Università degli Studi di Cagliari
Delft University of Technology

Dipartimento di Ingegneria Elettrica ed Elettronica
Delft Center for Systems and Control

Corso di Laurea Specialistica in Ingegneria Elettronica

Andrea Pinna

Modeling, Calibration and Validation of Highway Traffic Networks

Relatori: Prof. Ing. Alessandro Giua
Prof. Ing. Bart De Schutter
Dr. Ing. Andreas Hegyi
Ing. Monique van den Berg

Anno Accademico 2006/07

Sommario

Un buon modello di una rete stradale è uno strumento fondamentale per il progetto di misure di controllo del flusso del traffico, per la pianificazione di lavori stradali, e per la modifica della struttura della suddetta rete di traffico stradale. Il modello, infatti, è quindi utilizzato per la simulazione del traffico in tali condizioni sconosciute, affinché gli effetti sul traffico delle modifiche possano essere valutati.

Affinché il modello possa essere utilizzato per questi scopi, è necessario eseguire la calibrazione dei suoi parametri e quindi validare i risultati ottenuti. Il processo di calibrazione determina i valori dei parametri del modello che permettono di ottenere il miglior adattamento tra le uscite del modello (dopo una simulazione del traffico) e le misure reali. Il processo di validazione valuta se i parametri ottenuti nel processo di calibrazione forniscono una buona corrispondenza tra le misure reali e le uscite delle simulazioni eseguite con scenari differenti da quello utilizzato per la calibrazione.

Questo lavoro di tesi consiste nel costruire tramite Paramics (un software per la simulazione su scala microscopica del traffico) il modello di una porzione rettilinea di 10 chilometri dell'autostrada A12 situata nei Paesi Bassi: su tale rete verrà simulato il comportamento di ciascun veicolo. Dopo la costruzione del modello, i suoi parametri verranno calibrati tramite diverse tecniche di ottimizzazione (programmazione non lineare, algoritmi genetici, metodi diretti). Infine, il modello ottenuto verrà validato, affinché esso possa, in seguito, essere utilizzato come banco di prova per il progetto di tecniche di controllo del traffico.

Abstract

A good model of a traffic network is a fundamental tool for the design of control measures of the traffic flow, for the planning of road works, and for the modification of the structure of the given traffic network. The model, in fact, is then used for the simulation of the traffic behavior in these unknown conditions, in order to evaluate their effects on the traffic flow.

In order for the model to be used for these purposes, it is necessary to perform the calibration of its parameters and to validate the result. The process of calibration determines the values of parameters of the model that yield the best approximation or fit between the model outputs and the actual measurements (e.g., provided by the detector loops). The process of validation evaluates whether the parameters obtained in the calibration process provide a good correspondence between the actual measurements and the outputs of the simulation when performed for different scenarios.

This MSc project deals with constructing the Paramics model, which is a microscopic traffic simulation model, in which individual vehicles are modeled, of a 10 km stretch of the A12 highway in The Netherlands. Then, the model parameters will be calibrated by means of different optimization techniques (such as constrained nonlinear programming, genetic algorithm, and pattern search). At last, the resulting model will be validated, so that the model can afterwards be used as a test bench for control traffic design.

Contents

Chapter 1

Introduction

The Dutch traffic network is, already since some years, now inadequate to sustain the current traffic demand. Moreover, it will be totally incapable of satisfying the future traffic demand, which is predicted to be growing strongly. For this reason, a partial reorganization of the above-mentioned network is in operation, so that it will be capable of satisfying the traffic demand for decades to come [16, 17].

The scheduling of the works is assisted by simulation software; with these information media it is possible to model the part of the network of interest and to simulate the behavior of the vehicles. In this way it is possible to predict the effects of the changes that are going to be performed. Of course, so that the prediction will be reliable, the model of the network should be as much as possible similar to the real network.

At this point, the procedures of model calibration and validation enter the scene. The first one enables to determine the model parameters so that a strong correspondence will exist between actual traffic measures and the same ones calculated by the traffic simulator. The second one allows to eventually state that the parameters found in the calibration procedure could be really usable for the model simulation.

This project deals with the modeling of a 10 kilometres stretch of the A12 highway, in the proximity of the city of Utrecht, and, in particular, with the calibration and the validation of this model. In order to perform the modeling of this network, the software Paramics (developed by Quadstone Limited) has been used, while for the parts regarding the processes of calibration and validation both Paramics and Matlab (developed by The MathWorks) have been employed.

1.1 Overview

This is the content of the next chapters.

Chapter 2 is an introduction to the problem of modeling and control of the traffic on highways. Some measures of control for managing the flow of vehicles are presented, and different typologies of traffic modeling software are illustrated. Chapter 3 explains the ideas of calibration and validation, showing also the mathematical methods (in particular the optimization techniques and the measures of fitness) involved in these two processes. In Chapter 4, the case study network and its Paramics modeling are described in details. Chapter 5 illustrates how the algorithm for the calibration and the validation of the Paramics model of the traffic network has been developed. In Chapter 6 the results of the simulations, executed by means of the above-mentioned algorithm, are presented. Finally, Chapter 7 reports the conclusions on this project and some ideas for related future researches.

Chapter 2

Traffic Flow Modeling and Control

In this chapter the underlying notions about traffic flow modeling and control will be discussed briefly.

2.1 Actual Network Situation

The Dutch network is no more able to satisfy the more and more increasing traffic demand, as it is happening worldwide (Figure 2.1). In the Netherlands, in particular, since 1990 there have been a strong increase of the traffic demand, that is going to grow further in the next years (Figure 2.2). Thus it is necessary to apply some changes, enhancing the capacity of the network, with special care at strategic points (e.g., intersections): crossroads, on-ramps, and off-ramps belong to this category. In fact, in the nearness of these zones the traffic jams, that are

Figure 2.1: Highway congestions in Los Angeles and Sydney.

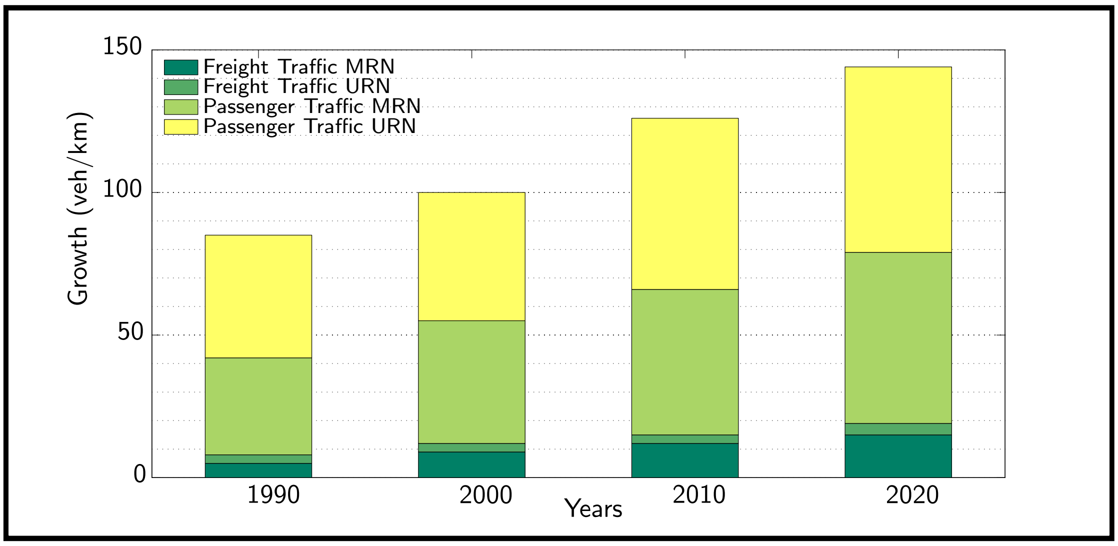


Figure 2.2: Density of vehicles (freight and passenger traffic) on the Dutch main (MRN) and underlying (URN) networks, normalized with year 2000 [11].

obviously an indication of the network deficiency, are more likely to occur. The matter is that, during the progress of the works, it will be necessary to close roads or lanes, or parts of them. The discomfort for drivers is going to be, for a some time, even worse compared to the actual condition.

2.2 Traffic Flow Control

In order to reduce a similar unpleasantness, it is possible to apply some appropriate control methods, based for example on Model Predictive Control [4, 9]. Prior to that, we obviously need to have a mathematical model of the system (that should be calibrated), to know which are the relevant variables, to find a way for measuring them, to transfer these data to a controller, and finally to make the designed control strategy work.

2.3 Traffic Flow Sensors

The main variables [28] that allow the traffic status to be described in the network are, for example: the **traffic flow** (veh/h), which measures the number of vehicles that pass through a road (or a lane); the **mean speed** (km/h) of the vehicles flow; the **traffic density** ($veh/km/lane$), which measures how many vehicles, for every kilometer, take up a lane; the **occupancy**, that is the relative time

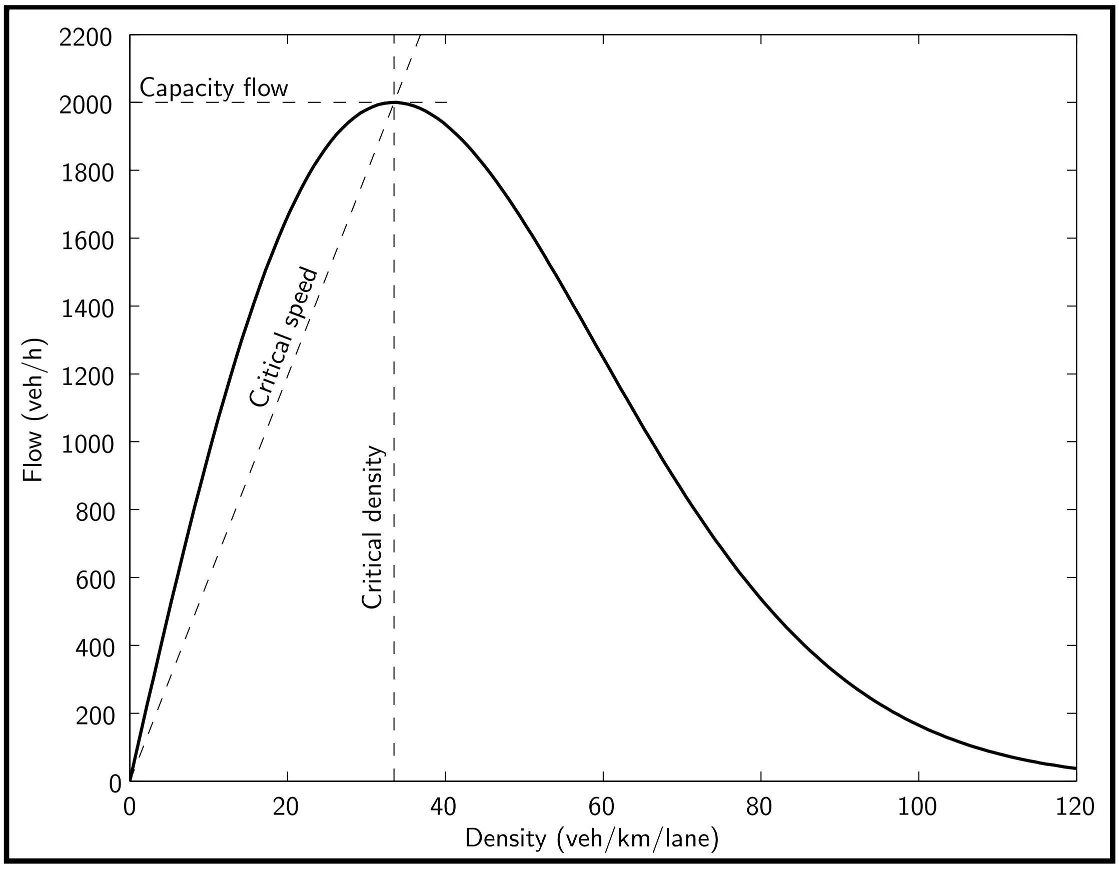
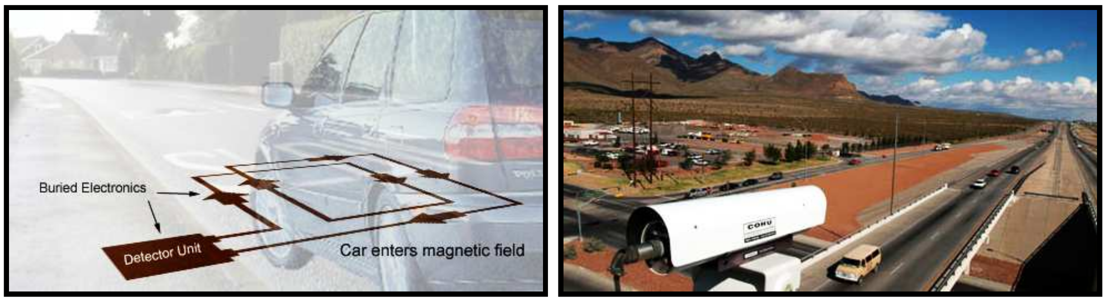


Figure 2.3: An example of the fundamental diagram, that represents the relationship between the flow and the density of the vehicles in a section of a highway.

(in percentages) during which the traffic sensor is detecting a vehicle; the **time headway** (h) and the **distance headway** (km) between vehicles, etc. An example of the relationship between the traffic flow and the traffic density is depicted in Figure 2.3: the plot is known as the fundamental diagram of the highway traffic.

These values could be measured or derived by means of various sensors [1, 2, 13, 29] (Figure 2.4). The one which is mainly used is the **inductive loop**; it is a device, placed just under the pavement, capable to detect the passage of vehicles thanks to a frequency shift. The double loop configuration allows the traffic engineer to obtain more accurate measurements, and in particular it enables the calculation of the vehicles length, and so it is possible to distinguish between a car and a truck. The **traffic cameras** are another especially used device. They are positioned over the motorway (for example on a overpass) and they send their images to a processor which is capable, by means of a recognition algorithm, to count how many



(a) Scheme of an inductive loop. (b) Traffic camera.

Figure 2.4: Some traffic flow sensors.

and which sort of vehicles pass through the highway. The accuracy is larger than the one provided by inductive loops, except in the event of bad weather conditions because the pictures will be corrupted and so they will be hardly analyzable by the recognition algorithm. The cheapest sensor is the **pneumatic tube**: it is laid down directly on the asphalt and recognizes the pressure produced within by the transit of the vehicle. Unfortunately, these devices wear down very quickly, and so they are available only for short-term operations. Their useful characteristic is the installation readiness, and so it is possible to perform various measurements with the same tube in different workspaces and in a brief time.

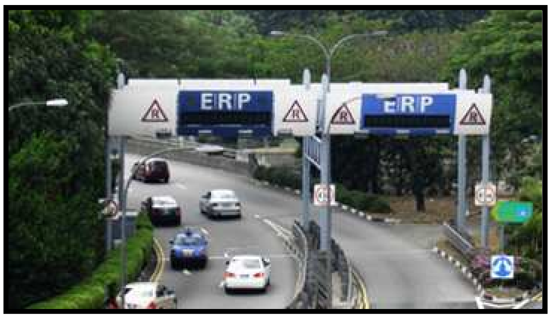
The distributed sensors over the network can be linked with a centralized controller or with more distributed controllers, depending on the network topology and on the sort of control method implemented. However, although nowadays data transmission from sensors to controllers is not liable for errors anymore, the same sensors may not be totally reliable, and so they could send wrong or incomplete data. Therefore it is necessary to take into account this eventuality, and to develop techniques that are able to detect errors and to estimate the missing data.

2.4 Traffic Control Actuators

Control strategies are put in practice by means of one or more techniques (Figure 2.5). Between these, we have the so-called **peak lanes** (which are lanes that are opened or closed depending on the actual traffic demand), **tidal flows** [31] (one or more lanes are opened downstream or upstream depending on the traffic demand), **speed harmonization** (an automatic system that gives advice to drivers about the cruising speed to follow for avoiding the creation of traffic jams), **intelligent speed adaptation systems** [5, 26] (which suggest to drivers the maximum speed to follow depending on traffic state, weather, inter-vehicles distance, etc.),

(a) Tidal flow lane in the Netherlands.

(b) Variable speed limits in Germany.



(c) Electronic road pricing in Singapore.

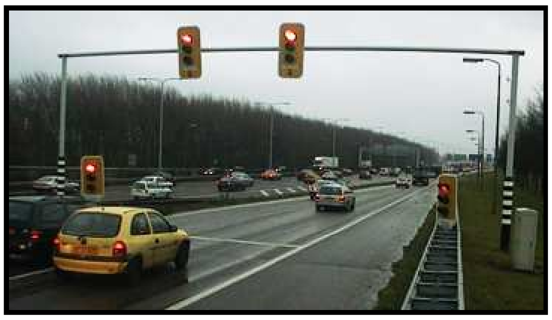

(d) Ramp metering in the Netherlands.

Figure 2.5: Some measures for controlling the traffic flow of vehicles in the highways.

electronic road pricing (the road toll is raised if the transit in a particular road is advised against), **traffic re-routing** (it consists in suggesting an alternative route, if available, in order to distribute more fairly the vehicles on the network), **ramp metering** [33] (where traffic lights limit the number of vehicles entering the highway) and **traffic signals** (used principally in urban networks). The first three control strategies are mostly used in the United States, while the others are applied world-wide.

2.5 Modeling Software

Simulation software is used for predicting the results of a given control strategy. With these software packages it is possible to model the network (even in the smaller details) and to verify the performance of the control action. The benefits of simulation are manifold: it allows the engineer to investigate the reliability and the performance of the control strategy before it will be put in practice, and it is cheaper than a field test, which usually is not even practicable. In fact we are not interested in simulating the traffic behavior in an existing network, because the required data can be more easily gathered via field measurements.

Simulation is therefore used to predict the traffic status in unknown circumstances; for example, it would be useful to find out the degree of discomfort caused by the closing of a lane or by a road accident, to know how much the vehicle flow will improve thanks to a roadway enlargement, or how effective a control strategy is. Simulation is also used as a planning tool, because its results could be helpful during the elaboration of a control strategy. The reliability of a simulator could be quantified depending on its ability to yield results similar to the actual ones. Amongst several traffic simulation software [22], the main ones are, in alphabetical order, *Aimsun*, *Contram*, *Corsim*, *Cube Dynasim*, *Dynasmart*, *Freflo*, *Kronos*, *Linsig*, *Lisa+*, *Netsim*, *Paramics*[1], *Saturn*, *Simtraffic*, *Transims*, *Transyt*, *Vissim*.

2.6 Traffic Models Classification

Commercially there exists a wide variety of software packages capable to model a network and to simulate the flow of vehicles on it. Similar models could be distinguished depending on four different classifications: the physical interpretation, the level of detail, the deterministic or stochastic nature, and the discrete or continuous nature of the model [3, 15, 22].

2.6.1 Physical Interpretation

There exist three types of models [24], noticeable depending on the kind of relationships between system inputs and outputs. The **white box** modeling is characterized by well-known relationships between the inputs and the outputs, as well between the different internal states of the system. This sort of modeling is also called deductive approach. The **black box** modeling owns its name to the fact that the relationships between inputs and outputs are unknown. The engineer has to accomplish the identification of these relationships and has to build an approximate model of the network. The **grey box** modeling is obviously a tradeoff between the two previous models; generally speaking, the form and the structure of the equations is known, but some parameters need to be calibrated in order to obtain a strong fit between the measured outputs and the simulated outputs, on the same inputs.

[1]There are two distinct products which share the name of *Paramics*, and they are separately developed by *SiAS* and *Quadstone*.

2.6.2 Level of Detail

A distinction between models could be carried out also depending on how detailed they are [18]. **Microscopic** models (e.g., *Aimsun*, *Corsim*, *Paramics*, *Vissim*) have a very high level of detail; in fact they consider the characteristic of every single vehicle and their mutual interactions. All that involves therefore a particular burden in computation, but for the current processors it is not a problem calculating such simulations and at the same time displaying as an animation the traffic flow in the network. The latter characteristic is particularly useful because it allows the traffic engineer to have an instantaneous feedback about the control strategy, without the additional burden of necessarily analyzing the numerical outputs of simulation. **Macroscopic** models (e.g., *Saturn*, *Transyt*) are those with the lower level of detail, and so they are used for describing very large networks. These models represent the traffic by means of aggregate variables, such as the flow, the density, and the mean speed, and therefore they do not allow the behavior of every single vehicle to be analyzed in the same way as microscopic models. Finally, the **mesoscopic** (e.g., *Contram*, *Transims*) models turn out to be a tradeoff between microscopic and macroscopic models. Usually they describe the behavior of groups of vehicles and hence they are suitable for larger networks than those simulated with microscopic models.

2.6.3 Deterministic or Stochastic

Deterministic models (e.g., *Dynasmart*, *Freflo*) are characterized by a deterministic input-output relationship. Then, every time a simulation is executed with the same inputs, the outputs will be obviously identical. This will not occur in **stochastic** models (e.g., *Netsim*, *Paramics*, *Vissim*) because they contain at least a stochastic variable which is described by a statistical distribution. Even using the same inputs, the outputs of different simulations will be always mutually different, and so it is necessary to perform a sequence of runs in order to draw valuable conclusions.

2.6.4 Discrete or Continuous

Almost all traffic models are **discrete**, because the time and space independent variables are discretized in order to be managed by computers and, therefore, to be numerically solved by means of simulation. **Continuous** models are not appropriate to be simulated, because they are too much complex to be analitycally solved, in particular if the size of the network is large.

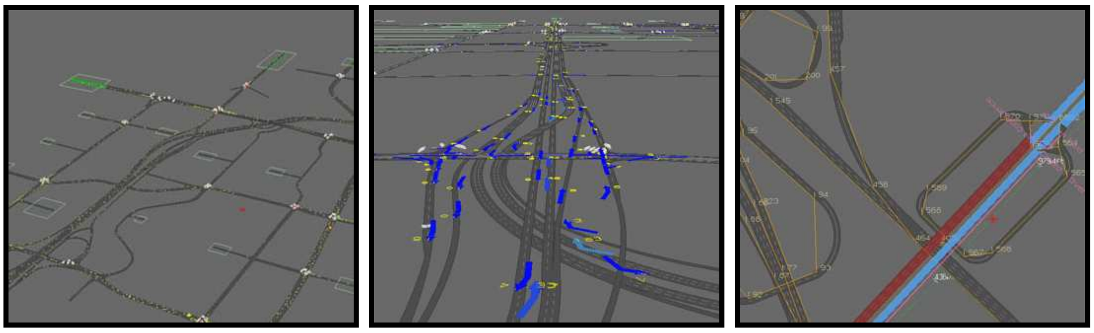


Figure 2.6: Screenshots from Paramics Modeller V6.

2.7 The Paramics Simulation Model

The stretch of the A12 highway will be modeled with the software Paramics [34], developed by the Scottish company Quadstone Limited. Paramics traffic models, according to the previous categorizations, are grey box, microscopic, stochastic and discrete. Paramics is also indicated among the best commercially microscopic simulators [7]; in fact it is characterized by a set of modules, and each of them performs a particular role [36]. The main module is **Modeller** (Figure 2.6), because it deals with the building of the model and with the traffic flow simulation. The module **Processor**, thanks to an innovative graphical interface, allows the model parameters to be easily set and, in particular, to determine how to modify, run after run, the parameters of interest. **Analyser** is the tool that allows the simulation outputs to be graphically displayed and to be statistically analyzed. The module **Programmer** is specialized in the customization of many model characteristics, for example the tuning of the behavior parameters of the vehicles and the replication of actual conditions. Finally, **Monitor** is specific for the calculation of the pollutant emission level caused by the traffic in the study network.

As a mathematical model, Paramics needs certain inputs in order to produce its outputs (Figure 2.7). Generally speaking, these inputs consist of the layout and the description of the network, the (statistical or deterministic) distribution in time of the number of vehicles that enter the network (demand) and their supposed route within the network, the parameters that describe the characteristics of the vehicles (type, dimensions, maximum speed and acceleration, etc.) and the behavior of their drivers (aggressiveness, awareness, etc.), and the technical parameters for the simulation (seed, simulation step, etc.). All these inputs allow Paramics to simulate the traffic flow of the vehicles and, therefore, to obtain the measurements that characterize the traffic flow (mean flow, speed, and density of the vehicles, etc.), minute by minute, through the detector loops placed along the

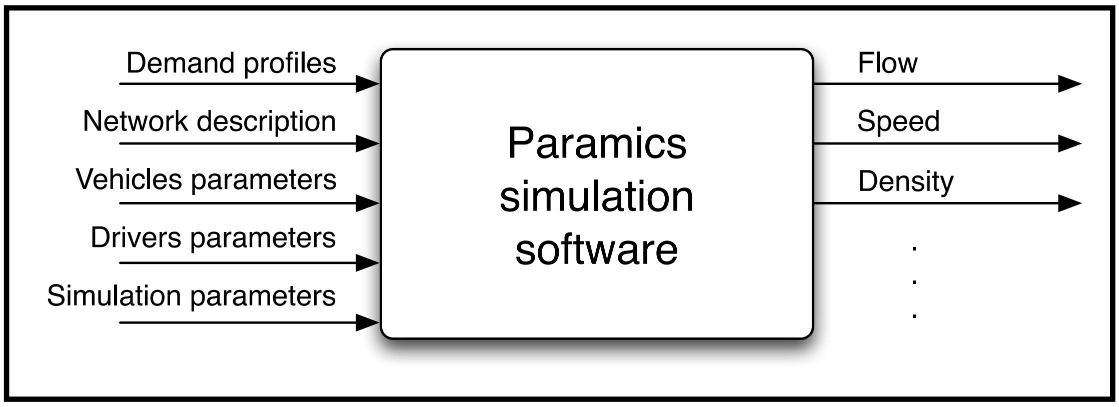


Figure 2.7: Inputs and outputs for the Paramics simulation software.

model network.

2.8 Summary

In this chapter some ideas about the traffic flow control have been introduced. The main variables that describe the traffic in a network are the traffic flow (veh/h), the mean speed (km/h) and the traffic density (veh/km), that could be measured by means of inductive loops, traffic cameras and pneumatic tubes. The traffic control strategies could be applied by means of various methods: peak lanes, speed harmonization, intelligent speed adaptation systems, electronic road pricing, ramp metering. A prediction of the effects yielded by the selected control measures is achievable by means of a traffic simulation software, that could be able to reproduce the real flow of vehicles with good approximation.

Chapter 3

The Processes of Calibration and Validation

Calibration and validation procedures [14, 19, 35, 37, 38] are essential if some guarantee about the model fit with the real network and, therefore, about the reliability of the simulation outputs is needed. In fact, models are only an approximation of what happens in reality (in this case study, models are a mathematical approximation of the networks and of the behavior of the drivers).

A generic procedure for calibrating and validating a model is the following. Initially it is necessary to gather as much field data as possible, concerning the network and its traffic. As the model has been developed, an estimation of its parameters needs to be performed. These parameters have to be corrected by means of calibration. In the end, model validation is executed to check the capacity of the model in correctly predicting the traffic behavior with different sets of inputs, which have not been used in the calibration process. If the validation outcome is positive, it is possible to assert under which conditions the model could be successfully used.

3.1 Calibration

Paramics is capable of modeling the drivers behavior, the criteria that rule the choice of the route, and the origin-destination (OD) matrix. These three models have to be calibrated [10] and validated.

Since the model is a statistical model, the estimation of the number of runs to execute in order to obtain helpful results has to be done. Then the variance of the model has to be known, and so it is necessary to execute a certain number n_t of trial runs to determine this value. When the variance has been obtained, it is time to determine how many simulation runs should be executed; from these simulations can be drawn the data that will enable the model calibration. The number of simulation runs could be estimated by the following formula [8, 23, 39]:

$$n_r \geq \left(t_{\alpha/2} \frac{\overline{\sigma}}{\overline{\mu}\epsilon} \right)^2$$

where $\overline{\mu}$ and $\overline{\sigma}$ are, respectively, the estimate of the real mean value μ and of the real standard deviation σ of the selected output of the simulation (e.g., flow, speed), both obtained by means of n_t trial runs; $t_{\alpha/2}$ is the distribution threshold value for a confidence interval of $100(1-\alpha)\%$ and ϵ is the maximum allowable percentage error of the estimated mean value $\overline{\mu}$.

The driver behavior model includes the car-following and the lane-changing models: both need to be calibrated for every zone of the network. On the other hand, the route-choice model needs to be calibrated for the overall network. Regarding the OD matrix, its estimation (or a simple adjustment, if the source that has provided the OD matrix is reliable) could involve several sub-steps, and it could be done after the calibration of the car-following and the lane-changing models. Usually we are interested in calibrating the traffic network during the peak hours, but in order to achieve a better result it is best to run the simulation over a longer time interval (for example, from an hour before to an hour after the peak hours).

In practice, a model calibration consists in the search for the best values of the model parameters. The goal is to minimize the difference between the observed outputs and the simulated outputs. The objective function could be as follows:

$$z = \sum_{i=1}^{n} |y_i - \theta_i|^u$$

where y_i is the simulated output, θ_i is the observed output (i.e., the queue length, the travel time, the density, or the speed), u is a positive integer (usually $u = 2$), while n is the number of measurements, and i is the time index. Obviously θ_i is function of the model parameters, and so it is more proper to turn this dependence into an explicit one, writing $\theta_i = \theta_i(x_1, x_2, \ldots, x_m)$, being $m \leq m_r$ the number of parameters that needs to be calibrated: m is not necessarily equal to the number m_r of model parameters, because that number could be too high, or some of those parameters could have less sensitivity compared to others and hence they could be dismissed to lighten the computational complexity. The minimization of such functions could be therefore obtained by means of optimization algorithms (as in the case of sequential quadratic programming and genetic algorithms [21, 25]).

3.2 Optimization Techniques for Calibration

In this section the ideas of sequential quadratic programming, genetic algorithm and direct search are presented.

3.2.1 Sequential Quadratic Programming

Sequential quadratic programming (SQP) [6] is an efficient method for solving nonlinear bounded optimization problems with smooth objective function. The most general form of such problems is:

$$\begin{array}{lll} \text{minimize} & z = f(\boldsymbol{x}) & \\ \text{subject to:} & g_i(\boldsymbol{x}) = 0 & i = 1, \dots, m_e \\ & g_i(\boldsymbol{x}) \leq 0 & i = m_e + 1, \dots, m \\ & \boldsymbol{x_l} \leq \boldsymbol{x} \leq \boldsymbol{x_u} & \end{array}$$

where the vector of the model parameters $\boldsymbol{x} = [x_1, x_2, \dots, x_n]^T$, the objective function to be minimized $f(\boldsymbol{x})$, the m equality and inequality functions $g_i(\boldsymbol{x})$, and the lower and the upper bounds of parameters $\boldsymbol{x_l}$ and $\boldsymbol{x_u}$ are shown.

The basic idea of SQP (which is an iterative algorithm, as its name suggests), is the modeling of the nonlinear problem into a quadratic programming sub-problem, for every step k; a new starting point could be obtained from its solution $\boldsymbol{x_k}$, in order to implement a new sub-problem that will yield a new solution $\boldsymbol{x_{k+1}}$, which hopefully is going to be more accurate than the previous one. While iterating the algorithm, a sequence of solutions is generated; this sequence should converge towards an optimum $\boldsymbol{x}^*$.

Two relevant characteristics of this method are that the SQP does not need a feasible starting point (both for the first and for the other iterations). This is a real strong point, because usually in nonlinear problems the task of finding a feasible point may be nearly computationally as hard as the resolution of the optimization problem itself. The second property is the existence of fast and accurate algorithms for the solving of quadratic programming problems, thus making the whole procedure viable.

Given a generic optimization problem, the main idea is to build a quadratic programming sub-problem, which rests upon a quadratic approximation of the Lagrangian function:

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i \cdot g_i(\boldsymbol{x})$$

where λ_i are estimations of the Lagrange multipliers. The constraints of the latter are obtained while linearizing the nonlinear ones. To obtain the sub-problem objective function, it is necessary to calculate the Hessian matrix of the scalar values function $f(\boldsymbol{x})$, the components of which are defined as follows:

$$H_{i,j} = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_i \partial x_j}$$

At every step k, the shape of the sub-problem is:

$$\begin{array}{lll} \text{minimize} & \frac{1}{2}\boldsymbol{d^T H_k d} + \boldsymbol{\nabla f(x_k)^T d} & \\ \text{subject to:} & \boldsymbol{\nabla g_i(x_k)^T d} + g_i(\boldsymbol{x_k}) = 0 & i = 1, \dots, m_e \\ & \boldsymbol{\nabla g_i(x_k)^T d} + g_i(\boldsymbol{x_k}) \leq 0 & i = m_e + 1, \dots, m \end{array}$$

Such problems are solved by any algorithm for quadratic programming; from its solution, a new iterate is calculated:

$$\boldsymbol{x_{k+1}} = \boldsymbol{x_k} + \alpha_k \boldsymbol{d_k}$$

where the parameter α_k is selected in order to produce a sufficient decrease of an appropriate merit function.

At each iteration, the Hessian matrix is usually updated (in the Matlab implementation of the algorithm) according to the BFGS method [27]:

$$\boldsymbol{H_{k+1}} = \boldsymbol{H_k} + \frac{\boldsymbol{q_k q_k^T}}{\boldsymbol{q_k^T s_k}} - \frac{\boldsymbol{H_k^T H_k}}{\boldsymbol{s_k^T H_k s_k}}$$

where:

$$\begin{array}{l} \boldsymbol{s_k} = \boldsymbol{x_{k+1}} - \boldsymbol{x_k} \\ \boldsymbol{q_k} = \boldsymbol{\nabla f(x_{k+1})} + \sum_{i=1}^{n} \lambda_i \cdot \boldsymbol{\nabla g_i(x_{k+1})} - (\boldsymbol{\nabla f(x_k)} + \sum_{i=1}^{n} \lambda_i \cdot \boldsymbol{\nabla g_i(x_k)}) \end{array}$$

3.2.2 Genetic Algorithms

Genetic algorithms [12, 32] are adaptive methods that could be used for solving research and optimization problems, in particular when it is difficult to find a solution with the classical methods. It is a robust method, which could be successful where other techniques fail (e.g., if they find multiple local minima). Genetic algorithms do not guarantee the finding of an optimal solution, but usually they do not have any trouble in finding an acceptable solution in a relatively short time.

The working of a genetic algorithm is the following. Given a population of *individuals*, each one representing a possible solution of the problem and characterized with a *fitness score*, the algorithm selects the best individuals and gives them the chance to procreate. This way, the *offsprings* are going to have, with a high probability, a better genetic equipment, and so forth for the next generations. If the algorithm has been well designed, the population will converge towards the optimal solution of the problem. The genetic features of each individual are equivalent to a parameter set of the model, and so to each gene one of these parameters is linked. The objective function $f(\boldsymbol{x})$ to be minimized is calculated based on the model parameters. The fitness score of each individuals i is compared with that of the other population members: in fact, given the objective functions $f(\boldsymbol{x}^{(i)})$ for each individual and given its mean value for the whole population $f_A(\boldsymbol{x})$, the fitness score of the individual is given by the ratio between these two values. The algorithm selects, in a probabilistic fashion, the best individuals and it performs a cross-over of their genetic dowry in order to create a new population. After cross-over, sometimes a gene mutation could be executed (usually with a low probability). By means of mutation one or more genes of the individuals are modified, in order to obtain further alterations of the genetic features: these changes could be hard to happen performing solely the cross-over. The procedure is iterated until the algorithm finds the optimal solution or until the maximum number of steps is reached.

3.2.3 Direct Search Methods

Direct search methods [20] are unconstrained optimization techniques that do not explicity use derivatives[1]. They are an effective option, and sometimes the only option, for several varieties of difficult optimization problems.

A simple example of direct search method could be found in the so-called algorithm of compass search, that could be exemplified as follows for a minimization problem with two variables. From a starting point (x_0, y_0), move one step respectively to north $(x_0, y_0 + k_0)$, south $(x_0, y_0 - k_0)$, east $(x_0 + k_0, y_0)$, west $(x_0 - k_0, y_0)$, and calculate the objective function in these points. If one of these steps yields a reduction in the function, the point relative to the maximum improvement becomes the new iterate (x_1, y_1). If none of these steps yields improvement, try again with steps half as long $(k_1 = k_0/2)$.

Such methods are easy to describe and easy to implement (as it could be seen with the compass search method), and usually they make rapid progress toward the solution. On the negative side, when the iterates are near a minimizer will only become apparent as the algorithm reduces the length of the trial steps. That

[1]As *fmincon*, for example.

is, the algorithm may quickly approach a minimizer, but it may be slow to detect this fact: this is the price of not explicitly using gradient information, and so, the asymptotic rate of convergence could be slow[2]. Nonetheless, slow asymptotic convergence rates are not necessary an issues, especially in the context of the problems to which direct search methods are the most applicable. A common situation in practice is that one wants improvement rather than full-blown optimality. The user's goal may be only one or two correct digits, either because this is good enough for the application, or else because the values that can be obtained for the function to be optimized are sufficiently inaccurate that seeking higher accuracy from a solution would be pointless. Furthermore, direct search methods have limitations concerning the number of variables: they are best suited for problems with a small number of these (even if they have been successfully used on problems with a few hundred variables).

Direct search methods are especially suitable for simulation-based optimization problems[3]. In such situations, a computer simulation must be run, repeteadly, in order to compute the various quantities needed by the optimization algorithm. Each simulation may in turn involve the execution of several indipendent programs in sequence, such as a geometry generator, a mesh generator, and a partial differential equations solver. Furthermore, the resulting simulation output must then be post-processed to arrive finally at values of the objective and contraints functions. These complications can make the obtaining of the derivatives for gradient-based methods at the very least difficult, even when the underlying objective and constraint functions are smooth (i.e., continuously differentiable). Since the associated functions are not expressed in algebraic or analytical form, the computed results may resemble the plot in Figure 3.1, characterized by low amplitude and high-frequency oscillations. Such oscillations could diminish with successive adaptations, but the computed objective never becomes smooth, even near the minimizer. In this example, derivative estimates with a small finite-difference interval are wildly inaccurate. In general, features such adaptive algorithms, if-then-else logic, stopping tests in iterative schemes inside the simulation, and the inevitable effects of floating point arithmetic are among the culprits that cause smooth problems to appear to be nonsmooth.

[2]Direct search methods are, in a precise sense, asymptotically slower than the steepest descent method.

[3]Simulation-based optimization problems are deemed as the methodology in which complex physical systems are designed, analyzed, and controlled by optimizing the results of computer simulations.

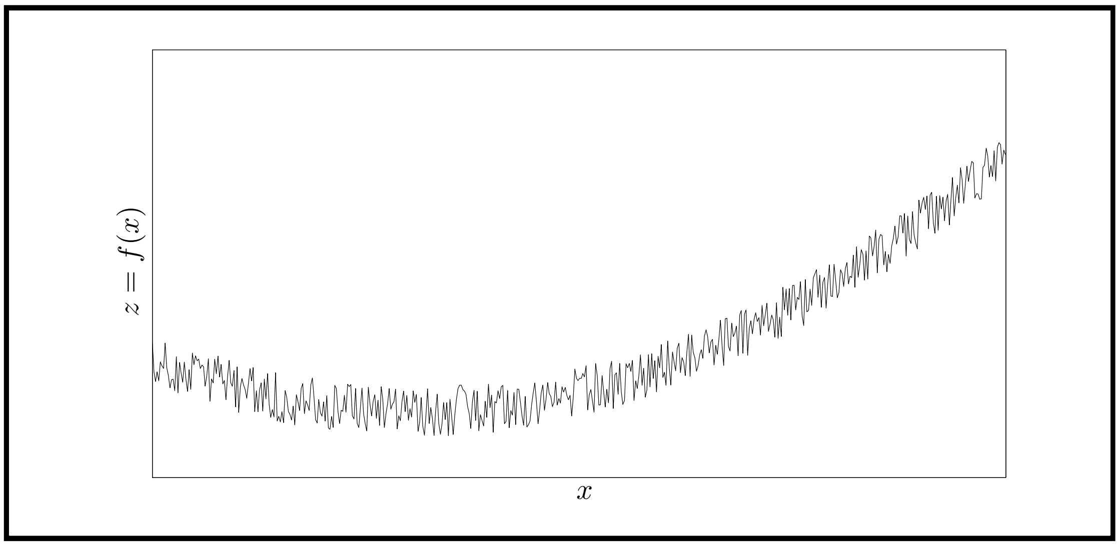


Figure 3.1: Exemplificative plot for the objective function of a complex simulated-based optimization problem.

3.3 Validation

The validation [38] of a model assesses the accuracy of its outputs and it is necessary if the model has to be used for prediction. Usually validation is executed comparing the measured outputs with the simulated outputs (obtained with the parameters generated by the calibration process) using a different set of inputs, which has not been used within the calibration procedure. If the difference between these outputs is relatively low (to be more precise, if it is lower than a certain threshold value for which the engineer could consider himself satisfied) for a large amount of simulation runs, the traffic flow model is deemed as validated. If this is not the case, it will be necessary to repeat the calibration and validation process.

Another common technique used for validation is the so-called dual sampling scheme [38]. According to this method, given two set of inputs, at the beginning the model is calibrated with the first set and is validated with the second, and then it is calibrated with the second set and is validated with the first one. So two groups of parameters are obtained, and the ultimate set could be obtained by finding a tradeoff between the two (e.g., by the calculation of the mean value).

To quantify the performance of the outcomes, various error measurement methods could be used, such as the Root Mean Square Percentage Error ($RMSP$), the correlation coefficient (r), and Theil's inequality coefficient (U) [19].

3.4 Measures of Fitness

Amongst the measures [19] used to estimate the fit between the simulation outputs and the actual measurements, there is the **Root Mean Square Percentage Error** ($RMSP$) which provides a good estimate of the total percentage error between the measured values y_i and the simulated values θ_i. It is defined as:

$$RMSP = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\theta_i - y_i}{y_i}\right)^2}$$

This method should be avoided when $y_i \approx 0$, because it might yield numerical problems (e.g., division by zero, overflow).

The **correlation coefficient** r, which measures the strength of the linear association between the actual y_i and the simulated θ_i traffic measurements, is:

$$r = \frac{1}{n-1}\sum_{i=1}^{n}\frac{(\theta_i - \overline{\theta})(y_i - \overline{y})}{\sigma_\theta \sigma_y}$$

where $\overline{y}$ and $\overline{\theta}$ are the mean values, σ_y and σ_θ are the corresponding standard deviations, n is the number of measurements, and i is the time index that indicates at which instant the measurement has been performed.

A more efficient goodness-of-fit measurement is **Theil's inequality coefficient**, because it is more sensitive and accurate than the previous two. It is defined as follows:

$$U = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \theta_i)^2}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n} y_i^2} + \sqrt{\frac{1}{n}\sum_{i=1}^{n}\theta_i^2}}$$

The square root of the numerator could be decomposed in three terms:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \theta_i)^2 = (\overline{y} - \overline{\theta})^2 + (\sigma_y - \sigma_\theta)^2 + 2(1-r)\sigma_y\sigma_\theta$$

Three more parameters could be drawn from Theil's coefficient U; these values evaluate different aspects of the error between simulated and actual data: U_m is the bias proportion (a measure of systematic error[4], that could be used for discovering potential errors in the counting of vehicles); U_s is the variance proportion (a

[4]Systematic errors are biases in measurements which lead to measured values being systematically too high or too low. They always affect the results of a simulation in the same direction, and they are caused by the environment and by the employed instruments.

measure of the ability of the simulated measurements to replicate the fluctuation in the actual measurements), and U_c is the covariance proportion (a measure of unsystematic error[5]). These parameters are defined as:

$$U_m = \frac{n(\overline{y} - \overline{\theta})^2}{\sum_{i=1}^{n} (y_i - \theta_i)^2}$$

$$U_s = \frac{n(\sigma_y - \sigma_\theta)^2}{\sum_{i=1}^{n} (y_i - \theta_i)^2}$$

$$U_c = \frac{2n(1 - r)\sigma_y \sigma_\theta}{\sum_{i=1}^{n} (y_i - \theta_i)^2}$$

and it could be easily realized, from the equation that shows the decomposition of the square root of the numerator of U in three terms, that $U_m + U_s + U_c = 1$.

3.5 Summary

In this chapter the ideas of calibration and validation of a traffic network model (which include the car-following and the lane-changing models) have been presented. The process of calibration consists, basically, in the estimation of the model parameters in order to minimize an objective function: the minimization task is performed through optimization algorithms (e.g., sequential quadratic programming and genetic algorithms). Finally, the model could be considered validated if the measure of fitness (e.g., Theil's inequality coefficient) associated to the calibrated parameters satisfies certain conditions.

[5]Unsystematic errors represent the unavoidable statistic fluctuation between the mean amd the single measurement. They are caused by unknown factors, and/or by known but no removable factors.

Chapter 4

Description and Modeling of the Traffic Network

4.1 Network Description

The traffic network under examination is a simple 10 kilometres stretch, being part of the A12 highway between the sites of Veenendaal and Maarsbergen, in the province of Utrecht, The Netherlands (Figure 4.1). The A12 highway starts from the city of Den Haag, passes by the cities of Utrecht and Arnhem, until it arrives at the German border. From there, the highway continues, but under a different name. It should be specified that the traffic flow of this stretch will be modeled only for the east-west direction, that is from Arnhem to Den Haag.

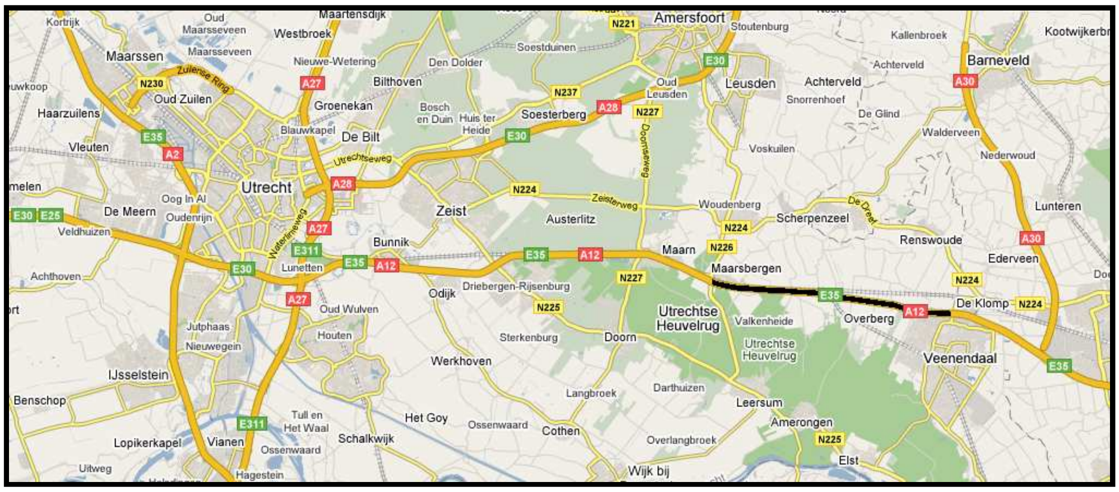


Figure 4.1: Traffic network in the surrounding of Utrecht (©2007 Google^{TM}).

This segment of highway has been chosen due to the absence of any kind of intersections with other roads for 10 kilometres, meaning that there are as few on-ramps and off-ramps as possible. This particular configuration is a good case study in order to test on the network new and improved control measures. The area at issue is, also, one of the more congested in The Netherlands, because it is densely populated, in particular during the peak hours.

The aim of this project is to create an algorithm which could be able to calibrate and validate the Paramics model of a traffic network. The simplicity of this network should also enable an easy modeling, while the frequent traffic jams are a necessary condition if we want to test control measures which are going to be applied during congestions. Unfortunately, after this network has been chosen for the project, the presence of a petrol station has been discovered just in the middle of the stretch, introducing one off-ramp and one on-ramp. Thus, the flow of the vehicles in the highway is perturbed by the vehicles that, stopping for a lay over, enter and exit the service station. In particular, the vehicles entering the highway from the on-ramp, adding up to the already strong flow of vehicles in the main stream, could cause congestions and traffic jams.

4.1.1 Details

The traffic network that is going to be modeled is the segment of A12 highway (Figure 4.2), starting at km 91 (after the intersection near Veenendaal) and ending at km 81 (before the intersection near Maarsbergen). Only one direction will be regarded (the one from Veenendaal to Maarsbergen) because there the traffic volumes are larger than in the opposite direction, due to vehicles going towards Utrecht. In fact the city, capital of the homonym province, is the fourth biggest agglomerate of The Netherlands, and this implies a huge number of vehicles moving in its proximity. Hence, the chronic problems of traffic congestion.

According to data supplied by the Dutch Ministry of Transportations (Ministerie van Verkeer en Waterstaat), in the 35 kilometres segment from km 65 until km 110[1] there are 77 detector loops, which continuously measure the traffic variables (flow, density, speed, etc.). To each of these detector loops is associated a position in the highway; unfortunately, this position is not always accurate. In order to have a good model of the network, it is necessary to know the exact position of the detector loops. Thanks to the service provided by Google Earth, it has been

[1]There is no mistake, since from km 65 to km 110 there are 35 kilometres of highway and not 45, probably due to a modification of the track; in fact, kilometres from 92.0 to 101.9 do not exist, and km 91.9 is directly followed by km 102.0: a *hole* of 10 kilometres is therefore created.

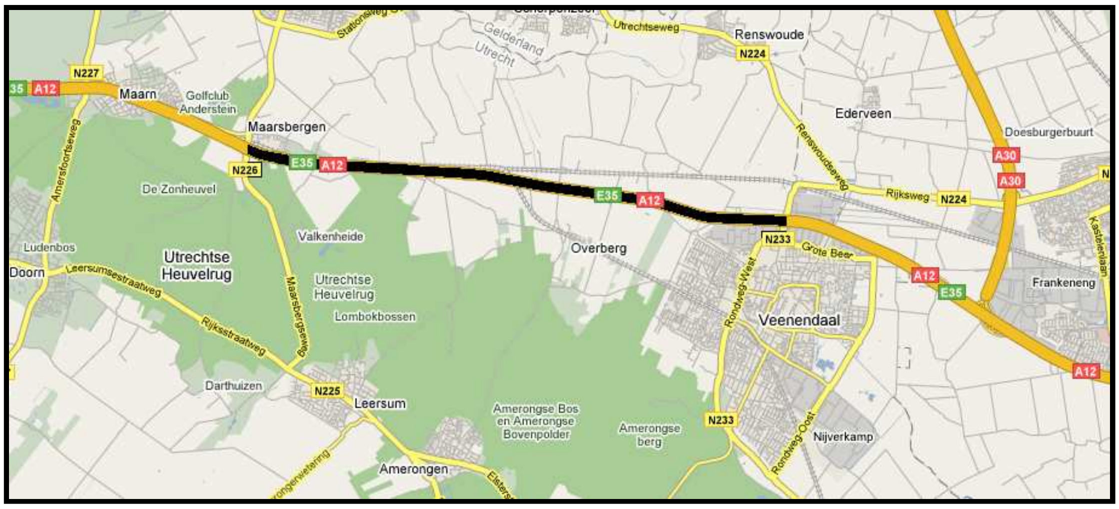


Figure 4.2: Segment of the A12 highway between Veenendaal and Maarsbergen (©2007 Google$^{\text{TM}}$).

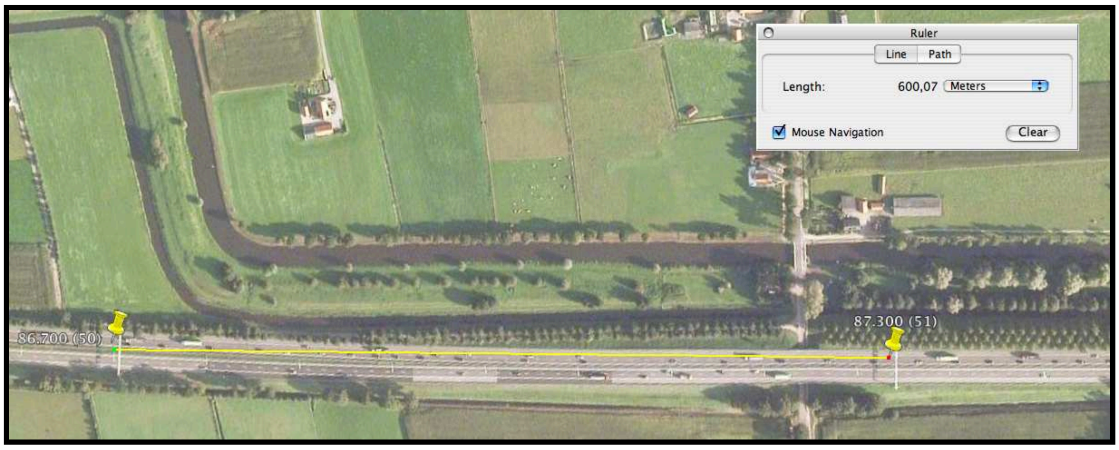


Figure 4.3: The ruler function on Google Earth (©2007 Google$^{\text{TM}}$).

possible to measure the true reciprocal distances between the detector loops (Figure 4.3). Most of the positions were really accurate, but some detector loops are positioned some dozens or even some hundreds of metres far from the point stated by the Ministry, and so the benefit of Google Earth has been critical. In fact, if this check had not been performed, the model would have had its detector loops in wrong positions along the highway: this would have triggered bigger errors in the estimate of the parameters, because the actual measurements would have been compared with the simulated measurements gathered from detector loops placed in different points of the highway. As an example, the actual flow at km 81.1

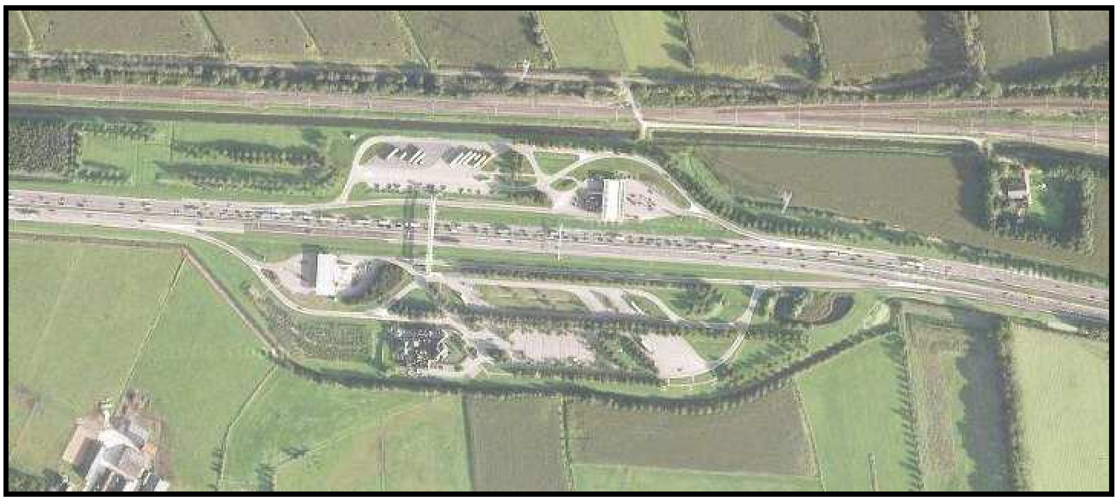


Figure 4.4: Service station at km 85 (©2007 Google^{TM}).

would have been compared with the simulated flow at km 81.4, and a such situation should be avoided.

Detector loops are usually placed few metres after control panels, which are clearly visible in the pictures offered by Google Earth. In the case study segment there are 25 detector loops, but only 17 control panels: hence, it is not possible to verify the exact position of 8 detector loops.

4.1.2 The Service Station

The service station is located about the 85th kilometer of the A12 highway; its entering ramp starts at 85.950 km, while its exiting ramp ends at 84.780 km (Figure 4.4). The presence of a petrol station affects the flow of the vehicles in the highway, due mainly to the vehicles which leave the station and enter the main road. In particular, it causes further congestion and traffic jams (Figure 4.5). Because of a lack of data regarding the vehicles passing through the service station[2], the accuracy of the model will be further on compromised[3].

[2]In Section §4.3.1, it will be shown how an approximation of the demand of the vehicles to and from the service station has been found.

[3]Actually, this is not a relevant concern: since the aim of this project is the design of an algorithm for calibrating and validating the Paramics model of a traffic network, it is not strictly necessary to test it on an accurate model.

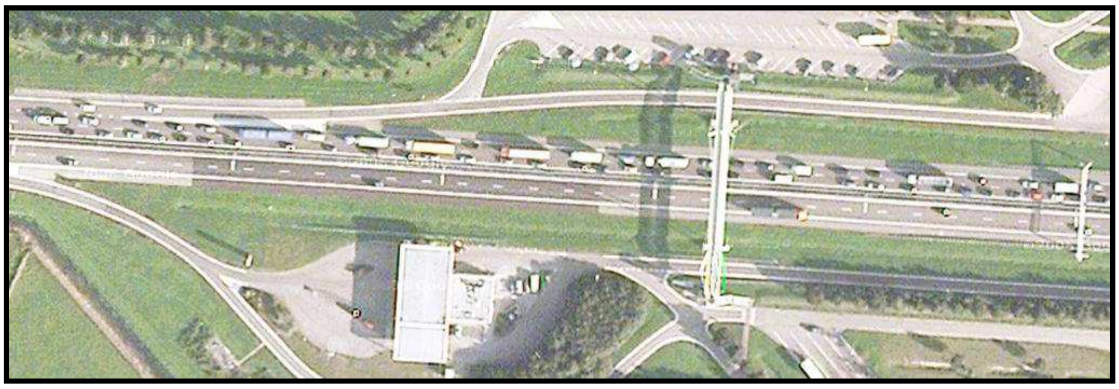


Figure 4.5: Traffic jam caught at the service station (©2007 Google^{TM}).

4.1.3 Available Actual Data

The data gathered by the 77 detector loops between the kilometres 65 and 110 are provided by the Ministry of Transportations; they consist of the flow (veh/h), the speed (km/h) and the density (veh/km) for each lane of the highway, of the average flow (veh/h/lane), the average speed (km/h/lane) and the average density (veh/km/lane) over all lanes, of the total flow (veh/h) and the total density (veh/km) over all lanes. These values are computed for every minute of the day, and the data for the whole month of January 2006 are available. The messages shown by control panels are also enclosed within these data; they are the suggested speeds (30, 50, 60, 70, 80, 90, 100, 120 km/h) or the service messages (such as the arrows that indicate to move from a lane to another, the red cross that warns about the closure of a lane, an abbreviation that informs about the end of the previous possible restrictions, etc.). Also the case in which the panels do not visualize any signal is registered in these messages. All these data are stored, for each day of the month of January 2006, in the variable *dat*, an about 70 MB cell containing several fields, each of these for a specific type of data (all the measurements detected and all the messages displayed).

4.2 Modeling of the Traffic Network

In order to be able to tackle the full task of the calibration and validation project within the allowed period of time, a rather simple model of the network has been built (Figure 4.6). This model is characterized by the presence of four zones: two of them are strictly origin zones, while the other two are strictly destination zones. The two main zones are the start of the freeway stretch near Veenendaal (zone 1) and the end before the intersection near Maarsbergen (zone 4). The entrance

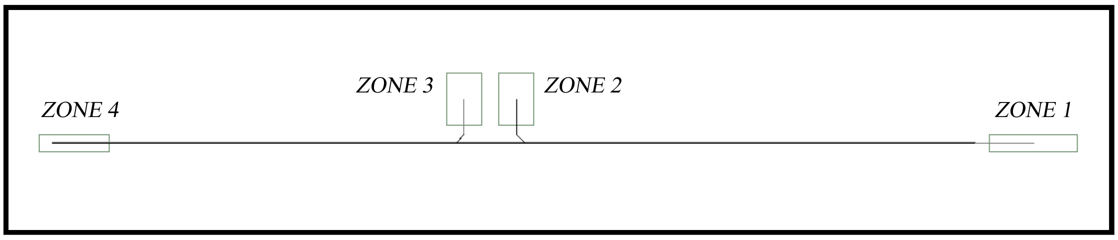


Figure 4.6: The Paramics model of the case study network.

(zone 2) and the exit (zone 3) of the petrol station are the remaining zones.

A 10 kilometres stretch connects zones 1-4, while zones 2-3 are linked to it through, respectively, an off-ramp and an on-ramp. The traffic network model will not consider the curvature of the road, which will be actually represented by a mere sequence of nodes and links.

It has been decided to associate one link for every detector loop, and vice versa, except for the border links of the model, which will be bereaved of them. Each detector loop will be set 50 metres before the exit node of its link, following the suggestion written in the Paramics manual [36]: *a detector loop should not be placed too close to the end of a link because it may cause some imprecisions during the transit of vehicles through a node.* Regarding the length of detector loops, it has been chosen as 20 metres[4] the distance between the downstream and the upstream edge of the detector loop. This distance allows the sensor to detect also the fastest vehicles (moving at 144 km/h) while the discrete time step for the simulation is set at 0.5 seconds (maximum possible value). The hazard of counting two or more times the same vehicle (when its speed is low) is avoided thanks to the Matlab plugin [30] for Paramics, which has the functionality to set an identifier for each vehicle that has been counted by the detector loop for the first time.

After this considerations, it is possible to build the model. The first node is positioned within zone 1, at km 91.600; while the first detector loop is placed at km 90.985, the second node (which ratifies the end of the link 1:2) will be 50 metres beyond, at km 90.935. Proceeding with the same approach, the stretch will be built until reaching the last detector loop (at km 81.210) and the last node (km 81.160). A further link is necessary (and then another node) so that the last detector loop does not lie within the destination zone; otherwise, the vehicles would not be detected by the last detector loop, because Paramics deems the zones as an external part of the network, and so the vehicles that pass there would not be

[4]In real world, the distance between the two edges of a detector loop is about $3 \div 5$ metres.

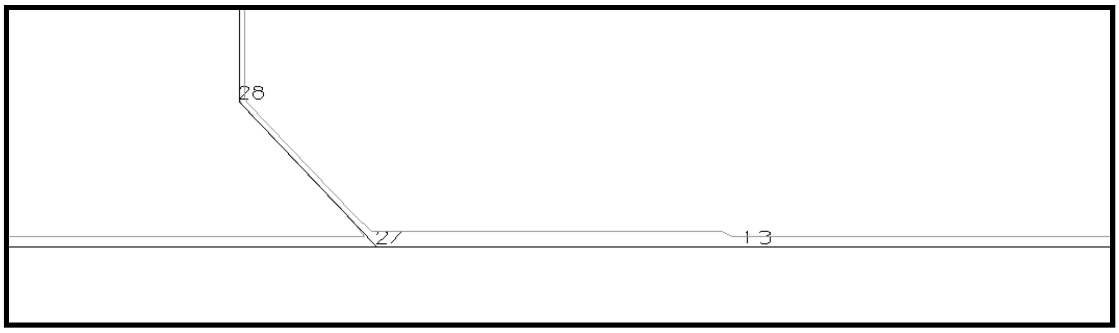


Figure 4.7: The off-ramp to the service station (zone 2).

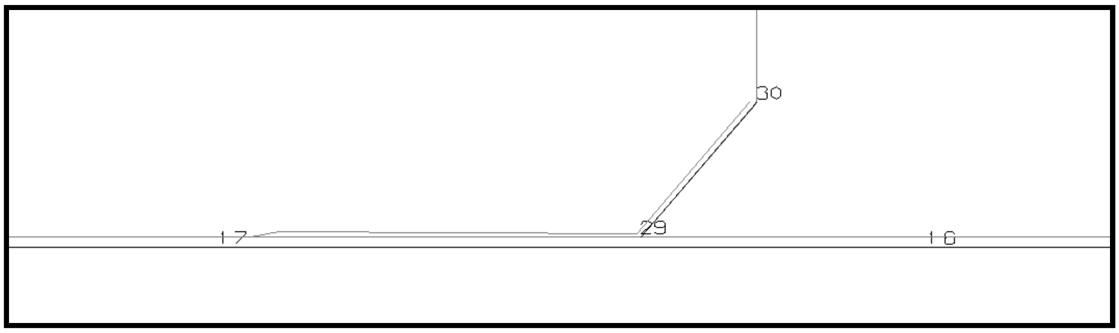


Figure 4.8: The on-ramp from the service station (zone 3).

counted by any potentially installed detector loop.

4.2.1 Modeling of The Service Station

The setting up of the ramps (for and from the service station) is, instead, more elaborate. Concerning the off-ramp (Figure 4.7), it has been sufficient to add a new node (27) in the exact point in which the off-ramp completes itself, then to specify the length of the ramp (240 metres) in the *links* file and, at last, to connect this node with the previous (13) and with the next (not in the picture) ones, and with the entrance of the service station (28).

Relatively to the on-ramp (Figure 4.8), the procedure is completely different: a new node (29) has to be created in the proximity of the intersection between the ramp and the highway, then some parameters have to be specified (length of the ramp, headway factor, minimum ramp time), but the node does not have to be linked with the closest ones (16 and 17) in the *links* file.

4.2.2 Speed Limits on the Network

In real highways and during the circumstances of congestion, the control panels advise the drivers about the suggested speed, in order to maximize the flow of vehicles and to minimize the negative effects of traffic jams. This dynamic speed limit, when applied, could be passed to the traffic network model by means of the Speed Limit plugin [30] for Paramics, so that its vehicles will be enforced to follow the suggested speed. This ploy allows the real traffic conditions of the network to be more faithfully reproduced than not providing speed limits.

The speed limits can be stored in the *speedlimits* matrix (Table 4.1), which has two dimensions: the first is equal to the number of links of the model, as reported in the *links* file (this number is twice than expected, because for every link it is necessary to include also the opposite direction; an an example, given the link A:B, there will be the link B:A that will be marked as barred), while the second matches with the number of minutes of the simulation. Therefore there will be a speed limit for each link at every minute.

Since the control panels do not always visualize a speed limit (usually they are turned off and sometimes they are displaying a particular message), a suitable speed value has to be assigned to the *speedlimits* matrix in these cases. When there are no speed limits displayed, the default speed limits of the highway are applied; unfortunately, these values are not available from the network data, so a standard speed limit of 120 km/h has been enforced. In the case of signals displayed during incidents, works and exceptional conditions, a speed limit of 10 km/h has been chosen[5].

The Speed Limit plugin [30] is able only to apply the speed limits in the whole link, and is not able to differentiate between different lanes. Since the speed limits are sometimes distinct from lane to lane (e.g., 50 km/h in the first lane and 70 km/h in the overtaking lane), it has been necessary to reach a compromise: the limit applied to both lanes will be the average of the two speed limits.

It has been opted, moreover, to apply the speed limits of the control panel (corresponding to the closer detector loop) positioned in the link i to the vehicles crossing through the link $i + 1$, because the detector loops are close to the end of the link and also because drivers need some time to adapt their speed to the one suggested.

[5]Since in the scenarios chosen for testing the algorithm for calibration and validation such events should not occur, the strict speed limit of 10 km/h has been selected in order to point out their potential and unpredicted presence.

Link \ Minute	...	17	18	19	20	21	22	23	24	25	...
⋮	⋱	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋰
11:12	...	120	120	120	120	120	120	120	120	120	...
11:9	...	0	0	0	0	0	0	0	0	0	...
12:13	...	120	120	120	120	120	120	120	120	120	...
12:11	...	0	0	0	0	0	0	0	0	0	...
13:27	...	120	120	120	120	120	120	120	70	70	...
13:12	...	0	0	0	0	0	0	0	0	0	...
15:16	...	120	120	120	120	70	70	70	50	50	...
15:27	...	0	0	0	0	0	0	0	0	0	...
16:17	...	120	120	120	120	70	70	70	50	50	...
16:15	...	0	0	0	0	0	0	0	0	0	...
17:19	...	120	120	70	70	50	50	50	50	50	...
17:16	...	0	0	0	0	0	0	0	0	0	...
19:21	...	120	70	50	50	50	50	50	50	120	...
19:17	...	0	0	0	0	0	0	0	0	0	...
21:23	...	90	50	50	50	50	50	120	120	120	...
21:19	...	0	0	0	0	0	0	0	0	0	...
23:24	...	50	50	50	50	120	120	120	120	120	...
23:21	...	0	0	0	0	0	0	0	0	0	...
24:25	...	50	50	120	120	120	120	120	120	120	...
⋮	⋰	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Table 4.1: Extract from the *speedlimits* matrix. Speed limits applied in minutes $17 \div 25$ of the simulation, in links $12 \div 30$.

Regarding the link related to the last detector loop, the measured speed has been strictly applied. This choice will be better explained in Section §4.3.1. At last, on both ramps the speed limit of 50 km/h has been applied.

4.2.3 Selection of the Simulation Parameters

The following parameters are requested by Paramics in order to run a simulation, and they could be set in the *configuration* file of the network. When a parameter is going to be used always with the same value for all the simulation runs that will be executed for this project, such value will be highlighted between brackets.

Start Time Is the time at which the simulation starts, in the hh:mm:ss format.

Simulation Time Specifies the length of the simulation, in the hh:mm:ss format.

Demand Weight (100.0) Sets the dynamic demand for the simulation, ranging from 0 to 200% of the demand defined in the appropriate file. In this way, the demand could be easily increased or decreased from its starting value.

Seed Indicates the starting value used by the random number generator; identical networks simulated with the same seed (on the same computer) will yield the same results. If the selected seed is 0, a random seed will be generated.

Generator (0) Specifies which Random Number Generator is being used by the simulation: Marsaglia (0) or Mersenne (1).

Loop Length (2.0 m) Is the default length of a detector loop, and it is applied if it is not otherwise specified in the *detectors* file.

Speed Memory Sets the number of simulation time steps, previous to the current time step, that all vehicles record in order to implement the driver reaction time, by basing the change in speed of a following vehicle on the speed of the leading vehicle at a time in the recent past.

Curve Speed Factor (0.0) Defines how much the vehicles slow down due to the curvature of the road.

Hand Drive (Right) Could be left or right, depending on the side of the carriageway used by vehicles (it is the right one in The Netherlands).

Units (Metric) Specifies the unit convention for display in Paramics (USA, UK, Metric), although all calculations are made in metric measurements and eventually converted to the selected unit type.

Time Step Detail (2) Defines the number of discrete simulation intervals that are simulated every second.

Mean Headway Time Is the global mean target headway, expressed in seconds, between a vehicle and its following vehicle.

Mean Reaction Time Defines the lag in time between a change in speed of a vehicle and the reaction of the following vehicle to this change.

Gap (2.0 m) Is the minimum distance between vehicles in a queue.

Queue Speed (7.2 km/h) Defines the maximum speed a vehicle could move in order to be counted as being in a queue.

Queue Distance (10.0 m) Sets the maximum distance between vehicles so that they could be considered in a queue.

Weight Heavy (3.0 t) Specifies the minimum weight for a vehicle to be considered a heavy one.

4.2.4 Available Simulation Data

While the traffic simulation is performed by Paramics and the computations for calibration are done in Matlab, the Matlab plugin [30], exploiting the data collected by the detector loops of the model of the traffic network, could calculate several kinds of variables. These values are saved in the three-dimensional matrix *detectordata*, contained in the file *detector.mat*, which can be read by Matlab. The first two dimensions of the matrix identify the minute of simulation and the detector at which the values stored are referring to, while the third dimension contains the indices that select the measurement; for example, in *detectordata(57,14,5)* is stored the average flow measured, during the 57th minute of the simulation, at the 14th detector loop of the model. Amongst these measurements, there are the estimated and the approximated space mean densities (veh/km) and space mean speeds (km/h), the time mean density (veh/km), the time mean speed (km/h), the average flow (veh/h), the number of vehicles that passed through the detector loop at a certain minute (veh), and the total number of vehicles that passed through it since the start of the simulation (veh).

4.3 Modeling of the Demands

In order to complete the Paramics model and, therefore, to start a simulation run, it is necessary to evaluate the traffic demand: thus, the incoming flow of the vehicles could be imposed at the origin zones of the network.

For this, the Paramics simulation model needs three input files: the first of these displays the origin-destination matrix (demands), the second describes the distributions (profiles) over time of the vehicles entering the network during the length of the simulation, the third associates to each demand the concerning profile.

4.3.1 Demand Estimation

The traffic demand can be set in the *demands* file (Table 4.2); it consists of a ZxZ matrix (since Z is the number of the zones of the network) that contains the number of vehicles that are going to move from a zone to another one during the

```
Demand Period 1
Matrix Count 1
Divisor 1

Matrix 1
From 1:  0  100  0  2448
From 2:  0    0  0     0
From 3:  0    0  0   107
From 4:  0    0  0     0
```

Table 4.2: Example of *demands* file. The OD matrix contains three non-zero elements: $\mathrm{OD}(1,2) = 100$, $\mathrm{OD}(1,4) = 2448$, $\mathrm{OD}(3,4) = 107$.

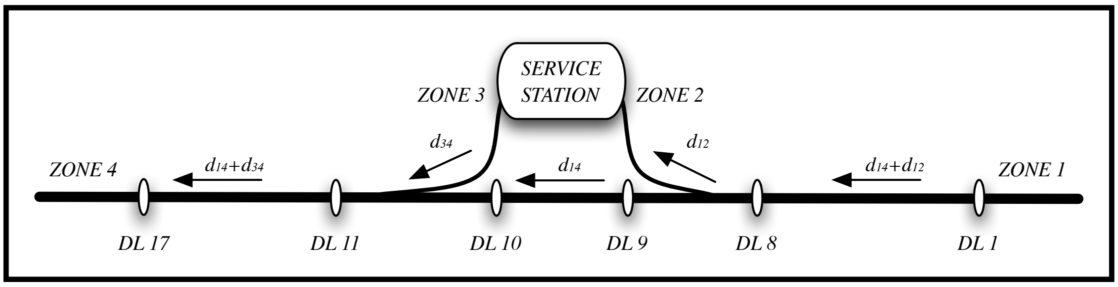


Figure 4.9: Scheme of the model of the traffic network; the most relevant detector loops and the demands from the OD matrix are highlighted.

simulation. This matrix is also known as the origin-destination (OD) matrix; for example, the element $\mathrm{OD}(1,4)$ indicates the amount of vehicles leaving from zone 1 and directed to zone 4. In this case study, there exist only three flows of vehicles: those directed from zone 1 to zone 4, those from zone 1 to zone 2, and the ones from zone 3 to zone 4.

Given the available measurements collected from the actual detector loops (Section §4.1.3), it has been decided to use the measured flow of vehicles in order to extrapolate the demands needed for the Paramics model. In fact, the flow is defined as the number of vehicles that pass through the detector loop in one hour; since this value is calculated every minute, if the flow is divided by 60, the number of cars that cross the detector loop in that minute will be obtained. Therefore it is easy to count the quantity of vehicles at each detector and at every minute. Referring to Figure 4.9, a brief description of how the origin-destination matrix has been built follows.

OD(1,:) As a first approximation, the total demand from zone 1 ($d_1 = d_{14} + d_{12}$) could be obtained as the number of vehicles passing through the first detector loop (DL 1 at km 90.985) from the first to the last minute of the simulation.

OD(1,2) Subtracting the number of vehicles measured at the detector loop subsequent the off-ramp (DL 9 at km 85.400) for the service station from the one of the detector loop prior the same ramp (DL 8 at km 86.100) yields the demand d_{12} from zone 1 to zone 2.

OD(3,4) The same operation realized between detector loops DL 10 at km 85.270 (before the on-ramp) and DL 11 at km 84.780 (after the on-ramp) yields the demand d_{34} from zone 3 to zone 4.

OD(1,4) Finally, to obtain the demand d_{14} from zone 1 to zone 4, the demand d_{12} from zone 1 to zone 2 have to be subtracted from the total demand d_1 from zone 1.

The other elements of the demands matrix are zero, because vehicles could travel only on the three routes already mentioned.

Finally, since it is not possible to impose the flow of the vehicles exiting the highway from zone 4, a further and different input has been added to the model of the traffic network: the speed of the vehicles in the last link has been forced to be equal to the speed measured by the last detector loop (DL 17) of the real network.

4.3.2 Demand Profiles

The statistical distribution over time of the demand of vehicles for the simulation is defined in the *profile* file: every minute a certain percentage of vehicles, as a fraction of the total demand, is going to leave the origin zone.

Referring to the exemplifying *profile* file in Table 4.3, the simulation starts at 06:15, and two one-minute basis profiles are defined: since the simulation lasts for 50 minutes, there are so many elements in each profile. The sum of these integer numbers, divided by the parameter *divisor* (that in this case is equal to 100), is 100, as requested by Paramics. Each element indicates the percentage of vehicles, calculated from the demand associated to the profile, which will leave the origin zone in that minute. As an example, the first element of the first profile (associated to $\mathrm{OD}(1,4) = 2448$ as in Table 4.2) is equal to 263: this means that, in the first minute of the Paramics simulation, the 2.63% of 2448 vehicles ($\simeq$ 64) will leave zone 1 directed to zone 4.

It is simple to calculate the demand profile for zone 1: using the flow from the first real detector loop, it is easy to obtain the number of vehicles for each minute of

```
Demand Profile
Profile Count 2

Profile 1
Period Count 1
Divisor 100
Interval 1
Period 1 Start 06:15:00
263  192  247  184  338  326  251  255  173  220
243  267  314  228  377  228  106  133  232  228
228  239  263  259  239  251  271  239  239  232
290  251  216  216  133  200  181  188  177  161
208   39   20   20    0   24   35   20   90  266

Profile 2
Period 1 Start 06:15:00
  0    0    0    0    0    0    0    0    0    0
283  207  266  198  364  351  271  275  186  237
262  288  338  246  406  246  114  143  250  246
246  258  283  279  258  271  292  258  258  250
313  271  233  233  143  216  195  203  191  172
```

Table 4.3: Example of *profile* file. Two different demand profiles are defined for this model.

the simulation; this quantity, divided by the total demand from zone 1, yields the percentage of vehicles leaving zone 1 for every minute. This profile could be used for both the demands from zone 1.

Regarding the profile of the vehicles that leave the service station directed to Utrecht, the question is more complex. It is not possible to obtain the profile directly from the data of the real detector loops near the on-ramp, due to the different speeds of the vehicles in distinct positions of the network, an event caused mainly by congestion. The adopted solution is the use of the demand profile from zone 1 shifted by some minutes: this lag time could be estimated as the sum between the time needed by the vehicles to reach the service station (starting from zone 1) and the time to refuel. Both travel time and stop time have been chosen to be equal to 5 minutes, respectively due to observations of simulation runs and due to an estimate of the average stop time at a service station; hence the shift time will be equal to 10 minutes. The choice to use the same profile from zone 1 is justified because the vehicles which leave zone 3 arrive at the petrol station with

Matrix Count 1				
Profile Matrix 1				
From 1:	1	1	1	1
From 2:	1	1	1	1
From 3:	1	1	1	2
From 4:	1	1	1	1

Table 4.4: Example of *matrix* file. Two different profiles are associated to the 16 possible routes of the model.

a similar statistical distribution.

4.3.3 Association between Demands and Profiles

The *matrix* file (Table 4.4) is used to link each demand of the origin-destination matrix with a profile. In this model, the demands $OD(1,2)$ and $OD(1,4)$ will be associated to the first profile, while the demand $OD(3,4)$ will be associated to the second profile. The other elements of the OD matrix, since they are referring to routes with null demand, could be indifferently linked to any available profile.

4.4 Summary

This chapter has been divided into three main sections. The first one illustrates in detail the studied stretch on the A12 highway, explains the problem introduced by the presence of the service station and gives a summary of the available data from the measurements taken during the month of January 2006. The second section describes the modeling of the network in Paramics, from the main structure to the selection of the configuration parameters, the enforcing of the advised speed limits and the gathering of the simulated data through the Speed Limit and Matlab plugins. The last section deals with the modeling of the traffic demand through the possible routes of the network, and with the distribution profiles associated to these demands.

Chapter 5

The Algorithm for Calibration and Validation

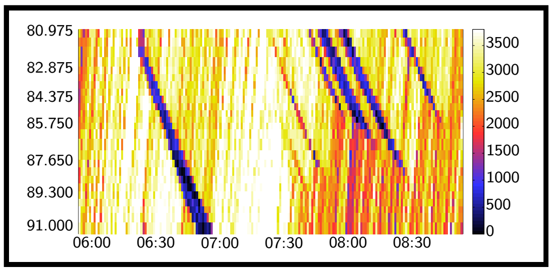
In this chapter it will be shown, first, how the calibration and validation scenarios have been chosen; then, three different algorithms will be illustrated: the main algorithm for *calibration and validation*, and the secondary algorithms for *cross-validation* and for *calibration with simulated data.*

5.1 Selection of the Traffic Scenarios

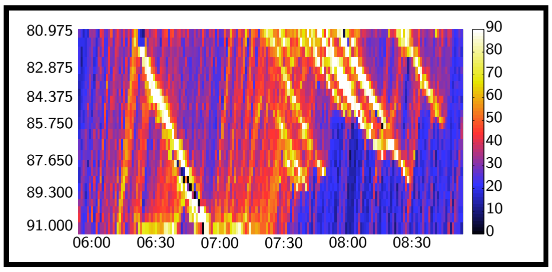
5.1.1 Calibration Scenario

The model of a traffic network will be completed once the scenario it has to represent will be chosen. A scenario deemed interesting for the calibration process is the one in which, in the stretch between Veenendaal and Maarsbergen and for a limited amount of time, particular traffic conditions verify, so that the algorithm for calibration and validation could be effectively tested. In fact, all the possible traffic states should be present in the scenario, in order to calibrate all the possible effects that cause such states.

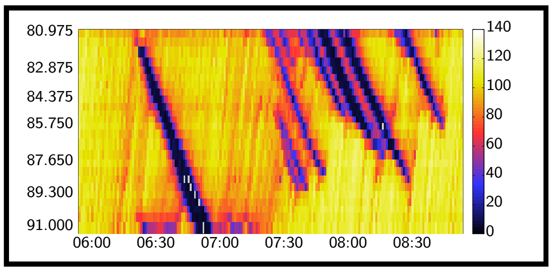
Therefore, the chosen scenario for the simulation will be characterized by both free flow traffic and congestions, in order to obtain strongly variable traffic measurements (as flow, speed and density) along the whole network and for the entire duration of the simulation: these are the required particular traffic conditions. Moreover, the congestion state is interesting because it is the most difficult to describe since many effects play a role; finally, it is a situation that should be improved by means of a further use of the calibrated and validated model.



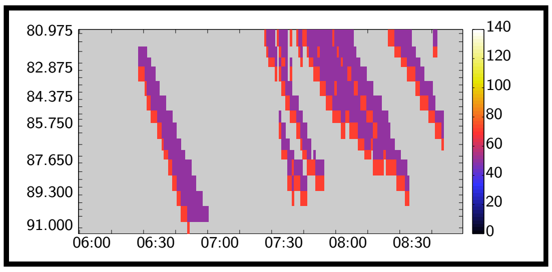
(a) Flow (veh/h).



(b) Density (veh/km).



(c) Speed (km/h).



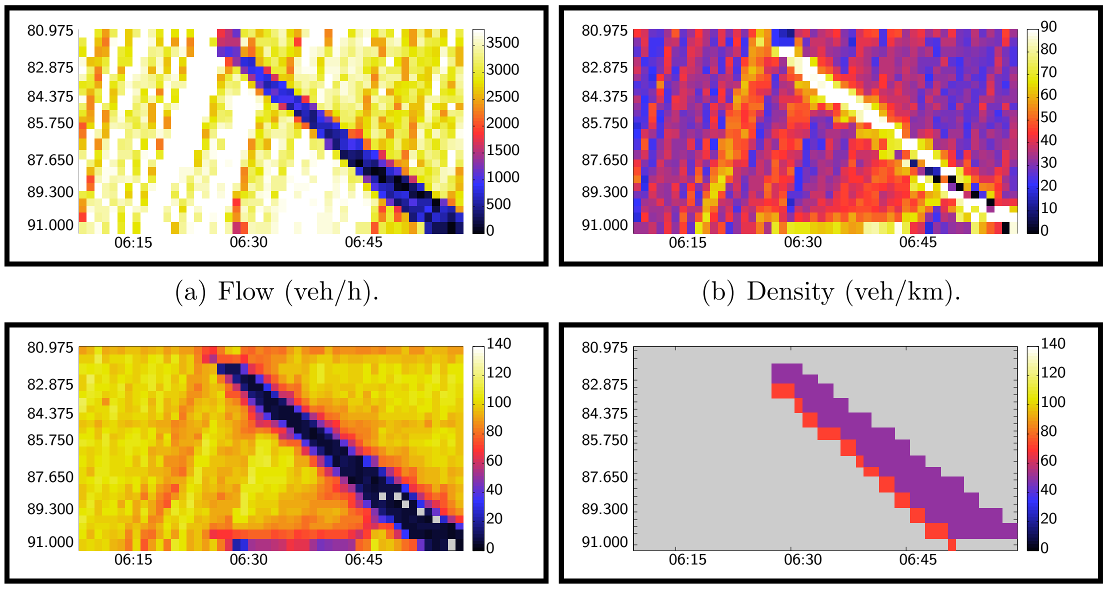
(d) Speed Limits (km/h).

Figure 5.1: Real measurements in the stretch, detected between 6:00 and 9:00 on January 30th, 2006.

For this reason, by evaluating the available information about the traffic scenarios for each day of the month of January 2006, the day of Monday 30th has been selected as the calibration scenario, due to the presence of continuous and heavy traffic jams during the whole morning. This sequence of congestions[1] emerges, respectively, at 6:25 at km 81.4, at 6:30 at km 91.0, at 7:15 at km 77.1, at 7:25 at km 80.6 and at 7:40 at km 86.7; two shock waves resolve at 8:15 at km 86.1 and at 9:00 at km 81.4. All these traffic situations are featured in Figure 5.1, that shows (by means of a chromatic scale) the detected values of flow, density, speed and the applied speed limits on the stretch, from the first detector loop at km 90.400 to the last one at km 81.400, and for each minute within the selected period of time. In other days there are traffic jams and congestion too, but usually they are originated by incidents or road works, or they arise in other points of the A12 highway, outside of the considered stretch (from km 91 to km 81).

The most representative event (Figure 5.2) for the case study is the traffic jam

[1]Congestions usually move backwards for the following reason: at a certain instant, due to any cause, a congestion may arise, and so the incoming vehicles start to enqueue in succession; meanwhile, the queue developes itself backwards. Then, when the forward vehicles can eventually move, they leave the head of the queue, but at the same time other vehicles enqueue themselves in the queue, and so its head and its tail move backwards.



(a) Flow (veh/h). (b) Density (veh/km).

(c) Speed (km/h). (d) Speed Limits (km/h).

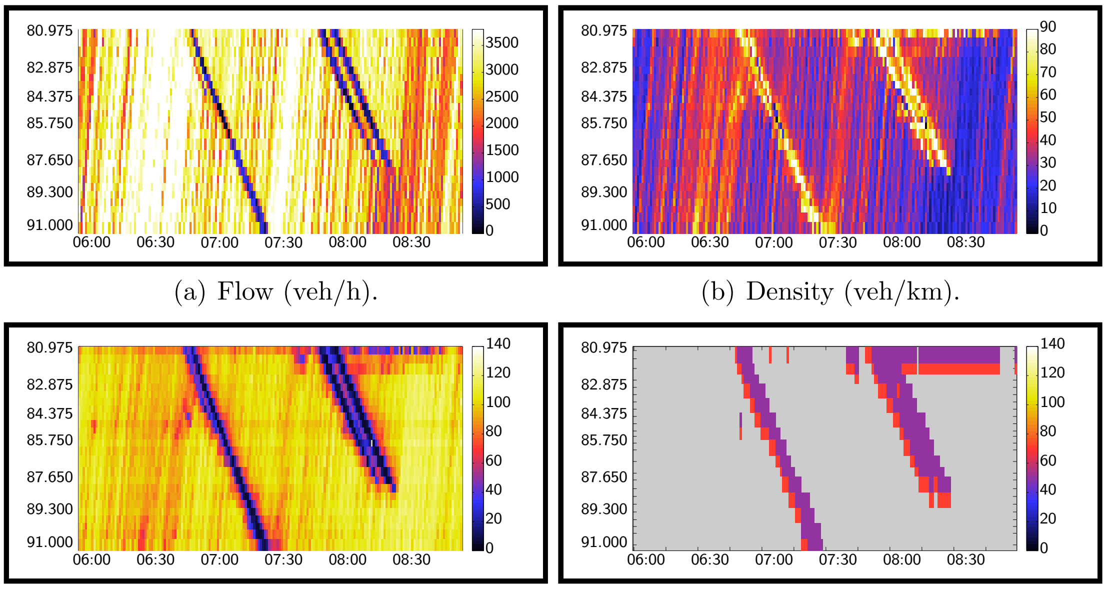
Figure 5.2: Real measurements in the stretch, detected between 6:10 and 7:00 on January 30th, 2006.

which starts at 6:25 at km 81.4, that is approximately the position of the one but last detector loop of the modeled network. It means that the congestion begins at the intersection nearby Maarsbergen, because of the vehicles entering the highway or changing their lane in order to facilitate the insertion of the new vehicles in the freeway. This traffic jam spreads itself upstream, and after about thirty minutes the congestion reaches the end of the stretch at km 91. Therefore, the chosen scenario consists of a spontaneous holdup, caused by an excessive flow of vehicles and not by other factors, as accidents or road works. Since the traffic jam which is intended to be simulated starts at 6:25 and ends after 30 minutes, a simulation interval from 6:10 to 7:00 has been selected[2], so that the vehicles have enough time to spread themselves through the whole network, as they only enter the network from the zones of origin.

5.1.2 Validation Scenario

The selected scenario (Figure 5.4) for the validation process is January 31st, 2006, that is the day after the day chosen as the calibration set; the most relevant

[2]Due to the large simulation time needed by Paramics, only a short period can be used, although simulating within a larger period of time could have yielded more accurate and realistic results but also an unacceptable duration of this project.



(a) Flow (veh/h). (b) Density (veh/km).

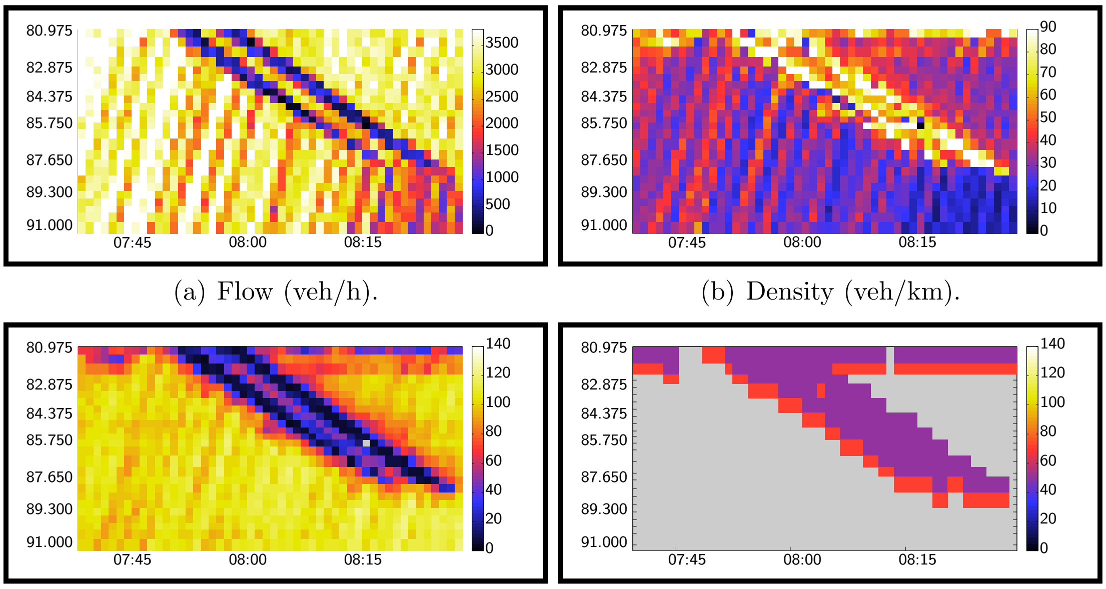
(c) Speed (km/h). (d) Speed Limits (km/h).

Figure 5.3: Real measurements in the stretch, detected between 6:00 and 9:00 on January 31st, 2006.

congestions appear during the morning peak hours: a traffic jam from about 7:00 to 7:30, and two parallel shock waves around 8:00. In order to have both the lengths of the calibration and validation simulations of the same duration (although this is not requested), the simulated period for the validation process starts at 7:40 and ends at 8:30, so that it could include the parallel shock waves (Figure 5.4). This scenario has been selected because it is similar to the one used for the calibration process, as a traffic jam happens between km 81.0 and km 88.5. In point of fact, the validation scenario should be completely different from the calibration scenario, in order to effectively test whether the calibrated parameters (the best-fitting parameters for the calibration scenario) also fit for a totally different traffic situation. But, due to the testing purposes of this project, the validation scenario has been simply chosen similar to the calibration one.

5.1.3 Data Preparation

In order to extract the requested data (i.e., those obtained by the detector loops placed in the 10 kilometres stretch during the selected period of simulation) from the file containing the measurements of the whole day by means of the function *cv_data*, some parameters have to be selected, both for the calibration and the validation process.



(a) Flow (veh/h). (b) Density (veh/km).

(c) Speed (km/h). (d) Speed Limits (km/h).

Figure 5.4: Real measurements in the stretch, detected between 7:40 and 8:30 on January 31st, 2006.

The first parameters are the paths of the files that contain the information about the chosen scenarios: they will be stored in the variables *cal_set* and *val_set* as strings. The starting and the final instants of both simulations are set in the variables $start\_cal = [06\;10]$, $end\_cal = [07\;00]$, $start\_val = [07\;40]$, $end\_val = [08\;30]$: all these instants are defined in the format $[hh\;mm]$.

Then, from these variables, the total duration of the simulations in minutes, $n\_cal = 50$ and $n\_val = 50$, could be easily obtained[3]. Both n and the number $m = 17$ of detector loops present in the model will be one of the main parameters in order to build the majority of the matrices used during the run of the algorithm.

In order to extract the required information from the chosen scenario (e.g., from the calibration scenario), the variables $data\_Y_1 = 13$, $data\_Y_2 = 11$ and $data\_Y_3 = 14$ allow the algorithm to select respectively the total flow (veh/km), the average speed (km/h) and the total density (veh/km) detected.

Once the (calibration or validation) scenario has been selected, together with the other parameters, the function *cv_data* extracts, from the data available for the

[3]The lengths of the simulations of the two scenarios do not necessarily have to be equal.

whole day and for the whole network stored in the structure *dat* (Section §4.1.3), the measurements (flow and speed) that later will be used for the calculation of the objective function, only for the m detector loops and for the n minutes of the simulation. This operation allows the above-mentioned measurements to be stored in two simple matrices with m rows and n columns ($\boldsymbol{Y_1}$ and $\boldsymbol{Y_2}$), instead of using the whole *dat* structure, whose size is about 70 MB. This solution consents to save time (because the huge data are not going to be loaded anymore) and memory (the *dat* structure is removed from the primary memory when the routine *cv_data* terminates).

The function *cv_data* prepares, moreover, the structure *message*, where the messages appeared on the control panels in the period chosen for the simulation are stored[4]. Finally, the variable l records the number of links of the model, as indicated in the file *links*. Both l and *message* are used for the setting up of the *speedlimits* matrix (Section §4.2.2).

5.2 Calibration and Validation

In this section, the algorithm that performs the calibration and the validation of the model of the traffic network will be explained (Figure 5.5). First of all, a detailed overview of the input parameters for the algorithm will start this section. Then, it will be explained why the mean headway time and mean reaction time have been chosen as the calibrating parameters. Next, the available optimization functions are illustrated and the computation of the objective function is explained. Finally, the validation process is commented.

5.2.1 Input Data

In order to be able to run the algorithm for *calibration and validation*, some parameters have to be previously selected.

Data Preparation Parameters

As it has been seen in Section §5.1.3, the calibration and the validation scenarios have to be described (with their paths and their starting and final instants), along

[4]A defect in the provided data has been fixed: sometimes a NaN (Not a Number, in Matlab) value appears within the speed measurements. Comparing these values with the measurements of the previous and the next instants, it is clear that the vehicles are idle in a traffic jam, and that the value of NaN should be replaced by a 0, since the mean speed of the vehicles in that minute and in that detector loop is 0. If this correction had not been implemented, the matrix *speedlimits* would not have been usable by the Speed Limit plugin [30] for Paramics.

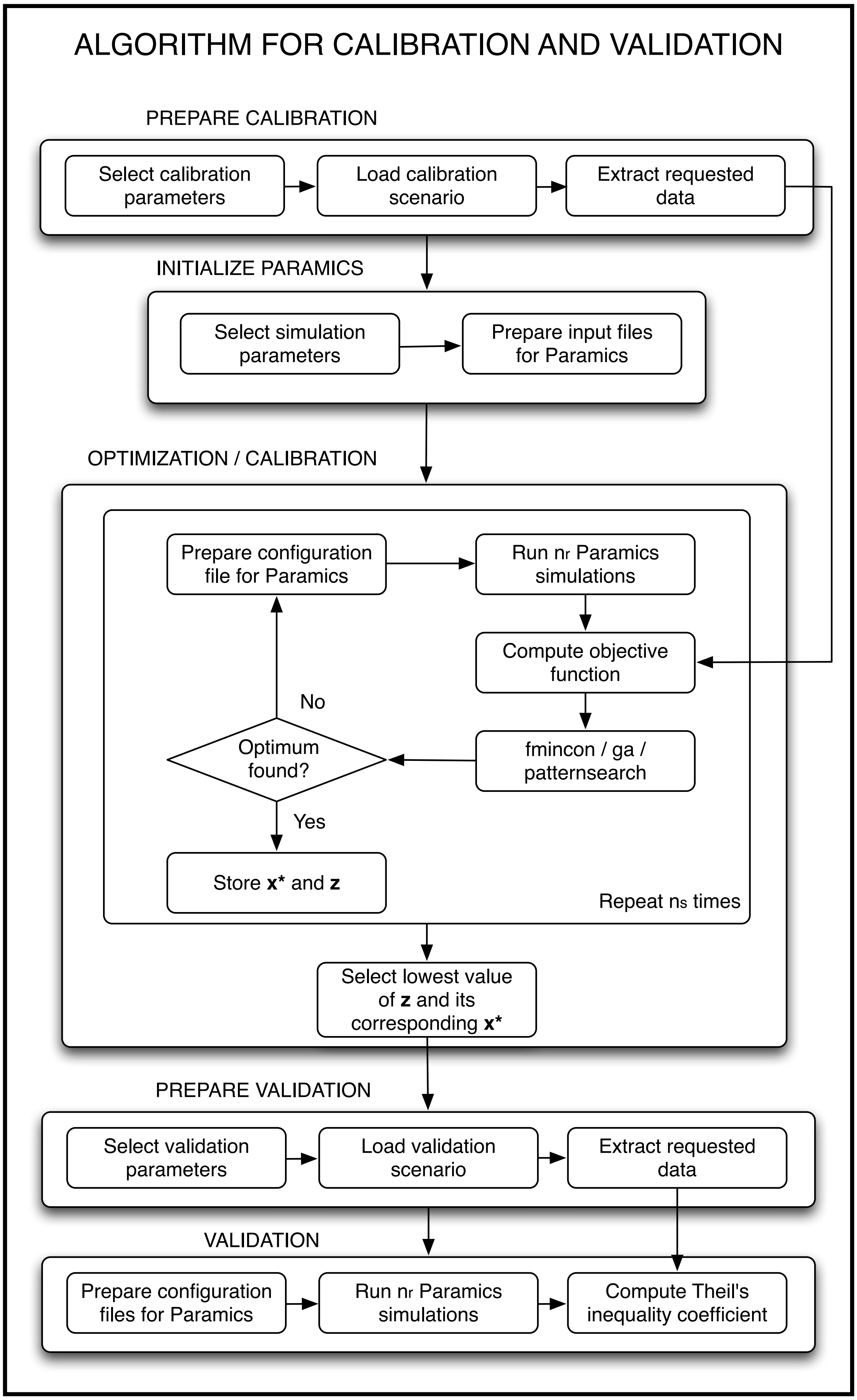

ALGORITHM FOR CALIBRATION AND VALIDATION
PREPARE CALIBRATION
Select calibration parameters
Load calibration scenario
Extract requested data
INITIALIZE PARAMICS
Select simulation parameters
Prepare input files for Paramics
OPTIMIZATION / CALIBRATION
Prepare configuration file for Paramics
Run nr Paramics simulations
Compute objective function
fmincon / ga / patternsearch
Optimum found?
No
Yes
Store x* and z
Repeat ns times
Select lowest value of z and its corresponding x*
PREPARE VALIDATION
Select validation parameters
Load validation scenario
Extract requested data
VALIDATION
Prepare configuration files for Paramics
Run nr Paramics simulations
Compute Theil's inequality coefficient

Figure 5.5: Flow chart of the *calibration and validation* procedure.

with the variables that allow the algorithm to extract the selected measurements. It could be specified that both m and l are fixed for any simulation regarding this network, but they could be changed if the model of the network is expanded or the missing detector loops are added to it.

Paramics Input Files Parameters

The following parameters need to be selected, so that some of the input files for Paramics (*configuration*, *profile*, *speedlimits.mat*) could be created.

The parameter *timestep*, to be specified in the $configuration$ file (Section §4.2.3) has been set equal to 2, so that the computational complexity of the Paramics simulations will be held at a low level, due to the length of the calibration runs and to the availability of a short period of time for the fulfillment of the project.

The variable t_shift is, in minutes, the shift time already mentioned (Section §4.3.2), that is the estimated lag time between the profile of demand of the vehicles of zone 1 and zone 3. The pair *multiplier* and *divisor* are variables used to settle the accuracy of the profile of demands (it increases as *multiplier* grows): as it has been seen previously (Section §4.3.2), in the *profile* file the sum of the elements of each of its matrices have to be equal to *multiplier* (in this case study it has been set to 10^4) so that this number divided by *divisor* (*divisor* = *multiplier*$/10^2$) is equal to 100, as requested in Paramics.

The values of the speed limits assigned to the signals of the control panels which, in effect, do not represent a real speed limit, are stored in the structure $signals$. To these signals (BLK, <–, –>, VGP, -X-, ERE, nvt)[5] has been associated the fictitious speed limit of 10 km/h, except for BLK (no signal on the control panel) and ERE (end of restrictions) to which has been associated the speed limit of 120 km/h, usually adopted on the Dutch freeways. This ploy has been used because it is necessary to associate a numerical value to each signal in order to create a valid *speedlimits* matrix (Section §4.2.2); the speed limit of 0 km/h has been avoided because vehicles would have stopped. This should not be a major issue, since the chosen scenario is not characterized by incidents and road works that could have caused the displaying of such signals, but it could be useful for future considerations.

[5]The arrows suggest the driver, respectively, to move from the current lane to the left or the right one; VGP and -X- indicate that the lane is, respectively, going to be opened or closed; nvt indicates a fault connection between the control panel and the relative detector loop.

Optimization Routine Parameters

The parameters involved in the characterization of the optimization routines and in the computation of the objective function are subsequently described.

In order to investigate more accurately the objective function, several calibration runs will be executed, each being associated with one of the n_s randomly generated starting point $\boldsymbol{x_0}$.

Another important parameter is the number of simulation runs n_r to be executed in order to compensate the fact that the model is stochastic: it means that the objective function is calculated using the outputs of n_r simulation runs. Since a large value of n_r would have implied a very long operating time for the whole algorithm, a small value has been chosen ($n_r = 5$), based on the experience and the insight gained from several training runs.

If the variable *seed* (Section §4.2.3) is set equal to 0, the seed used by the random number generator will be chosen randomly for each Paramics simulation run. In this case (i.e., a random seed has been selected), more simulation runs ($n_r \geq 2$) should be executed before the results could be statistically significant. Otherwise, if the variable *seed* is set equal to 1 (or to another positive integer), all the simulation runs will have the same seed, and their outputs will be the same if the other parameters do not change: so, for each fixed set of parameters, only one Paramics simulation run ($n_r = 1$) is requested.

The variables $data\_T_1 = 5$, $data\_T_2 = 7$ and $data\_T_3 = 3$ (analogue to those already illustrated in Section §5.1.3) correspond, respectively, to the measurements of total flow (veh/h), average speed (km/h) and total density (veh/km), and they are used to select and to draw these data from the output file of each simulation, so that they could be compared with the actual measurements in order to calculate the objective function.

By means of the variable *t_rise*, the time (in minutes) during which the model is deemed to reach the full performance state, is defined. In the case study, this value has been estimated as $t\_rise = 15$, as this period considers both t_shift and the time needed by the vehicles leaving the gas station to reach the last detector loop (i.e., the end of the stretch). The objective function is computed using only the measurements obtained after the first *t_rise* minutes of the simulation.

Regarding the optimization algorithm, selected in the string *method*, it is possible to define the tolerance on the calibrating parameters *tol_x* and the tolerance on the objective function tol_fun. The variable *cds* defines the number of decimal digits for the calibrating parameters. At last, $\boldsymbol{l_b}$ and $\boldsymbol{u_b}$ are the arrays in which are defined the lower and upper bounds, respectively, for the calibrating parameters. The values for these variables could change, depending on the desired effort for each simulation, as it will be explained later.

5.2.2 Calibration Parameters

Due to the experimental purpose of this project, only few parameters could have been chosen for the calibration process. Amongst several of these, the mean headway time and the mean reaction time have been selected as the calibrating parameters, since they turn out to be the most relevant ones in the car-following and the lane-changing behaviors, and the outputs of the Paramics simulation depend mostly by them [36].

In fact, the mean headway time is the mean temporal distance between two consecutive vehicles, depending by the characteristics assigned to the following vehicle[6]; therefore, it regulates the distance between next vehicles and also their current lane-changing strategy, since vehicles decide to move to another lane when they desire to overtake a slower preceding vehicle and when, in the overtaking lane, the other vehicles are sufficiently distant. The mean reaction time contributes in the car-following behavior since it can regulates the accurate simulation of the backward travelling shock waves[7].

In the calibration and validation processes, the mean headway time and the mean reaction time have been respectively called x_1 and x_2, because they turn out to be the variables, stored in the vector $\boldsymbol{x}$, of the optimization problem.

5.2.3 Preparing the Optimization Process

After all the requested parameters have been assigned to the variables of the algorithm, the routine *cv_record* creates the files in which the main variables and the outputs of the Paramics simulations will be stored; it is characterized by the

[6]For a particular vehicle and in certain zones of the traffic network, the mean headway time could be different from the selected value: in fact, it could be multiplied by a factor f (usually $0.5 \leq f \leq 1.8$). As an example, if the vehicle is a heavy one, then $f = 1.8$, or if the vehicle is passing through a road where manteinance works are being executed in a lane, then $f = 1.5$. Generally speaking, it can be said that the mean headway time regulates the aggressiveness and the awareness of the vehicles.

[7]Observations on shock waves have shown that they travel upstream at, approximately, the speed of 11 km/h.

establishment of a unambiguous string, which recall the instant during which the simulation has been started. For example, the string *20070703_093211* will be used within the name of the files (also figures and graphics) concerning the simulation started at 09:32:11 of July 3rd, 2007. This routine returns the file identifier fid_rec of the record file, in which is printed the information regarding the actual algorithm run (the calibration set, the validation set, the initial instant and the duration of the simulation, etc.), the origin-destination matrix, the profiles of demands, the options of the used optimization routine, the data for each evaluation of the objective function (the value of $\boldsymbol{x}$, the same objective function and various informations about it), and finally the same kind of data for the validation process (except the objective function part).

Finally, the matrices $\boldsymbol{Y_1}$ and $\boldsymbol{Y_2}$ will be used for the creation of the files *speedlimits.mat*, *demands* and *profile*, respectively in the routines *cv_speedlimits*, *cv_demands* and *cv_profile*; at the same time, the file *matrix* is created too.

5.2.4 Optimization Functions

The field is now ready for the call of the optimization function. Amongst those available on Matlab, it has been decided to solve the calibration problem using one of the following three: *fmincon*, *ga* or *patternsearch*. All these functions are able to solve problems of the form:

$$\begin{array}{ll} \text{minimize} & z = f(\boldsymbol{x}) \\ \text{subject to:} & \boldsymbol{Ax} \le \boldsymbol{b} \\ & \boldsymbol{A_{eq}x} = \boldsymbol{b_{eq}} \\ & \boldsymbol{C}(\boldsymbol{x}) \le \boldsymbol{0} \\ & \boldsymbol{C_{eq}}(\boldsymbol{x}) = \boldsymbol{0} \\ & \boldsymbol{l_b} \le \boldsymbol{x} \le \boldsymbol{u_b} \end{array}$$

in which linear and nonlinear constraints appear.

Other available optimization routines do not fit with this kind of problems: as an example, *fminbnd* can find the minimum of a single-variable function (in this case study there are two variables), *fminsearch* and *fminunc* can find the minimum of a unconstrained multivariable function (in this case study some boundaries will be applied to the variables), *fgoalattain* and *gamultiobj* can solve multiobjective goal attainment problems (this case study features a single objective function), while *bintprog*, *linprog* and *quadprof* can solve, respectively, binary integer, linear and quadratic programming problems (the case study does not belong to these categories).

Since it is not necessary to specify linear and nonlinear constraints for this optimization problem, the matrices $\boldsymbol{A}$ and $\boldsymbol{A_{eq}}$, the vectors $\boldsymbol{b}$ and $\boldsymbol{b_{eq}}$, and the functions $\boldsymbol{C}(\boldsymbol{x})$ and $\boldsymbol{C_{eq}}(\boldsymbol{x})$ will not be defined; only the lower and the upper boundaries of the variables x_1 and x_2 will be applied. These values are, for both parameters, $l_b = 0$ and $u_b = 3$, since the mean headway time and the mean reaction time cannot be negative and, as their default value is 1, a reasonable upper limit has been set. The optimization functions accept, as a starting point, the vector $\boldsymbol{x_0}$, whose components are randomly selected from a uniform distribution in the state space (i.e., between l_b and u_b). They also accept an *options* structure, in which many parameters could be set. The handle routine, used by the three optimization functions, is called *cv_paramics* and will be later explained in detail.

Constrained Nonlinear Optimization

The constrained nonlinear optimization (or nonlinear programming) in Matlab [27] is performed by means of the routine *fmincon*, which attempts to find a constrained minimum of a scalar function of several variables, starting at an initial estimate. It uses a sequential quadratic programming method, where the function solves a quadratic programming subproblem at each iteration.

Some options are selected by the *optimset* routine: the large-scale algorithm has been unselected in favor of the medium-scale one, the maximum number of function evalutations allowed has been set to 200, and the termination tolerance on the function value and on the vector of the variables have been set, respectively, to 0.001 and 0.01. The outputs of the *fmincon* routine consist of the array of calibrated parameters $\boldsymbol{x}$, of the objective function evalutated at $\boldsymbol{x}$, of the exit flag (a number that identifies the reason the algorithm terminated), and finally of a structure, whose fields are the number of iterations taken, the number of function evaluations, and the type of algorithm used.

Genetic Algorithm

The genetic algorithm [27] is the second optimization routine tested. It repeatedly modifies a population of initial solutions; at each step, it randomly selects individuals from the current population to be parents, and uses them to produce the children for the next generation. Over successive generations, the population evolves toward an optimal solution. The genetic algorithm does not use derivatives to detect descent in its minimization steps, and so it is a good choice for nondifferentiable problems. Because of the nature of the search done by the genetic algorithm, it is also often very effective at finding a good approximation of the global minimum.

Unlike nonlinear programming and pattern search, the genetic algorithm does not need a starting vector $\boldsymbol{x_0}$, but it simply needs to know the number of variables it has to deal with. Amongst the options, the termination tolerance on the fitness function value has been set to 0.001, and the time limit for the optimization process has been set to 10 hours. It has been possibile to set the tolerance on the variables x_1 and x_2 to 0.01 thanks to the functions *cv_int_pop* and *cv_int_mutation*: the first one is used to set a starting population composed by integer individuals; the second one allows the genetic algorithm to generate integer children. Both the starting population and the children are then divided by 100 in order to obtain individuals with two decimal digits. The outputs of the algorithm are the calibrated vector $\boldsymbol{x}$, the objective function, a string that explains the reason that caused the termination of the optimization algorithm, and a structure that contains informations about the number of generations computed, the number of evaluations of the fitness function and eventually the maximum constraint violation.

Pattern Search

Pattern search [27] is an alternative to the genetic algorithm, as it is often computationally less expensive and can minimize the same types of functions. Pattern search operates by searching a set of points called a pattern, which expands or shrinks depending on whether any point within the pattern has a lower objective function value than the current point. The search stops after a minimum pattern size is reached. Like the genetic algorithm, the pattern search algorithm does not use derivatives to determine descent, and so it works well on non-differentiable, stochastic, and discontinuous objective functions. Pattern search is also effective at finding a good approximation of the global minimum because of the nature of its search.

Also for the *patternsearch* routine, some options have been selected, as the termination tolerance on the objective function (0.001) and on the vector $\boldsymbol{x}$ (0.01), as well as the maximum number of function evaluations allowed (200).

5.2.5 Objective Function

The objective function is calculated inside the function *cv_paramics*, which is the handle function for the optimization routine. The selected objective function is:

$$z(\boldsymbol{x}) = \frac{\sum_{i=1}^{m}\sum_{j=m_r}^{n}\left(f_{i,j}-\hat{f}_{i,j}\right)^2}{\sum_{i=1}^{m}\sum_{j=m_r}^{n}\hat{f}_{i,j}^2} + \frac{\sum_{i=1}^{m}\sum_{j=m_r}^{n}\left(v_{i,j}-\hat{v}_{i,j}\right)^2}{\sum_{i=1}^{m}\sum_{j=m_r}^{n}\hat{v}_{i,j}^2}$$

where m is the number of detector loops on the network, m_r is the number of minutes after which the vehicles are considered spreaded over the whole model, n is the number of minutes of the simulation, $f_{i,j}$ and $\hat{f}_{i,j}$ are the actual flow and the simulated flow of the vehicles at the detector loop i at the minute j, and $v_{i,j}$ and $\hat{v}_{i,j}$ are the actual speed and the simulated speed.

Two kinds of measurements have been chosen in order to calculate the objective function: the mean flow and the mean speed, so that the calibration of parameters could be more accurate. This means that the objective function could be divided into two factors, one regarding the mean flow and the other concerning the mean speed; given this configuration, it is possible to evaluate the weight of each measurement within the objective function. In order to avoid large differences between the two factors (usually, a flow value is one deep order of magnitude higher than a speed one), the sum of the squares of the differences between the actual and the simulated values is divided by the sum of the squares of the simulated values, so that both parts could be at about the same order of magnitude.

Before the computing of the objective function, n_r simulation runs have to be executed. In order to do that, a *configuration* file has to be compiled for each run: this file will contain all the needed parameters. In this case study, most of them are fixed and their value has been left equal to the default one, while some could change at every run: the mean headway time x_1, the mean reaction time x_2, and *speed_memory*. The latter, in fact, depends directly by the mean reaction time, and should be equal at least at the 150% of this parameter (expressed in time steps). Sometimes it could happen that the mean reaction time, during the iterations of the optimization routine, is set to 0, which implies that also *speed_memory* is 0. To avoid this improper circumstance, this parameter is immediately set equal to 1 when it is recognized to be equal to 0.

As soon as the *configuration* file is ready, n_r simulation run will be executed while the Matlab plugin [30] for Paramics will gather all the data and store them in the *detector.mat* file. After each simulation run, some data are extracted in order to build the matrices $\boldsymbol{T_1}$ and $\boldsymbol{T_2}$, which will contain the simulated mean flow and the simulated mean speed. Now it is finally possible to calculate the objective function concerning the last run. When all the n_r simulation runs will be executed, the effective objective function (the partial output of the optimization routine) is calculated as the arithmetic mean between the n_r momentary objective functions. Then, these data are printed in the record file of the whole simulation and are stored in a save file, so that it will be possible, at the end of the calibration process, to draw the trend of the objective function as the optimization algorithm runs.

The *cv_paramics* routine will be repeated at each evaluation of the objective function until the optimization algorithm terminates for whatever reason. If the optimization has yielded a valid result, the calibrated parameters x_1 and x_2 will undergo the validation process.

5.2.6 Validation

The validation process uses the optimal parameters (the result of the calibration process) to simulate a new traffic scenario and to compare the results of this simulation with the real measurements available from the validation scenario: the goal of the validation process is to determine the validity of the calibrated parameters.

Therefore, as soon as the array $\boldsymbol{x}$ of calibrated parameters is available, the process of validation starts. It will test whether the calibrated parameters are suitable for another scenario.

After the setting of the parameters (they are the same used in the calibration section, except for these that specify the validation scenario) and the creation of the simulation files (*speedlimits.mat*, *demands*, *profile*, *matrix*), a fixed *configuration* file is set, since the mean headway time and the mean reaction time are the values already obtained from the calibration process.

Finally, a sequence of n_r Paramics simulation runs will be executed. For each simulation run, Theil's inequality coefficient U [19] is calculated:

$$U = \frac{\sqrt{\frac{\sum_{i=1}^{m}\sum_{j=m_r}^{n}\left(f_{i,j}-\hat{f}_{i,j}\right)^2}{m(n-m_r+1)}}}{\sqrt{\frac{\sum_{i=1}^{m}\sum_{j=m_r}^{n}(f_{i,j})^2}{m(n-m_r+1)}}+\sqrt{\frac{\sum_{i=1}^{m}\sum_{j=m_r}^{n}\left(\hat{f}_{i,j}\right)^2}{m(n-m_r+1)}}} + \frac{\sqrt{\frac{\sum_{i=1}^{m}\sum_{j=m_r}^{n}(v_{i,j}-\hat{v}_{i,j})^2}{m(n-m_r+1)}}}{\sqrt{\frac{\sum_{i=1}^{m}\sum_{j=m_r}^{n}(v_{i,j})^2}{m(n-m_r+1)}}+\sqrt{\frac{\sum_{i=1}^{m}\sum_{j=m_r}^{n}(\hat{v}_{i,j})^2}{m(n-m_r+1)}}}$$

where the variables are the same listed in Section §5.2.5.

As the objective function, also the fitness coefficient is composed of two factors, one regarding the mean flow and the other concerning the mean speed. This distinction could allow the engineer to evaluate which facet needs more effort in order to improve the accuracy of the traffic network model.

Theil's inequality coefficient may vary from 0, which means a perfect fit between actual and simulated measurements, to 1 in the opposite condition (in this case study the reachable maximum value of U is 2 because the coefficient is composed of two factors, and each of them could reach the unitary maximum value).

5.3 Cross-Validation

A more comprehensive procedure is the algorithm for *cross-validation*: it provides a stricter test-bench for the calibrating parameters. In fact, given a set of n traffic scenarios, each of these will be used as calibration set and its calibrated parameters will be validated by means of the remaining $n-1$ scenarios. After this process has been successfully performed for all the n possible combinations, a final array of parameters will be estimated from the initial n sets of calibrated and validated parameters.

In this case study, the algorithm for cross-validation (Figure 5.6) consists in the implementation of the dual sampling scheme [38]: as two scenarios have been selected, the algorithm for calibration and validation (Section §5.2) is performed using one scenario as the calibration set and the other one as the validation set, in order to obtain a first array of calibrated parameters $\boldsymbol{x_A}$. The procedure is repeated after the calibration set has been swapped with the validation set, and another array of parameters, $\boldsymbol{x_B}$, is obtained. If both procedures are accomplished (i.e., both values of Theil's inequality coefficients are satisfactory), a final estimate of the mean reaction time and the mean headway time could be drawn from the previous obtained values:

$$\boldsymbol{x} = \frac{\boldsymbol{x_A} + \boldsymbol{x_B}}{2}$$

In order to assure a completely validated traffic network model, the cross-validation process should be executed with several scenarios that range within a wide variety of traffic conditions.

5.4 Calibration with Simulated Data

The last presented algorithm is the so-called algorithm for *calibration with simulated data*, that is a variation of the algorithm for *calibration and validation*, already seen in Section §5.2. It is an appropriate tool for researching the effectiveness of the various optimization routines, in order to focus the greatest computational effort on the most effective amongst them.

The algorithm works as follows (Figure 5.7): after the data of the calibration scenario have been loaded and the measurements requested for executing a simulation run have been extracted, a single Paramics simulation run is executed with

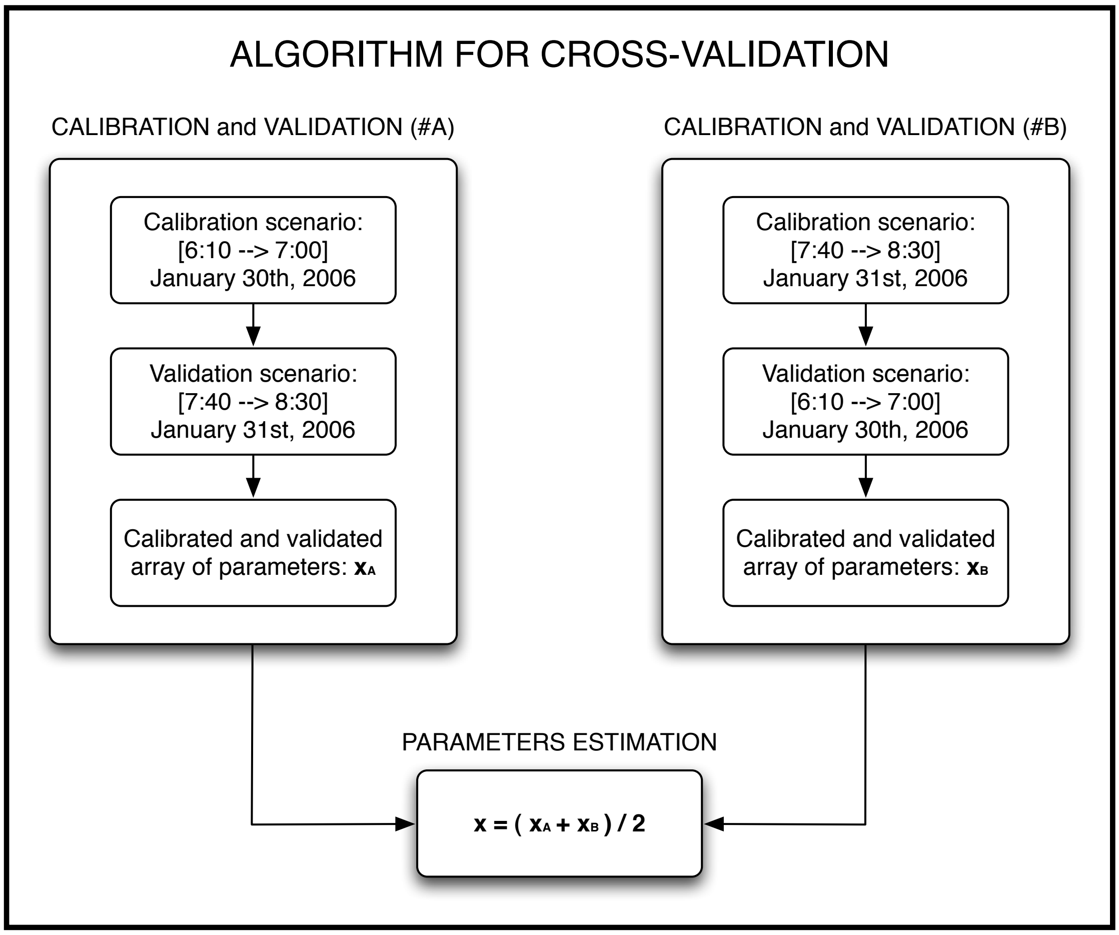

ALGORITHM FOR CROSS-VALIDATION
CALIBRATION and VALIDATION (#A)
Calibration scenario: [6:10 --> 7:00] January 30th, 2006
Validation scenario: [7:40 --> 8:30] January 31st, 2006
Calibrated and validated array of parameters: xA
CALIBRATION and VALIDATION (#B)
Calibration scenario: [7:40 --> 8:30] January 31st, 2006
Validation scenario: [6:10 --> 7:00] January 30th, 2006
Calibrated and validated array of parameters: xB
PARAMETERS ESTIMATION
x = (xA + xB) / 2

Figure 5.6: Flow chart of the *cross-validation* procedure.

known parameters (i.e., $\boldsymbol{x} = \boldsymbol{x_0}$) and its outputs (flow and speed) are stored aside.

At last, the optimization process starts and yields the calibrated values of mean headway time and mean reaction time. Since the objective function is computed by comparing the outputs of the simulation with known parameters $\boldsymbol{x_0}$ with the outputs of the simulations executed during the optimization process, the global minimum $z = 0$ is obtained when a positive integer k, such that $\boldsymbol{x_k} = \boldsymbol{x_0}, exists$[8]: this is the main difference from the ordinary algorithm for calibration. So, it is possible to investigate how much a given optimization algorithm (*fmincon*, *ga* and *patternsearch*) could be able to approach the known parameters $\boldsymbol{x_0}$, both in terms of accuracy and elapsed time; this process, therefore, suggests upon which

[8]The global minimum could also be obtained for a different value of $\boldsymbol{x_k}$, but in this case study it is very unlikely to happen.

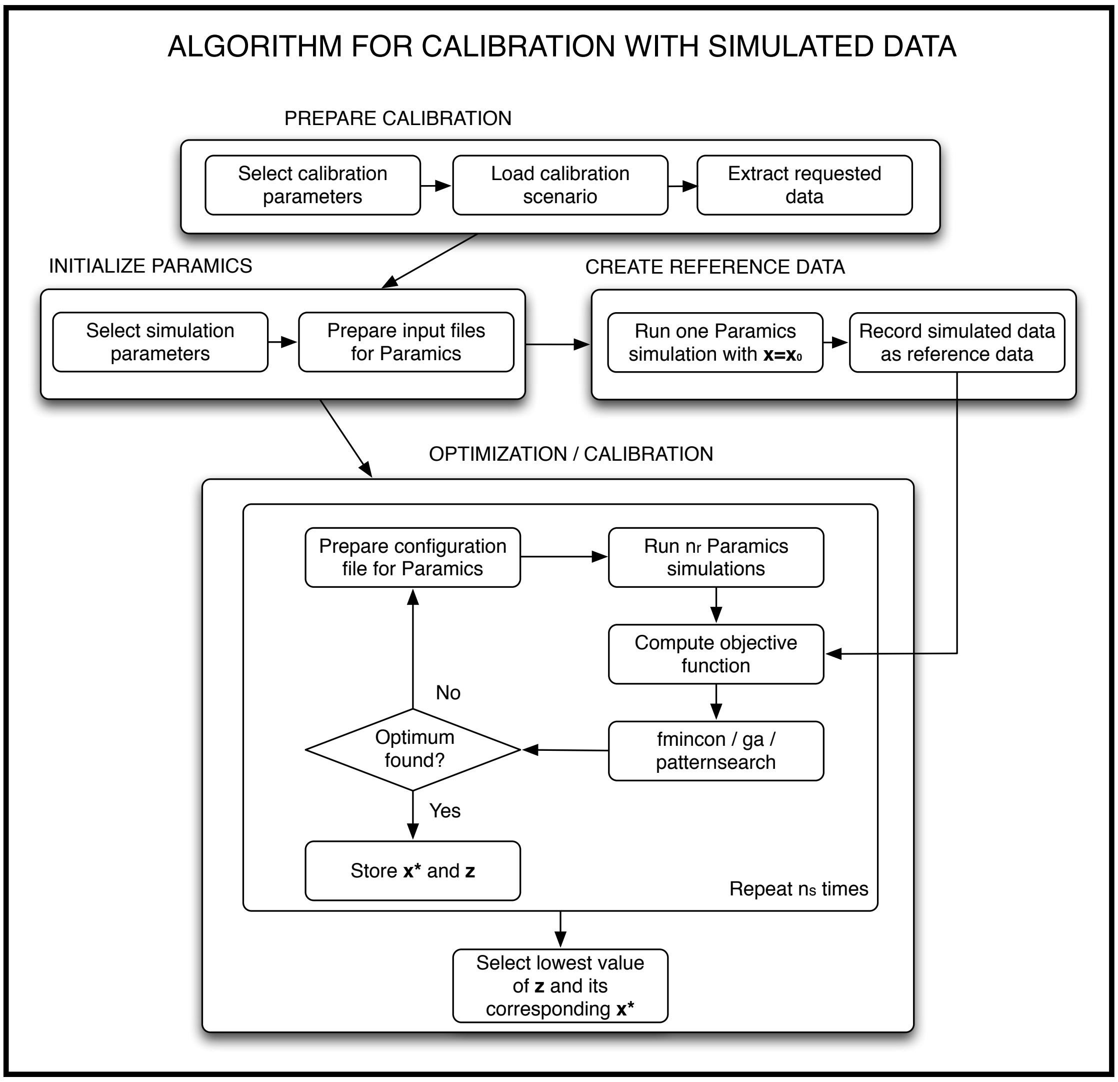

ALGORITHM FOR CALIBRATION WITH SIMULATED DATA
PREPARE CALIBRATION
Select calibration parameters
Load calibration scenario
Extract requested data
INITIALIZE PARAMICS
Select simulation parameters
Prepare input files for Paramics
CREATE REFERENCE DATA
Run one Paramics simulation with x=x0
Record simulated data as reference data
OPTIMIZATION / CALIBRATION
Prepare configuration file for Paramics
Run nr Paramics simulations
Compute objective function
fmincon / ga / patternsearch
Optimum found?
No
Yes
Store x* and z
Repeat ns times
Select lowest value of z and its corresponding x*

Figure 5.7: Flow chart of the *calibration with simulated data* procedure.

optimization routine the main efforts of time and resources should be focused.

5.5 Summary

In this chapter the designed algorithm for *calibration and validation* has been presented. First of all, the reasons regarding the selection of the traffic scenarios have been explained; then, the choice of the simulation and calibration parameters is commented, as well as the instructions that precede the optimization routine. Three Matlab functions for the optimization (*fmincon*, *ga* and *patternsearch*) are illustrated, and then it is explained how the objective function is calculated. Next, the algorithm for validation is shown. Finally, a quick explanation of the algorithms of *cross-validation* and *calibration with simulated data* is given.

Chapter 6

Results

In this section the results accomplished by the execution of various simulations[1] will be presented.

The first sequence of simulations consists in the testing of the following Matlab optimization functions: *fmincon*, *ga*, and *patternsearch*. In fact, the algorithm for *calibration with simulated data*, applied to these functions, is able in determining which, amongst these, is/are the best suited for the calibration process. The second set of simulations performs the process of *calibration and validation*, using the routine(s) selected from the previous simulation runs as the optimization function(s). The third block of simulations is a test of the *cross-validation* algorithm.

For each optimization function, two main kinds of simulations could be run. The first is performed with the use of a fixed seed for the simulation run; it means that, if all the parameters set in the *configuration* file are not changed, each simulation run will yield the same outputs: therefore, for the evaluation of the objective function only one simulation run is needed. Hence, amongst the parameters of the simulation, there will be $seed = 1$ and $n_r = 1$. The second type of simulation, in contrast with the first, is characterized by the use of a random seed for each simulation run; then, it is necessary to run multiple simulations before evaluating the objective function. So, the concerning parameters will be $seed = 0$ and $n_r = 5$.

Then, the tolerances on the variable $\boldsymbol{x}$ and on the objective function, that are some of the parameters requested by the optimization routines, have to be evaluated. In order to evaluate which value should be used for the tolerance on the variables of the calibration problem, the shape of the objective function, calculated

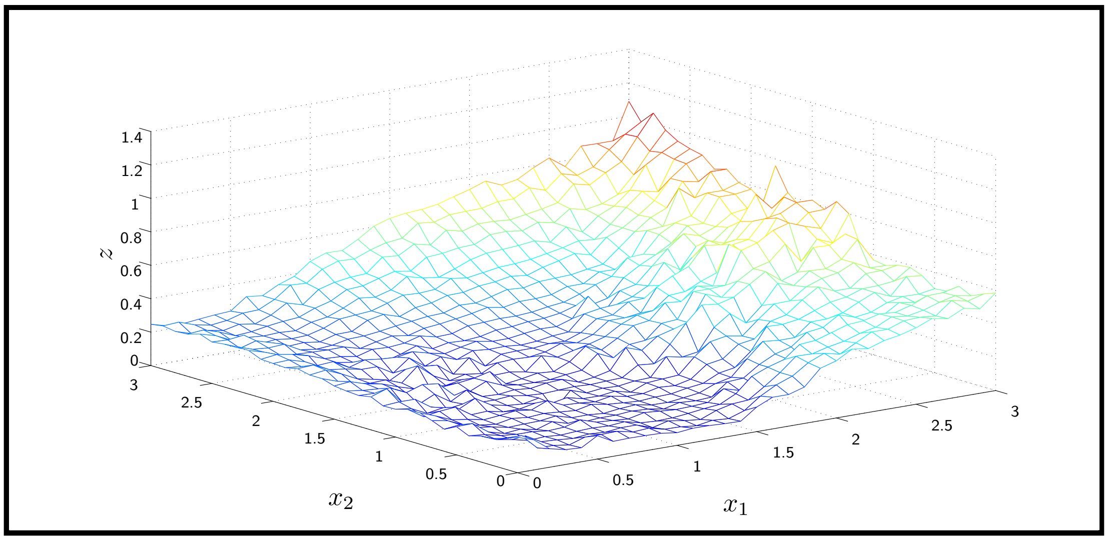
[1]All the simulations have been run on Dagorlad, a shared machine located at the Delft Center for Systems and Control, whose processor is an Intel®Pentium®4 at 3.20 GHz, with a main memory of 1 GHz.

using simulated data as reference, with $\boldsymbol{x} = [1.00, 1.00]$ as the optimum, has been plotted. The plot of the objective function on the whole state space shows how it decreases until it reaches a wide and flat zone (Figure 6.1(a)); adjacent values of x_1 (or x_2) differ of a step equal to 0.1. Shortening the plot in the nearby of the optimum and evaluating the objective function in the interval from 0.9 to 1.1 for both variables (Figure 6.1(b)), the shape appears to be nonsmooth, as each point seems to represent a local maximum or minimum. Further reducing the interval from 0.99 to 1.01 (Figure 6.1(c)), the plot shows a behavior similar to the previous one; it means that the computational noise (Section §3.2.3) is altering the shape of the objective function from smooth to nonsmooth. Therefore, it means that there is no significant difference if the optimization routine looks for the optimum by means of variables with two or three decimal digits: for this reason, the tolerance on the variables x_1 and x_2 has been set equal to $tol\_x = 1 \cdot 10^{-2}$. Concerning the tolerance on the objective function, it has been set to the value $tol\_fun = 1 \cdot 10^{-3}$, since some trial simulation runs have suggested that searching for a bigger accuracy on the minimum would have simply increased the length of the optimization process, in the absence of any significant improvement on the results.

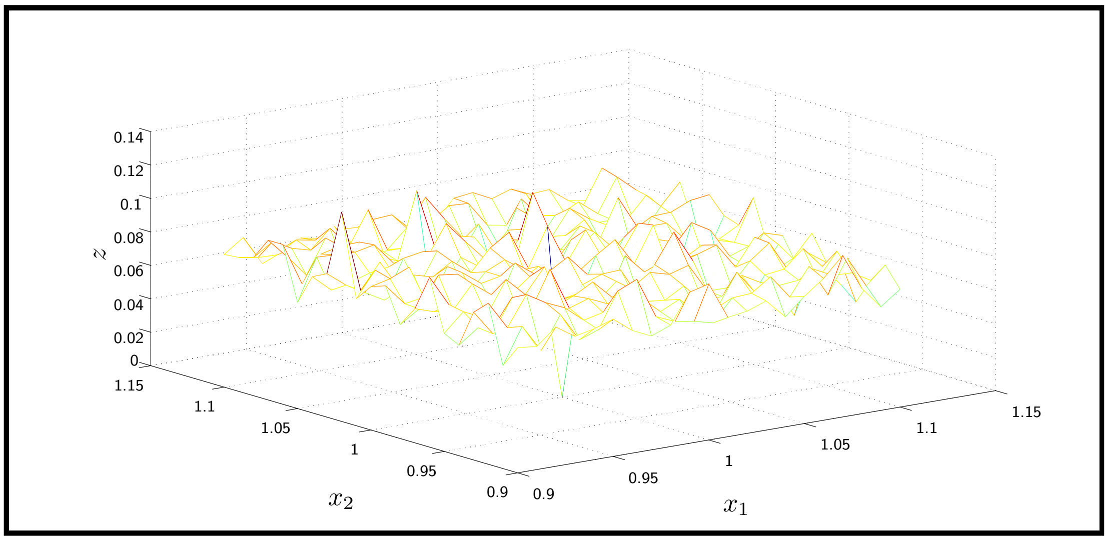
6.1 Calibration with Simulated Data

The algorithm for *calibration with simulated data* (Section §5.4) yields a first opinion about the efficiency and the effectiveness of the optimization algorithm in solving a problem of this sort. Therefore, it is possible to observe which optimization routine could (eventually) converge to an optimum solution, which one is the faster, and which one yields the best tradeoff between accuracy and rapidity.

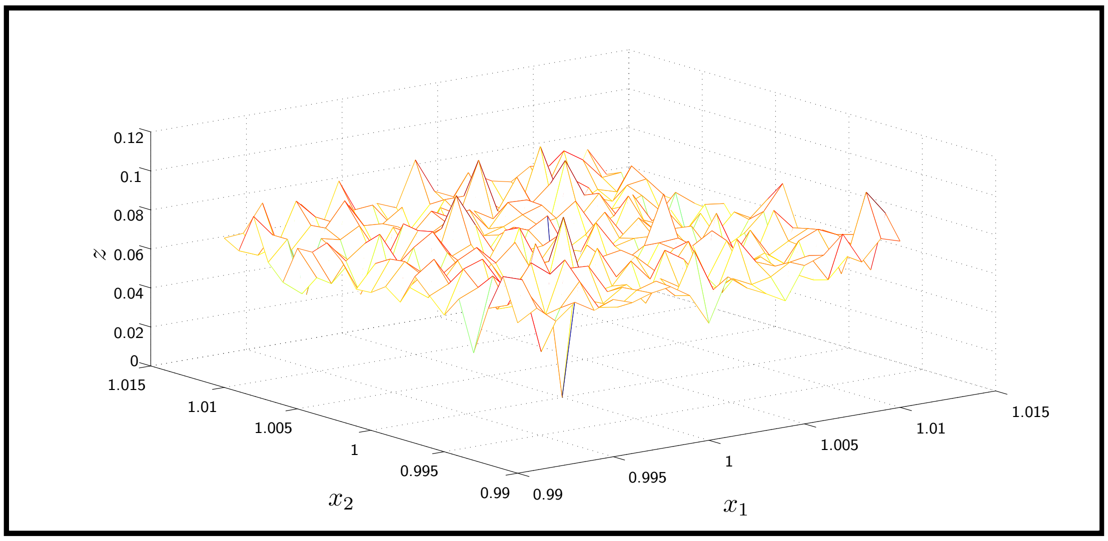
The reference network represented by the demands and the profiles extrapolated from the data of the calibration set is replaced by the network characterized by the demands and the profiles obtained from the outputs of a simulation of the network performed using the calibration set. It entails that, given that the outputs of the reference network are themselves the outputs of a Paramics simulation, it should be easier for the optimization routine to find a parameter set that could bring to the achievement of a very low value of the objective function. The selected value for both parameters (mean headway time and mean reaction time), in order to build the reference measurements, is $x_1 = x_2 = 1.00$; to be more precise, they are equal to the Paramics default values for these parameters.



(a) Objective function plotted for $x_1, x_2 \in [0.0 : 0.1 : 3.0]$.



(b) Objective function plotted for $x_1, x_2 \in [0.90 : 0.01 : 1.10]$.



(c) Objective function plotted for $x_1, x_2 \in [0.990 : 0.001 : 1.010]$.

Figure 6.1: Plot of the objective function $z = f(x_1, x_2)$, calculated referring to the measurements obtained by simulating the traffic flow of vehicles with values of mean headway time and mean reaction time equal to $x_1 = x_2 = 1.00$.

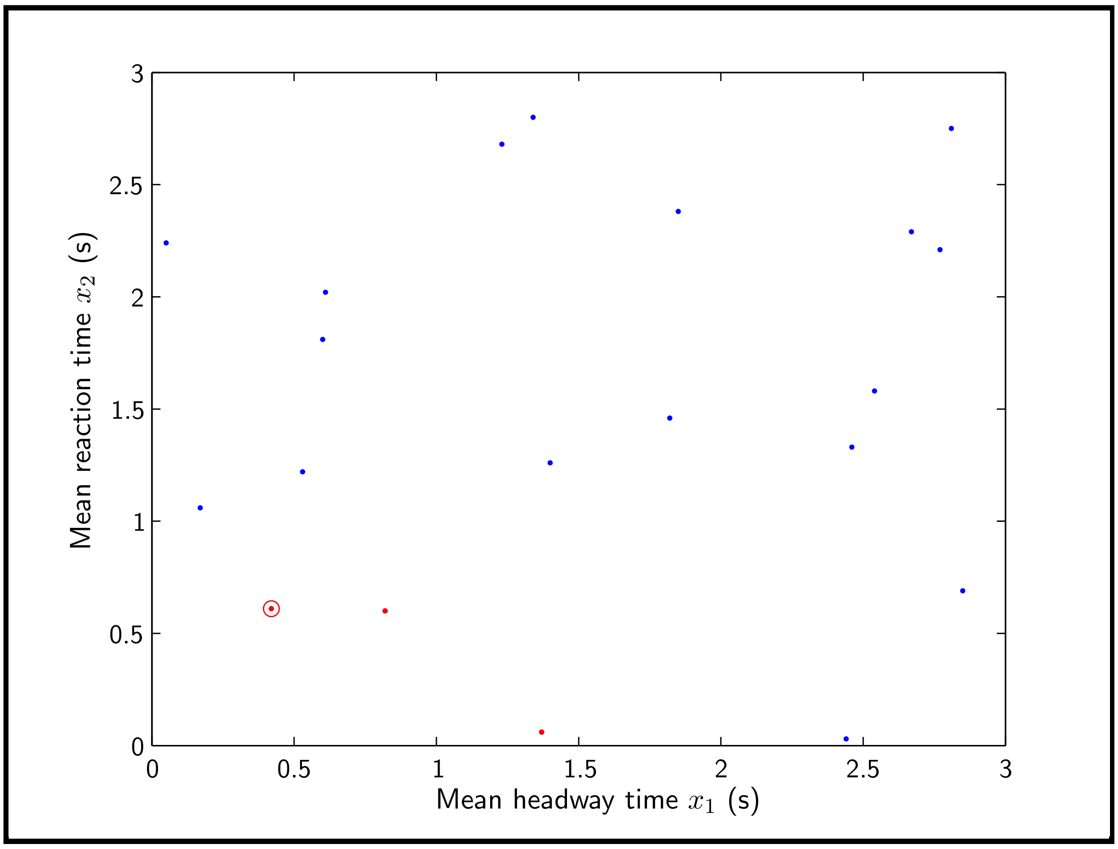


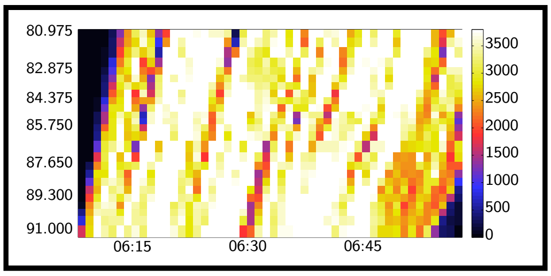
Figure 6.2: Optimum points in the state space for the 20 simulation runs executed with *fmincon* (calibration with simulated data, fixed seed). The global optimum is marked with a red dot and a red circle, while the two other best minima are marked with a red dot.

6.1.1 Simulations with Fixed Seed

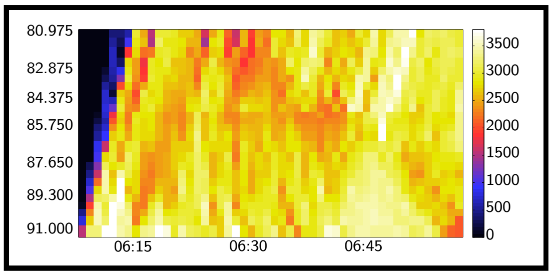
Constrained Nonlinear Optimization

The first optimization function that has been tested is *fmincon*; 20 simulation runs have been executed, but all of these terminated after few minutes at the first iteration, explaining the reason of the termination with the following output message: *first-order optimality measure less than options.TolFun and maximum constraint violation is less than options.TolCon.* Obviously, any improvement on the objective function has been achieved, since the optimum is deemed as the objective function calculated at the starting point[2] (Figure 6.2). It would be worthwhile to

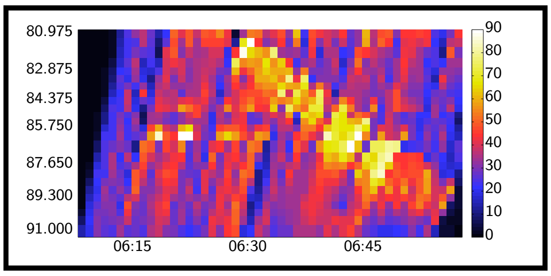
[2]Nevertheless, it could be noticed that the points that yield the lowest values for the objective function stay in the lower left corner of the state space, as anticipated by the plot in Figure 6.1(a).



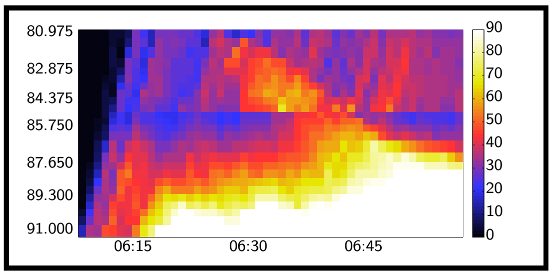
(a) Flow (veh/h).



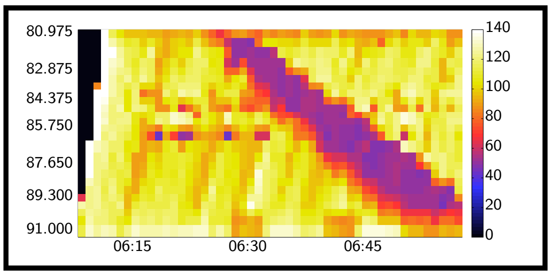
(b) Flow (veh/h).



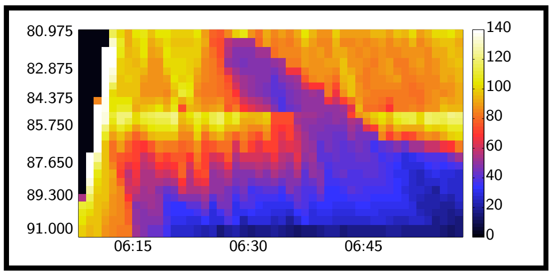
(c) Density (veh/km).



(d) Density (veh/km).



(e) Speed (km/h).



(f) Speed (km/h).

Figure 6.3: Simulated measurements for $\boldsymbol{x} = [1.00, 1.00]$ (reference measurements, left) and $\boldsymbol{x} = [2.85, 0.69]$ (one of the optimum found with the *fmincon* optimization algorithm, right).

decrease the tolerance on the objective function and to set a value for the tolerance on the constraints but, due to the short time available for the project, this will not be done.

However, it could be interesting to show (Figure 6.3) the chromatical difference between the mean flow, the density and the speed of the vehicles on the network for both the reference measurements (simulated with $\boldsymbol{x} = [1.00, 1.00]$) and, as an example, the measurements from the first simulation run ($\boldsymbol{x} = [2.85, 0.69]$).

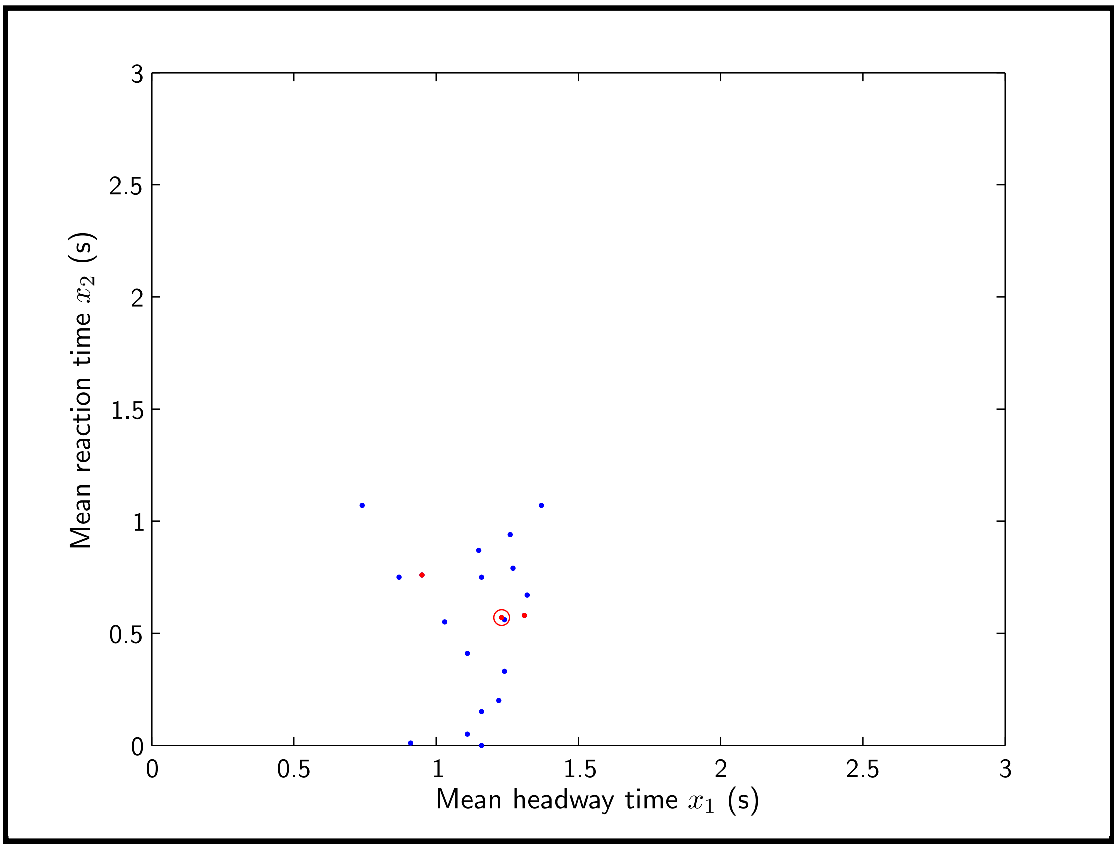


Figure 6.4: Optimum points in the state space for the 20 simulation runs executed with *ga* (calibration with simulated data, fixed seed). The global optimum is marked with a red dot and a red circle, while the two other best minima are marked with a red dot.

Genetic Algorithm

Also the *ga* optimization routine has been tested with 20 different starting points. The global optimum achieved is $z = 0.0535$ in $\boldsymbol{x} = [1.23, 0.57]$, and it has been obtained only once; however, the real optimum $z = 0$, achievable in $\boldsymbol{x} = [1.00, 1.00]$, has not been found, though it stays within the small area where most of the other optima have been found (Figure 6.4).

Most of the simulation runs (60%) are terminated (*Optimization terminated: average change in the fitness value less than options.TolFun*) after 51 iterations and so 1020 evaluations of the objective function (since the population is composed of 20 individuals), while the remaining 40% terminates because the stall time limit (10 hours) has been reached (Table 6.1). By the way, the average number of iterations is 48.05, and 961 is the average number of evaluation of the objective

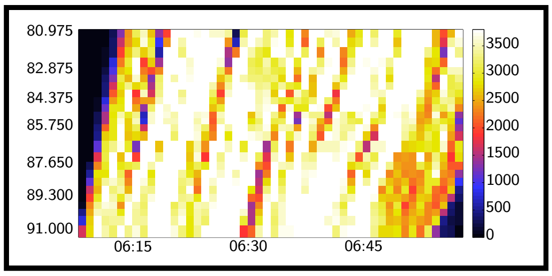
Run	$x_{0,1}$	$x_{0,2}$	x_1	x_2	z	N.It.	N.F.E.
1	2.51	0.06	0.95	0.76	0.0546	50	1000
2	1.32	0.28	1.11	0.41	0.0553	40	800
3	0.46	1.45	1.32	0.67	0.0570	51	1020
4	0.12	2.66	0.87	0.75	0.0570	51	1020
5	1.24	2.19	1.24	0.33	0.0642	51	1020
6	2.80	1.50	0.74	1.07	0.0633	51	1020
7	0.82	2.42	1.27	0.79	0.0589	48	960
8	1.62	1.81	1.16	0.00	0.0561	44	880
9	0.66	0.84	1.22	0.20	0.0593	51	1020
10	1.30	0.40	1.31	0.58	0.0550	41	820
11	2.40	0.64	1.26	0.94	0.0569	47	940
12	1.98	2.89	1.16	0.75	0.0573	51	1020
13	1.28	1.73	1.03	0.55	0.0599	51	1020
14	0.22	0.08	1.16	0.15	0.0591	51	1020
15	2.90	0.33	0.91	0.01	0.0634	36	720
16	0.15	2.96	1.15	0.87	0.0607	51	1020
17	2.56	2.12	1.11	0.05	0.0552	51	1020
18	**1.42**	**2.67**	**1.23**	**0.57**	**0.0535**	**51**	**1020**
19	2.06	0.27	1.37	1.07	0.0633	43	860
20	2.95	2.74	1.24	0.56	0.0594	51	1020

Table 6.1: Summary table with the results of the 20 simulation runs executed with *ga* (calibration with simulated data, fixed seed). The bolded row coincides with the simulation run that has yield the global optimum.

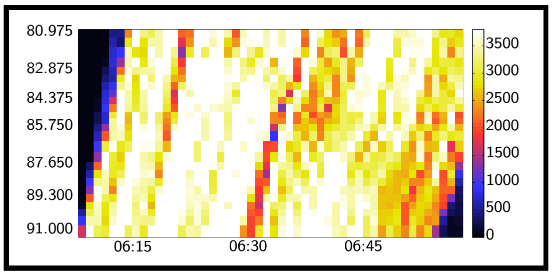
	$x_{0,1}$	$x_{0,2}$	x_1	x_2	z	N.It.	N.F.E.
μ	1.5385	1.5020	1.1405	0.5540	0.0585	48.05	961.00
σ^2	0.8971	1.1063	0.0272	0.1151	$1.05 \cdot 10^{-5}$	21.63	$8.65 \cdot 10^3$

Table 6.2: Mean and variance values for the outputs of the 20 simulation runs executed with *ga* (calibration with simulated data, fixed seed).

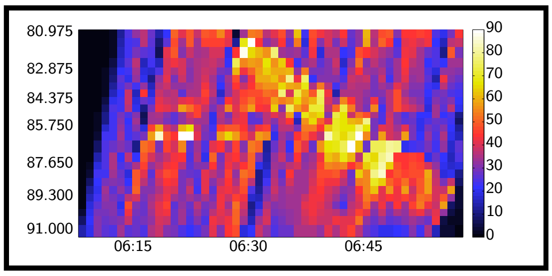
function (Table 6.2). Regarding the calibrating parameters x_1 (mean headway time) and x_2 (mean reaction time), their mean value is respectively $\mu_{x_1} = 1.1405$ and $\mu_{x_2} = 0.5540$, and their variance is $\sigma_{x_1} = 0.0272$ and $\sigma_{x_2} = 0.1151$. Indeed, as shown in Figure 6.4, the optimal values for the mean headway time are more condensed (from about 0.9 to 1.4) than the optimal values for the mean reaction time (from 0 to about 1.1). The values obtained for the objective function are really close to each other, as indicated by the variance: $\sigma_z = 1.05 \cdot 10^{-5}$. Actually, the mean of these values ($\mu_z = 0.0585$) is not so far from the optimum $z = 0.0535$



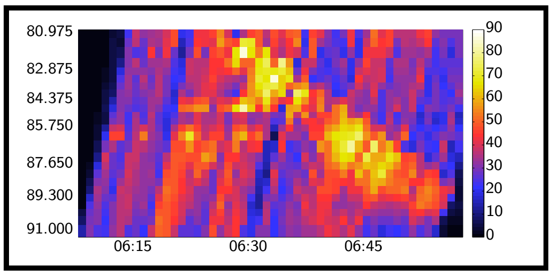
(a) Flow (veh/h).



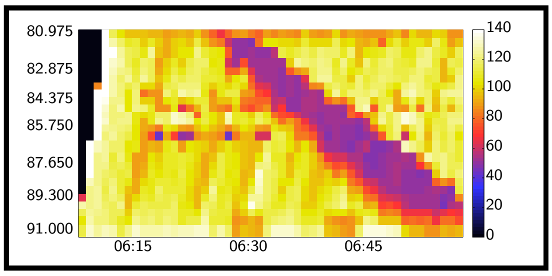
(b) Flow (veh/h).



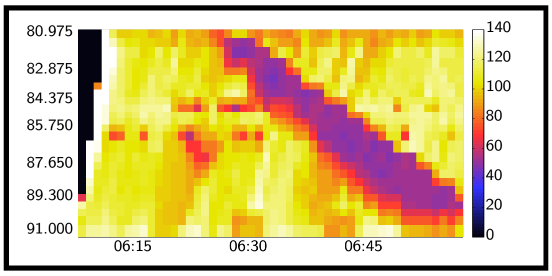
(c) Density (veh/km).



(d) Density (veh/km).



(e) Speed (km/h).



(f) Speed (km/h).

Figure 6.5: Simulated measurements for $\boldsymbol{x} = [1.00, 1.00]$ (reference measurements, left) and $\boldsymbol{x} = [1.23, 0.57]$ (optimum found with the *ga* optimization algorithm, right).

(+1%).

The graphical comparison between the measurements of flow, density and speed of the reference simulation ($\boldsymbol{x} = [1.00, 1.00]$) and of the simulation that yielded the global optimum ($\boldsymbol{x} = [1.23, 0.57]$) shows that qualitatively the behavior of the vehicles in the network is quite similar (Figure 6.5).

Pattern Search

Since, during some trial simulation runs, it has been observed that the *patternsearch* optimization routine reaches earlier than the others a solution, this minimization

Run	x_1	x_2	z	Run	x_1	x_2	z
2	1.32	0.46	0.0526	110	1.11	0.41	0.0553
126	1.26	0.66	0.0529	139	1.11	0.41	0.0553
48	1.15	0.98	0.0537	105	1.19	0.74	0.0555
78	1.04	0.65	0.0541	124	1.07	0.10	0.0558
168	1.10	0.53	0.0543	171	1.07	0.10	0.0558
182	1.10	0.53	0.0543	35	1.27	0.60	0.0561
9	1.31	0.56	0.0546	17	1.34	0.80	0.0561
68	1.24	0.49	0.0550	146	1.34	0.80	0.0561
177	1.24	0.49	0.0550	100	1.07	0.98	0.0563
92	0.96	0.43	0.0551	108	1.21	0.56	0.0565
10	1.11	0.93	0.0551	149	1.21	0.56	0.0565
69	1.11	0.93	0.0551	56	1.13	0.64	0.0566
119	1.10	0.93	0.0551	130	1.30	0.32	0.0567
38	1.11	0.05	0.0552	7	1.33	0.40	0.0567
79	1.11	0.41	0.0553	132	1.33	0.40	0.0567

Table 6.3: Summary table with the best results of the 200 simulation runs executed with *patternsearch* (calibration with simulated data, fixed seed).

algorithm has been intensively tested with 200 different starting points.

The best optimum value that has been found is $z = 0.0526$ in $\boldsymbol{x} = [1.32, 0.46]$, while the real minimum $z = 0$ in $\boldsymbol{x} = [1.00, 1.00]$ has not been found. Since there has been only one recurrence of the optimum (Table 6.3) and one recurrence too for the second, the third and the fourth best values, and since it took about 60 hours for all these simulation to run, it has been decided to execute other 1000 runs with the same parameters. A better optimum value has been found (although it is still not the real optimum $z = 0$) in $\boldsymbol{x} = [1.18, 0.39]$: $z = 0.0509$. Unfortunately, only one simulation run yielded this result, while the previous optimum ($z = 0.0526$) has been found in other four runs, and so on for the other best values of the objective function (Table 6.4). Through the analysis of the distribution of the optima in the state space, it could be seen how the nearly totality of these points stay in the area delimited by $x_1 \in [0.5, 1.5], x_2 \in [0.0, 1.5]$, while the best values stay in a smaller area, delimited by $x_1 \in [1.0, 1.4], x_2 \in [0.3, 1.0]$ (Figure 6.6). This is confirmed too by the variance value of the mean headway time ($\sigma_{x_1} = 0.0561$), that is smaller compared to the one of the mean reaction time ($\sigma_{x_2} = 0.1344$). Regarding the minima values, their average is $\mu_z = 0.0635$, with a variance $\sigma_z = 3.17 \cdot 10^{-5}$ (Table 6.5).

Run	x_1	x_2	z	Run	x_1	x_2	z
335	1.18	0.39	0.0509	414	1.04	0.65	0.0541
2	1.32	0.46	0.0526	614	1.04	0.65	0.0541
338	1.32	0.46	0.0526	814	1.04	0.65	0.0541
538	1.32	0.46	0.0526	1078	1.04	0.65	0.0541
738	1.32	0.46	0.0526	168	1.10	0.53	0.0543
1002	1.32	0.46	0.0526	182	1.10	0.53	0.0543
126	1.25	0.66	0.0529	504	1.10	0.53	0.0543
270	1.25	0.66	0.0529	518	1.10	0.53	0.0543
462	1.25	0.66	0.0529	704	1.10	0.53	0.0543
662	1.25	0.66	0.0529	718	1.10	0.53	0.0543
862	1.25	0.66	0.0529	904	1.10	0.53	0.0543
1126	1.25	0.66	0.0529	918	1.10	0.53	0.0543
263	1.13	0.54	0.0534	1168	1.10	0.53	0.0543
999	1.13	0.54	0.0534	1182	1.10	0.53	0.0543
48	1.15	0.98	0.0537	282	1.15	1.19	0.0545
384	1.15	0.98	0.0537	9	1.31	0.56	0.0546
584	1.15	0.98	0.0537	345	1.31	0.56	0.0546
784	1.15	0.98	0.0537	545	1.31	0.56	0.0546
1048	1.15	0.98	0.0537	745	1.31	0.56	0.0546
78	1.04	0.65	0.0541	1009	1.31	0.56	0.0546

Table 6.4: Summary table with the best results of the 1200 simulation runs executed with *patternsearch* (calibration with simulated data, fixed seed).

	$x_{0,1}$	$x_{0,2}$	x_1	x_2	z	N.It.	N.F.E.
μ	1.5402	1.5037	1.1052	0.6810	0.0635	15.14	38.40
σ^2	0.6665	0.7206	0.0561	0.1344	$3.17 \cdot 10^{-5}$	13.35	92.84

Table 6.5: Mean and variance values for the outputs of the 1200 simulation runs executed with *patternsearch* (calibration with simulated data, fixed seed).

Finally, a comparison between the reference measurements and the simulated measurements for the global optimum ($\boldsymbol{x} = [1.18, 0.39]$) shows how the vehicles cross the network in a similar fashion (Figure 6.7).

Comparison of Simulations with Fixed Seed

Since the *fmincon* optimization routine has not produced acceptable results, it is possible to compare the performances of *ga* and *patternsearch* only.

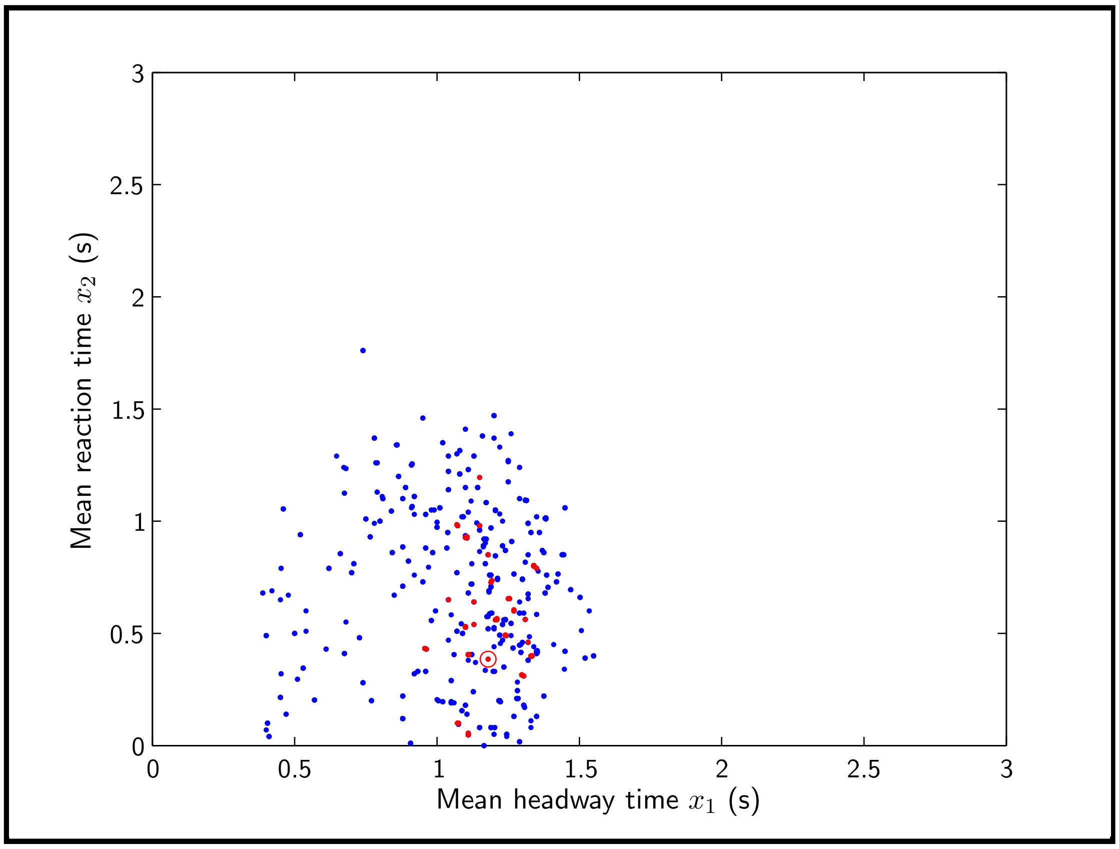
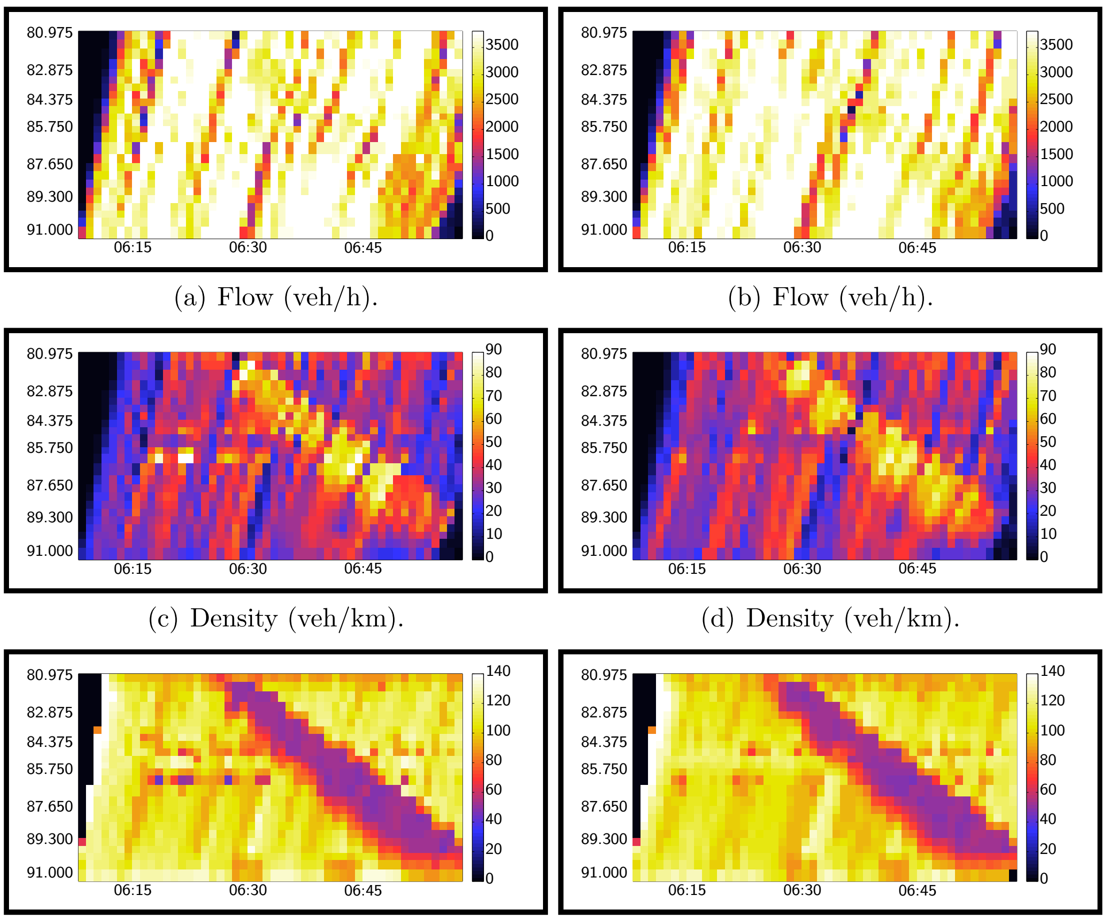


Figure 6.6: Optimum points in the state space for 1200 simulation runs executed with *patternsearch* (calibration with simulated data, fixed seed). The global optimum is marked with a red dot and a red circle, while the other best minima are marked with a red dot.

With regard to Table 6.6, the *ga* optimization routine resulted the most accurate, as it is demonstrated by the lower values of $\sigma^2_{x_1}$, $\sigma^2_{x_2}$ and σ^2_z. Oppositely, in a shortest time it has been possibile to execute several more simulation runs with *patternsearch*, and this has permitted to find a better global minimum, although the mean and variance values of its minima are larger than the ones found with *ga*. In fact, with *patternsearch* it has been possible to execute 1200 simulation runs in about 15 days, while *ga* took about 6 days for executing only 20 simulation runs: it means that *patternsearch* has been about 25 times faster than *ga* in reaching the optimum for a single optimization run.



(a) Flow (veh/h). (b) Flow (veh/h).

(c) Density (veh/km). (d) Density (veh/km).

(e) Speed (km/h). (f) Speed (km/h).

Figure 6.7: Simulated measurements for $\boldsymbol{x} = [1.00, 1.00]$ (reference measurements, left) and $\boldsymbol{x} = [1.18, 0.39]$ (optimum found with the *patternsearch* optimization algorithm, right).

	x_1	x_2	μ_{x_1}	μ_{x_2}	$\sigma^2_{x_1}$	$\sigma^2_{x_2}$
ga	1.23	0.57	1.1405	0.5540	0.0272	0.1151
patternsearch	1.18	0.39	1.1052	0.6810	0.0561	0.1344

	z	μ_z	σ^2_z	$\mu_{n.it.}$	$\mu_{n.ev.}$
ga	0.0535	0.0585	$1.05 \cdot 10^{-5}$	48.05	96.10
patternsearch	0.0509	0.0635	$3.17 \cdot 10^{-5}$	15.14	38.40

Table 6.6: Comparison of the results obtained from the optimization runs executed with *ga* and *patternsearch* (calibration with simulated data, fixed seed).

Run	$x_{0,1}$	$x_{0,2}$	x_1	x_2	z	N.It.	N.F.E.
1	2.80	2.95	0.0000	0.6592	0.1692	4	24
2	2.25	1.86	1.3213	0.0000	0.0777	6	30
3	0.17	1.02	0.8775	1.5150	0.0873	4	89
4	0.72	1.60	0.3544	0.7875	0.0811	3	202
5	2.03	0.66	1.6312	1.8729	0.1869	5	36
6	1.99	0.36	0.0000	0.0000	0.1693	2	26
7	2.56	0.50	0.0000	3.0000	0.2186	2	38
8	1.84	1.91	0.0000	0.5764	0.1607	7	33
9	2.31	1.64	0.0000	0.0000	0.1769	2	19
10	**1.22**	**0.64**	**1.2200**	**0.6400**	**0.0651**	**2**	**202**
11	1.88	2.50	2.0199	2.2210	0.2735	4	28
12	2.29	2.29	1.1869	0.0000	0.0684	7	37
13	2.10	2.45	0.0000	3.0000	0.2433	2	10
14	0.56	0.59	1.2011	0.3403	0.0699	7	202
15	0.63	2.30	0.3949	0.0000	0.0771	5	25
16	2.74	0.69	1.7006	0.8750	0.1113	8	202
17	1.04	2.10	0.7421	1.9162	0.1225	11	122
18	0.05	1.68	0.7491	1.3449	0.0785	8	202
19	1.82	1.35	1.2201	1.5742	0.1106	7	62
20	0.09	2.97	0.0147	0.0000	0.1732	3	12

Table 6.7: Summary table with the results of the simulation runs executed with *fmincon* (calibration with simulated data, random seed). The bolded row coincides with the simulation run that has yield the global optimum.

6.1.2 Simulations with Random Seed

In this section, the results of the simulation runs executed with random seeds are analyzed and, then, compared.

Constrained Nonlinear Optimization

This time it has been possible to obtain useful results from the *fmincon* optimization routine, since it did not terminate at the first iteration. By analyzing the outputs of the 20 simulation runs, the optimum[3] $z = 0.0651$ has been found for $\boldsymbol{x} = [1.22, 0.64]$ (Table 6.7). However, the average of the optima is $\mu_z = 0.1361$,

[3]It has to be remembered that, for the current algorithm for calibration, the objective function is obtained as the average of 5 objective functions calculated with the same mean headway time and mean reaction time but different seed, and so it is normal to find bigger values for the objective function, compared to the previous section.

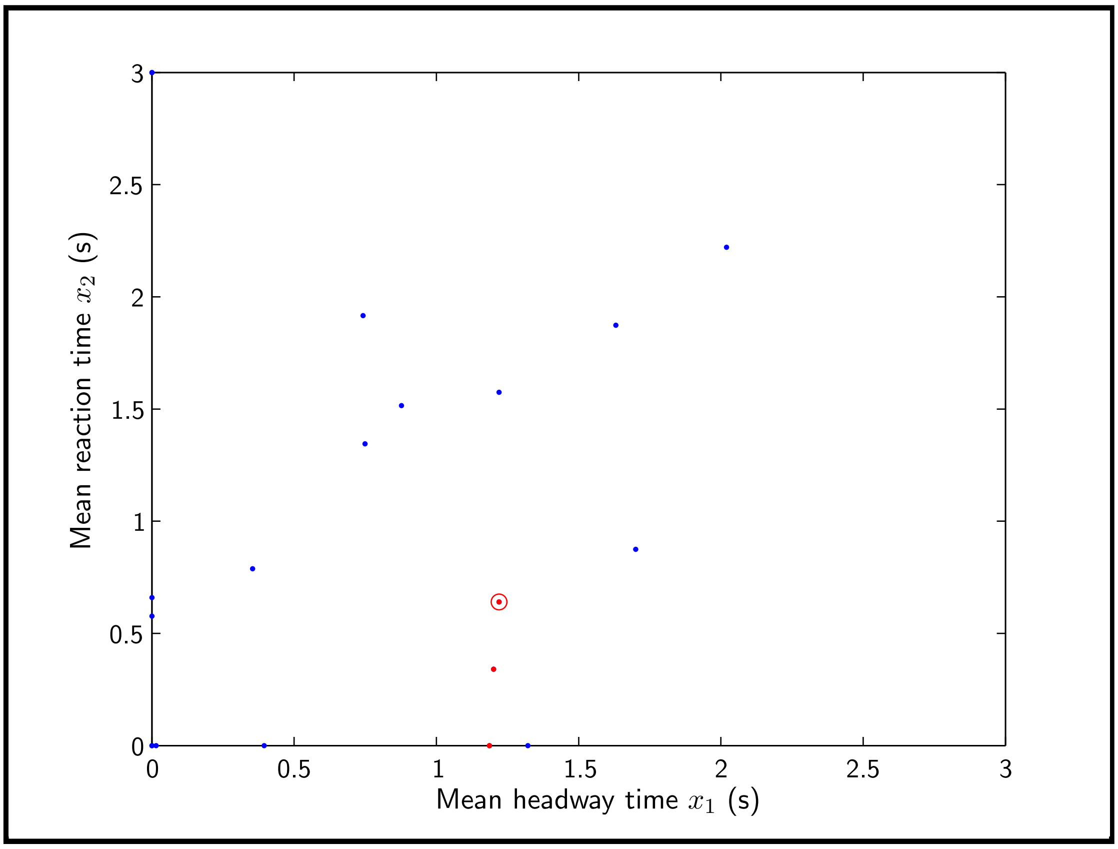


Figure 6.8: Optimum points in the state space for the 20 simulation runs executed with *fmincon* (calibration with simulated data, random seed). The global optimum is marked with a red dot and a red circle, while the two other best minima are marked with a red dot.

	$x_{0,1}$	$x_{0,2}$	x_1	x_2	z	N.It.	N.F.E.
μ	1.5545	1.6030	0.7317	1.0161	0.1361	4.95	80.05
σ^2	0.8303	0.7063	0.4560	0.9854	$4 \cdot 10^{-3}$	6.68	$5.89 \cdot 10^3$

Table 6.8: Mean and variance values for the outputs of the simulation runs executed with *fmincon* (calibration with simulated data, random seed).

with a variance $\sigma_z = 4 \cdot 10^{-3}$. Regarding the calibrating parameters, their mean values are $\mu_{x_1} = 0.7317$ and $\mu_{x_2} = 1.0161$, with variance values of $\sigma^2_{x_1} = 0.4560$ and $\sigma^2_{x_2} = 0.9854$: in fact, the points of minimum are sparse in the state space (Figure 6.8). Other parameters could be seen in Table 6.8.

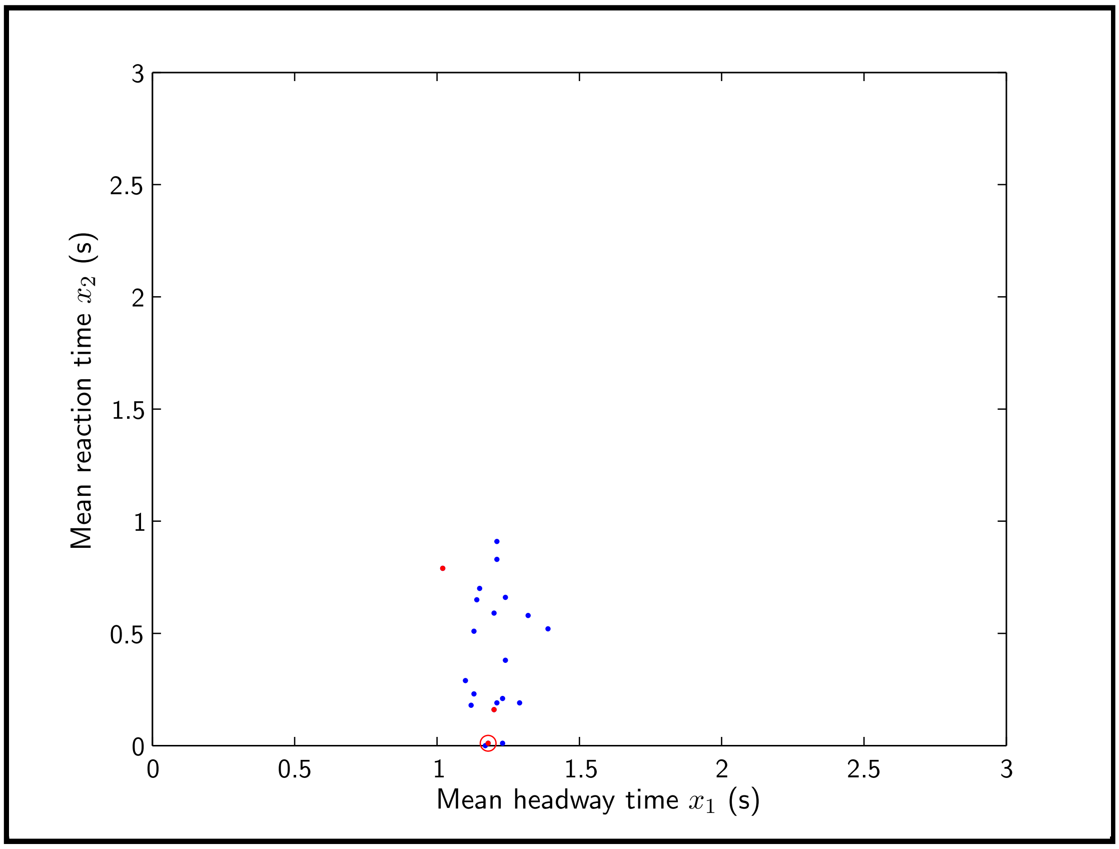


Figure 6.9: Optimum points in the state space for the 20 simulation runs executed with *ga* (calibration with simulated data, random seed). The global optimum is marked with a red dot and a red circle, while the two other best minima are marked with a red dot.

Genetic Algorithm

In 20 simulation runs, the optimum has been found close to the border (Figure 6.9): $z = 0.0601$ in $\boldsymbol{x} = [1.18, 0.01]$ (Table 6.9). The fact that the minima have been all found in a small area, delimited by $x_1 \in [1.0, 1.5], x_2 \in [0.0, 1.0]$, means that the values of variance are quite small, and the same could be said for the variance of the minima (Table 6.10). On the other hand, each simulation had to be run until the *average change in the fitness value is less than options.TolFun*: in fact it took about 30 days to execute the 20 optimization runs.

Pattern Search

The 20 simulation runs with *patternsearch* (Table 6.11) have been executed in just 28 hours, with an average of 15.3 iterations and 37.2 evaluations of the objective

Run	$x_{0,1}$	$x_{0,2}$	x_1	x_2	z	N.It.	N.F.E.
1	2.38	2.87	1.21	0.19	0.0753	51	1020
2	0.50	0.51	1.12	0.18	0.0656	51	1020
3	0.40	1.10	1.23	0.21	0.0687	51	1020
4	2.78	1.54	1.29	0.19	0.0734	51	1020
5	**0.09**	**2.29**	**1.18**	**0.01**	**0.0601**	**51**	**1020**
6	2.27	2.72	1.32	0.58	0.0750	51	1020
7	1.42	2.56	1.15	0.70	0.0656	51	1020
8	0.39	2.04	1.17	0.00	0.0644	51	1020
9	1.93	2.90	1.10	0.29	0.0808	51	1020
10	2.66	2.51	1.21	0.83	0.0826	51	1020
11	0.98	0.98	1.13	0.23	0.0638	51	1020
12	1.97	0.10	1.39	0.52	0.0828	51	1020
13	1.16	1.90	1.21	0.91	0.0730	51	1020
14	2.56	2.72	1.02	0.79	0.0623	51	1020
15	2.54	2.20	1.13	0.51	0.0807	51	1020
16	2.85	0.69	1.20	0.59	0.0661	51	1020
17	2.80	2.95	1.24	0.66	0.0819	51	1020
18	1.89	0.96	1.23	0.01	0.0658	51	1020
19	0.48	1.75	1.24	0.38	0.0755	51	1020
20	1.23	1.09	1.14	0.65	0.0722	51	1020

Table 6.9: Summary table with the results of the simulation runs executed with *ga* (calibration with simulated data, random seed). The bolded row coincides with the simulation run that has yield the global optimum.

	$x_{0,1}$	$x_{0,2}$	x_1	x_2	z	N.It.	N.F.E.
μ	1.6640	1.8190	1.1955	0.4215	0.0718	51	1020
σ^2	0.8960	0.7969	0.0068	0.0832	$5.46 \cdot 10^{-5}$	0	0

Table 6.10: Mean and variance values for the outputs of the simulation runs executed with *ga* (calibration with simulated data, random seed).

function. The optimum $z = 0.0603$ is achieved in $\boldsymbol{x} = [1.11, 0.46]$; mean and variance values could be seen in Table 6.12. In this case too, the points of optimum are concentrated in the same specific area (Figure 6.10).

Comparison of Simulations with Random Seed

The outputs from the previous optimization runs are summarized in Table 6.13. It can be easily seen how *fmincon* is not suitable for calibrating the parameters of

Run	$x_{0,1}$	$x_{0,2}$	x_1	x_2	z	N.It.	N.F.E.
1	2.85	0.69	1.35	0.75	0.0684	13	33
2	1.82	1.46	1.32	0.58	0.0656	15	37
3	2.67	2.29	1.17	0.42	0.0614	15	33
4	1.37	0.06	1.12	1.06	0.0658	15	44
5	2.46	1.33	0.96	0.33	0.0801	21	45
6	1.85	2.38	1.35	0.63	0.0780	15	35
7	2.77	2.21	1.15	0.21	0.0700	27	66
8	0.53	1.22	1.16	0.22	0.0642	13	32
9	2.81	2.75	1.37	1.00	0.0724	17	39
10	1.23	2.68	1.23	0.18	0.0838	11	27
11	0.17	1.06	1.17	1.06	0.0748	13	30
12	2.44	0.03	1.25	0.03	0.0732	23	45
13	0.42	0.61	0.42	0.61	0.0770	7	20
14	0.60	1.81	1.16	0.31	0.0691	17	43
15	0.82	0.60	1.26	0.60	0.0831	11	32
16	0.05	2.24	1.05	1.24	0.0692	17	36
17	1.34	2.80	1.09	0.80	0.0614	11	29
18	1.40	1.26	1.40	0.51	0.0673	13	38
19	2.54	1.58	0.67	0.58	0.0829	13	31
20	**0.61**	**2.02**	**1.11**	**0.46**	**0.0603**	**19**	**49**

Table 6.11: Summary table with the results of the simulation runs executed with *patternsearch* (calibration with simulated data, random seed). The bolded row coincides with the simulation run that has yield the global optimum.

	$x_{0,1}$	$x_{0,2}$	x_1	x_2	z	N.It.	N.F.E.
μ	1.5375	1.5540	1.1375	0.5790	0.0714	15.30	37.20
σ^2	0.9354	0.7507	0.0562	0.1088	$5.62 \cdot 10^{-5}$	21.17	96.17

Table 6.12: Mean and variance values for the outputs of the 20 simulation runs executed with *patternsearch* (calibration with simulated data, random seed).

such a traffic model: the variance values of mean headway time and mean reaction time are too high to be considered trustworthy. This could have been predicted with a look at the shape of the objective function (Figure 6.1(c)), similar to the phenomenon observed in Section §3.1.

On the other hand, *ga* and *patternsearch* have comparable values of variance for the calibrating parameters. While the ones from *ga* could be slightly better than the ones obtained from *patternsearch*, the length of their optimization runs are not

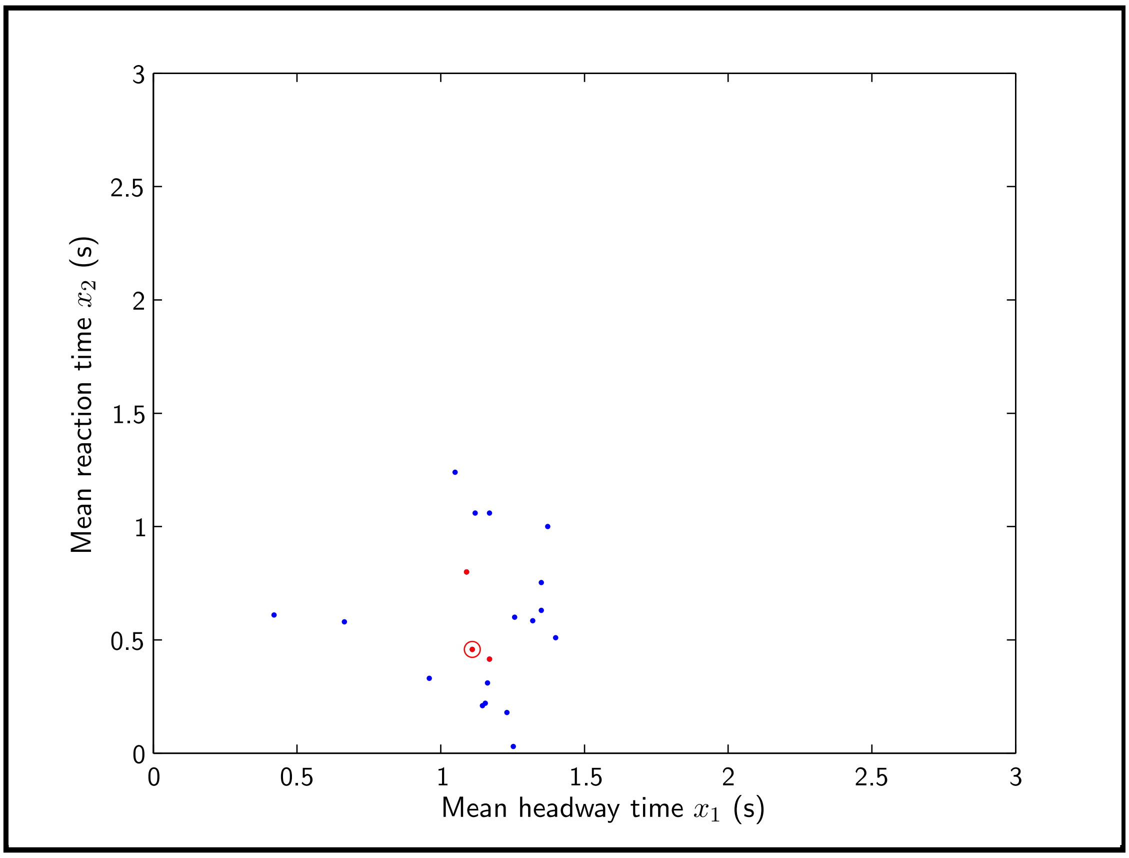


Figure 6.10: Optimum points in the state space for the 20 simulation runs executed with *patternsearch* (calibration with simulated data, random seed). The global optimum is marked with a red dot and a red circle, while the two other best minima are marked with a red dot.

	x_1	x_2	μ_{x_1}	μ_{x_2}	$\sigma^2_{x_1}$	$\sigma^2_{x_2}$
fmincon	1.22	0.64	0.7317	1.0161	0.4560	0.9854
ga	1.18	0.01	1.1955	0.4215	0.0068	0.0832
patternsearch	1.11	0.46	1.1375	0.5790	0.0562	0.1088

	z	μ_z	σ^2_z	$\mu_{n.it.}$	$\mu_{n.ev.}$
fmincon	0.0651	0.1361	$4.00 \cdot 10^{-3}$	4.95	80.05
ga	0.0601	0.0718	$5.46 \cdot 10^{-5}$	51.00	1020.00
patternsearch	0.0603	0.0714	$5.62 \cdot 10^{-5}$	21.17	96.17

Table 6.13: Comparison of the results obtained from the optimization runs executed with *fmincon*, *ga* and *patternsearch* (calibration with simulated data, random seed).

commensurable: *patternsearch* is able to reach a solution at least 25 times faster than *ga*.

Therefore, for the simulations involving the calibration and validation algorithm, *patternsearch* will be the selected optimization routine.

6.2 Calibration and Validation

As it has been seen in the previous section, the process of calibration and validation will be executed using *patternsearch* as the optimization routine.

6.2.1 Calibration Process

After 400 optimization runs, *patternsearch* found the minimum $z = 0.1477$ of the objective function in $\boldsymbol{x} = [1.55, 0.62]$: this value is bigger than the minima found in the previous sections because, in this case, the simulated measurements are compared with the real measurements and not with other simulated measurements.

Most of the optima have a value for the mean headway time at about $x_1 = 1.5$, while for the mean reaction time the values are spread in the interval $x_2 \in [0.0, 1.0]$ (Figure 6.11). This is also confirmed in Table 6.14: most of the values for the mean headway time are comprised between 1.50 and 1.60, while the values for the mean reaction time are comprised between 0.20 and 0.60, an interval four times wider. So, it could be confirmed (since this situation has been already seen in Section §6.1) that the mean headway time is a more sensitive parameters, than the mean reaction time, for the evaluation of the objective function, and so x_1 is more influential than x_2 on the flow of the vehicles on the network. However, the Paramics model of the network is only able to barely replicate the flow, the density, and the speed of the vehicles detected in the real network, as they could be compared in Figure 6.12: it is noticeable that the simulated vehicles are, in average, faster than the real vehicles, in particular during the occurrence of the traffic jam. This fact could be justified by the imperfect modeling of the network, especially in its critical parts as, for example, the connections to the service station.

The *patternsearch* algorithm could be still regarded as an absolutely fast optimization routine for this kind of simulation, since it is able to find an optimum, averagely, in just 18 iterations (Table 6.15).

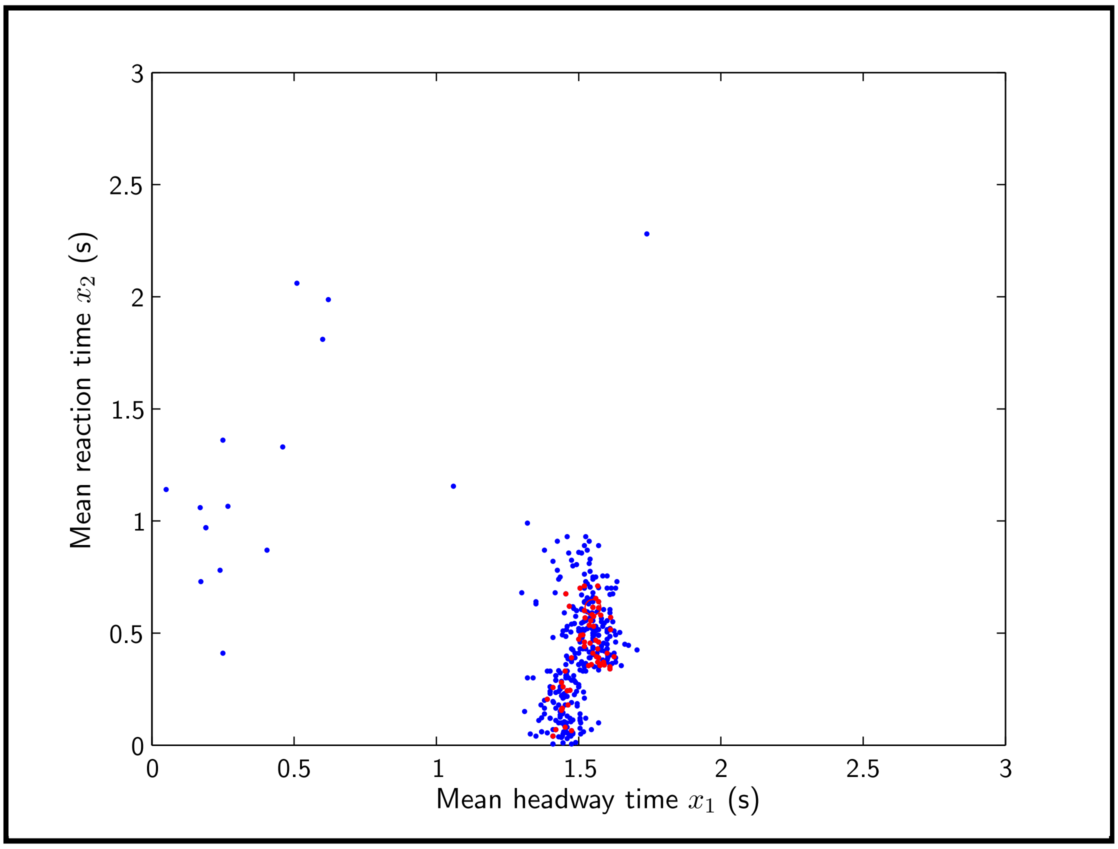
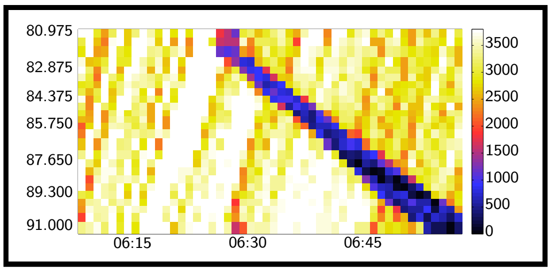


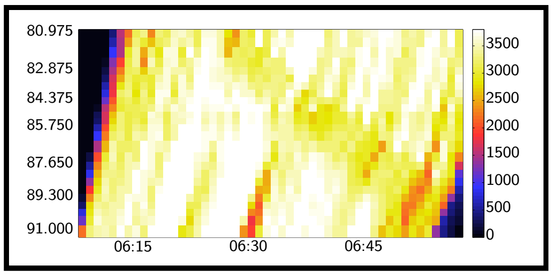
Figure 6.11: Optimum points in the state space for the 400 simulation runs executed with *patternsearch* (calibration and validation, random seed). The global optimum is marked with a red dot and a red circle, while the other best minima are marked with a red dot.

Run	x_1	x_2	z	Run	x_1	x_2	z
16	1.55	0.62	0.1477	232	1.58	0.36	0.1509
330	1.56	0.66	0.1485	310	1.54	0.56	0.1509
329	1.39	0.21	0.1491	213	1.52	0.71	0.1510
73	1.59	0.37	0.1500	113	1.54	0.36	0.1511
119	1.54	0.46	0.1502	142	1.57	0.39	0.1511
307	1.52	0.46	0.1502	366	1.52	0.49	0.1511
34	1.47	0.62	0.1503	195	1.54	0.35	0.1512
91	1.61	0.35	0.1505	64	1.47	0.25	0.1513
359	1.57	0.43	0.1506	255	1.54	0.53	0.1513
239	1.58	0.35	0.1507	351	1.56	0.40	0.1513

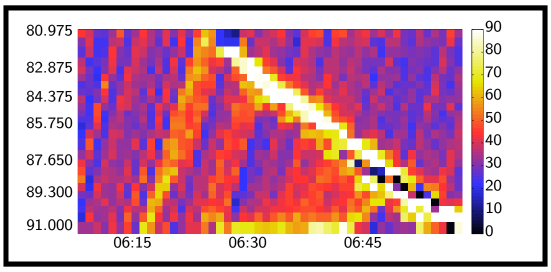
Table 6.14: Summary table with the best results of the 400 simulation runs executed with *patternsearch* (calibration and validation, random seed).



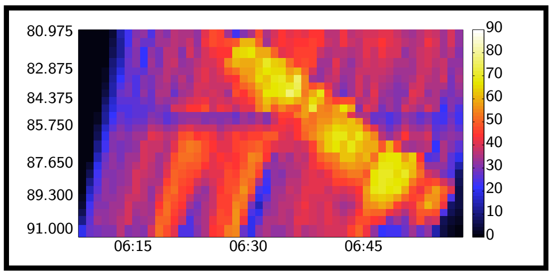
(a) Flow (veh/h).



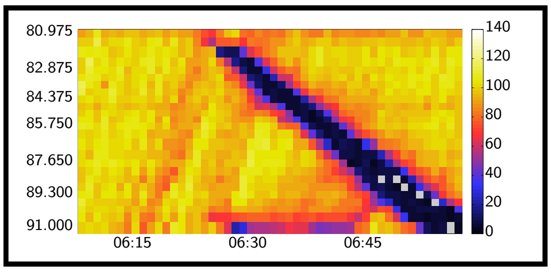
(b) Flow (veh/h).



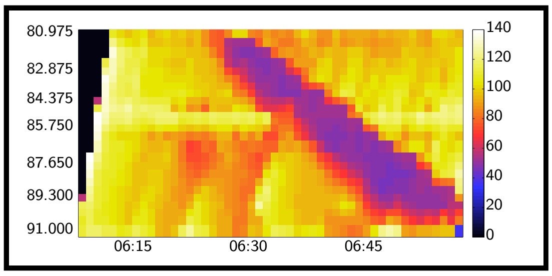
(c) Density (veh/km).



(d) Density (veh/km).



(e) Speed (km/h).



(f) Speed (km/h).

Figure 6.12: Reference measurements (left) and simulated measurements for $\boldsymbol{x} = [1.55, 0.62]$ (optimum found with the *patternsearch* optimization algorithm, right).

	$x_{0,1}$	$x_{0,2}$	x_1	x_2	z	N.It.	N.F.E.
μ	1.4976	1.4827	1.4626	0.4358	0.1563	17.99	45.84
σ^2	0.6968	0.7104	0.0547	0.0857	$1.97 \cdot 10^{-4}$	20.19	151.89

Table 6.15: Mean and variance values for the outputs of the 400 simulation runs executed with *patternsearch* (calibration and validation, random seed).

6.2.2 Validation Process

Once the calibration of the model parameters has been executed, the process of validation could be finally started. Since the optimum for the calibration scenario has been found in $\boldsymbol{x} = [1.55, 0.62]$, this couple of parameters will be used to simulate the flow of the vehicles in a different scenario (Section §5.1.2): the simulated measurements will be, in fact, compared with the measurements detected on the real network on January 31st, 2006, from 7:40 to 8:30.

As it could have been predicted due to the insufficient accuracy of the model, the calculated Theil's inequality coefficient is higher than any likely satisfactory value it could have been desired[4]: for $\boldsymbol{x} = [1.55, 0.62]$, the simulation has yielded $U = 29.01\%$. But, since in this case study the coefficient is composed by two terms ($U = U_f + U_v$), an acceptable value would be smaller than 20%; in this case, $U_f = 17.33\%$ and $U_s = 11.68\%$.

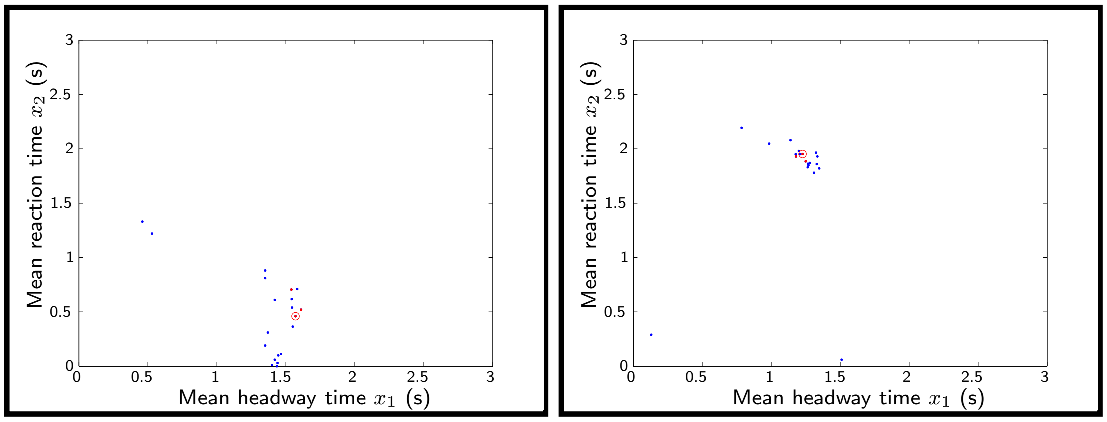
Since Theil's inequality coefficient is too large, the model of the network could not be considered as validated, and therefore such model should be improved and re-calibrated.

6.3 Cross-Validation

The cross-validation process (Section §5.3) has been executed using *patternsearch* as the optimization routine involved in the calibration of the mean headway time and the mean reaction time. The procedure is divided in two parts: the first one consists in calibrating the parameters through the calibration scenario (January 30th, 2006) and in validating them through the validation scenario (January 31st, 2006); the second part is performed in the same manner, with the only difference being the swapping of the scenarios: the calibration scenario will be the one selected from January 31st, while the validation scenario will be the one selected from January 30th.

The first set of simulation runs, as expected, has yielded an optimum and its relative parameters similar to the ones already obtained in Section §6.2: $z_A = 0.1506$ in $\boldsymbol{x_A} = [1.57, 0.46]$; the optima found are shown in Figure 6.13(a). The second set of simulations, instead, has produced very different values for the calibrating parameters, in particular for the mean reaction time: $\boldsymbol{x_B} = [1.23, 1.95]$, corresponding to $z_B = 0.1296$ (Figure 6.13(b)). The simulated measurements may be compared with the real ones obtained from the validation scenario in Figure 6.14. Comparing the values from both the simulations (Table 6.16), it could be seen that the Paramics model of the network is better suited for the scenario of January 31st (selected as the validation scenario), than for the one of January 30th (selected as the calibration scenario).

[4]Theil's inequality coefficient should be less than $5 - 10\%$.



(a) Calibration set. (b) Validation set.

Figure 6.13: Optimum points in the state space for the 2 sets of 20 simulation runs executed with *patternsearch* (cross-validation, random seed). The global optimum is marked with a red dot and a red circle, while the two other best minima are marked with a red dot.

Set	x_1	x_2	μ_{x_1}	μ_{x_2}	$\sigma^2_{x_1}$	$\sigma^2_{x_2}$
#A	1.57	0.46	1.3687	0.4790	0.0963	0.1537
#B	1.23	1.95	1.1761	1.7543	0.0821	0.3024

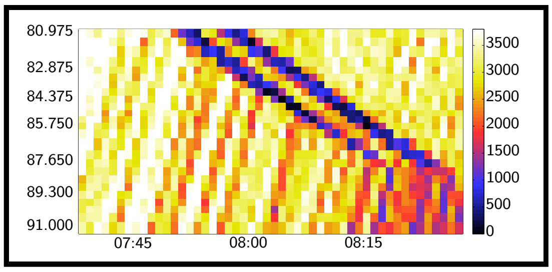
Set	z	μ_z	σ^2_z	$\mu_{n.it.}$	$\mu_{n.ev.}$
#A	0.1506	0.1585	$9.31 \cdot 10^{-5}$	17.70	43.95
#B	0.1296	0.1433	$1.57 \cdot 10^{-4}$	17.80	45.85

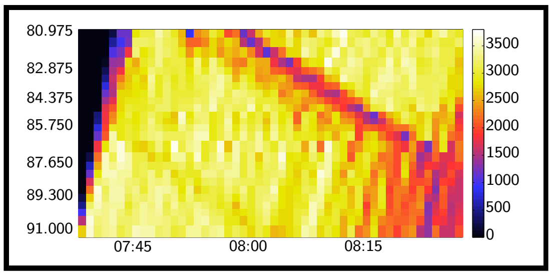
Table 6.16: Comparison of the results obtained from the first set and from the second set of optimization runs with *patternsearch* (cross-validation, random seed).

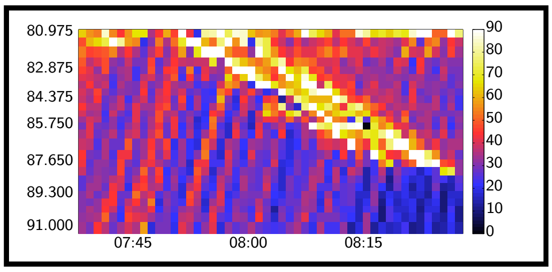
The ultimate parameters, that should fit for both the scenarios, are estimated by means of $\boldsymbol{x} = 0.5 \cdot (\boldsymbol{x_A} + \boldsymbol{x_B})$; unfortunately, these values are quite distant from the original ones, especially for the mean reaction time: $x_1 = 1.40$, and $x_2 = 1.21$. By simulating again the traffic of the vehicles for both scenarios, this time using $\boldsymbol{x} = [1.40, 1.21]$, it could be seen how the flows (Figures 6.15(e), 6.15(f)) are different from the flows obtained with the optima values of the parameters (Figures 6.15(c), 6.15(d)), and obviously more different from the real measurements (Figures 6.15(a), 6.15(b)).

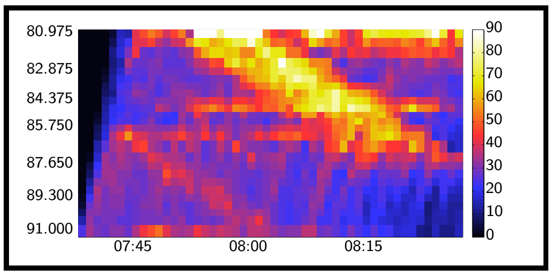
6.4 Summary

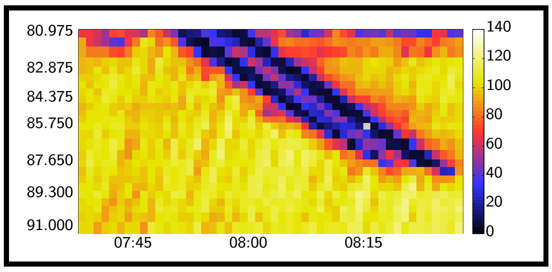
The optimization routines of *fmincon*, *ga* and *patternsearch* have been compared through the algorithm for *calibration with simulation data*, and *patternsearch* is


(a) Flow (veh/h).


(b) Flow (veh/h).


(c) Density (veh/km).


(d) Density (veh/km).


(e) Speed (km/h).

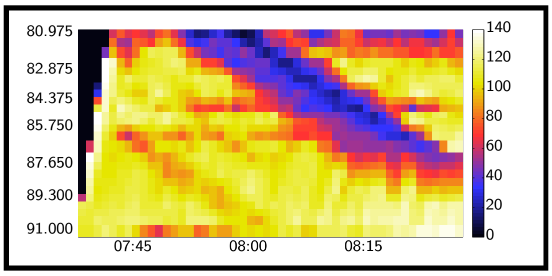
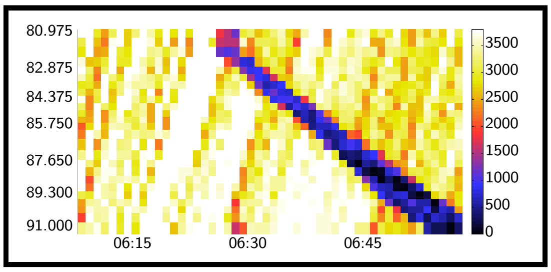
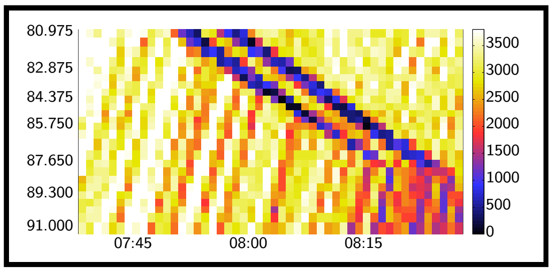

(f) Speed (km/h).

Figure 6.14: Real measurements (left) and simulated measurements for $\boldsymbol{x} = [1.23, 1.95]$ (optimum found with the *patternsearch* optimization algorithm, right), concerning the validation scenario.

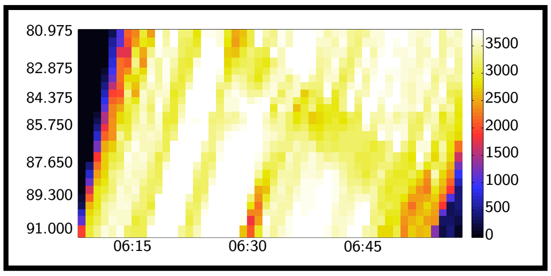
arisen to be the most effective of them. Afterward, the algorithm for *calibration and validation* has been executed: the process of optimization has been successfully performed by means of *patternsearch*, while the model could not have been validated due to the excessive value achieved for the measure of fitness. Finally, a short test of the algorithm for *cross-validation* has been completed, showing how much the calibrated parameters could be dissimilar to each other, depending on the selected scenario.



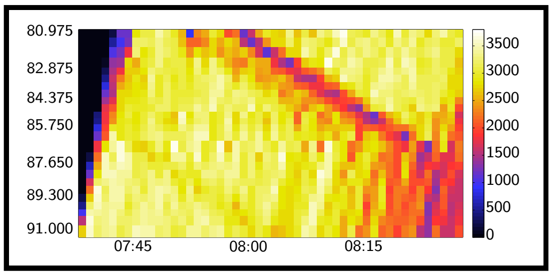
(a) Real measurements (calibration scenario).



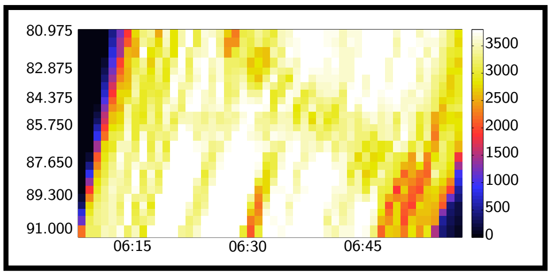
(b) Real measurements (validation scenario).



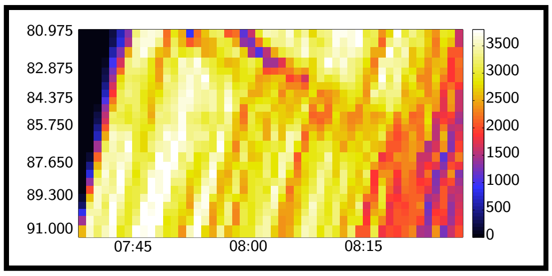
(c) Simulated measurements ($\boldsymbol{x} = [1.57, 0.46]$).



(d) Simulated measurements ($\boldsymbol{x} = [1.23, 1.95]$).



(e) Simulated measurements ($\boldsymbol{x} = [1.40, 1.21]$).



(f) Simulated measurements ($\boldsymbol{x} = [1.40, 1.21]$).

Figure 6.15: Comparison between the real and the simulated measurements of the flow of the vehicles (calibration scenario on the left, validation scenario on the right).

Chapter 7

Conclusions and Future Research

7.1 Conclusions

The purpose of this MSc project is the design of an algorithm that is able to calibrate and validate the parameters of the model of a traffic network.

At first, the Paramics model of a 10 kilometres highway stretch has been built, in order to provide the case study for the calibration and the validation processes; the model includes the layout of the network and the service station, the evaluation of the origin-destionation matrix, and the estimation of the demand profiles.

Then, the algorithm for the calibration of (some of) the parameters that regulate the flow of the vehicles on the network has been formulated, and several runs have been executed. By means of the algorithm for calibration with simulated data, the optimization routine *patternsearch* has been selected as the most efficient for a such task, as it prevailed on *fmincon* and *ga*. In fact, the accuracy on the results obtained with *patternsearch* and with *ga* is quite bigger compared to the one obtained with *fmincon*: for example, the variance of the objective function is about 100 times smaller than the one obtained by using *fmincon*. Then, *patternsearch* has prevailed on *ga*, since the routine based on direct search is able to reach a valuable solution of the optimization problem about 10 times faster than the genetic algorithm.

With *patternsearch* selected as the most effective optimization routine, the algorithm for the calibration of the mean headway time x_1 and of the mean reaction time x_2 has been run. The results have shown that the mean headway time is, amongst the two parameters, the most sensitive one, since the values obtained from the several simulation runs oscillate around $x_1 = 1.5$, while the values for the mean reaction time are spread in a relatively wide interval: $x_2 \in [0.0, 1.0]$. Therefore, it has not been possible to select a particular point as the global optimum, but only a well-defined area in the state space where the optimum value of the objective function could be likely found.

Finally, the algorithm for validation stated the inefficiency of the model: the performance of the traffic flow, simulated with the calibrated parameters, is not close enough to the real behavior of the vehicles, since the simulated measurements and the actual measurements are not comparable yet, as pointed out by the high value of Theil's inequality coefficient: an improvement of the model is still needed, and some related suggestions are described in Section §7.2. In fact, the model is not recommended to be used for other purposes (e.g., dynamic speed control), at least until it is successfully validated.

7.2 Future Research

In this section the improvements, that could be applied on the whole algorithm submitted in this thesis project, are discussed in order to quickly achieve better results (short term research). Furthermore, some hints, guidelines and suggestions for another MSc project (medium term research) and for a PhD project (long term research) on this same topic are presented.

7.2.1 Short Term Research: Improvements on this Project

Modeling of the Traffic Network

A moderate improvement on the minumum value of the objective function could be achieved with a more accurate modeling of the highway: as an example, if the position of the missing detector loops could be obtained, they could be inserted in the model of the network and their measurements could be compared with the outputs of the Paramics simulation.

Another little improvement on the model could be obtained by accurately modeling the structure of the road: the network, in fact, is not an exact stretch, since it presents some large curvatures. If these curvatures were modeled, the vehicles would moderately brake in such parts of the highway as it happens in the real world. The same could be said regarding the width of the lanes, which is unknown and has been only estimated for this model.

Finally, by a quick analysis of Figures 6.12 and 6.14, it could be seen how the flow of the vehicles in the simulations is, on average, higher than the flow obtained from the real measurements. This may have been caused by the absence of the subsequent part of the stretch, which contains an on-ramp: the vehicles of the model, since they do not know about the probable traffic jams caused by the vehicles entering the highway from the on-ramp, are not subjected to the effects of these traffic jams and so they may continue to flow until the end of the model network without being perturbed. This undesidered effect is reduced by the presence of the forced speed limits at the end of the network, but probably it is not totally nullified.

Modeling of the Service Station and Demand Profiles

The biggest obstacle for the building of a truthful model is the presence of the service station: the data regarding the vehicles that enter and leave the service station are not available, and this does not allow the algorithm to build accurate origin-destination matrix and demand profiles for such vehicles.

A different method (from the one used in this project) for the evaluation of these demand profiles is based on a dynamic estimation of the number of vehicles that enter and leave the highway for the service station. As an example, for each minute of the simulation period, the number of vehicles that enter the service station could be estimated as follows:

$$n_v(j) = \frac{f_8(j) - \{a(j) f_9(j) + [1 - a(j)]\, f_9(j+1)\}}{60}$$

where j is the j-th minute of the simulation, f_8 and f_9 are, respectively, the measured flows at detector loop 8 (the one before the off-ramp) and at detector loop 9 (the one after the off-ramp), and $a(j) \in [0 \div 1]$ is a coefficient that is determined by the speed of the vehicles in the link between the above-mentioned detector loops. This method works only if the mean speed is such that the vehicles could move from the 8th detector to the 9th in a period of time smaller than one minute. In order to overcome this complication, the coefficient $a(j)$ may also depend by the average time needed by the vehicles to cover the distance from the 8th to the 9th detector loop. In this way, it could be possible to have three different profiles instead of the one used for the three paths of the OD matrix.

Speed Limits

In this project a speed limit of 120 km/h has been applied to the whole stretch, as it is the standard speed limit on the Dutch highways. However, since the speed limits applied on the real stretch are not available in the data supplied by the Ministry of Transportations, it could be useful to travel across the case-study stretch in order to discover which are the real speed limits, hypothesizing that they have not been changed since January 2006; elsewhere, the speed limits could also be estimated from the speed measurements detected during conditions of free-flow traffic.

Length of the Simulation Interval

The selected simulation period (50 minutes) is probably too short for obtaining accurate results but, due to the excessive length of the Paramics simulations, it has not been possible to extend this period: an improvement on the results could be obtained if the selected simulation period is longer than the current one (e.g., 1 hour and a half, 2 hours, or more, instead of 50 minutes).

Simulation Parameters

Some parameters of the simulation could be modified to improve the accuracy of the model. As an example, t_shift could be decomposed in two parts[1]: $t\_shift = t_1 + t_2$, where t_1 is the average time needed by the vehicles to reach the service station from the origin of the stretch, and t_2 is the average time for fueling. Furthermore, t_2 could increase if the service station includes a supermarket and/or a bar. The parameter t_rise should be modified too, since it depends on t_shift.

Regarding the Paramics simulations, the parameter *timestep* could be increased in order to obtain bigger accuracy; on the opposite, this would increase the simulation time too.

Then, an accurate estimation of the number of simulation runs needed for a correct evaluation of the objective function should be performed, as described in Section §3.1. By this approach, a higher value for the parameter n_r (instead of $n_r = 5$) will be probably found, and the total simulation time will be further increased, as a tradeoff between accuracy and quickness.

Objective Function

The objective function could be easily extended so that it could also include the effect of the density measured at the detector loops, as this already occurs with the flow and the speed of the vehicles:

$$z(\boldsymbol{x}) = a_f z_f(\boldsymbol{x}) + a_v z_v(\boldsymbol{x}) + a_d z_d(\boldsymbol{x})$$

where the weights a_f, a_v, a_d do not necessarily have the same value.

[1]This parameter would not be any longer necessary if the technique for modeling the demand profiles, presented previously in Section §7.2.1, is applied.

Measure of Fitness

Theil's inequality coefficient could be decomposed in three parts (Section §3.4), that could be separately studied: this would help in understanding how the error has been originated, and in which part of the model it is suggested to address the biggest effort in order to improve the measure of fitness.

7.2.2 Medium Term Research: MSc Project

A future MSc project could complete the work started with this thesis. Through the algorithm developed in Section §5.2 and improved as previously suggested in Section §7.2.1, a larger and more complex traffic network could be calibrated and validated (as an example, it could be the whole A12 highway or an urban network). If the network measurements are gathered together in files organized in the same manner as the files used for this project, the algorithm for calibration and validation would be immediately available in order to start the process.

In the future project, the cross-validation procedure could also be executed by using a wide set of scenarios or by studying another simple stretch (for example, without service station). Another task could be the selection of new calibrating parameters (perhaps it could be done after having studied their sensitivity on the objective function), as well as the calibration of the origin-destination matrix and, eventually, of the demand profiles.

The thesis could be completed with a study on the evaluation of the best parameters for the optimization routines (e.g., the tolerances on the objective function and on the optimization variables, the poll method in *patternsearch*, the selection function in *ga*, and so on), in order to make the most of their efforts. It could be also studied whether the optimization routines could be run in succession: as an example, one optimization algorithm could be used to reach more quickly a satisfactory value of the parameters, that could be then improved by means of another algorithm.

7.2.3 Long Term Research: PhD Project

A long term project could deal with the task of better modeling the characteristics of the flow of the vehicles through the network, by also using other simulation software packages. They could be more successful than Paramics in modeling, for example, the car-following or the lane-changing behavior, and so a lower value of objective function could be found.

Once the network is deemed to have been calibrated and validated, finally some control measures could be applied to the vehicles in order to maximize the performance of the traffic flow (e.g., to decrease the traveling time, the length of the queues, the pollutant emissions). The control measures (e.g., ramp metering, route guidance, dynamic speed limits) can be ruled by different control strategies (e.g., Model Predictive Control), and they could be compared in order to evaluate which one is the most effective for the case-study network.

Other aspects that could be investigated are the applicability of on-line control strategies, and of distinct control strategies to be applied during exceptional circumstances, as accidents, or during critical weather conditions (e.g., heavy rain, snow).

Bibliography

[1] Tom Bellemans. *Traffic Control on Motorways.* PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium, 2003.

[2] Tom Bellemans, Bart De Schutter, and Bart De Moor. On Data Acquisition, Modeling and Simulation of Highway Traffic. In *Proceedings of the 9th IFAC Symposium on Control in Transportation Systems 2000 (CTS 2000)*, pages 22–27, Braunschweig, Germany, 2000.

[3] Tom Bellemans, Bart De Schutter, and Bart De Moor. Models for Traffic Control. *Journal A*, 43(3-4):13–22, 2002.

[4] Tom Bellemans, Bart De Schutter, and Bart De Moor. Anticipative Model Predictive Control for Ramp Metering in Freeway Networks. In *Proceedings of the 2003 American Control Conference*, pages 4077–4082, Denver, CO, USA, 2003.

[5] Richard Bishop. A Survey of Intelligent Vehicle Applications Worldwide. In *Proceedings of the 2000 IEEE Intelligent Vehicles Symposium*, pages 25–30, Dearborn, MI, USA, 2000.

[6] Paul T. Boggs and Jon W. Tolle. Sequential Quadratic Programming. *Acta Numerica*, 4:1–51, 1995.

[7] Sharon Adams Boxill and Lei Yu. An Evaluation of Traffic Simulation Models for Supporting ITS Development. Technical Report SWUTC/00/167602-1, Center for Transportation Training and Research, Texas Southern University, Houston, TX, USA, 2000.

[8] Wilco Burghout. A Note on the Number of Replication Runs in Stochastic Traffic Simulation Models. Technical Report CTR2004:01, Centre for Traffic Research, Royal Institute of Technology, Stockholm, Sweden, 2004.

[9] Eduardo F. Camacho and Carlos Bordons. *Model Predictive Control.* Berlin; New York: Springer, 1999.

[10] Lianyu Chu, Henry X. Liu, Jun-Seok Oh, and Will Recker. A Calibration Procedure for Microscopic Traffic Simulation. In *Proceedings of the 2003 International Conference on Intelligent Transportation Systems*, pages 1574–1579, Shanghai, China, 2003.

[11] Dutch Ministry of Transportations and Water Management. *Nota Mobiliteit*, 2004.

[12] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[13] Bruce A. Harvey, Glenn H. Champion, and Rick Deaver. Accuracy of Traffic Monitoring Equipment Field Tests. In *Proceedings of the IEEE 1993 Vehicle Navigation and Information Systems Conference*, pages 141–144, Ottawa, Canada, 1993.

[14] Rachel R. He and Bin Ran. Calibration and Validation of a Dynamic Traffic Assignment Model. *Transportation Research Record*, 1733:56–62, 1999.

[15] Andreas Hegyi. *Model Predictive Control for Integrating Traffic Control Measures*. PhD thesis, Delft University of Technology, Delft, The Netherlands, 2004.

[16] Bas H. Heutinck. Eindhoven Under Construction: Microscopic Traffic Simulation and Dynamic Alternative Route Guidance. Master's thesis, Delft University of Technology, Delft, The Netherlands, 2005.

[17] Bas H. Heutinck, Monique van den Berg, Hans Hellendoorn, and Ben Immers. Dynamic Route Guidance During Maintenance Works, a Case Study. In *Preprints of the 11th IFAC Symposium on Control in Transportation Systems*, pages 380–385, Delft, The Netherlands, 2006.

[18] Serge P. Hoogendoorn and Piet H.L. Bovy. Modelling Multiple User-Class Traffic Flow. *Transportation Research Record*, B34(2):123–146, 2000.

[19] John Hourdakis, Panos G. Michalopoulos, and Jiji Kottommannil. A Practical Procedure for Calibrating Microscopic Traffic Simulation Models. *Transportation Research Record*, 1852:130–139, 2003.

[20] Tamara G. Kolda, Robert Michael Lewis, and Virginia Torczon. Optimization by Direct Search: New Perspectives on Some Classical and Modern Methods. *SIAM Review*, 45(3):385–482, 2004.

[21] Der-Horng Lee, Xu Yang, and Phua Chandrasekar. Parameter Calibration for Paramics using Genetic Algorithm. In *Proceedings of the 80th Annual Meeting of Transportation Research Board*, Washington, DC, USA, 2001.

[22] Edward Lieberman and Ajay K. Rathi. *Traffic Simulation in Traffic Flow Theory*. Gartner and Messer, 1997.

[23] Roger V. Lindgren and Sutti Tantiyanugulchai. Microscopic Simulation of Traffic at a Suburban Interchange. In *Proceedings of the 2003 ITE Annual Meeting*, Seattle, WA, USA, 2003.

[24] Lennart Ljung. *System Identification: Theory for the User*. Prentice-Hall, 1987.

[25] Tao Ma and Baher Abdulhai. A Genetic Algorithm-based Optimization Approach and Generic Tool for the Calibration of Traffic Microscopic Simulation Parameters. *Transportation Research Record*, 1800:6–15, 2002.

[26] Nicola Manca. Design of Adaptive Cruise Controllers Using Hybrid Control Methods. Master's thesis, Delft University of Technology, Delft, The Netherlands, 2007.

[27] MathWorks. *MATLAB Optmization Toolbox 3, User's Guide*, 2007.

[28] Adolf D. May. *Traffic Flow Fundamentals*. Prentice Hall, 1990.

[29] Jose M. Menendez, Luis Salgado, Enrique Rendon, and Narciso Garcia. Motorway Surveillance through Stereo Computer Vision. In *Proceedings of the IEEE 33rd Annual 1999 International Carnahan Conference on Security Technology*, pages 197–202, Madrid, Spain, 1999.

[30] Mernout Burger. Control of Traffic Networks Using Intelligent Vehicles. Master's thesis, Delft University of Technology, Delft, The Netherlands, 2007.

[31] Frans Middelham. Dynamic Traffic Management. Technical report, Ministry of Transport, Public Works, and Water Management, Directorate-General of Public Works and Water Management, AVV Transport Research Centre, Rotterdam, the Netherlands, 2006.

[32] Melanie Mitchell. *An Introduction to Genetic Algorithms*. Cambridge, 1996.

[33] Markos Papageorgiou, Habib Hadj-Salem, and Jean-Marc Blosseville. ALINEA: a Local Feedback Control Law for On-Ramp Metering. *Transportation Research Record*, 1320:58–64, 1991.

[34] Paramics Quadstone Limited Website. *http://www.paramics-online.com.*

[35] Byungkyu Park and Hongtu Qi. Microscopic Simulation Model Calibration and Validation for Freeway Work Zone Network - A Case Study of VISSIM. In *Proceedings of the 2006 IEEE Intelligent Transportation Systems Conference*, pages 1471–1476, Toronto, Canada, 2006.

[36] Quadstone Limited. *Quadstone Paramics V5.1, User Guide and Reference Manual*, 2005.

[37] Hesham Rakha, Bruce Hellinga, Michel Van Aerde, and William Perez. Systematic Verification, Validation and Calibration of Traffic Simulation Models. In *Proceedings of the 75th Annual Meeting of Transportation Research Board*, Washington, DC, USA, 1996.

[38] Michael A.P. Taylor. Evaluating the Performance of a Simulation Model. *Transportation Research Record*, 13A:159–173, 1979.

[39] Tomer Toledo, Haris N. Koutsopoulos, Angus Davol, Moshe E. Ben-Akiva, Wilco Burghout, Ingmar Andreasson, and Christer Lundin. Calibration and Validation of Microscopic Traffic Simulation Tools: Stockholm Case Study. *Transportation Research Record*, 1831:65–75, 2003.