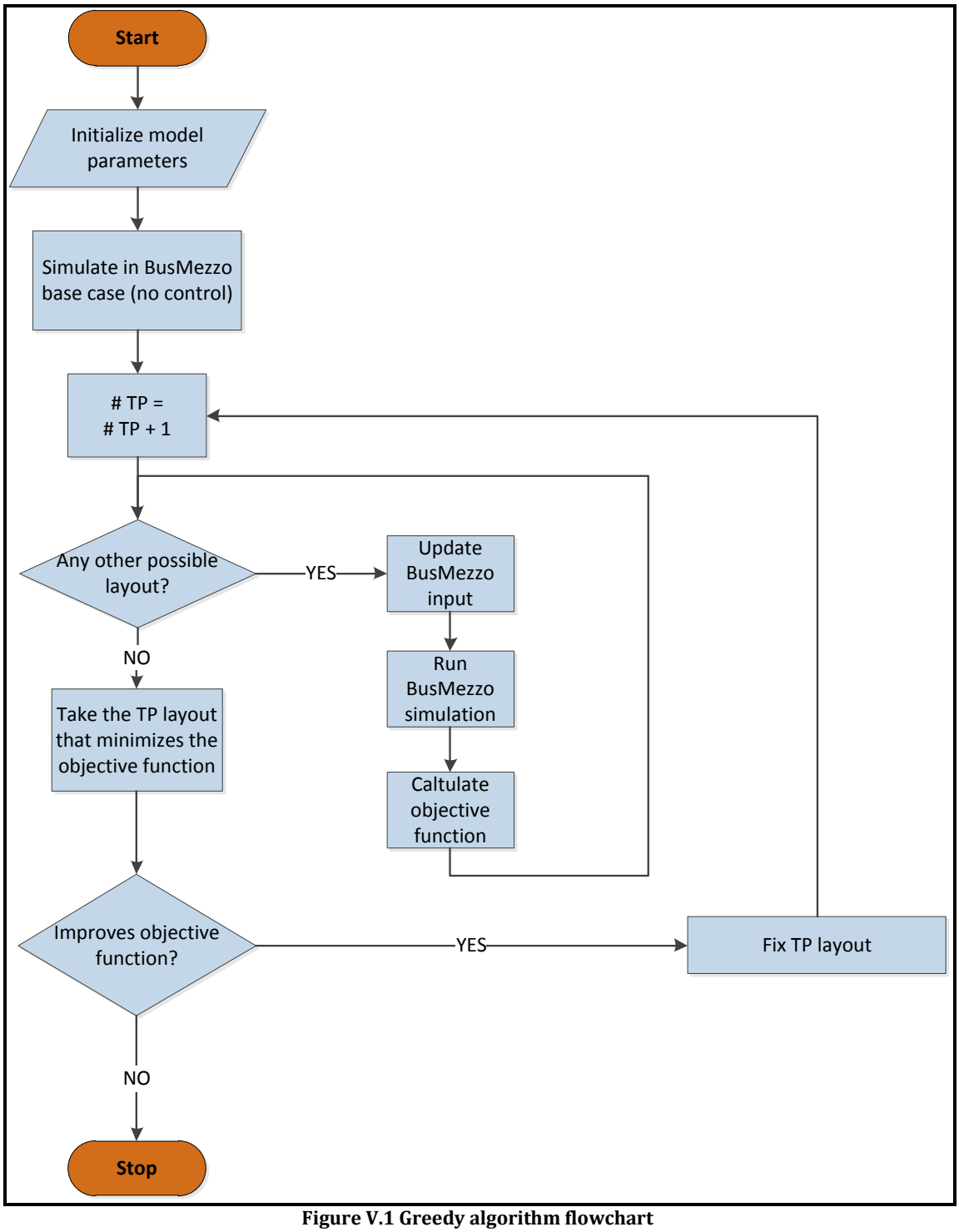

Figure V.1 Greedy algorithm flowchart

V.2.2 Genetic algorithm

A more elaborated approach to optimize the holding problem that takes into account the stochastic character of the data by using a metaheuristic approach is using a genetic algorithm. The function to optimize will be also $f_1(\bar{x})$. A flowchart of the algorithm running scheme is showed in Figure V.2.

When the algorithm starts, the parameters of the genetic algorithm have to be defined. These parameters include: number of individuals, initial population, scaling function, selection function, crossover and mutation functions and stopping criteria. In this study defining a maximum number of generations will be the stopping criterion.

An initial population is created from a predefined creation function. In the current algorithm a function that created population with a specified probability for every stop to be a time point was selected.

The initial population, as well as all other individuals that are created along the generations is evaluated with the same internal process of the previous algorithm: an internal subprocess that changes the input for BusMezzo, runs the application and extracts the output data for every case within the loop.

After the population members are evaluated the individuals are bred to create offspring, using the typical techniques of GA (fitness scaling, selection, crossover and mutation in this order). The offspring becomes the new population and the loop is repeated again until the chosen number of generations is reached.

The individuals in every generation will converge progressively to a space of solutions where the optimal solution should be found. However, because of the stochastic character of the data, it cannot be assured that the best value of the objective function will be found in the last generation. The best individuals will survive through generations because of the elitism technique but their objective function value will vary because they are evaluated again in every generation.

V.2.3 Multi-objective genetic algorithm

In this approach to the problem the two objective functions $f_1(\bar{x})$ and $f_2(\bar{x})$ will be minimized, generating a Pareto front as explained in III.3.5. The multiobjective optimization used is based on the GA; therefore, the scheme of the algorithm is almost the same as in the GA (Figure V.2). The few differences are the fact that two objective functions are evaluated per individual instead of only one and the introduction of the Pareto fraction function and distance function, both with the aim of guaranteeing diversity in the individuals and avoiding premature convergence.

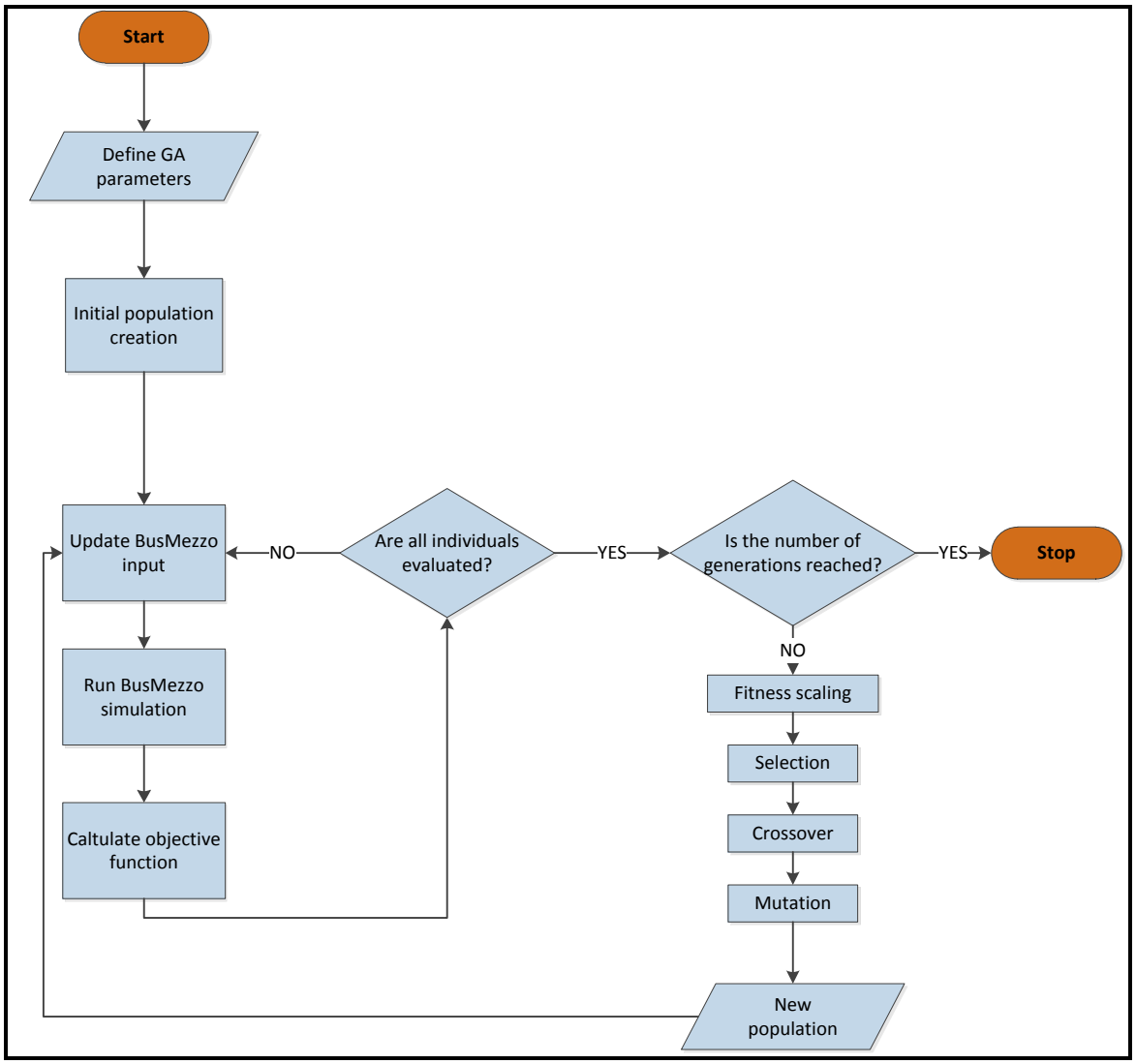

Figure V.2 Genetic algorithm flowchart

VI Results and discussion

The results obtained from the use of the three implemented algorithms are discussed and explained in this section of the thesis. All the algorithms return the value of the defined objective function as an output (the values of both objective functions in the case of the multi-objective optimization). The way these algorithms are implemented, not exclusively the total value of the objective function in the different scenarios is calculated, but also the components of the function, which represents more interesting data to analyze when discussing the results. Obviously, another desired output is the list of stops where a holding strategy will be applied (the holding points): it is a matter of study in this work to analyze their position in the line and the total number of them.

VI.1 Greedy algorithm results

The first results to assess will be the ones given by the complete run of the algorithm without stopping criteria, to fully reproduce the behaviour of the algorithm and contrast the output with the latter solution when the stopping criteria is considered.

Figure VI.1 shows the value of the objective function using the greedy algorithm, when no stopping criterion is used. As the algorithm works, in every iteration one more holding point is added in the position where the objective function is lower among the feasible positions. It can be seen how from the moment the first time point is added, the objective function value decreases almost an 11%. For less than ten time points, the value of the objective function remains low and similar to the value introducing the first time point. However, from the 11th time point on, the value increases more rapidly until the introduction of the last time point. Around the 20th time point approximately the objective function reaches the value of the base case with no holding control. The case with control in all stops represents a 14,7% increase comparing with the no holding strategy.

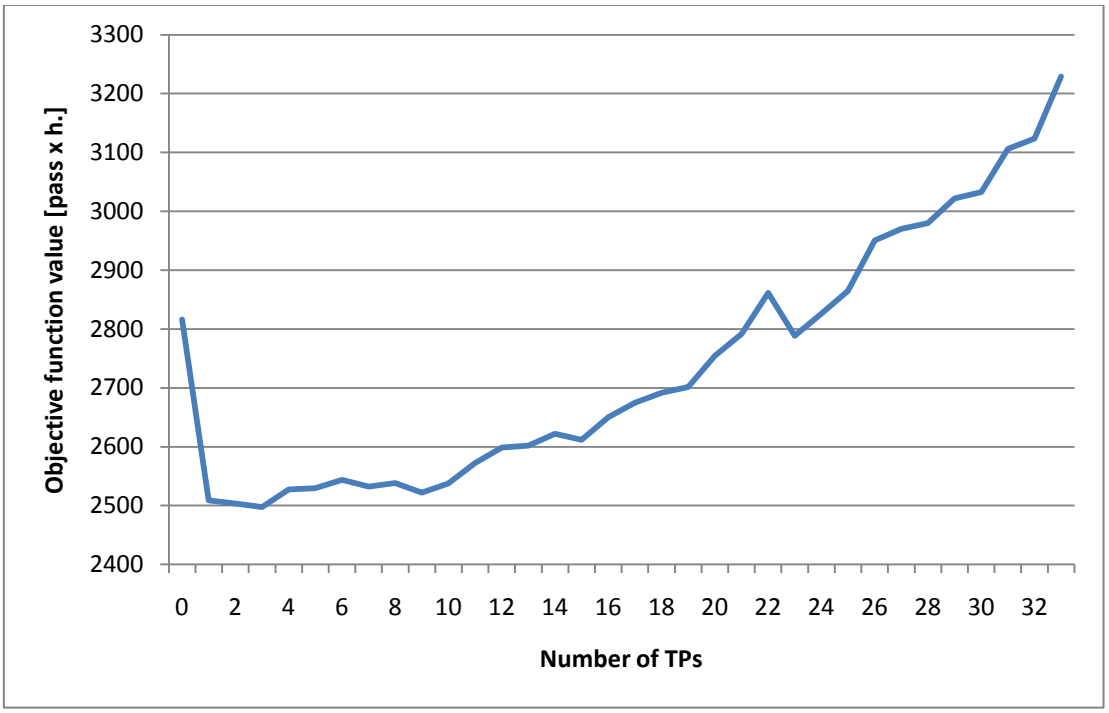

Figure VI.1 Objective function value for the complete run of the greedy algorithm

In order to thoroughly understand the behaviour of the function, its four components need to be studied separately. In figure VI.2 a representation of in-vehicle riding time and dwell time shows the very similar trend those two variables have. If the trend lines are observed, it can be noticed that the minimal values for riding time and waiting time are obtained for the cases of 2 and 3 time points, respectively. The values for these time components for the case with no time points and holding in all stops are very similar. Comparing the best and worst values for each function, a maximum improvement of 7,7% can be achieved in riding time and an improvement of a 12,6% in dwell time. Comparing the best values with the no control case, it exists an improvement of a 6,8% in riding time and a 7,6% in the dwell time. It is remarkable that the total passenger experienced dwell time is on average as much as an 85% of the riding time.

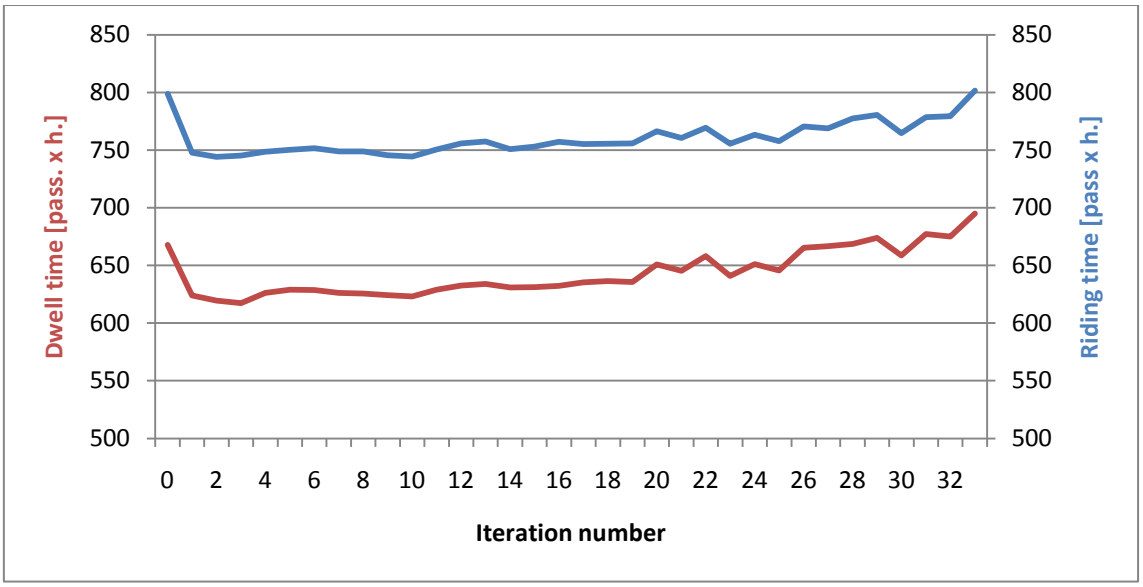

Figure VI.2 Dwell and riding time values for the complete run of the greedy algorithm

The other two components of the objective function are shown in figure VI.3, without the effect of their respective weights. As expected, holding time increases with the number of time points, although the value is quite stable from introducing one to ten holding points. From the tenth time point the increasing trend becomes clearer. On the contrary, the range of waiting time values is very limited since the moment the first time point is introduced to the case of all stops as holding points. The maximum improvement that can be obtained in the waiting time is for ten time points, and represents an improvement of the 23,4% in comparison with the no time points case.

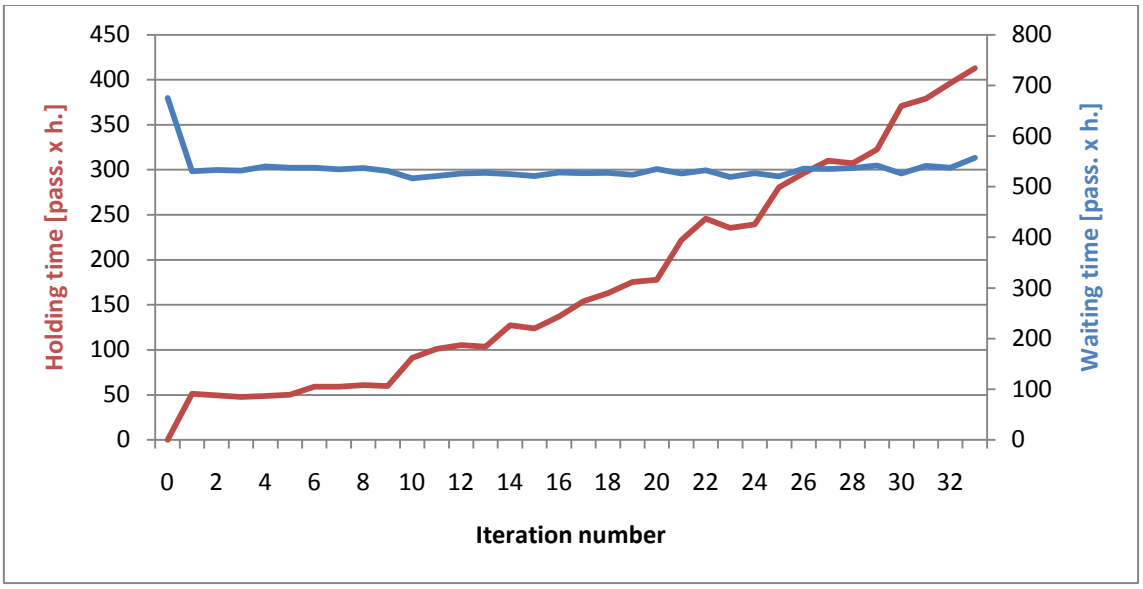

Figure VI.3 Holding and waiting time values for the complete run of the greedy algorithm

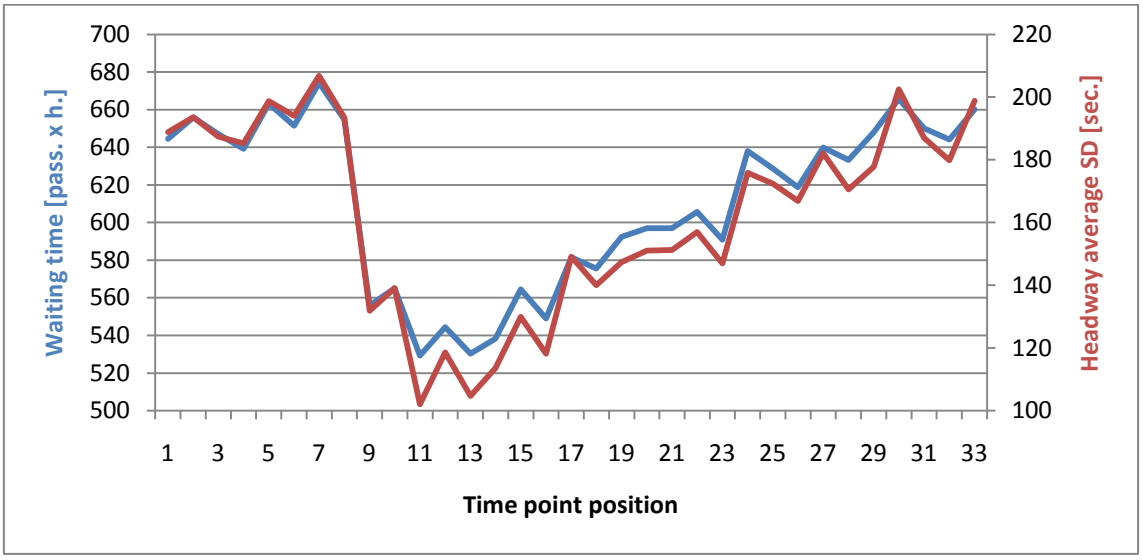

Figure VI.4 Total passenger waiting time vs. average SD of the headway for the 1st iteration

It is interesting to have a look at the first iteration results (choosing the most optimal position for the first holding point) in Figure VI.4 . If we look at the results of the average standard deviation of the headway and the passenger waiting time we realize that their trends are almost equal. In BusMezzo the two components are calculated separately; the simulator registers all the events sequentially in time and waiting time is calculated from

disaggregate data. The similar behavior of the functions can be explained with the following relationship:

$$E(WT) = \frac{H}{2}(1 + CV_h^2) \tag{6.1}$$

Where E(WT) is the expectation of waiting time and CV_h is the coefficient of variation of the headway. The coefficient of variation of the headway is used to reflect bus line reliability. The coefficient is defined as:

$$CV_h = \frac{SD_h}{\mu_h} \tag{6.2}$$

Hence, looking at the two functions behavior it can be concluded that the simulator captures the relationship between the two variables.

To evaluate the best location of the time points that can be obtained with the algorithm, Figure VI.5 shows the location of the chosen time point per iteration. To clarify the plot, in the first iteration the 13th stop becomes a time point and in the second iteration the 1st stop is added as a time point. This process is repeated until all 33 stops have become time points. Analyzing the locations along the iterations, it is remarkable the fact that in the first 10 iterations, the time points are allocated in the first half of the line, basically between stops 1-15. The fact that stops at the end of the line are also selected (31 and 33) is not as much because their improvement in the objective function, but mostly because since these stops are at the end of the line and not many passengers board, introducing these stops as time points does not affect much the value of the objective function.

It seems clear then, that using the greedy technique the most favorable stops to be time points are the ones located within the first half of the line and the ones located in the second half are not introduced as time points until the last iterations of the algorithm.

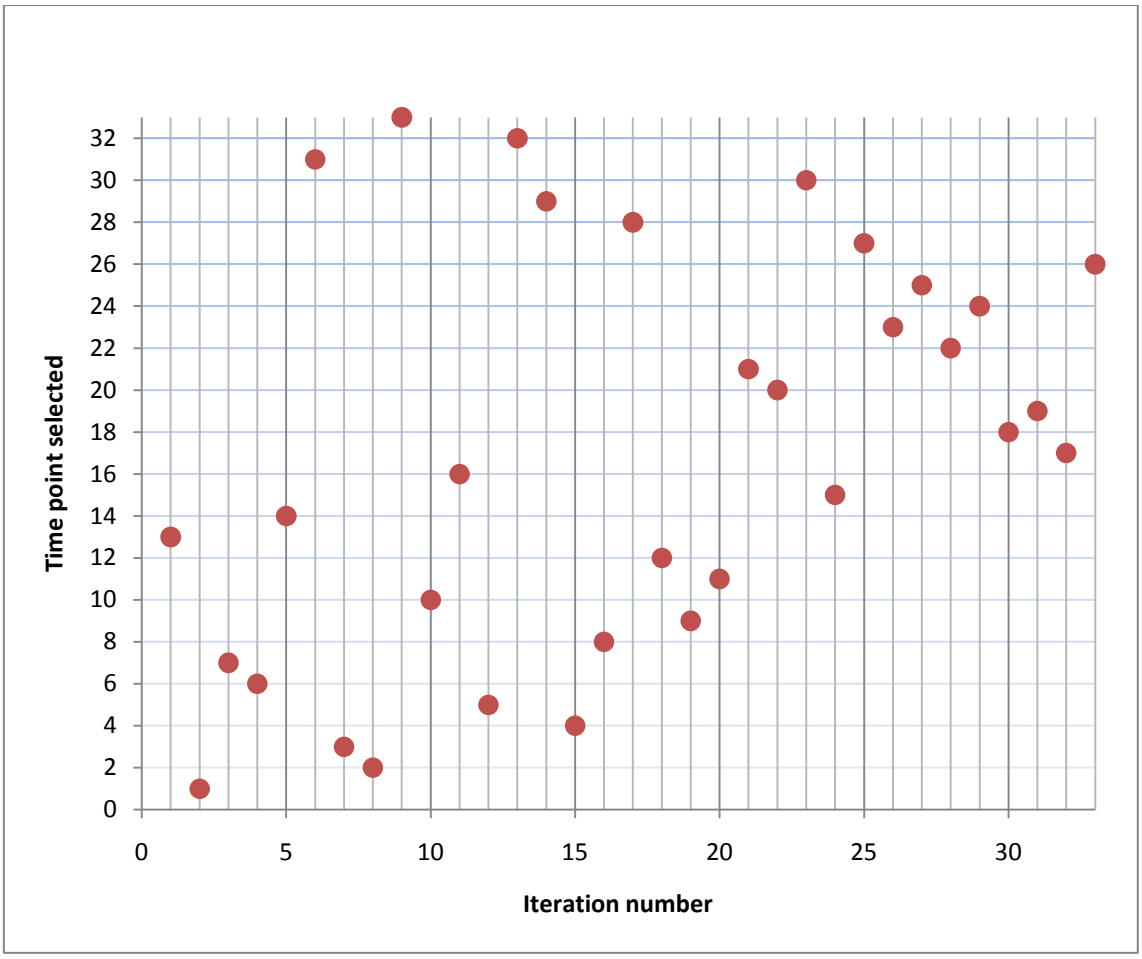

Figure VI.5 Best location of TPs per iteration in the greedy complete run

The use of more time points means more control which can be understood lower coefficients of variation. Figure VI.6 represents the evolution of this coefficient with the number of iterations. It is especially important the improvement that can be achieved when introducing the first time point; a reduction of almost a 50%. From that moment, the general trend is to lower the coefficient with the increase of time points as expected.

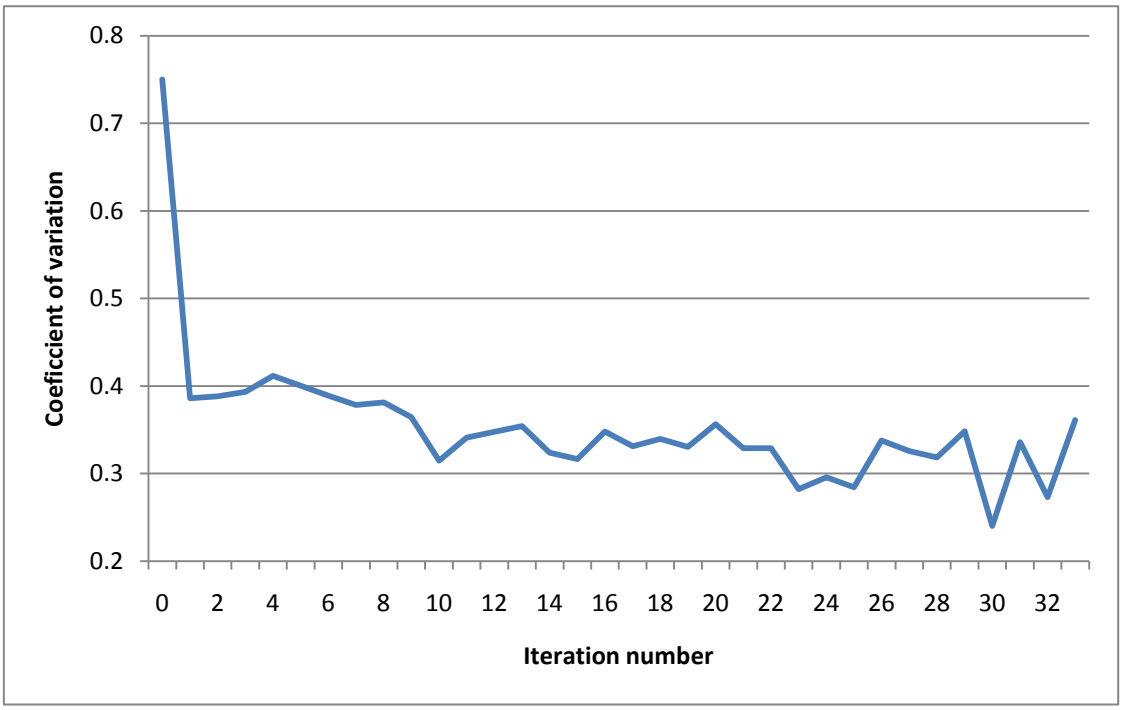

Figure VI.6 Evolution of the coefficient of variation of the headway for the complete run

- **Stopping criteria**

When the stopping criterion is introduced in the algorithm, it stops when the objective function cannot be improved by adding one more time point. In the case of the greedy algorithm run, this point is after the third iteration. Three time points have been set until that moment: 13th, 1st and 7th, in this order. It happens that the value of the objective function at that point corresponds to the minimum value of the objective function that the greedy algorithm can achieve along all iterations. Therefore, it can be concluded that the stopping criterion is successful. With the chosen combination of time points the value of the objective function is improved by a 11,3% comparing to the case of no time points. Although the chosen layout by the algorithm are the three stops (1, 7 and 13), there is no more than a 0,5% variation between choosing only the 13th stop or all three as time points. A penalty function could be added, which penalizes the introduction of more time points, because of the extra operation complexity that means having more holding points.

VI.2 Genetic algorithm results

The genetic algorithm implemented was run with the following parameters:

- An randomly generated initial population with a probability for each stop to be time point of 0,15.
- A population of 20 individuals per generation.
- A proportional scaling function
- Roulette wheel selection
- Elitism of two individuals per generation
- Uniform crossover with probability of 0,8.
- Flipping mutation with probability of 0,2.
- Generational updates for replacement
- A stopping criteria based on the number of generations. In this case, 25 generations plus the initial population.

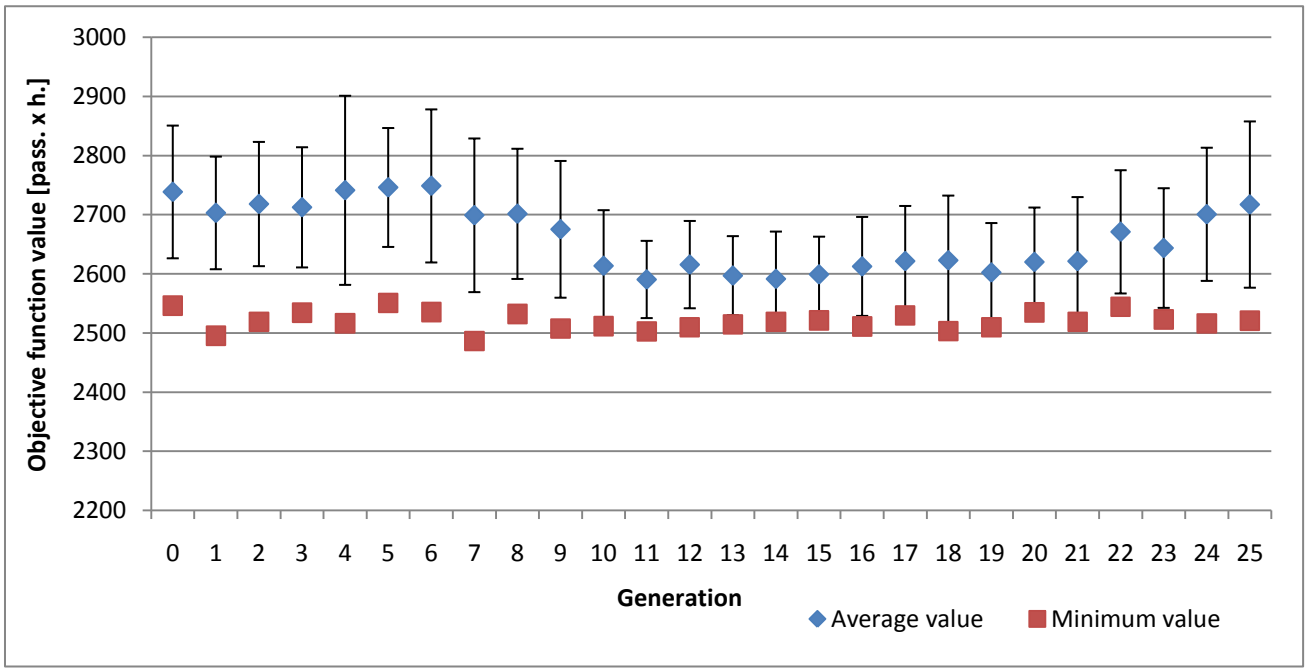

Figure VI.7 Average, SD and minimum objective function value for generation

Generating a random initial population with a probability of 0,15 will generate on average almost five holding points per population; an initial first approximation that exceeds the number of optimal time points according to the greedy algorithm. According to figure VI.7, very good individuals are created even in the initial population, according to the minimum value of the objective function for one of the individuals. With the progress of the algorithm, the average value of the objective function decreases until the 15th generation approximately. However, it can be noticed how from that point the average increases again, and so does the standard deviation. The reason for this behavior is the algorithm itself, which after some generations where crossover played a major part in the creation of new individuals falls into a minimum with little variability among the individuals. From that point, instead of coming to a standstill, mutation starts playing a major role, creating new genotypes which perform well and the algorithm continues exploring in those new directions.

Studying the components of the objective function, Figure VI.8 shows the evolution of the average waiting time per generation and its standard deviation. Although there is not much decrease in the average value it can be observed that it exists a trend to decrease slightly until the 11th generation, almost a 4% comparing that value with the one obtained from the initial population.

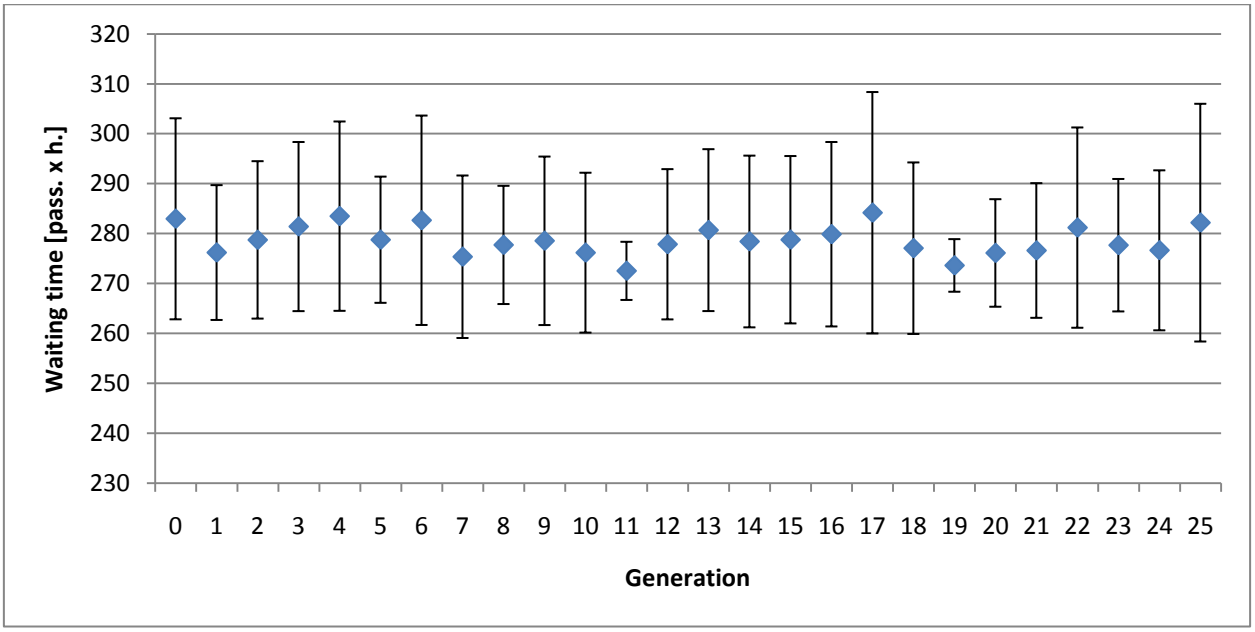

Figure VI.8 Average and standard deviation of the total passenger waiting time per generation

Focusing in the total passenger holding time (Figure VI.9), the decreasing trend of the average value is more marked than in the previous case, until the 15th generation approximately, where the algorithm starts exploring new solution spaces with mutation. This more marked decrease is up to a 19%, comparing the value of the initial population and the 14th one. Accordingly, the standard deviation decreases until the same point and increases again after then. The trend is consistent with the fact that the improvement in the objective function is mainly due to improvement in the holding time component, rather than waiting time.

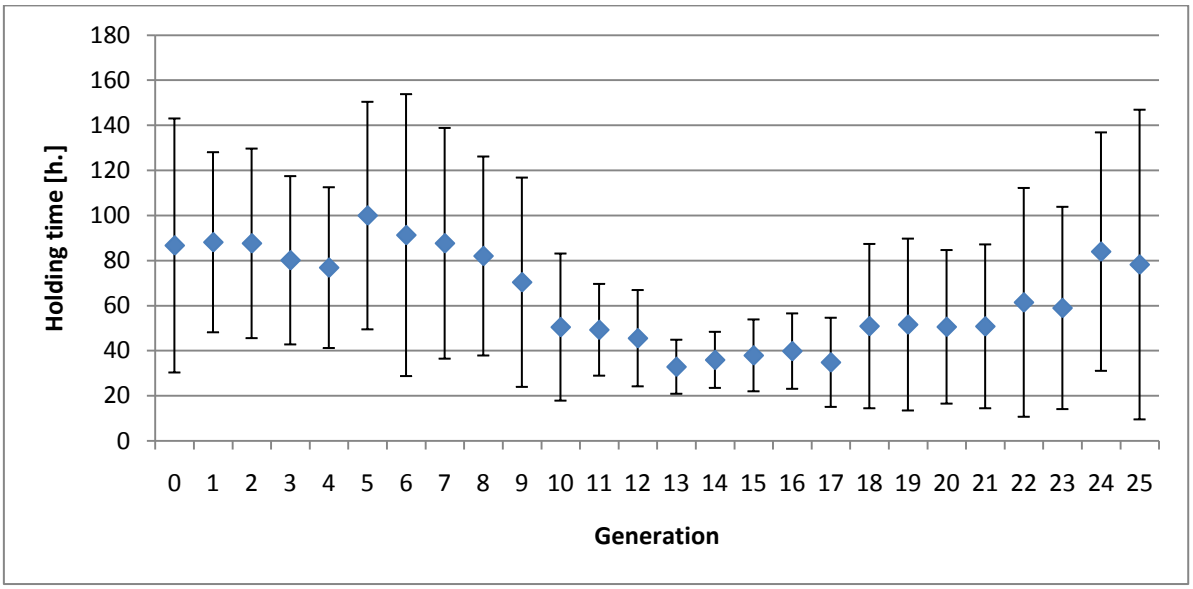

Figure VI.9 Average and standard deviation of the total passenger holding time per generation

When analyzing the results it might arise the question of which variable is more important when choosing a stop as a holding point: the total number of holding points or the location of the points.

Figure VI.10 represents the frequency of the time points according to their location in different generations (initial, 5th, 10th and 15th). It can be noticed how there are no favorite locations to set the time points in the initial population, and how progressively some locations are more prefered by the algorithm (e.g. 6th, 7th, 13th, 15th), having less variability with the advance of generations.

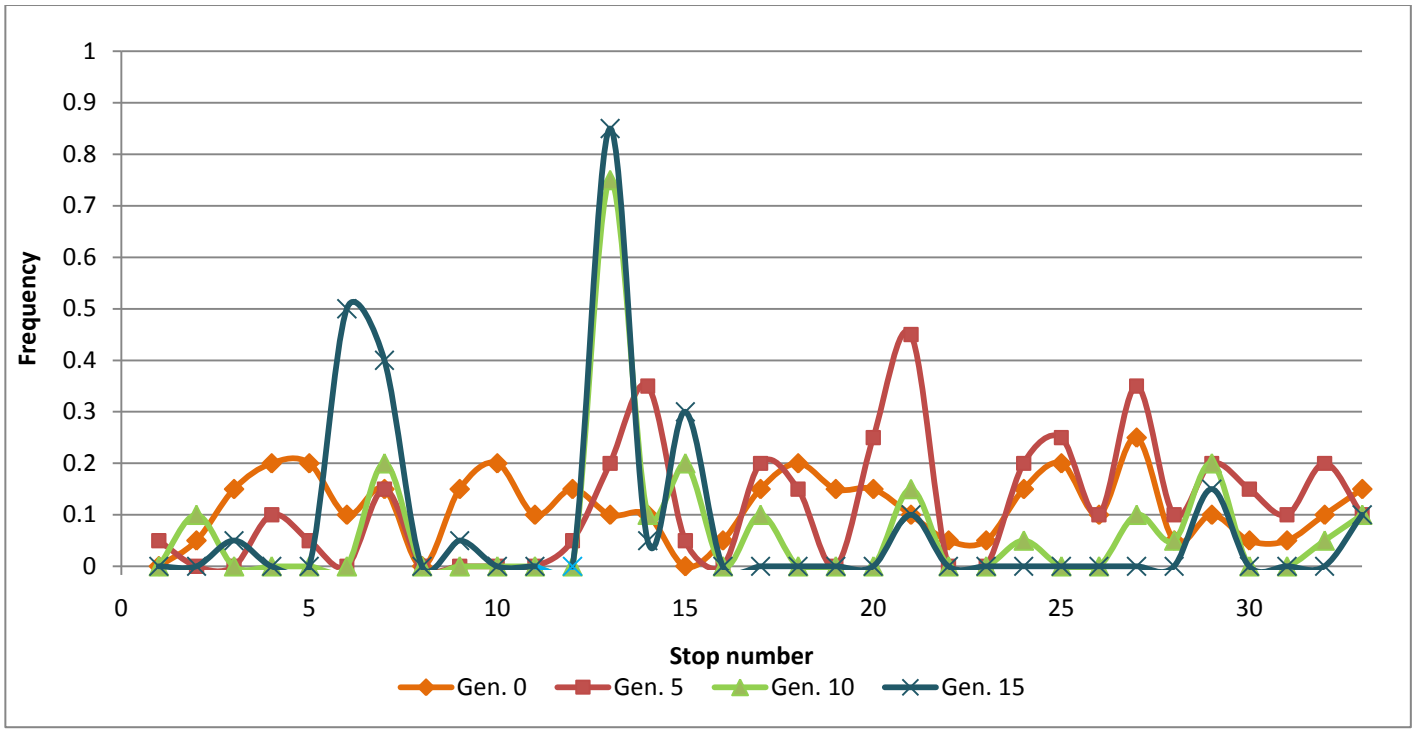

Figure VI.10 Frequency of time points in 0, 5, 10 and 15th generations

In table VI.1 is given explicitly the layout of the ten best individuals evaluated within all generations. Certainly, there are time point positions in common among all solutions, but the total number of time points is not coinciding. Most layouts include a holding point around de 6th-7th position, another a little before the middle of the line (positions 13th, 14th and 15th) and some cases another in the 21st. It can be noticed that this pattern is very similar to the current location (10-17-24) but shifted.

To study the influence of the total number of time points in the value of the objective function, the average number of points per generation is represented in Figure VI.11. Assigning a probability to create a time point in the first generation of 0,15 the expected number of time points in that generation is around five. However, in the current run the initial average number was 3,5 time points. This number decreases along the generations until the 13th one, where the value is less than two time points on average. From that point, as explained before, the algorithm searches for new solutions and the number of time points increases again.

Ranking position	Time points							Number of TP
1			13					1
2			13				21	2
3	6		13					2
4		7	13					2
5		7	13		15			3
6			13	14				2
7				14				1
8	6	7	13					3
9	6	7	13		15			4
10	6			14			21	3

Table VI.1 Rank of the 10 best GA individuals within all generations

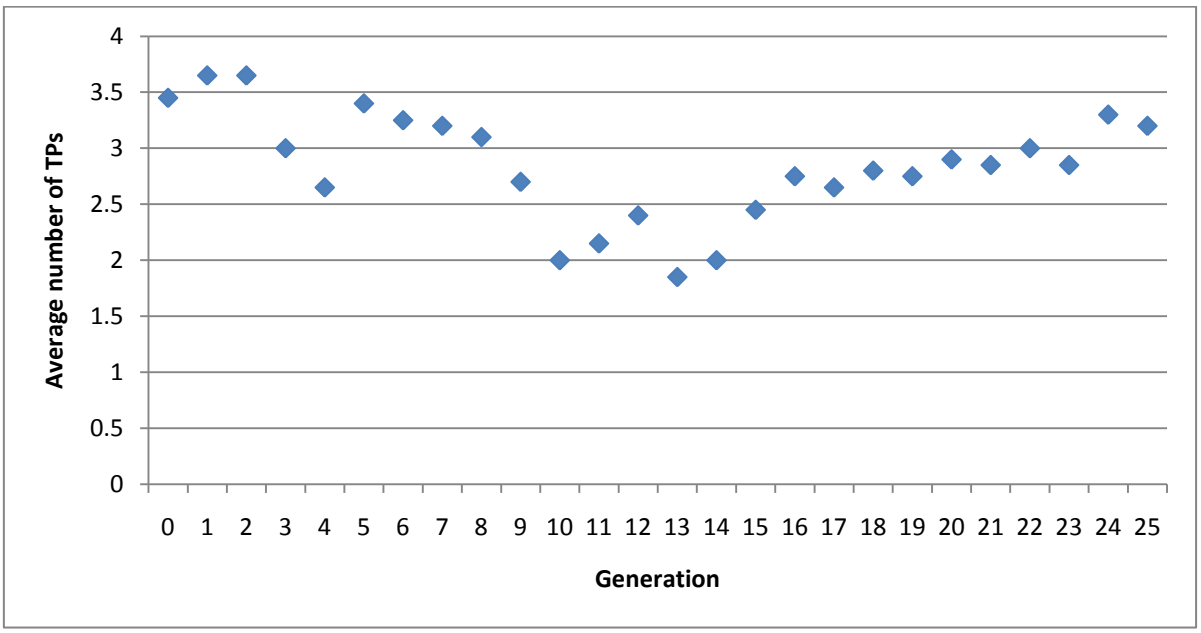

Figure VI.11 Evolution of the individuals' average number of TP per generation

Taking all individuals created by the algorithm along all generations, and analyzing their number of time points and their performance can be useful to judge the influence of the number of holding points. In Figure VI.12, four histograms corresponding to all the individuals with a total of one, two, three and four time points are plotted. Despite the different number of time points, the trend is similar in all cases, slightly shifted to the right. It is clear that there is higher variability of the objective function within a given number of time points than variability between different number of time points, which implies that location is more important than the number of time points.

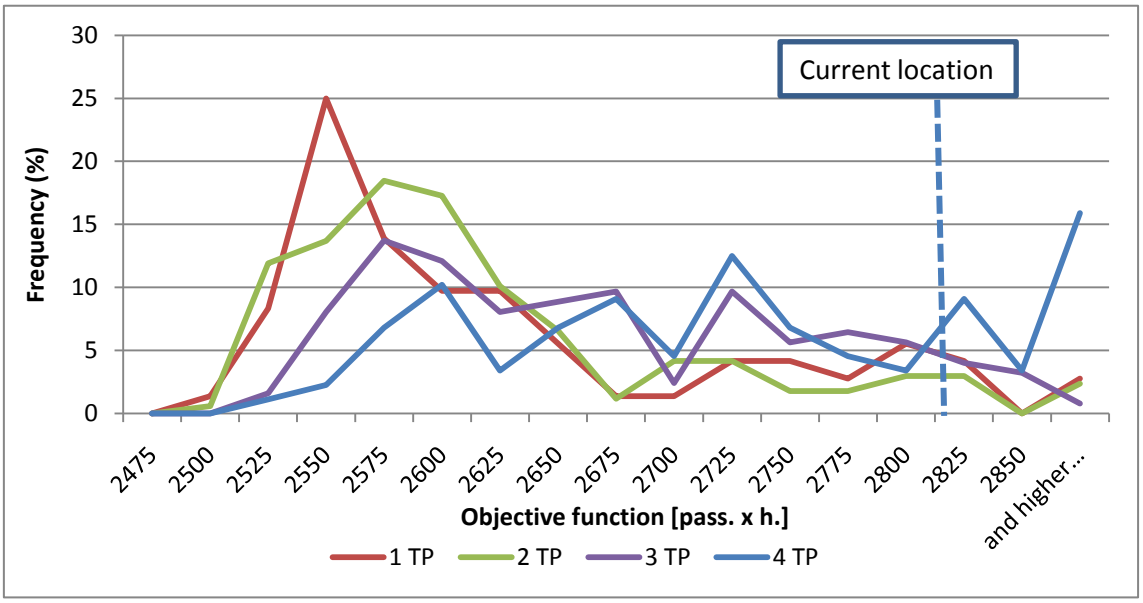

Figure VI.12 Histograms of the objective function value for different number of TP

VI.3 Multiobjective GA results

With the use of the multiobjective GA optimization the aim is to find the Pareto front of the two functions. Figure VI.13 contains a representation of all individuals from generations 0, 3, 6, 9, 12 and 15. The algorithm pursuits to create new individuals that are closer to the Pareto front in every generation. In the current run of the algorithm only two points are in the Pareto front, and both correspond to the same time point layout: both solutions come from different evaluations for the individual with a single time point in the 13th stop. Therefore, it can be concluded that there is no trade-off between the two objective functions, but a direct relationship that allows improving both values simultaneously. As a result, optimizing the function from the passenger perspective, also optimizes the function from the operator point of view, and vice versa.

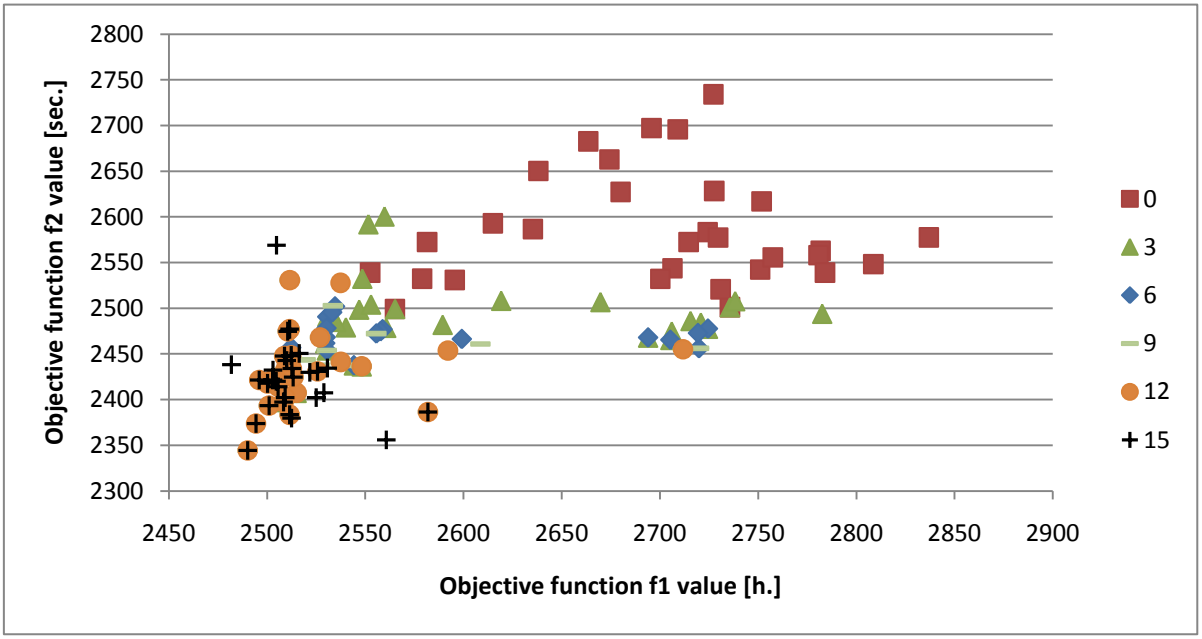

Figure VI.13 Individuals obtained by the multiobjective GA for different generations

VI.4 Comparison of the solutions

In this section, the results obtained with the different algorithms are compared with the current situation and with the case with no control. The Figure VI.14 shows the values of the four different time components for those cases. The greedy solution has three time points (1,7 and 13) and the genetic solution only one time point (13). It is noticeable the higher value of holding time of the current situation compared with the rest of the cases. However, the passenger waiting time is very similar for the three cases where holding is applied, independently of the time points.

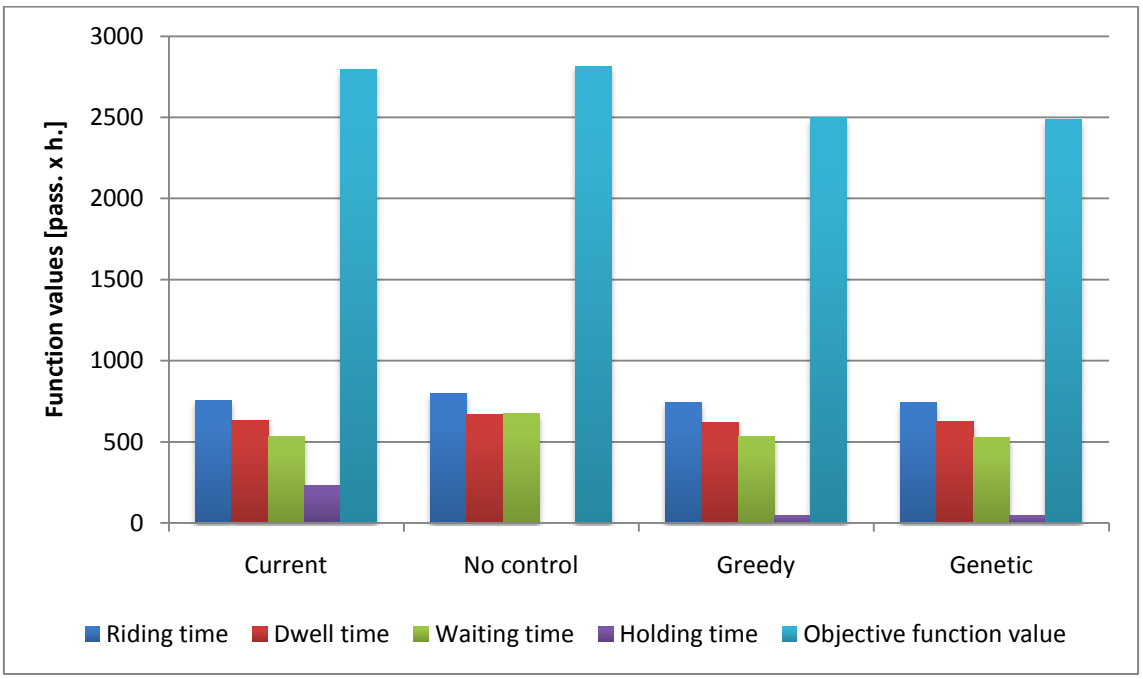

Figure VI.14 Time components for the best found solutions

Looking at the value of the objective function, the no control value is approximately the same as the value in the current situation. The layouts of the greedy and genetic algorithm produce also almost the same objective function value. The difference between the two first cases and the solution obtained differs by a 11%, a considerable value of improvement.

It is possible to calculate the objective function value for all possible combinations of two holding points, a total of 528 possibilities, and observe where the values obtained for the different algorithms lie. Figure VI.15 represents a histogram with the frequencies for all possible combinations of one and two time points. It can be seen that the no control value

lies in the right part of the histogram, by the 75th percentile of the values for two time points, approximately. As mentioned before, the current time point location objective value is very close to the no control value; therefore it can be considered that approximately a 75% two-point combinations are better than the current location. The histogram illustrates how important the location of the time points is, for example, showing how some solutions are worse than the no-control case or that the current location of the time points is worse than some random layouts.

Comparing the algorithms efficiency in terms of computational time, which depends mostly in the number of simulations required, the greedy algorithm with the stopping criterion reached the solution in the third iteration. Hence, the algorithm stopped after evaluating the possible layouts for the fourth iteration, evaluating in total 127 cases.

On the other hand, we considered that the genetic algorithm reached the best solutions in the fifteenth generation, with 20 evaluations per generation. This means a total of 320 individuals evaluated.

Because most of the computational time is due to the time needed to make the number of replications needed for the simulation, we can establish a direct relationship between the number of assessments and total computational time. Considering this, the time needed to reach the best solution of the genetic is 2,5 times the time required for the greedy. The total time of execution for a run (with 50 replications) was 25 seconds on a PC Intel Core 2 Duo, 2,13GHz, 4GB RAM running windows 7.

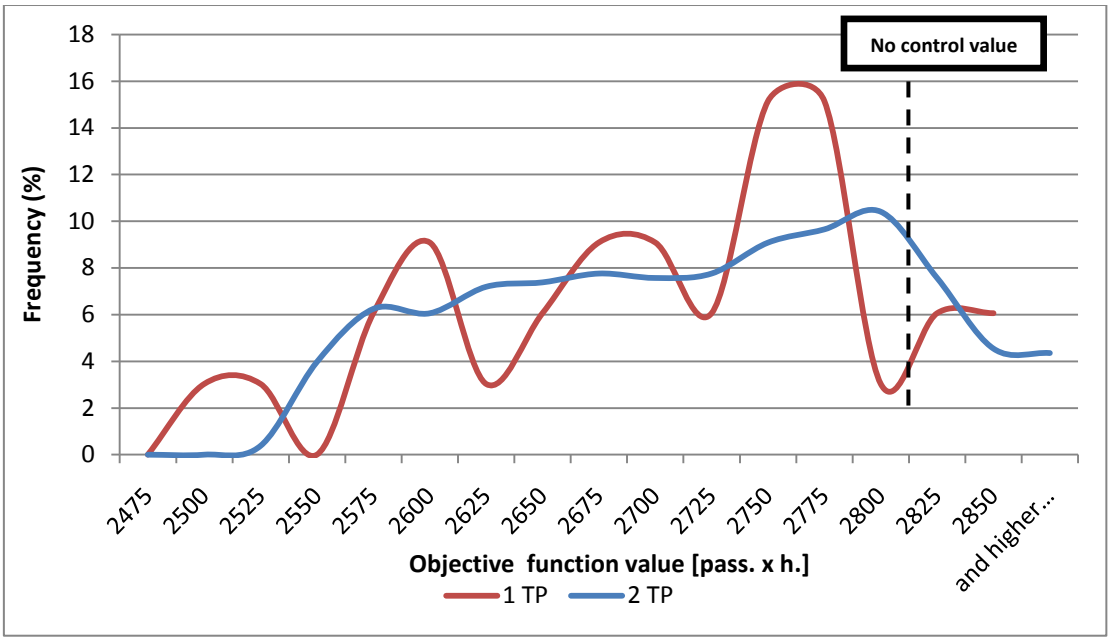

Figure VI.15 Histogram for all 1 TP (33) and 2 TP (528) combinations

In the last table (Table VI.2), the values for the time components of the objective function, the value of the objective function itself and the 90th percentile of the total travel time are given explicitly. The cases included are the current situation, the no control case, the best solutions given by the greedy (1st, 7th and 13th), genetic (13th) and another solution that corresponds to one of the best individuals of the genetic algorithm (6th, 14th and 21st stops as time points). The output of this last layout is solid with the rest, since is neither best than the best solutions found by the algorithms, nor worse than the current or no control situation. However, as one of the best solutions of the genetic, the results are closer to the best solutions than to the other cases.

The value of the last column corresponds to the total standing time per passenger, a crowding measure to assess how do different holding strategies and time point locations affect the total time on average a passenger is standing in the bus. Is calculated the following way:

$$TST = \frac{RT \cdot max(0, L-c) + DT \cdot \max(0, TP-c)}{Total\ boading\ passengers} \quad (6.3)$$

where RT and DT are riding and dwell times respectively, c the capacity of the vehicle, L passenger load and TP through passengers.

Looking at the total standing time it can be observed how regularity affects crowding and how less regular lines represent more crowded buses (the no control case has the higher value, for example).

	***RT* (pass. x h.)**	***DT* (pass. x h.)**	***WT* (pass. x h.)**	***HT* (pass. x h.)**	***Objective function* (pass. x h.)**	***90th perc. TTT* (min.)**	***TST/pass.* (sec.)**
Current	753.86	634.41	531.14	231.27	2797.45	39.2	95
No control	798.77	667.85	674.92	0.00	2816.46	34.5	103
Greedy	745.10	617.23	531.99	47.61	2497.72	34.6	89
Genetic	740.20	622.98	525.75	47.93	2486.58	34.7	88
6-14-21	747.20	624.23	526.84	59.70	2514.67	35.6	94

Table VI.2 Numerical results

VII Conclusions

VII.1 Summary of the main results

This thesis presents several optimization algorithms to optimize the number and location of time points in a bus line, given a headway-based holding strategy. To limit the scope of the problem, only one the holding problem in one direction is studied, although both directions are simulated and the trip chaining between the out- and ingoing directions is considered. The optimization is based in the output given by a mesoscopic transit simulator, BusMezzo, and the line modeled is a real-world one; more concretely the bus blue line number 1 of the city of Stockholm.

The optimization algorithms used are a greedy one, a genetic algorithm and a multiobjective genetic algorithm. With the first two, a function that represents the total travelling (riding, dwell and holding) and waiting times experienced by the passengers is optimized, taking into account a weight factors for the waiting and holding times, which are perceived as having more disutility or discomfort for the passenger. The multiobjective genetic optimization is used to optimize that function, which takes mainly into account the performance of the service from the travelers' perspective, and another function that optimizes the operation from the operator perspective. This latter function considers the 90th percentile of the total travel time for each bus trip, a measure that can be used for the operator to plan the fleet and trip chaining for the line to ensure certain levels of reliability.

The developed greedy algorithm achieved an improvement of the objective function higher than an 11% with the optimal solution having three time points. However, the difference between adding only the first time point and the solution with the three holding points is not significant. The introduction of more holding points means more complicated management of the line, for this reason; despite of the result of the greedy algorithm, it might be better to introduce only the first holding point obtained by the algorithm.

The optimization using the greedy algorithm leads to a best solution with only one time point. However, looking at the best individuals, it is possible to identify some dominant positions that appear more frequently among the individuals along the generations, and

the difference between the best individual and those alternative good solutions is less than 1,5%. Between those individuals there is a trend to locate mostly all time points in the first half of the line, almost evenly spread. The layouts resembles the current time point location slightly shifted.

When comparing the solutions found and the rules of thumb used by agencies nowadays it can be observed in the load profile that after the 13th stop there is a sequence of stops with important boarding numbers. This is consistent with a common rule of thumb. The other rule of considering the first stop as a time point is already implemented by default in BusMezzo with the dispatching from the origin terminal subject to schedule control and vehicle availability.

It arises the question then, about which factor is more decisive: the number of time points used or the location of the time points. To answer these questions, the best solutions given by the genetic algorithm have been analyzed to finally determine that among the best solutions brought by the algorithm there is some variability in the number of holding points between one and four time points, but not as much variability in the positions these time points are located.

VII.2 Future research recommendations

Some different research directions might come out from this thesis. From the point of view of the optimization algorithms, a more accurate study could be conducted about the two algorithms used: different greedy algorithms with different choosing criteria or a sensitivity analysis about the genetic algorithm parameters and operators could lead to more accurate solutions in less number of iterations (or generations), exploring a wider parameter search space. Because of the many parameters, operators and techniques applicable to the genetic algorithm, many possible combinations of these variables are possible, all of them with different efficiency. An analysis in depth of them, according to the problem conditions is also recommended.

It might be interesting to analyze the relationship between line characteristics such as load profiles, bus stop distances or headway variability and the results of the algorithms. A sensitivity analysis about how these parameters affect the best solution given, in terms of

number and location of time points, could allow to reformulating or reinforcing the rules of thumb used by public transport agencies nowadays and to enable to generalize the rules for determining their layouts.

As mentioned in the literature review, different authors have used different objective functions, choosing the parameters depending on the approach studied in their works. The variables and weights used in this thesis could be changed to study how much does the optimal solution vary with them. It might be advisable to introduce new components to the objective function, lik e a penalty for increased number of time points because of operating complexity, or a variable to take into account passenger comfort, for example.

This work took into account only one direction of the line, supposing no holding in the other direction but the problem of optimizing the number and location of time points in both directions should be studied to analyze how the solution changes because of the interaction of considering both ways.

References

Abkowitz, M., Engelstein, I., (1984). "*Methods for maintaining transit service regularity*". Journal of the Transportation Research Board 961, 1-8

Abkowitz, M., Eiger, A. and Engelstein, I. (1986). "*Optimal Control of Headway Variation on Transit Routes*", Journal of Advanced Transportation, Vol.20, No.1, pp. 73-88

Abkowitz, M., Slavin, H., Waksman, R., Englisher, L. and Wilson, N.H.M., (1978). "*Transit Service Reliability*", Technical Report UMTA-MA-06-0049-78-1, U.S. Transportation Systems Center, Cambridge, MA.

Barnett, A. (1974). "*On controlling randomness in transit operations*". Transportation Science 8(2) 102-16

Ben-Akiva, M. (1996). "*Development of a deployable real-time dynamic traffic assignment system, Task D*". Interim report: analytical developments for DTA system, MIT ITS Program, Cambridge (MA).

Ben-Akiva M., Lerman S., (1985). *"Discrete choice analysis*". The MIT Press, Cambridge Massachusetts.

Burghout, W. (2004a). "*Hybrid microscopic-mesoscopic traffic simulation"*. Doctoral thesis, Stockholm, Sweden.

Burghout, W. (2004b). "*A note on the number of replication runs in stochastic traffic simulation mode*". Center of Traffic Research working paper.

Burghout, W., Koustopoulos, H.N., Andreasson, I. (2006). "*A Discrete-event Mesoscopic Traffic Simulation Model for Hybrid Traffic Simulation*". Proceedings of the IEEE Intelligent Transportation Systems Conference, Toronto.

Cats, O., Burghout, W., Toledo, T., Koustopoulos, H.N. (2010a). "*Evaluation of real-time holding strategies for improved bus service reliability*". Submitted ti IEEE.

Cats, O., Toledo, T., Burghout, W., Koustopoulos, H.N. (2010b). "*Mesoscopic modeling of bus public transportation*". Proceeding and CD of the 89th Transportation Research Board Annual Meeting, Washington DC.

Ceder Avishai (2007). *"Public Transit Planning and Operation: Theory, Modeling and Practice*". Public transit planning and operation, Civil and Environmental Faculty, Transportation Research Institute, Technion-Israel Institute of Technology, Haifa, Israel.

Cortés, C.E., Saez, D., Milla, F., Núñez, A., Riquelme, M. (2009). "*Hybrid predictive control for real-time optimization of public transport systems' operations based on evolutionary multi-objective optimization*". Transportation Reseacrh Part C 18 (2010) 757-769.

Daganzo, C.F. (2009). "*A headway-based approach to eliminate bus bunching: Systematic analysis and comparisons*". Transportation Research Part B, 43(10):913-921.

Delgado, F., Muñoz, J.C., Giesen, R. and Cipriano, A. (2009). "*Real-Time Controlo f Buses in a Transit Corridor Based on Vehicle Holding and Boarding Limits*". Journal of Transportation Research Board No. 2090 pp.59-67.

Dessouky, M., Hall, R., Zhang, L., Singh, A. (2003). "*Real-time control of buses for schedule coordination at terminal*". Transportation Research Part A 37. 145-164.

Dorigo, M. (1992). "Optimization, Learning and Natural Algorithms". PhD Thesis. Politecnico di Milano, Milan.

Eberlein, X.J. (1995). "*Real Time Control Strategies in Transit Operations: Models and Analysis*". PhD Thesis. Massachusetts Institute of Technology, Cambridge, MA.

Eberlein, X.J., Wilson, N.H.M. and Bernstein D. (2001). "*The holding problem with real-time information available*". Transportation Science, 35(1), 1-18.

Fu,L., Yang, X. (2002). "*Design and implementation of bus-holding control strategies with real-time information*". Transportation Research Record 1791, 6-12.

Gawron, C. (1998). "*Simulation-Based Traffic Assignment; Computing User Equilibria in Large Street Networks*". PhD Thesis, University of Cologne, Cologne.

Glover F. (1989). "*Tabu-Search Part I"*. ORSA, Journal on Computing Vol 1, pp 190-206.

Goldberg, D. E. & Smith, R. E. (1987). "*Nonstationary Function Optimization using Genetic Algorithms with Diploidy and Dominance*". In J.J Grefenstette, editor, Proceedings of the Second International Conference on Genetic Algorithms, 59–68. Lawrence Erlbaum Associates.

Holland, J.H. (1975). "*Adaptation in Natural and Artificial Systems"*. University of Michigan Press.

Hickman, M.D. (2001). "*An analytic stochastic model for the transit vehicle holding problem*". Transportation Science 35(3), 215-237

Jayakrishnan, R., H.S. Mahmassani and T.Y. Hu (1994). "*An Evaluation Tool for Advanced Traffic Information and Management Systems in Urban Networks*". Transportation Research C 2C (3): 129-147.

Kemp, M A. 1973. "*Some evidence of transit demand elasticities*". Transportation 2 (1):25-51.

Kirkpatrick, S.; Gelatt Jr., C.D.; Vecchi, M.P. (1983). "*Optimization by Simulated Annealing*". Science 220 (4598)

Kittelson & Associates, KFH Group, Parsons Brinkkerhoff Quade & Douglass, Hunter-Zawarski, K. (2003). "*Transit Capacity and Quality of Service Manual*". Transit Cooperative Research Program, Report 100, Washington DC.

Koustopoulos, H.N. and Wang, Z. (2007). "*Simulation of urban rail operations*". Transportation Research Record, 2006, 84-91

Leonard, D.R., P. Power and N.B. Taylor (1989). "*CONTRAM: structure of the model*", Transportation Research Laboratory, Crowthorn.

Lesley, L.J.S. (1975). "The *role of the timetable in maintaining bus service reliability*". Proceedings of the Symposium on Operating Public Transport p. 36, University of Newcastle-Upon-Tyne, UK.

Liu, G., Wirasinghe, S.C. (2001). "*A simulation model of reliable schedule design for a fixed transit route*". Journal of Advanced Transportation 35 (2), 145-174.

Malzoumi, E., Currie, G., Rose, G., Ceder, A., Moridpour, S. (2010). "*A Practical Tool to Optimize Transit Schedule Timing Points and Slack Times: An Ant Algorithm Application*". Submitted for Presentation at 90th Transportation Research Board Annual Meeting.

Morgan, D.J. (2002). "*A Microscopic Simulation Laboratoy for Advanced Public Transport System Evaluation*". Master Thesis, Massachusetts Institute of Technology, MA.

Newell, G.F. (1974). "*Control of pairing of vehicles on a public transportation route, two vehicles, one control point*". Transportation Science 8(3) 248-264

Newell, G.F., Potts, R.B. (1964). "*Maintaining a bus schedule*". Proc., Second Conference Australian Road Research Board, Melbourne, volume 2, pp. 388-393.

Osuna, E.E., G.F. Newell (1972). "*Control Strategies for an idealized public transportation system*". Transportation Science 6 52-72.

Pilachowski, J.M. (2009). *"An Approach to Reducing Bus Bunching*". UCTC Dissertation No. 165, University of California, Berkeley.

Rossetti, M.D. and Turitto, T. (1998). "*Comparing static and dynamic threshold based control strategies*". Transportation Research Part A, 32(8), 607-620.

Turnquist, M.A. and S.W. Blume (1980). "*Evaluating potential effectiveness of headway control strategies for transit systems*". Transportation Research Record 746, 25-29, Washington DC.

Wirasinghe, S.C., Liu, G. (1995). "*Determination of the number and locations of time points in transit schedule design- Case of a single run*". Annals of Operations Research 60(1995) 161-191.

Zhao, J., Dessouky, M., Bukkapatnam, S. (2006). "*Optimal slack time for schedule-based transit operations*". Transportation Science 40(4), 529-539

- **Bibliography**

Introduction to genetic algorithms. Siranandam, S.N. and Deepa, S.N. Springer 2008

Introduction to algorithms. Thomas H. Cormenm Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. MIT Press 2001

Genetic Algorithm Toolbox. Andrew Chipperfield, Peter Fleming, Hartmut Pohlheim and Carlos Fonseca. Version 1.2, Department of Automatic Control and Systems Engineering, University of Sheffield.