Real-time forecasting of traffic conditions on road networks

Tiago Luis Abreu Pinho

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Prof. Rosaldo José Fernandes Rossetti (PhD)

July 28, 2014
Real-time forecasting of traffic conditions on road networks

Tiago Luis Abreu Pinho

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Prof. Carlos Manuel Milheiro de Oliveira Pinto Soares (PhD)
External Examiner: Prof. Artur José Carneiro Pereira (PhD)
Supervisor: Prof. Rosaldo José Fernandes Rossetti (PhD)

July 28, 2014
Abstract

In the present, most of the cities and metropolitan regions face the problem of having traffic congestions on their road networks, being this an important challenge and topic for the Intelligent Transport Systems (ITS). A hypothesis of solution to mitigate this problem is through the development of traffic forecasting techniques using historical data, that would help to take better traffic management decisions regarding the network’s state in a close future.

Thus, this dissertation focus on the study, test and analysis of techniques to forecast the traffic conditions on road networks (motorways) in real-time, based on Simulation (model-driven approach) and Artificial Neural Networks (data-driven approach), that would serve as basis for a new module in the Intelligent Transport System developed by Armis. The chosen simulation technique was the microscopic approach using the open-source simulator SUMO (Simulation of Urban Mobility). Regarding the data-driven approach, it was decided to use artificial neural networks (ANN), firstly developing a normal model and then creating a combined neural network with fuzzy logic, to learn and predict the traffic patterns.

For testing these techniques it was used real data collected from traffic detectors located in VCI (Via de Cintura Interna) motorway, a main inner-ring road of Porto. The executed tests were made using a test-bed corresponding to a junction node of VCI motorway.
Resumo

Hoje em dia, a maioria das cidades e grandes áreas metropolitanas enfrentam o problema de ter congestionamentos nas suas redes rodoviárias, sendo este um desafio importante para os sistemas de transporte inteligentes (ITS). Uma hipótese de solução para atenuar este problema é através do desenvolvimento de técnicas de previsão de tráfego usando dados históricos, que poderão ajudar a melhorar as decisões de gestão de tráfego tomadas, tendo em conta o estado da rede num futuro próximo.

Assim, esta dissertação foca-se no estudo, teste e análise de técnicas para previsão das condições de tráfego em redes rodoviárias (auto-estradas) em tempo real, baseando-se em Simulação (abordagem *model-driven*) e Redes Neuronais Artificiais (abordagem *data-driven*), que servirão como base para um novo módulo do ITS desenvolvido pela Armis. A técnica de simulação escolhida foi a abordagem microscópica através do uso do simulador *open-source* SUMO (Simulation of Urban Mobility). Em relação à abordagem *data-driven*, optou-se por utilizar redes neurais artificiais (RNA), desenvolvendo primeiro um modelo normal e depois um outro modelo combinado com lógica *fuzzy*, para aprender e prever os padrões de tráfego.

Para testar as técnicas desenvolvidas foram utilizados dados reais de sensores de tráfego existentes na auto-estrada VCI (Via de Cintura Interna), localizada na área metropolitana do Porto.
Acknowledgements

First of all, I would like to thank my supervisor, professor Rosaldo Rossetti, for the support and guidance given during the whole process of this thesis. I want also to thank to LIACC and its members for receiving me during the first stage of this project and for the knowledge and advices transmitted, as well to Armis which received me in its facilities in the last two months, in particular to Ricardo Coelho.

In addition, I am thankful to my colleges Eurico and Anya for the exchange of ideas and help at solving some problems during the implementation phase, and indubitably to Guilherme Soares and José Macedo for the great support and guidance given during my stay at Armis, where they were a kind of co-supervisors.

At last but not the least, I want to thank to the most important elements of my life, my family and friends.

Tiago Luis Abreu Pinho
“Any knowledge that doesn’t lead to new questions quickly dies out: it fails to maintain the temperature required for sustaining life”

Wisława Szymborska-Włodek
Contents

1 Introduction 1
   1.1 Context ............................................. 1
   1.2 Motivation and Goals ................................. 1
   1.3 Organisation of the Dissertation ................... 3

2 Literature Review 5
   2.1 Background ....................................... 5
      2.1.1 Simulation models on road traffic domain .. 5
      2.1.2 Data-driven models ............................ 11
   2.2 Related Works .................................. 12
      2.2.1 Macroscopic approach ........................ 12
      2.2.2 Microscopic approach ......................... 12
      2.2.3 Mesoscopic approach .......................... 13
      2.2.4 Nanoscopic approach .......................... 13
      2.2.5 Artificial Neural Networks ................... 13
      2.2.6 Fuzzy logic .................................. 14
   2.3 Summary ........................................ 14

3 Methodological Approach 15
   3.1 Problem Statement ................................ 15
   3.2 Requirement Analysis ............................... 15
   3.3 Proposed Solution ................................. 16
      3.3.1 Forecasting with Simulation .................. 16
      3.3.2 Forecasting with Artificial Neural Networks 16
   3.4 Technology Overview ............................... 18
      3.4.1 SUMO (Simulation of Urban MObility) .......... 18
      3.4.2 JOSM (Java OpenStreetMap Editor) ............ 20
      3.4.3 Encog Machine Learning Framework ........... 20
   3.5 Summary .......................................... 20

4 Implementation 21
   4.1 Network Description ............................... 21
   4.2 Forecasting with Simulation ....................... 23
   4.3 Forecasting with Artificial Neural Networks ....... 27
   4.4 Summary .......................................... 31
## CONTENTS

5 Tests and Evaluation 33  
5.1 Simulation Model 33  
  5.1.1 Functional Tests 33  
  5.1.2 Execution Tests 34  
5.2 Data-driven Model 46  
5.3 Summary 57  

6 Conclusions and Future Work 59  
6.1 Goals achieved and final remarks 59  
6.2 Future Work 60  

References 61  

A Data and configuration files used for testing the models 67  
A.1 Data used for modelling traffic with DFRouter 67  
A.2 Data used for modelling traffic with OD matrices 70  
A.3 Data used for incident scenario using OD matrices 70
List of Figures

2.1 The different simulation models ........................................... 6
2.2 Comparison between car-following and cellular automata models .... 8
2.3 Example of neural network structure ..................................... 12

4.1 Map of VCI network with the existing nodes and traffic detectors location ........................................... 22
4.2 Map of Paranhos junction node with traffic detectors location .......... 22
4.3 Aerial view of test-bed area .................................................. 23
4.4 Map of test-bed area modelled for SUMO .................................. 24

5.1 Illegal U-turn ................................................................. 34
5.2 Illegal U-turn on a crossing .................................................. 34
5.3 Results of 1st test of DFRouter - West (C) direction ..................... 35
5.4 Results of 1st test of DFRouter - East (D) direction ...................... 36
5.5 Results of 2nd test of DFRouter using "--disallowed-edges" - West (C) direction ........................................... 37
5.6 Results of 2nd test of DFRouter using "--disallowed-edges" - East (D) direction ........................................... 37
5.7 Results of 1st test using OD matrices - West (C) direction .............. 38
5.8 Results of 1st test using OD matrices - East (D) direction ............... 38
5.9 Results of 2nd test using OD matrices - West (C) direction .............. 39
5.10 Results of 2nd test using OD matrices - East (D) direction .............. 40
5.11 Results of 3rd test using OD matrices (4 hours) - West (C) direction .. 41
5.12 Results of 3rd test using OD matrices (4 hours) - East (D) direction .. 41
5.13 Results of 4th test using OD matrices with "--uniform" (4 hours) - West (C) direction ........................................... 42
5.14 Results of 4th test using OD matrices with "--uniform" (4 hours) - East (D) direction ........................................... 42
5.15 Results of 5th test using OD matrices (6 hours) - West (C) direction .. 43
5.16 Results of 5th test using OD matrices (6 hours) - East (D) direction .. 43
5.17 Results of 1st incident scenario test using vehicles stopping ............ 44
5.18 Map of accident area, with the defined speed limits ....................... 45
5.19 Results of 2nd incident scenario test using variable speed signs .......... 45
5.20 Results of the test of normal ANN with Sigmoid function - one hidden layer ........................................... 47
5.21 Results of the test of normal ANN with Sigmoid function - two hidden layers ........................................... 48
5.22 Results of the test of normal ANN with Tanh function - one hidden layer ........................................... 48
5.23 Results of the test of normal ANN with Tanh function - two hidden layers ........................................... 49
5.24 Results of the test of normal ANN using Tanh and Sigmoid function - two hidden layers ........................................... 50
5.25 Results of the test of normal ANN using Log function - one hidden layer ........................................... 50
5.26 Results of the test of normal ANN using Log function - two hidden layers ........................................... 52
5.27 Results of the test of ANN using an incident scenario ..................... 52
5.28 Results of the test of fuzzy ANN using Sigmoid function - one hidden layer ........................................... 54
LIST OF FIGURES

5.29 Results of the test of fuzzy ANN using Sigmoid function - two hidden layers . . . 54
5.30 Results of the test of fuzzy ANN with Tanh function - one hidden layer . . . . . . 55
5.31 Results of the test of fuzzy ANN with Tanh function - two hidden layers . . . . . 55
5.32 Results of the test of fuzzy ANN with Log function - one hidden layer . . . . . . 56
5.33 Results of the test of fuzzy ANN with Log function - two hidden layers . . . . . 57
List of Tables

3.1 Description of inputs used by the neural network ............................................. 17
3.2 Description of fuzzy sets for velocity input ...................................................... 18

5.1 Error rates obtained in the tests with DFRouter ................................................. 37
5.2 Error rates obtained in the 1st experiment with OD matrices ............................ 40
5.3 Error rates obtained in the 2nd experiment with OD matrices ........................... 43
5.4 Error rates obtained in the accident experiment with OD matrices .................. 46
5.5 Description of activation functions tested ......................................................... 46
5.6 Epochs and time spent per each ANN to finish the training - Sigmoid function .. 47
5.7 Epochs and time spent per each ANN to finish the training - Tanh function ........ 48
5.8 Epochs and time spent per each ANN to finish the training - Log function ......... 51
5.9 Error rates obtained in the tests with normal ANN ............................................. 51
5.10 Error rates obtained in the tests of an incident scenario with normal ANN ........ 53
5.11 Epochs and time spent per each fuzzy ANN to finish the training - Sigmoid function 53
5.12 Epochs and time spent per each fuzzy ANN to finish the training - Tanh function 55
5.13 Epochs and time spent per each fuzzy ANN to finish the training - Log function . 56
5.14 Specifications of computer used to test the simulation and the neural networks models ...................................................................................................................... 58

A.1 Data values used in the tests with dfrouter ......................................................... 67
A.2 OD matrix used in the experiments with OD2trips ............................................. 70
A.3 OD matrix used in the experiments of incident scenario with OD2trips ............... 71
Abbreviations

ANN  Artificial Neural Network
CSV  Comma Separated Value
GUI  Graphical User Interface
ITS  Intelligent Transport System
OD   Origin-Destination
OSM  Open Street Map
SUMO Simulation of Urban Mobility
VCI  Via de Cintura Interna
XML  Extensible Markup Language
Chapter 1

Introduction

1.1 Context

Nowadays, most of the cities and metropolitan regions face the problem of having traffic congestions on their road networks, affecting daily thousands of citizens in their home-job routes which waste time in traffic jams. Therefore, this is the main problem for urban transportation planners and the stakeholders of the traffic management, which seek to solve it through the use of systems for manage the traffic networks, formally called Intelligent Transport Systems (ITS) [RLT11].

These network traffic management systems are dynamic structures where a small disturbance or event may result in channels block which leads to time and money losses. Hence, it is required to take decisions regarding the network’s state in a close future. In order to do that it is necessary to use forecasted measures from historical data to predict how it will be the network in a near future. The first theories of traffic prediction applied to road networks date back to 1950s, however only from 1990s started to have a greater focus on traffic forecasting and the acknowledge of its potential use and benefits to road networks management.

Thus, this dissertation arises from the need of the current transport management systems to have a traffic forecast module. The main goal is to compare and evaluate two different forecasting techniques: simulation (model-driven approach) and neural networks (data-driven approach). The data used for testing these techniques was real traffic data from Porto motorway network, provided by Armis. This project was launched by Artificial Intelligence and Computer Science Laboratory (LIACC), which belongs to Faculty of Engineering of the University of Porto (FEUP) and had the collaboration of Armis, a company with a great experience and know-how in development of Intelligent Transport Systems (ITS).

1.2 Motivation and Goals

Forecast in real-time the traffic conditions on road networks has high importance and utility for Intelligent Transport Systems (ITS) as well for navigation systems because it allows to inform and
help the travellers to choose the best way (closest/fastest) to reach their destination, and consequently have a better management of the road networks.

Hence, the motivation for this project is supported in three topics: first and the main one is scientific since “Traffic Forecasting” as well “Traffic Management” are hot topics among scientific community, being the basis of several studies and projects developed since 1990s. In fact, more and more research projects focusing on this subjects have been carried out in the last decade, proving that it is a topic which is worth to explore, with still an open-space for improvements and innovations.

The second topic is the technological motivation, due to the fact that “Big-Data” and “Data-Mining” are concepts that have been very used nowadays and on the last 30 years. We live in an information era where the amount of data collected is increasing all the time and it is necessary to properly handle that data through the use of "data-mining" techniques, in order to analyse it and take the benefits of gathering such information. Also, currently, traffic management already makes use of "big-data" by collecting data about traffic conditions through the several existing tools for monitor and control road density, from traffic detectors to cameras or even radars.

And to finish there is the applicational motivation of having the challenge to improve Armis’s ITS by researching and analysing the addition of a new intelligent module that would be able to better assist in the decisions made about traffic management.

Therefore, this project aims to study, analyse and develop techniques for forecast the traffic conditions on road networks (motorways) in real-time based on Simulation and Artificial Neural Networks, that would serve as basis for a new module in the Intelligent Transport System developed by Armis. This new module should be able to work with a high volume of data coming in real-time, and with that predict traffic in any kind of road networks.

About the specific goals of the project, they are:

1. define a test-platform, using a pilot scenario case based on the junction node of Paranhos, which is part of VCI (Via de Cintura Interna) motorway.

2. implement the prediction algorithms firstly using SUMO (Simulation of Urban Mobility), a microscopic traffic simulator software, to deploy the model-based techniques. And later, focus on data-driven approaches using Artificial Neural Networks (ANNs) with Fuzzy logic to create a traffic predictor.

3. execute a set of tests to evaluate the two implemented models using the defined test-platform.

4. compare and analyse the efficiency of the developed algorithms with the expected results, in order to recommend which of those techniques are more suitable to Armis’ system.


1.3 **Organisation of the Dissertation**

Besides this first chapter, where the context, motivation and goals of the thesis were explained, this dissertation has also the following chapters:

In chapter 2, is presented the literature review made, describing the state-of-art among simulation and data-driven research communities, presenting the main existing techniques. The related works done in these subject are also presented.

In chapter 3, is presented the theoretical part of the thesis, by establishing the context and objectives, enumerating the technology requirements and describing the chosen technology.

In chapter 4 the implementation stage developed during the thesis is explained.

Chapter 5 covers the testing phase made, along with the analyse and evaluation of the achieved results.

To finalize, chapter 6 finishes the dissertation by presenting the final remarks and future work.
Introduction
Chapter 2

Literature Review

This section presents the concepts of Simulation applied to road traffic domain and Artificial Neural Networks which are covered by the project, in order to enumerate the differences between the existing methods and to help the reader to better understand the approached contents. It starts by defining the existing simulation models: macroscopic, mesoscopic, microscopic and nanoscopic, and their most used algorithms; as well the data-driven technique used in this project: Artificial Neural Network. Finally, it is presented a list along with a brief description of the past projects developed that are related to the subject of this thesis.

2.1 Background

2.1.1 Simulation models on road traffic domain

Simulation models can be classified into four different categories, according to the level of detail: macroscopic, mesoscopic, microscopic and nanoscopic. The macroscopic approach has the lowest level of detail, depicting traffic through its density per road section. In contrast the microscopic approach provides a higher level of detail by defining traffic through each existing pair of driver-vehicle in the network and the interaction between vehicles. Nanoscopic approach is a recent model that goes even further on detail by considering the vehicle’s characteristics and its parts like the type of engine, tyres used and fuel consumption. In figure 2.1 is shown the difference among the models:

Macroscopic Approach

This kind of model analyses traffic conditions by grouping data per each road section in the network, and describes the traffic flow by using vehicles’ density and average velocity. Thus, it is based on the traffic flow-rate at a certain part of network. Due to the fact that the analysis of the
network is made by road section and not by vehicle, it is a model that requires less computational resources than other models, but with the disadvantage of losing some accuracy in the forecasts. For that reason, macroscopic models are suitable to use with large networks where execution time is a priority rather than the level of detail.

Therefore, the macroscopic techniques identified during the literature review are:

**AW & Rascle** - it is a second-order model, composed by two equations: the first describes the flow conservation:

$$\partial_t \rho + \partial_x (\rho v) = 0 \quad (2.1)$$

where $\rho$ is the traffic density and $v$ represents the velocity.

The other formula defines the evolution of the driver’s speed according to the surrounding traffic state:

$$(\partial_t + vv_x)v + (\partial_t + vv_x)p(\rho) = 0 \quad (2.2)$$

where $p(\rho)$ an increasing function representing density.

It is a model identified as more suitable for road intersections. [AR00] [HB05]

**BLX (Berg, Lin, Xi)** – it is a model based on queues for each lane in the road, created by [vdBHDH07] and [LX08] but firstly theorized by [KS83] (urban part) and [MP90] (motorway part). Basically in each intersection, for any possible direction to take, there is a queue representing the number of vehicles that arrive there and the number that leave. At each time step, it forecasts traffic for each destination link. [LDXH12] Due to the high level of detail acquired by using queues in intersections, it is a reliable technique that can provide accurate results. However to provide the high level of detail it requires many computational resources, being this the main disadvantage.
**Literature Review**

**LWR (Lighthill, Whitman, Richards)** - originally created by Lighthill and Whitman [LW55] and then improved by Richards [Ric56], it is a time-continuum model based on the equations of mass conservation, using only a single continuity equation:

\[
\frac{\partial}{\partial t} \rho(x,t) + \frac{\partial}{\partial x} q(x,t) = 0
\]  

(2.3)

The variables used by the equation are the macroscopic variables: the average density, denoted by the formula \(\rho = \rho(x,t)\); and the average velocity of the vehicles, denoted by the formula \(v = v(x,t)\). The combination of these two formulas allows to obtain a formula to represent the traffic flow-rate, which is: \(q(x,t) = (x,t)v(x,t)\).

It is efficient when applied along road links rather intersections, where its performance and accuracy drops to lower levels. [RM13]

**S Model** - this model is a simplified version of BLX model, developed by Lin [LSXH09] with the goal of having a model that would require less computational resources. The main differences are in the cycle time used for simulation, that is bigger than in BLX model, allowing less computing consumption. Also, the data used in a queue is the average flow rate of vehicles, instead of actual number of vehicles. With these changes some accuracy in the forecasts is lost but a more feasible algorithm is obtained. [LDXH12]

**Microscopic Approach**

This model has the opposite methodology of the macroscopic one, by using each vehicle data (speed, position, direction) to analyse traffic, providing a detailed traffic examination. It simulates the traffic conditions in a more realistic way, however requiring more computational resources than macroscopic and mesoscopic models.

It can be divided into two different models, that vary in the way the network is "seen": Car-following and Cellular Automata. The first one, Car-following, is a space-continuous simulation model where the traffic is described in terms of fluid behaviour [RM13] and which is based on the interactions between two consecutive vehicles, by describing the way a driver adjusts its velocity and position according to the vehicle in front of it (acting like the leading vehicle). [BRK11] The other model, Cellular Automata, is a space-discrete simulation model where the road is divided into cells which have a length equal as the minimum distance between two cars. A cell can be either empty or occupied by one vehicle, and a vehicle's motion is made between adjacent cells. Figure 2.2 illustrates the difference between both methods.

Thus, the microscopic techniques identified during the literature review are:

**Gipps** - it is a car-following model developed by Gipps [Gip81], that aims to define a collision-free model, however still realistic, by focusing on the different behaviours that a driver has, as well the limitations of a vehicle, and using those behaviours as constraints in the model. These
Figure 2.2: Comparison between car-following and cellular automata models [KHRW02]

constraints lead to a maximum safe speed where the driver drives as fast as the vehicle can and without the risk of collision with the vehicle ahead. Initially the algorithm makes the calculation of the acceleration of a vehicle \( n \) at time \( t + \tau \), through the formula:

\[
a_n(t + \tau) = \frac{l_n [v_{n-1}(t) - v_n(t)]^k}{[x_{n-1}(t) - x_n(t)]^m} \tag{2.4}
\]

Using the previous formula as a starting point, the model is then defined through a set of constraints:

- \( a_n \): is maximum acceleration which the driver of vehicle \( n \) desires to achieve;
- \( b_n \): is the most severe braking that the driver of vehicle \( n \) desires to achieve (\( b_n < 0 \));
- \( s_n \): is the length of the vehicle plus a margin where the following vehicle is not allowed to invade;
- \( V_n \): is the velocity of vehicle \( n \) at which the driver desires to go;
- \( x_n(t) \): is the location of the vehicle \( n \) at the time \( t \);
- \( v_n(t) \): is the velocity of vehicle \( n \) at the time \( t \);
- \( \tau \): is the reaction time, which is constant to all vehicles.

A vehicle \( n \) will not exceed the driver’s desired speed, its free acceleration is increased first with the speed and then decreases to zero as the vehicle acquires the desired velocity:

\[
v_n(t + \tau) \leq v_n(t) + 2.5a_n\tau(1 - v_n(t)/V_n)(0.025 + v_n(t)/V_n)^{1/2} \tag{2.5}
\]

When a vehicle \( n - 1 \) starts to brake as hard as desirable at time \( t \), it will be at position \( x_{n-1}^* \):

\[
x_{n-1}^* = x_{n-1}(t) - v_{n-1}(t)^2/2b_{n-1} \tag{2.6}
\]

where \( b_{n-1} \) is negative.

With the previous constraint, the vehicle \( n \) moving right behind will not react until \( t + \tau \), so it will not stop before reaching the point \( x_n^* \):

\[
x_n^* = x_n(t) + [v_n(t) + v_n(t + \tau)]\tau/2 - v_n(t + \tau)^2/2b_n \tag{2.7}
\]
It is also considered a safety margin for braking by supposing that a driver might not be able to brake within the time. Thus, it is defined a braking constraint that considers the true reaction time $\tau$ and the safety reaction time $\tau + \theta$:

$$x_{n-1}(t) - v_{n-1}(t)^2/2b_{n-1} - s_{n-1} \geq x_n(t) + \left[1 + \frac{v_n(t) + v_n(t + \tau)}{\tau}ight]/2, v_n(t + \tau)\theta - v_n(t + \tau)^2/2b_{n-1}$$

(2.8)

From the previous formula, a driver in a real life situation can evaluate by observation the safety braking distance, but not $b_{n-1}$. Thus $b_{n-1}$ is replaced by $\hat{b}$ which is an estimate value, being the new formula:

$$-v_n(t + \tau)^2/2b_n + v_n(t + \tau)(\tau/2 + \theta) - [x_{n-1}(t) - s_{n-1} - x_n(t)] + v_n(t)\tau/2 + v_{n-1}(t)^2/2\hat{b} \leq 0$$

(2.9)

Moreover, if the delay of braking, $\theta$ is equal to $\tau/2$ and if the desired of previous driver to brake hard was not underestimated, then a vehicle moving at a safe speed and distance is able to keep in a safety state indefinitely. With that, the velocity equation can be rewritten in:

$$-v_n(t + \tau)^2/2b_n + v_n(t + \tau)\tau - [x_{n-1}(t) - s_{n-1} - x_n(t)] + v_n(t)\tau/2 + v_{n-1}(t)^2/2\hat{b} \leq 0$$

(2.10)

Summing up, the final equation for the new velocity of a vehicle $n$ is:

$$v_n(t + \tau) = \min\{v_n(t) + 2.5a_n(1 - v_n(t)/V_n)(0.025 + v_n(t)/V_n)^{1/2}, b_n(t) + \sqrt{b_n^2\tau^2 - b_n[2[x_{n-1}(t) - s_{n-1} - x_n(t)] - v_n(t)\tau - v_{n-1}(t)^2/\hat{b}]}\}$$

**IDM (Intelligent Driver Model)** - it was developed by Treiber, Hennecke and Helbing [THH00], and it is a car-following model, where the traffic conditions are defined by the data of all vehicles in the network. That data are: the position, velocity of the vehicle, vehicle’s length, as well the lane where the vehicle moves. By focusing on each vehicle behaviour it is highly depend on vehicle’s movement and the interaction with other vehicles. The vehicle’s acceleration can be denoted in a simplified way by the equation [BRK11]:

$$a_{n+1}(t) = a\left(1 - \frac{v_{n+1}(t)}{V_1}\right)^\delta - \left(S^s_{n+1}(v_{n+1}, S^s_{n+1})/s_{n+1}(t)\right)^2$$

(2.11)

where

$$S^s_{n+1}(t) = z_0 + z_1\sqrt{\frac{v_{n+1}(t)}{V_1}} + T_{min}v_{n+1}(t) + \frac{v_{n+1}(t)s^s_{n+1}}{2\sqrt{ab}}$$

(2.12)

$v_{n+1}$: the free speed of the vehicle

$\delta$: the maximum acceleration

$T_{min}$: the minimum time headway

$z_0$ and $z_1$: the minimum space headway and stationary distance

$b$: the comfortable deceleration
Literature Review

δ: the exponent acceleration

**Linear** - it is a car-following model as well, where the acceleration of the vehicle in the back is an additive function that combines relative speed, space headway and the velocity of the following vehicle [BRK11]. It is defined by the equation:

\[\ddot{x}_{n+1}(t) = c_1(\dot{x}_n(t) - \dot{x}_{n+1}(t)) + c_2(x_n(t) - x_{n+1}(t) - L - h\dot{x}_{n+1}(t))\]  

(2.13)

\((c_1, c_2)\): the control parameters of speed and spacing;

\(L\): the length of a vehicle;

\(h\): the minimum time headway;

\(x_n\) and \(x_{n+1}\): the positions relative to points of reference.

**Nagel-Schreckenberg Model** - it is one of the most used "cellular automata" algorithms that was originally defined by Nagel and Schreckenberg [NS92]. The division of the road into cells is the basis of the algorithm, where the vehicle’s velocity is measured by cells per time step, and it is in a range from 0 to \(v_{max}\). At each time step in the simulation cycle, the motion of a vehicle \(i\) is defined by four update rules [SNW01]:

- **Acceleration**: when the velocity \(v\) of a vehicle is lower than \(v_{max}\) and the distance to next vehicle is greater than \(v + 1\), the speed is increased by one \([v \rightarrow v + 1]\).

- **Deceleration**: when a vehicle is at cell \(i\), and the next vehicle’s position is \(i + j\) (with \(j \leq v\)), its speed is decreased to \(j - 1\) \([v \rightarrow j - 1]\).

- **Randomisation**: with a certain probability \(p\), the vehicle’s velocity is decreased by one \([v \rightarrow v - 1]\).

- **Motion**: a vehicle moves \(v\) cells

These rules of simulation are applied at the same time to all vehicles of network.

**Mesoscopic Approach**

This model is a combination between macroscopic and microscopic approach, where the traffic flow is described with high detail, like microscopic approach, and the flow interactions and behaviour are defined in a low level of detail, similar to macroscopic techniques. This more detailed way allows to better depict the traffic conditions than macroscopic models, and on the other hand consuming less computational resources than microscopic models.

This kind of approach was first presented by Tolujew and Alcala [TA04] as a pedestrian flow simulation model, and by Savrasovs and Tolujew [ST07] as queue system, and to summarize it
the forecasting is made through the use of mathematical formulas, continuous functions, in every step $\Delta t$. Comparing with microscopic approach, the difference is that it controls the traffic data by grouping vehicles (acting like in a queue) instead of analysing one vehicle each time. And it differs from macroscopic technique in the way that it can analyse several groups of vehicles at the same time, their behaviour and the existing interactions among them.

**Nanoscopic Approach**

It is a model that has a higher level of detail than the microscopic approach, focusing not only in each vehicle-driver pair in the network but also in the structure of the vehicle such as the type of engine, the brakes characteristics or even the fuel consumption. This approach explores in a further way the three elements used by microscopic simulation: the vehicle modelling, the vehicle movement modelling, and modelling of the driver’s behaviour [DP08]. Generally this high level of detail is not necessary for the majority of traffic simulations, from the fact that the benefits between the level of detail and the computational consumption do not make it worth. Therefore, it is more suited to use in small systems, especially in robotic and automation applications domain.

### 2.1.2 Data-driven models

The data-driven model used in this dissertation is the Artificial Neural Networks (ANNs) combined with Fuzzy logic, taking advantage of the good capacity that a neural network has to define a pattern and behaviour of some data set, and the ability of Fuzzy logic to normalize values that are out of the desired range.

An artificial neural network is a machine-learning technique based on the human nervous system, aiming to function and learning like the human brain does through the synapses between neurons. Thus, it is composed by a set of connected nodes (neurons) divided into three layers: input, hidden and output, interacting among each others and producing the learning network. The input layer is used to introduce the values into the network, then the hidden layer is responsible to connect the neurons from input and output layers, having an important role during the learning process, by measuring and combining the weights of the nodes; and finally the result acquired from the training stage is given through the output layer. Each connection between the nodes has an associated weight which defines the importance of the input values for the neuron, that is used in the learning phase by the activation function, the function in charge for computing the input values and produce the resulting knowledge [RN10]. An example of a neural network structure is presented in figure 2.3.

When applied to traffic forecasting, an ANN uses the historical data from past forecasts as an input to learn (train) how the network’s traffic normally behaves, building up a knowledge base. Then, it forecasts the traffic values for the next time interval by receiving the traffic data in the current moment. A great asset of ANNs is the good capacity for identify patterns, making this data-driven technique a useful tool to be used in traffic forecasting and to determine patterns (eg: if a road is congested or not) from traffic data [Dou95].
About fuzzy logic, it is a probabilistic reasoning technique first presented by [Zad65], which considers data as knowledge under uncertain domain, making use of vagueness and focusing on the way an object belongs to a vague description group. This approach is suitable to use with non-deterministic challenges.

2.2 Related Works

Once introduced in the previous section a description about the types of simulation and data-driven techniques to be used in the thesis context, this section presents the past projects developed whose field of study is related to the subject covered by this project.

2.2.1 Macroscopic approach

This paper [HB05] presents a new extension of the macroscopic model Aw & Rascle, that aims to improve the efficiency of the original model when applied on junction nodes of roads. The BLX and the S model were the target of a comparison with the methodology MPC (Model Predictive Control) made by [LDXH12]. In the paper [RM13] it is described a model that was the result of the combination between the macroscopic model LWR and the machine-learning technique of Artificial Neural Network, showing that it is feasible to combine model and data-driven techniques and apply it to traffic forecasting.

2.2.2 Microscopic approach

Microscopic techniques have been widely used among traffic forecasting projects. The papers [ES97] and [SNW01] focused on the creation of a simulation tool based on the Nagel-Schreckenberg Cellular Automata model that would improve the accuracy of the original one. A comparison between two microscopic models, the IDM (Intelligent Driver Model) and Linear was made by [BRK11], in order to identify the strengths and weakness of each, and it concluded that there is not...
such a model suitable to describe fully the traffic state, varying from each case in study. Another test and comparison was made by [HHST02], but this time between the microscopic car-following model IDM and a macroscopic technique.

An innovative hybrid model tested on a Tokyo motorway was brought by [KFT05], that combined the microscopic technique of Cellular Automata with the data-driven approach of Fuzzy Clustering to forecast in short-term. In [TLJ10] a road network data model of Memphis city was developed using the simulation technique IDM. Also, in [RI12] a comparison between two microscopic simulators, AIMSUM and SUMO was made, using a junction node of a motorway in Stockholm.

2.2.3 Mesoscopic approach

About the use of mesoscopic techniques among traffic forecasting projects, there is the work developed in the paper [Sav11] where an example of using mesoscopic approach combined with a technique commonly used in logistics, called discrete-rate approach, was implemented using an urban scenario based in the map of Riga city. The paper [Sav12] focus also in the analysis of that subject, presenting some examples.

2.2.4 Nanoscopic approach

A nanoscopic approach is more suitable for automation and robotic applications than traffic forecasting. An example of it is the simulation model presented in [MSK11], which is a nanoscopic approach to simulate the power-train of an electric bus. The framework is composed by a nanoscopic model which is used to depict the power-train of the bus which is coupled to the microscopic SUMO simulator, responsible for depicting the surrounding traffic scenario (the other vehicles in the network). This highly detailed simulation allows to depict the bus in a great realistic way, by considering all the characteristics of the vehicle, for instance the engine efficiency, the friction forces against the vehicle, the vehicle’s mass or the braking energy generated. In paper [DP08] it was presented an innovative agent-based model, focused on driver’s behaviour, to be used in nanoscopic simulation. The model makes use of fuzzy neural networks to depict the human decisions made when a person is driving a vehicle. Also, an approach over the potential advantages of using nanoscopic techniques was carried out by the study presented in the paper [Ni06].

2.2.5 Artificial Neural Networks

The use of neural networks in traffic forecasting has become a prominent option among the research community since 1990s, when the papers [DKB93] and [DKB94] presented the first studies about using neural networks to recognize and detect traffic patterns. Also, in paper [DC97] it was explored the development of a model to predict traffic on motorways using neural networks. [Dia01] also tested the use of ANNs in short-term (up to 15 minutes) traffic forecasting on a short segment from a motorway in Australia. In Turkey, [BB10] focused to predict traffic in mid-term (15 minutes to 1 hour) on a congested urban scenario. [Lop12] deeply explored the use of ANNs
to forecast traffic anomalies (caused by incidents) on a motorway, by testing different training techniques like back-propagation, resilient back-propagation and radial basis function (RBF); also experimented different architectures for the network (Linear, Space-based and Time-based).

Furthermore, as referred before, more recently it has been developed hybrid models that combine ANNs with macroscopic techniques in order to predict traffic. A good example of it is the work developed by [FC13], that created an integrated model that uses ANN to predict the traffic state under normal conditions, and the second-order macroscopic algorithm Aw & Rascle to forecast the traffic conditions when an anomaly occurs. A slight different model was presented by [RM13], where an ANN is used to get trained from the historical data and its output is then simulated using the macroscopic model LWR.

2.2.6 Fuzzy logic

Along with the use of ANNs in traffic prediction, several papers have recently focused on testing the combination of ANNs with fuzzy logic, and taking advantage of the capacity to cluster data by pattern sets. An example is the model presented by [YWXW02], where a neural network system combined with fuzzy logic was used to predict traffic on urban roads. The fuzzy logic was used within a model that was in charge for clustering the input data into groups according to traffic patterns, and then pass the clustered sets as an input to the ANN. In paper [ZY08] the fuzzy logic was used in a different way, since the input data for the fuzzy model was the error rate of time-series forecasting methods. Basically the traffic was forecasted using two different time-series techniques and the combination of the obtained error rates was categorized into a fuzzy set.

The application of fuzzy sets in traffic domain can be not only to forecast traffic but also to incidents detection like the model created by [XKHP98], where an adaptive on-line fuzzy neural network was developed to identify traffic incidents. A similar project was later presented in [XHM08], where it was implemented a fuzzy logic-based system to detect incidents on a motorway, by using the velocity and its variation between adjacent measures to group data and then infer the probability of an incident to occur. Another example of a different application of fuzzy logic is the model created by [CQL09] for traffic control. The model consists in an intelligent traffic lights control system that builds its knowledge base from the traffic state that is characterized by fuzzy sets.

2.3 Summary

Once finished the bibliographic review, in this chapter it was described the four existing simulation techniques, the advantages and disadvantages of their usage, as well the running procedure behind neural networks and fuzzy logic, and the potential benefits when applied to traffic forecasting. It was also made a presentation of the state-of-art, and what has been developed so far at each of this subjects of simulation and data-driven domains. After making the state-of-art review, it is stated that there is open-space to explore in the area of traffic forecasting on road networks, which is the subject of this project.
Chapter 3

Methodological Approach

This chapter starts by presenting a detailed description of the problem, and then exploring the analysis made to identify the requirements. Afterwards, the proposed solution for the implementation of the model- and data-driven forecasting is introduced, presenting the plan and the way to do it. In the end, it is shown an overview of the technologies to be used.

3.1 Problem Statement

Traffic prediction in real-time on road networks has acquired high importance in the current ITS, due to the fact that if there is the chance of being able to predict, with a fair degree of confidence, the state of network in a close future that would lead to an improvement in the performance of road networks management. Basically, the existing techniques to implement traffic forecasting are through the use of simulation or data-driven approaches.

This dissertation aims to develop and test some techniques from these two different approaches, in order to analyse the benefits of applying traffic forecasting on ITSs. Hence, the selected simulation technique was the microscopic approach because, despite the high computational consumption that it might require, it is the approach that allows to simulate traffic and its behaviour in a more realistic way. Regarding data-driven approaches, the option made was to use artificial neural networks (ANN) with fuzzy logic, since it is a model that can be trained and "designed" according to the actual data, and when properly trained it has a high level of reliability [RM13]. To conclude, both approaches will be tested using real data and their performance compared.

3.2 Requirement Analysis

Before starting the implementation, it was necessary to identify the technology requirements for this thesis. Starting with the traffic simulation, the traffic simulator chosen was SUMO, which is one of most used microscopic simulators among research community, and the fact that is
Methodological Approach

lightweight and open-source made it the best option. The road network used was the Porto motorway, Via de Cintura Interna (VCI), and a further description of it can be found in section 4.1. Its map was loaded using the file format OpenStreetMap (.osm extension), through the open-source maps service OpenStreetMaps [Fou14]. An edition of the map was made, to correct the number of lanes in some road segments, add some missing traffic signals and information in some road sections. This edition process was made using the JOSM (Java OpenStreetMap) software, an editor for maps with .osm format.

To develop the artificial neural network, it was decided to use the machine-learning framework Encog, an open-source tool that provides several training methods and architectures for implementation. The version used was the one for Java, through the existing plugin for the IDE (Integrated Development Environment) Eclipse. To conclude, for testing both models it was used actual data, collected from real traffic detectors placed in VCI, and provided by Armis.

3.3 Proposed Solution

3.3.1 Forecasting with Simulation

As referred before, for implementing the traffic forecasting using simulation it was concluded that the microscopic approach was the model more suitable for this thesis context, since it allows to depict traffic in a very deep and realistic way. Thus, for testing the simulation model it was decided to use a test-bed area, with small dimensions, that consisted in a segment of the complete network. The selected area was the Paranhos junction node, which is part of VCI motorway, and its access ramps. This node was considered to be a good option for testing, from the fact that is a generic node, located in a busy area of Porto with a high density of traffic, that serves as an access to an university campus, a hospital, a wide residential area, also as a route in direction of city’s centre.

Regarding the data used for test, it was gathered a set of measures corresponding to the traffic state during rush-hour periods from normal days i.e. where there was no reported incidents neither it was weekend or holidays season. A detail description can be seen in chapter 5.

Therefore, the implementation phase was defined to be composed of four phases:

1. Conversion of the map from OpenStreetMap format (.osm) to SUMO format network (.net.xml) and making the necessary adjustments in some road segments;
2. Definition of traffic demand that would be used as input for the simulation;
3. Execution and test of the developed solution;
4. Calibration and validation of the model.

3.3.2 Forecasting with Artificial Neural Networks

About the implementation of forecasting using artificial neural networks it was also decided to develop and test it with the selected test-bed area of Paranhos junction node. Concerning the
structure of the network, the ANN was constructed with the Multi-layer feed-forward model, which is a multi-layer version of perceptron network model and uses supervised learning to train the network. The learning technique chosen was the supervised training method called Resilient Back-propagation. This decision was made regarding the conclusion presented by [Hea09] where it is proved that generally a resilient propagation is easier to implement, due to it does not require the definition of extra training parameters and it has a better performance during the training stage.

For defining the architecture of the neural network, it was taken in account the conclusions obtained from the test made by [Lop12]. In this paper a comparison among three different architectures: linear, space-based and time-based was made; and the space-based architecture proved to be the best for speed prediction which is the type of measure that will be used in this forecasting model. Thus, considering all this factors, the chosen architecture was the space-based model, defining an ANN for each way of the motorway, one for the traffic in direction of West and the other towards East.

In respect to the inputs and outputs of the network, the ANN receives thirteen inputs and produces one value as output, corresponding to the predicted speed for the next time interval \((t + 1)\). A detailed definition of the inputs is presented in table 3.1. In that table it is shown that the ANN receives the five last measures which, considering a time interval of 5 minutes, corresponds to the measures obtained in the previous 25 minutes before the current moment.

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>weekday</td>
<td>from 1 to 7 corresponding to the day of the week (1-Monday, 7- Sunday)</td>
</tr>
<tr>
<td>holiday</td>
<td>a flag that can be 0 or 1, corresponding to either the day is a holiday or not</td>
</tr>
<tr>
<td>holidays season</td>
<td>a flag that can be 0 or 1, corresponding to either is holidays season or not</td>
</tr>
<tr>
<td>day</td>
<td>from 1 to 31 corresponding to the day of the month</td>
</tr>
<tr>
<td>month</td>
<td>from 1 to 12 corresponding to the month</td>
</tr>
<tr>
<td>hour</td>
<td>from 0 to 23 corresponding to the hour</td>
</tr>
<tr>
<td>minute</td>
<td>from 0 to 59 corresponding to the minutes</td>
</tr>
<tr>
<td>average speed (t-5)</td>
<td>from 0 to 130 corresponding to fifth past measure</td>
</tr>
<tr>
<td>average speed (t-4)</td>
<td>from 0 to 130 corresponding to fourth past measure</td>
</tr>
<tr>
<td>average speed (t-3)</td>
<td>from 0 to 130 corresponding to third past measure</td>
</tr>
<tr>
<td>average speed (t-2)</td>
<td>from 0 to 130 corresponding to second past measure</td>
</tr>
<tr>
<td>average speed (t-1)</td>
<td>from 0 to 130 corresponding to first past measure</td>
</tr>
<tr>
<td>average speed (t)</td>
<td>from 0 to 130 corresponding to current measure</td>
</tr>
<tr>
<td>(t)</td>
<td>current moment of time</td>
</tr>
</tbody>
</table>

Table 3.1: Description of inputs used by the neural network

Besides this base neural network, it was also decided to develop an ANN with fuzzy logic, using the fuzzy sets for grouping the speed values that are used as input, into categories that represent a textual description of traffic state. The fuzzification was classified into four groups, each one representing a state of traffic, composed by a range of velocities and the corresponding input value given to the network. Table 3.2 describes the defined fuzzy sets.
Methodological Approach

<table>
<thead>
<tr>
<th>Traffic state</th>
<th>Range of values</th>
<th>Input value for ANN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stopped (red colour)</td>
<td>speed within 0-40 km/h</td>
<td>1</td>
</tr>
<tr>
<td>Stop-and-go (orange colour)</td>
<td>speed within 40-55 km/h</td>
<td>2</td>
</tr>
<tr>
<td>Slow (yellow colour)</td>
<td>speed within 55-70 km/h</td>
<td>3</td>
</tr>
<tr>
<td>Normal (green colour)</td>
<td>speed within 70-∞ km/h</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 3.2: Description of fuzzy sets for velocity input

For training the ANN it was decided to use four weeks of normal traffic data (from Monday to Sunday) without incidents to train the network, and one week of data to test it. The data used was desired to represent four weeks (28 days) of normal traffic conditions, in order to train the network with the common patterns of traffics. For that reason it should be noted that the data used was not necessarily from a continuous period of days in a week, from the fact that in some weeks there were days where incidents happened. Thus, the training set was composed by four entries of each day of week, composing a four weeks dataset. The traffic data belongs to the measures collected from VCI motorway during April and May of 2013.

To finalize, the development of the neural network was structured into six phases:

1. Implementation of the neural network;
2. Execution and test of the developed solution (by varying the number of neurons and hidden layers used);
3. Calibration and validation of the model;
4. Implementation of the neural network with fuzzy logic;
5. Execution and test of the developed solution (by varying the number of neurons and hidden layers used);
6. Calibration and validation of the model;

3.4 Technology Overview

3.4.1 SUMO (Simulation of Urban MObility)

SUMO is a microscopic open-source traffic simulator created by the German Aerospace Centre (DLR) in 2001, with the goal to provide a tool that would be fast in run-time and portable, as well able to support new algorithms developed by its users [KHRW02]. It is currently in the version 0.20 and it has been widely used among traffic simulation community. An important feature is the fact that SUMO is a multi-modal simulator, being able to not only depict the cars’ movements but also the other kind of vehicles that usually circulate within a city, like motorcycles, bicycles or pedestrians, as well public transport vehicles: bus, train, tram or taxi. The simulator is available
Methodological Approach

through command-line or through a GUI (Graphical User Interface) for those who need a graphical presentation.

The microscopic algorithm used is the car-following model extension of Gipps algorithm, developed by Krauß and Janz [Kra98], which is a collision-free model since a vehicle adapts its velocity at each simulation step according to the speed of the leading vehicle (the vehicle that is ahead). The vehicle’s speed is defined using two auxiliary equations. The first one is the safe velocity, which is the maximum speed the vehicle can go in order to not collide with the vehicle ahead:

\[ v_{safe}(t) = v_l(t) + \frac{g(t) - v_l(t)\tau}{\bar{v} + \tau} \]  

(3.1)

\( v_l(t) \): speed of the leading vehicle in time t;

\( g(t) \): gap to the leading vehicle in time t;

\( \tau \): the driver’s reaction time (usually 1s);

\( b \): the deceleration function.

Then there is the desired velocity, which is the speed the driver wants to have:

\[ v_{des}(t) = \min[v_{safe}, v(t) + a, v_{max}] \]  

(3.2)

And the final velocity is calculated through a third equation that binds a random factor, in order to simulate the fact that the driver might not be able to achieve the desired speed.

\[ v(t) = \max[0, \text{rand}[v_{des}(t) - a, v_{des}(t)]] \]  

(3.3)

The format of the files used by SUMO is XML-based, varying on the extension used according to the type of file. For instance, the configuration file for the simulation is saved with a .sumo.cfg extension; the network is defined into a file using .net.xml extension; or even the routes file which uses the .rou.xml extension. Besides the simulator, SUMO has also a set of small applications included in its package that work as auxiliary tools for tasks such as: generating the network to be used (netconvert, netgenerate), defining and computing the routes of vehicles (darouter, jtrrouter, dfrouter), inputting the demand of vehicles in the simulation (od2trips), as well for other uses (eg: polyconvert, edgesinDistrict). [KHWR02] [KHH*05]

The network to be used in simulation can be generate randomly with netgenerate or by using an existing map either previously defined "by hand" or from a maps service like OpenStreetMaps. In this case the application used is netconvert which is an importer that reads a network definition from a file and can be in a variety of formats.

To define the traffic demand there are three different generators: the first, duarouter, is the simplest since it uses the shortest path computation as the way to generate the routes, being the Dijkstra model the algorithm used. There is also the option to use A* algorithm instead. The application jtrouter uses the turning percentages at junctions to compute the routes, being suitable to use in a urban road network scenario. There is also the dfrouter which computes the routes by
Methodological Approach

using actual data collected from induction loop detectors and which is inserted into the network through a few "source points". Those "source points" represent the actual existing detectors. This is suitable in a motorway scenario and when there are induction loop detectors installed in the real network, that can provide a real and accurate set of data. [BBEK11]

3.4.2 JOSM (Java OpenStreetMap Editor)

JOSM is an open-source desktop editor for maps with the file format OpenStreetMaps (with .osm extension), written in Java, and created by Immanuel Scholz [Sto14]. Is is currently in the version 7000. The purpose of using this software in the thesis was to edit the map that would serve as basis for generating the network of VCI motorway which was used in the forecasting with simulation.

3.4.3 Encog Machine Learning Framework

Encog is a machine learning framework created by Jeff Heaton through its research institute, Heaton Research Inc., in 2008 [Hea13]. It has versions for the programming languages: C/C++, C# and Java; and it supports several artificial intelligence algorithms, such as Bayesian Networks, Genetic Algorithms, Neural Networks, Support Vector Machines.

In this thesis context it was only used the neural network feature for Java, which is currently in the version 3.1.0. This neural network module supports both Supervised and Unsupervised learning, and allows the implementation of different types of architecture like ADALINE Neural Network, Feedforward Neural Network (Perceptron), Radial Basis Function Network (RBF). As well, several training techniques are available such as: Backpropagation, Resilient Propagation (RProp), Levenberg–Marquardt or Manhattan Update Rule Propagation.

3.5 Summary

Initially in this chapter it was presented the statement of the problem and the goals of this thesis, as well the analysis of the requirements, which revealed to be crucial in order to obtain a robust plan and set of procedures for the implementation phase. Also, a brief but concise explanation of the proposed solution for the two models was made, providing an overall view of the chosen methods. Before finish the covering of the adopted methodologies, it was presented an overview of the technologies used, with a definition of its characteristics and features.
Chapter 4

Implementation

This chapter presents a detailed view about the steps and decisions made during the implementation process, which was divided into two phases: firstly the goal was to implement the traffic forecasting techniques using the microscopic simulation model (model-driven stage), and secondly the traffic forecast would be made using Artificial Neural Networks with Fuzzy logic (data-driven stage).

4.1 Network Description

The network used as base for implementing both simulation and data-driven models was the Via de Cintura Interna (VCI) motorway, in particular one of its junctions nodes called Paranhos, which served as test-bed for the initial developments made. VCI is a main urban motorway located in Porto metropolitan area with an extension of 21 kilometres. It has the shape of a ring which passes through East, North and West parts of Porto city and the North part of Gaia, being composed by twenty junctions nodes and connected to six inter-urban motorways (A1, A3, A20, A28 A43 and A44). The motorway is covered by several induction-loop detectors, property of the Portuguese Traffic Authority - Estradas de Portugal\(^1\), which help to control and evaluate the traffic flows.

Thus, in this thesis context it was used the data collected by the traffic detectors located near the Paranhos junction node. A network map, containing the junction nodes and traffic detectors location, can be seen in figure 4.1. As referred before, the implementations and tests were made using a test-bed based on the Paranhos junction node, which is covered by two traffic detectors, both located in the junction node boundaries of the motorway. This node is connected to three access streets, two of them serving as origin and destination, and another that works only as destination for the vehicles which come from VCI. Its representation can be seen in figure 4.2.

\(^1\)http://www.estradasdeportugal.pt/
Implementation

Figure 4.1: Map of VCI network with the existing nodes and traffic detectors location

Figure 4.2: Map of Paranhos junction node with traffic detectors location
Implementation

4.2 Forecasting with Simulation

As said in chapter 3 the implementation of forecasting with simulation was divided into four stages, starting by creating and using a test-bed area, that would depict the Paranhos junction node of VCI motorway, and its access ramps. An aerial view of the test-bed area can be seen in figure 4.3.

Thus, the first step of implementation was to convert the map of test-bed area from the OpenStreetMap format to a format readable for SUMO. This conversion was made with the SUMO application netconvert which allows to read a map from .osm format and convert it to a format of the network file used by SUMO (.net.xml). In order to get the network properly created, the following options were used: --ramps.guess and --ramps.min-highway-speed 25.00 which informs the application to create access ramps for the motorway by guessing it through the topology and the minimum speed defined on the road file; --junctions.join which was used to combine the existing traffic lights in the access roads; and the parameter --no-turnarounds which defines the prohibition of having u-turns on any road of the network.

Listing 4.1 presents the full command used with netconvert. For the sake of simplicity and since the focus area is primarily the motorway and its accesses, an existing crossing in the accesses in the north part of the area, called "Igreja de Paranhos", was simplified and considered to be just one single road instead of the three actual streets. In the figure 4.4 it is shown the resulting network loaded in SUMO.

```
netconvert.exe --osm-files map_v2_1_1.osm --output-file paranhos.net.xml --ramps.guess --ramps.min-highway-speed 25.00 --junctions.join --no-turnarounds
```

Listing 4.1: Command to generate the test-bed network in SUMO format

Figure 4.3: Aerial view of test-bed area - Paranhos junction node

2 Image retrieved from http://www.flashearth.com
Implementation

After creating the network model for SUMO simulator, the next step was the definition of traffic demand. And for that, it was tested two ways for modelling the traffic demand into the simulation model according to the formats accepted by SUMO. One of the techniques is through the use of Origin-Destination (OD) matrices, which is a traffic demand definition technique deeply used not only in traffic forecast but in the whole transportation area. This way of modelling traffic has the advantage of allowing a detailed definition of traffic and the existing generic routes. However, in order to have this level of detail it is necessary to have a full description of traffic from the data measured by traffic detectors. Therefore, for passing the data to the simulator using OD matrices it is necessary to firstly define the zones within the network that represent each row (district) of the OD matrix, which are designed by traffic assignment zone (TAZ). This TAZ definition is made using the XML-based file format called .taz.xml, and to simplify the process it was used the SUMO auxiliary tool edgesInDistricts.py, which is a program that loads a file in the format .poi.xml containing the TAZ definition through x,y coordinates in the map, and converts into the format .taz.xml. The complete command used is presented in the listing 4.2, which resulted in the creation of a file containing the taz definition (districts.taz.xml).

```
edgesInDistricts.py -n paranhos.net.xml,taz.poi.xml --verbose
```

Listing 4.2: Command to generate the districts from an OD matrix

With the OD districts defined, the next step was to generate the vehicles' trips from the traffic demand imported from an OD matrix. SUMO provides an useful application for importing an OD matrix called OD2trips. It accepts two ways of passing the matrix, and the chosen method was the simplest which is called O-format and can be defined using a text file. An example of the OD matrix file is presented by listing 4.3. Thus, the matrix importation was made using the command defined in listing 4.4.

```
1 OR;D2
2 * From-Time To-Time
3 0.00 1.00
```
Implementation

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Factor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>comments</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>2</td>
<td>188.00</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>3</td>
<td>4032.00</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>1</td>
<td>118.00</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>3</td>
<td>217.00</td>
</tr>
</tbody>
</table>

Listing 4.3: Example of OD matrix passed to simulator in O-format

```
Listing 4.4: Command to import the traffic demand from an OD matrix

od2trips.exe -n districts.taz.xml -d od_matrix.txt -o trips.trips.xml --departlane best --departpos free --ignore-errors --verbose
```

Listing 4.4: Command to import the traffic demand from an OD matrix

The phase after importing was to convert the created trips file (`trips.trips.xml`) into a file of routes that will be used by the simulator to inject the vehicles during the simulation. This was done by using the application `duarouter.exe`, which converts OD trips to routes file (`routes.rou.xml`). The command used can be seen in the listing 4.5.

```
Listing 4.5: Command to generate vehicles’ routes

duarouter.exe -n paranhos.net.xml -d districts.taz.xml -t trips.trips.xml -o routes.rou.xml --ignore-errors --verbose --with-taz
```

Listing 4.5: Command to generate vehicles’ routes

After this steps were made, the simulation was ready to be executed and along with that several executions and a variety of tests were made, as can be found in detail, in the testing report presented in chapter 5.

The other tested method for modelling traffic demand, was through the use of `dfrouter` application. This application aims to depict the traffic scenario as in reality by using virtual detectors that emulate the existing real detectors on road networks. Thus, the vehicles demand is input through the virtual detectors, which should be located in the network as the actual position that the real detector is on the road. This characteristic is a promising feature to be used in a scenario where the real data is collected from traffic detectors since the traffic data is passed exactly how is measured by detectors. Nevertheless, as it was concluded during the testing phase (see chapter 5), this way of defining demand leads to that neither the trips nor the routes are pre-defined (unlike with an OD matrix) since the amount of vehicles and the average speed are the only parameters passed to the simulator. Further analysis and conclusions can be found in "Tests and Evaluation" chapter.

For implementing this method it was first necessary to define the virtual detectors, by identifying their position and the type of detector (source or sink) so that the simulator will "release" vehicles from source detectors in direction to sink detectors. The traffic demand has to be passed
Implementation

through a CSV (Comma Separated Value) file, containing the id of the detector, the time of the measure, the quantity of vehicles that have passed ($qpkw$) and the average speed ($vpkw$) in km/h. It must be passed one CSV file per each detector. The format of the file is present on listing 4.6.

```plaintext
1 detector;time;qpkw;vpkw
2 a3_c;0;73;78.95
3 a3_c;5;86;81.01
4 ....;....;....;...
```

Listing 4.6: Example of CSV file used to pass traffic demand to DFRouter

Finally, to generate the vehicles’ routes used by the simulation, it was executed the `dfrouter.exe` command with the following options: `--time-step`, to distribute the vehicles along the time interval; and `--disallowed-edges` to avoid the creation of routes that do not exist in real-life scenario. With this command it was generated a file containing the routes (`routes.rou.xml`), and an emitters file (`emitters.add.xml`) which will be responsible to create the vehicles during the simulation. The listing 4.7 shows the complete command.

```plaintext
1 dfrouter.exe --net-file paranhos.net.xml -d real_detectors.xml -f real_data/vals_paranhos_c.csv,real_data/vals_paranhos_d.csv,real_data/vals_faria_c.csv,real_data/vals_faria_d.csv,real_data/vals_igreja_c.csv,real_data/vals_igreja_d.csv,real_data/vals_a3_c.csv,real_data/vals_a3_d.csv --time-step 300 --disallowed-edges 13798242#1 --routes-output rotas.rou.xml --emitters-output emitters.add.xml --detectors-poi-output pois.add.xml
```

Listing 4.7: Command to generate vehicles using dfrouter.exe

After the development and test of the simulation model to predict traffic, it was made an experiment to emulate an incident scenario. For that two methods were used: stopping a vehicle on a lane during the simulation; and through the imposition of speed limits on the lanes and road segments located in the accident area. Thus, two vehicles were set to halt, which was made through the use of `stop` attribute on the vehicles definition, in the routes file `routes.rou.xml`, as it is presented in listing 4.8.

```xml
1 <vehicle id="34392">
2  <route>
3   <stop lane="29036509#2_0" endPos="297" duration="9000"/>
4  </route>
5 </vehicle>
```

Listing 4.8: Example of a definition to stop a vehicle during simulation

Afterwards, the implementation of the speed limits on the lanes was done using the Variable Speed Sign feature available on SUMO, which allows to set speed limitations for a defined period
Implementation

on any road. This is done with two types of files: one with the extension .add.xml which carries the
description of the speed signs and the lanes where they are allocated, and another with the
extension .def.xml containing the values of the speed limit and the time step when it is valid. An
eexample of each type of file is presented in listings 4.9 and 4.10 respectively.

```xml
<additional xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
noNamespaceSchemaLocation="http://sumo-sim.org/xsd/additional_file.xsd">
  <variableSpeedSign id="vss" lanes="29036509#2_0" file="vss_def.def.xml"/>
  <variableSpeedSign id="vss2" lanes="29036509#2_1" file="vss_def2.def.xml"/>
</additional>
```

Listing 4.9: Example of description of speed signs

```xml
<vss>
  <step time="200" speed="9.73"/>
  <step time="4800" speed="12.5"/>
</vss>
```

Listing 4.10: Example of definition of speed signs limit and time of duration

To conclude, once implemented the two variants for forecasting traffic using simulation, an
evaluation and analysis of their performance was carried out, which is presented in chapter 5.

4.3 Forecasting with Artificial Neural Networks

The implementation of the Artificial Neural Network (ANN) was made using the machine-learning
framework Encog for Java as it was said in section 3.3.2. Starting by defining the way to input the
data in the ANN, it was decided to do it by importing from a CSV file, containing the time of the
measure and the average speed measured. An example of the CSV file format used is shown in
listing 4.11.

```csv
<table>
<thead>
<tr>
<th>time</th>
<th>avg_speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013-04-15</td>
<td>84.79</td>
</tr>
<tr>
<td>2013-04-15</td>
<td>84.51</td>
</tr>
</tbody>
</table>
```

Listing 4.11: Example of CSV file loaded by the neural network

In order to do that it was created the function `LoadCSV(String filename, double weekday)` to
load the traffic data from a CSV file. This function receives the path of CSV file and a double
type variable corresponding to the day of the week when the measure was made, and it returns an
`ArrayList<LinkedList<Double>>`, containing all the inputs of each measure in a double type
format. In listing 4.12 is presented a excerpt of the function code. To properly load the imported
Implementation

data and convert it to an usable format from the ANN, it was developed an auxiliary function called `CSVtoArray(ArrayList<ArrayList<Double>> csv_array, int mode)`, which converts the loaded CSV file into a array of arrays of double type.

```java
String line = "";
String cvsSplitBy = ",";
...

while ((line = br.readLine()) != null)
{
    ArrayList<Double> line_values = new ArrayList<Double>();

    // use comma as separator
    String[] data = line.split(cvsSplitBy);
    String speed = data[1];

    String[] time = data[0].split(" ");
    String date = time[0];
    String[] date2 = date.split("-");
    String month = date2[1];
    String day = date2[2];

    String hours = time[1];
    String[] hours2 = hours.split(:");
    String hour = hours2[0];
    String minutes = hours2[1];

    line_values.add(weekday);
    line_values.add(Double.parseDouble(day));
    line_values.add(Double.parseDouble(month));
    line_values.add(Double.parseDouble(hour));
    line_values.add(Double.parseDouble(minutes));
    ...
```

Listing 4.12: Code fragment of LoadCSV function

Since the activation functions used have a restrict domain of values which work with, it was necessary to normalize the input values to be within the required interval, since they are in a variety of ranges. The interval chosen was \([-1, 1]\), which is the range of values of the Sigmoid activation function co-domain. To do that, a set of normalizing functions were created, making use of the Encog class `NormalizedField` which allows to create a normalize method by simply defining the interval values. The definition of those functions can be seen in listing 4.13.

```java
//Used to normalize day of the week from a range of 1-7 to 0-1.
```
With the dataset prepared to be read by the ANN, it was time to create the neural network. Hence, it was initially defined an ANN with one hidden layer composed by fifteen neurons plus a bias neuron to provide some adjustments on the weights of the neurons and help during the learning phase. Several tests were done using a different number of hidden layers as well by varying the number of neurons used. This was easily done by just adding a new layer or changing the value of the variable *hidden_neurons*. Listing 4.14 presents the code used to create the neural network, and the results obtained with the tests of different configurations of the ANN can be seen in chapter 5.

```
static int inputs_number = 13;
static int ideals_number = 1;
static int hidden_neurons = 15;
...
BasicNetwork network = new BasicNetwork();
network.addLayer(new BasicLayer(null, true, inputs_number));
network.addLayer(new BasicLayer(new ActivationSigmoid(), true, hidden_neurons));
network.addLayer(new BasicLayer(new ActivationSigmoid(), false, ideals_number));
network.getStructure().finalizeStructure();
network.reset();
```

Listing 4.14: Example of code for creating the neural network
Implementation

Once the neural network was created it was time to train it, and the train process was made, like referred before, with the learning method of resilient back-propagation. Therefore, it was constructed the training set, composed by the input values (variable `inputs_arr`) and the values that are expected to be obtained with the train (variable `ideals_arr`). The next step was to start the iteration of the learning process, which was set to execute until a error rate lower than 0.1% was reached. The code implemented is shown in listing 4.15. After finishing the training stage, the neural network was tested with a different traffic data set composed by a non-continuous series of seven days of measures. Listing 4.16 shows the code used for testing.

```java
MLDataSet trainingSet = new BasicMLDataSet(inputs_arr, ideals_arr);
final ResilientPropagation train = new ResilientPropagation(network,trainingSet);

int epoch = 1;
System.out.println("Starting training..!");

while (train.getError() > error_rate){
    train.iteration();
    System.out.println("Epoch #" + epoch + " Error:" + train.getError());
    epoch++;
} while(train.getError() > error_rate);
train.finishTraining();
```

Listing 4.15: Example of code to train the neural network

```java
MLDataSet evaluationSet = new BasicMLDataSet(inputs_arr_eval, ideals_arr_eval);
System.out.println("\nNeural Network Results:");
for (MLDataPair pair: evaluationSet )
{
    final MLData output = network.compute(pair.getInput());
    System.out.println(normH.deNormalize(pair.getInput().getData(5)) + ":" +
    normMin.deNormalize(pair.getInput().getData(6)) + ", actual=" + normSpeed.
    deNormalize(output.getData(4)) + ", ideal=" + normSpeed.deNormalize(pair.
    getIdeal().getData(4)));
}
```

Listing 4.16: Example of code to test the neural network

Following the planned implementation, after the development and test of the basic neural network the next step was to implement an ANN with fuzzy logic. As described in chapter 3 the use of fuzzy logic consisted in grouping the speed values into velocity range sets, where those input values would be fuzzyfied, then computed by the neural network and later the output would be defuzzyfied. In terms of implementation, the task is simply to change the way the speed values are normalized and subsequently passed to the network; and make a verification of in which range
the value is. Thus, the velocity value is normalized in a scale according to the range defined (see Table 3.2). The code used for the fuzzyfication is present in listing 4.17

```java
public static NormalizedField normSpeedFuzzy = new NormalizedField(
    NormalizationAction.Normalize, "speed_fuzzy", 4, 1, 1, 0);
...
if (elem.get(5) > 0 && elem.get(5) <= 40)
{
    inputs_arr[ind][12] = normSpeedFuzzy.normalize(1.0);
}
else if (elem.get(5) > 40 && elem.get(5) <= 55)
{
    inputs_arr[ind][12] = normSpeedFuzzy.normalize(2.0);
}
else if (elem.get(5) > 55 && elem.get(5) <= 70)
{
    inputs_arr[ind][12] = normSpeedFuzzy.normalize(3.0);
}
else if (elem.get(5) > 70 && elem.get(5) <= 120)
{
    inputs_arr[ind][12] = normSpeedFuzzy.normalize(4.0);
}
```

Listing 4.17: Example of code to implement fuzzy logic

As it was done in the traffic forecast using simulation, after developing the normal neural network and the version with fuzzy logic, it was implemented an ANN aimed to detect the traffic patterns when an incident occurs. This model uses the data from two adjacent detectors in order to learn the correlation of the traffic state between both detectors, and it was created using the same structure of the normal neural network. Thus, besides considering the last five measures and the last measure as input, this model also considers the last five measures and last measure of the second detector, having in total 19 inputs. There was also a change in the outputs, with the network producing 2 outputs, one corresponding to the traffic forecast for the next time interval in the first detector, and the other matching to forecast of traffic for the next time interval in the second detector.

4.4 Summary

A detailed view about the steps taken during the implementation phase was presented in this chapter, which started by describing the road network used for testing. Then, it focused on the developed model for forecasting through the use of simulation and the way it was implemented using SUMO simulator. After that, it were presented the procedures taken for developing the data-driven prediction model using neural networks, using the Encog framework.
Implementation
Chapter 5

Tests and Evaluation

After the presentation in the previous chapter of the implementation details, this chapter explores the tests made during the development phase for both forecasting models: the simulation as well the data-driven model. In the end, is described the environment where the tests were made in.

5.1 Simulation Model

During the first stage of implementation, which focused on the configuration of the software, a few functional tests were executed in order to evaluate if a proper configuration was made. Later, when the initial simulation models were ready, it was carried out a set of execution tests with the goal to evaluate the performance of the techniques developed.

5.1.1 Functional Tests

The functional tests made were carried out after defining the first network model of the test-bed area, and it consisted of testing the injection of vehicles in the network in order to detect potential mistakes and errors in the network topology. The data used for testing was a set of random trips generated with the SUMO’s auxiliary tool randomTrips, which receives as input the network and the time interval of simulation, and generates vehicles with random routes.

This test allowed to identify a few deadlock situations existing on certain road segments, where the cars got stuck provoking a block on simulation. The reason for this to happen was the fact that there was no restriction of U-turns so that in every street the vehicles were allowed to turn back due to the fact that by default netconvert inserts U-turns in the end of every urban road edge. Similar errors were detected and described in the project presented in the paper [DAS+13]. In figure 5.1 is an example where we can see from the white arrows that the vehicles are allowed to turn around in the end of the road segment. Figure 5.2 shows a similar situation on a crossing with traffic lights where the vehicles could turn around, when in real-life that movement is forbidden. This error
Tests and Evaluation

![Figure 5.1: Illegal U-turn](image1)

![Figure 5.2: Illegal U-turn on a crossing](image2)

was solved by adding the option --no-turnarounds when creating the network from the OSM map with netconvert.

Also, some of the existing traffic lights in the access streets were not properly defined when the map importing was made. An example of it can be seen in figure 5.2, where one of the traffic lights is placed in the middle of the crossing, and it is not synchronized with the rest of traffic lights. A solution found was to use the parameter --junctions.join when using the netconvert program to import the network. This option forces the program to search the existing traffic lights on every crossing and synchronize (join) them into only one traffic light system per intersection.

### 5.1.2 Execution Tests

Once the functional errors related to U-turns movements and traffic lights were detected and corrected, it was time to test the execution of the developed simulation model. The tests covered two different ways for modelling traffic demand: first using the dfrouter application, and then with OD matrices. The data used for testing came from the measures collected by real traffic detectors and the time interval used was of several hours in some cases, and of one hour long during rush-hour period (early morning) in other cases. Since the VCI motorway is only partially covered by traffic detectors, not having detectors in some road segments neither existing on the access ramps, it was necessary to infer the data related to the trips between the motorway and the neighbour streets. This inference was decided to be made through an equal split-rate for all the origin and destination links. Indeed, this method might lead to a loss of accuracy, however in the case of study of this thesis it is acceptable since the main percentage of traffic flows occurs within the motorway which is the ultimate area of study.

For testing the simulation performance, the measurement unit was the number of vehicles passed within a time interval, that consisted of 5 minutes in some cases and of 10 minutes in others. Also, to improve the calibration and evaluation of the developed models it was used a function to calculate the error rate between the number of expected vehicles to be simulated at each time interval and the vehicles that are actually simulated. This function was based on the formula presented in [KHWR02], where it was used the variation between simulated and real
Tests and Evaluation

Travel times to calibrate the model. A model would be considered to have better performance as lower is its error rate. The formula is denoted by:

\[
\text{error} = \frac{|V_{\text{sim}} - V_{\text{obs}}|}{V_{\text{obs}}} \quad (5.1)
\]

\(V_{\text{sim}}\): number of simulated vehicles within the time interval;
\(V_{\text{obs}}\): number of observed vehicles from real data within the time interval.

In order to retrieve those measures from the simulation, it was used a SUMO feature for testing the simulation, which allows to define virtual detectors in the simulation that function as the real-life induction-loop detectors. These detectors count the number of vehicles that passed as well as calculate the average speed registered during the defined time interval, saving the results in an XML file.\(^1\)

**Modelling traffic with DFRouter**

The first experiment was made using the application `dfrouter` to model the traffic demand. As referred in chapter 4 this way of modelling the traffic demand is made through the use of virtual detectors that act as the actual traffic detectors. The dataset used for test was a time period of six hours composed by 5-minutes measures, belonging to the interval between 0h00 and 6h00 of the day March 1, 2013, which was a typical day without the occurrence of incidents. Then, the time of the simulation was 22000 seconds, corresponding to six hours plus an extra time to warm-up the data for simulation. A further description about the data used and the configuration file can be found in appendix A.1. The interval of measures used was of 300 seconds (5 minutes) and the measurements were made using the induction loop detectors (E1 type) of SUMO which were placed just-after the source detectors, and just-before the sink detectors. As said in beginning of this section, it was necessary to infer the data related to the trips between the motorway and the adjacent streets. In this case, it was used a 50 /50% split-rate for each of the neighbour streets (Paranhos and Faria Guimarães).

![Graphs of travel times](image)

(a) A3 detector - C direction (source)  (b) Paranhos detector - C direction (sink)

Figure 5.3: Results of 1st test of DFRouter - West (C) direction

\(^1\)http://sumo-sim.org/userdoc/Simulation/Output/Induction_Loops_Detectors_(E1).html
Tests and Evaluation

In figure 5.3 are presented the results obtained from the detectors located in direction of West (designated by letter C), and observing it we can see that all the expected cars were released, and within the time, from the source detector (figure 5.3a), however only nearly 40-50% of them arrived in the sink detector (figure 5.3b). On the opposite way, East direction (designated by letter D), in the source detector (figure 5.4a) most of the vehicles were released in accordance to the expected number with a small variation but still within the expected time. On the other hand the sink detector (figure 5.4b) registered a higher value for the simulated vehicles than the expected ones.

![Figure 5.4: Results of 1st test of DFRouter - East (D) direction](a) Paranhos detector - D direction (source)  (b) A3 detector - D direction (sink)

Summing up, the reason for this difference between the values measured in the source and in the sink detectors, either in West or East direction, is from the fact that the trips and routes generated with dfrouter cannot be defined by the user, since it only receives the number of expected vehicles and the average speed. Thus, the program computes the trips and routes accordingly to its own criteria, having the only milestone as to follow the defined number of vehicles. This leads to an unrealistic set of routes, that do not match with the actual trips.

A solution found to solve this situation and calibrate the simulation model was to use the option --disallowed-edges when importing and generating the traffic demand with dfrouter. This parameter commands the program to not use certain edges when calculating the routes, forcing the trips to have a "defined" route and though avoid the existence of unrealistic routes. Using this option led to an improvement in the results, especially with the source detector "Paranhos D" and the sink detector "Paranhos C".

Figure 5.5 shows the improvement achieved in terms of vehicles passing, reaching in average around 60% of the expected number, and in figure 5.6 can be seen that all the expected vehicles were release into the simulation. It should be noted that in the first test as well as in the second, the number of vehicles that reached the sink detector of "A3 D"(figure 5.6b) was superior to the expected value. The identified reason for this happening was again due to the impossibility to define the trips and routes generated by dfrouter. This provokes a considerable number of random trips, where the dfrouter tries to generate the enough number of trips from different source points and ends up creating more trips to a certain destination than it should have.
Tests and Evaluation

Regarding the error rates of the two experiments, in table 5.1 are presented the average rates registered at each detector. Observing it, we can see that there was a clear improvement in the second test, especially in the values collected at detector "Paranhos (C)", where the error rate dropped from around 57% in the first test to 21.4% in the second experiment. Also, both source detectors (A3 (C) and Paranhos(D) ) achieved a 0% error rate in the experiment using --disallowed-edges (2nd test). In contrast, during the second test was registered a higher error rate for the sink detector A3 (D), whose likely reason for happening was described before.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>A3 (C)</th>
<th>Paranhos (C)</th>
<th>Paranhos (D)</th>
<th>A3 (D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>0%</td>
<td>56.8%</td>
<td>5.5%</td>
<td>22.4%</td>
</tr>
<tr>
<td>#2 (using --disallowed-edges)</td>
<td>0%</td>
<td>21.4%</td>
<td>0%</td>
<td>29.1%</td>
</tr>
</tbody>
</table>

Table 5.1: Error rates obtained in the tests with DFRouter

After evaluating the results obtained and taking in account the error rates achieved, the arisen conclusion was that it was not the best method to be used in this thesis context since it revealed to be unable to fully emulate the real routes from the source and the sink points. Albeit, its ability to depict the traffic flows from real data measurements makes it a potential useful method to be used in a small network scenario, where having the right number of vehicles at each detector point is more important than the routes taken by the cars.
Tests and Evaluation

Modelling traffic with OD matrices - first experiment

Once tested the modelling of traffic with DFRouter, it was time to try another method to do it, specifically, using OD matrices. With this method the network is divided into origin-destination zones and it is necessary to input the number of trips between each zone. For running the test it was used a dataset of one hour of measures during the morning rush-hour period (8h-9h) in a normal day without accidents. Thus, the simulation period was of 4000 seconds, consisting of one hour plus an extra time used to warm-up the simulation. The testing data came from an observation experiment, therefore it was accurate for all the origins and destinations, including the neighbour streets, without the need of being inferred. Despite the time interval for the data was of one hour, corresponding to have only one measure for the whole simulation period, the data was input in a time interval of 10 minutes, with a total of six measurement periods. In appendix A.2 is presented the OD matrix used as well as the configuration file.

![Graphical representation of the test results](image)

(a) A3 detector - C direction (origin)  
(b) Paranhos detector - C direction (destination)

Figure 5.7: Results of 1st test using OD matrices - West (C) direction

The results obtained led to the identification of a problem in the road network which was the difficult of input all the vehicles within the simulation time. Nearly only half of the expected vehicles were released in the right time, requiring almost the double of simulation time (4000 seconds more) to insert all the cars in the network. This can be seen in figure 5.7 where it took 2400 seconds more than expected to complete the simulation of all vehicles in the West (C) direction. By observing the graphics of figure 5.8 it can be stated that in the opposite direction (East -

![Graphical representation of the test results](image)

(a) Paranhos detector - D direction (origin)  
(b) A3 detector - D direction (destination)

Figure 5.8: Results of 1st test using OD matrices - East (D) direction
Tests and Evaluation

D direction) the results were even lower, by taking around 4400 seconds more to conclude the simulation of all vehicles.

The identified reason for this happening was the fact that the network did not have enough space to be able to release the entire amount of expected cars within the desired time. A solution found to this problem was to extend the road edges by adding a dummy road that would be capable to inject all the vehicles in the simulation in useful time. Therefore, the motorway road sections related to the two main sources of vehicles (Paranhos and A3) were extended each one with a dummy road composed by six lanes. This allowed to insert more vehicles, requiring a time interval (around 600 seconds of simulation time) to warm-up the simulation.

Furthermore, it was necessary to reduce the simulation time-step, from the default value of 1 second to 0.5 seconds, and to change the definition of the vehicles’ characteristics. By reducing the time-step it was possible to insert more vehicles per second, and changing the minimum gap between vehicles, as well as the driver’s reaction time, provided a better usage of the space in road sections, allowing to introduce more vehicles in the same area without provoking collisions. The changes of the vehicles characteristics were made by defining a vehicle type for all the vehicles and which is presented in listing 5.1, where \( \sigma \) is the driver’s imperfection, \( \text{minGap} \) is the required minimum space between two vehicles, the \( \text{speedDev} \) is the deviation of speed value which allows the vehicles to vary their speed, and \( \tau \) is the driver’s reaction time, in seconds. Although the use of \( \sigma \) parameter with the zero value might lead to the loss of some realism in simulation, this loss was minimal when compared to the benefits achieved.

```
<vttype id="myType" \sigma="0" \text{minGap}="1" \text{speedDev}="0.1" \tau="0.6"/>
```

List 5.1: Definition of vehicle type used to increase the number of vehicles released

After implementing the defined changes, the model was tested again with an improvement in the results and all the vehicles were released within the time of simulation as can be seen in the graphics of figure 5.9. By observing it, is possible to ascertain that in the route toward West(C) direction, the expected cars to be released during the first interval were all injected into

![Graphs showing vehicles passed vs time for A3 and Paranhos detectors](image)

(a) A3 detector - C direction (origin)  (b) Paranhos detector - C direction (destination)

Figure 5.9: Results of 2nd test using OD matrices - West (C) direction
the simulation, as well as in the last time intervals, having only an average variation of 70 vehicles in the intervals between the time steps of 600 and 3000 seconds.

In the opposite direction, East (D) direction, the number of vehicles released in the expected time were not so accurate as in West direction, releasing 50 vehicles less than the real data during the first time interval (0-600 seconds). However, this difference has decreased in the following intervals of time, reaching a constant arriving ratio in the sink detector (A3 detector) practically equal to the actual ratio, as can be observed in the graphics presented in figure 5.10.

Figure 5.10: Results of 2nd test using OD matrices - East (D) direction

In table 5.2 are presented the error rates acquired during the two experiments and we can see the high error ratio of the first experiment, with rates around 38% in the direction toward West (C), and about 50% in East (D) direction. The use of the dummy road lead to a great progress, where the average error rates were around 3% at the origins (A3(C) and Paranhos(D) detectors) and of 5% at the destinations (Paranhos(C) and A3(D)).

<table>
<thead>
<tr>
<th>Experiment</th>
<th>A3 (C)</th>
<th>Paranhos (C)</th>
<th>Paranhos (D)</th>
<th>A3 (D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>38.4%</td>
<td>37.1%</td>
<td>55.3%</td>
<td>46.3%</td>
</tr>
<tr>
<td>#2 (with dummy road)</td>
<td>3.2%</td>
<td>5%</td>
<td>3.5%</td>
<td>4.1%</td>
</tr>
</tbody>
</table>

Table 5.2: Error rates obtained in the 1st experiment with OD matrices

Modelling traffic with OD matrices - second experiment

The first experiment using Origin-Destination matrices focused on testing the insertion of all vehicles within a useful time and after successfully accomplish it, the next step was to extend the time interval of simulation used, by inserting more vehicles, in order to get a duly calibrated model that could depict how the traffic normally behaves in a certain time of the day. This was made by repeating the dataset used in the first experiment (which corresponds to one hour of measures) for more hours and then determine when the model reach a constant ratio of vehicles simulated.

Initially it was used a time interval of four hours (14400 seconds), corresponding to a repetition of the testing dataset during four hours. This first trial got a set of irregular patterns in the way vehicles were released, injecting some times more vehicles than expected and other times less
Tests and Evaluation

cars, and having an average split of 50 vehicles than the expected data. In the West (C) direction, during the initial interval were released 100 vehicles less than desired, proving that it is required to have a warm-up time to set and prepare the data for the simulation. This deficit was balanced by later releasing more vehicles in the next time intervals, as can be observed in the graphics of figure 5.11.

![Image](a) A3 detector - C direction (origin) ![Image](b) Paranhos detector - C direction (destination)

Figure 5.11: Results of 3rd test using OD matrices (4 hours) - West (C) direction

About the traffic flow toward East (D) direction there was also, in the first interval, a difference between the vehicles released and the expected, but this time with a value of around 25 vehicles less. Unlike what happened in the West (C) direction, where the source point used was able to release later more cars than expected to balance the deficit of vehicles simulated, in this way the compensation for this shortfall was made by inserting the rest of the vehicles in the last time interval, as can be seen in figure 5.12.

![Image](a) Paranhos detector - D direction (origin) ![Image](b) A3 detector - D direction (destination)

Figure 5.12: Results of 3rd test using OD matrices (4 hours) - East (D) direction

A justification for this irregular ratio of the vehicles inserted was the fact the simulator was probably not distributing the vehicles in a uniform rate as it was expected. Therefore, the problem was solved by forcing SUMO to split the vehicles insertion in a uniform way through the use of the parameter "--spread.uniform" in the od2trips application. This led to a homogeneous correlation between the expected number of vehicles and the released ones as can be noticed in the graphics of traffic toward West (C) direction, presented in figure 5.13. In this segment, there was only a deficit of vehicles during the first time interval, which was interpreted as the time required to warm-up
the simulation, since in the following intervals the number of cars inserted per time interval was equal to the real data values.

![Graph](image1)

(a) A3 detector - C direction (origin)  
(b) Paranhos detector - C direction (destination)

Figure 5.13: Results of 4th test using OD matrices with "--uniform" (4 hours) - West (C) direction

In the reverse way, East (D) direction, the simulation took also the time of the first interval to warm-up and then starting to release cars in an uniform pace. Once warmed-up, the number of vehicles simulated was similar to the amount of expected vehicles, being in average 10 cars less than the actual data. This kept uniform along the time, and the identified reason for this small split was the fact that this road segment was not capable to deal with all the vehicles (833 per each time interval of 600 seconds), reaching its limit for the supported cars. Still, this split was not considered to be problematic since the difference was minimal. In figure 5.14 is presented the graphics related to the results of this way of the motorway.

![Graph](image2)

(a) Paranhos detector - D direction (origin)  
(b) A3 detector - D direction (destination)

Figure 5.14: Results of 4th test using OD matrices with "--uniform" (4 hours) - East (D) direction

After analysing the results obtained, the model could be considered as properly calibrated since it has simulated the expected amount of vehicles within the simulation time and in the desired time intervals. In order to extend the evaluation coverage for this model and perhaps improve even more the results, it was decided to make another test under the same circumstances as the first one but for a simulation time of six hours (21600 seconds). The results achieved were identical to the experiment with the simulation of four hours, having the simulation in both ways taken the first time interval to warm-up. The traffic flow in direction of West (C) obtained as well as a correct equivalence between the simulated and expected vehicles as presented in the graphics in
Tests and Evaluation

Figure 5.15. Also, the traffic toward East (D) direction had similar results as the experiment with four hours, where there was a minor split between the expected and the simulated cars and which can be observed in figure 5.16.

![Figure 5.15: Results of 5th test using OD matrices (6 hours) - West (C) direction](image)

(a) A3 detector - C direction (origin)  
(b) Paranhos detector - C direction (destination)

![Figure 5.16: Results of 5th test using OD matrices (6 hours) - East (D) direction](image)

(a) Paranhos detector - D direction (origin)  
(b) A3 detector - D direction (destination)

It is possible to see in detail the improvements achieved by observing the error rates results presented in Table 5.3. The average rates obtained, which were lower than 1% in the West direction (A3 (C) and Paranhos (C)), and 2% in direction toward East (Paranhos (D) and A3(D)), are satisfying results which prove the calibration of the model. Also, the increase to 6 hours of simulation lead to minimal improvements, around only 0.1% lower error rates, which reinforces that the minimum error rate to have a duly calibrated model was achieved.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>A3 (C)</th>
<th>Paranhos (C)</th>
<th>Paranhos (D)</th>
<th>A3 (D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1 (4h)</td>
<td>2.6%</td>
<td>3.1%</td>
<td>2.7%</td>
<td>2.6%</td>
</tr>
<tr>
<td>#2 (4h - using uniform)</td>
<td>0.4%</td>
<td>0.9%</td>
<td>2%</td>
<td>1.9%</td>
</tr>
<tr>
<td>#3 (6h - using uniform)</td>
<td>0.3%</td>
<td>0.7%</td>
<td>1.9%</td>
<td>1.7%</td>
</tr>
</tbody>
</table>

Table 5.3: Error rates obtained in the 2nd experiment with OD matrices

Once finished the tests and observing the results and error rates it was evident the advantage achieved by using OD matrices to model traffic demand, by obtaining a calibrated model which
Tests and Evaluation

was capable to simulate the traffic pattern for a defined time of the day. Thus, the use of OD matrices proved to be more suitable for this thesis study than dfrouter application.

Modelling traffic with OD matrices - using an incident scenario

The conclude the tests with the simulation model a final test was made, this time focusing on the prediction of the network state when an incident occurs. Thus, for testing it it was used a data set of measures from a time interval where an incident was reported to have happened in the test-bed area (Paranhos junction node), in East-West direction. Since the incident happened only in one direction of the motorway, it were used only traffic measures from the two detectors located in West (C) direction. The day in question was 28th of January 2014 from 10h until 12h30, which was the time the accident was reported to last, and it affected the most-right lane. A description about the OD matrix used and the configuration file is presented in appendix A.3.

The goal of carrying this test was to evaluate if at some point the simulator was able to depict the network’s behaviour when an incident occurs, more specifically when one or more lanes of the road are blocked. Therefore, the implementation of the accident in the simulation was made through two methods: stopping a car on the lane during the simulation; and by defining speed limits on the lanes and road segments close to the area of the accident. These methods are the recommend ways to simulate an accident since SUMO so far does not have an exclusively feature to depict accidents. ²

So, in the first experiment two vehicles were halted in the middle of the network on the right-most lane for the whole time of simulation. Their function was to act as the vehicles involved in the accident do, which leads to force the vehicles behind them to change their route by moving to the adjacent lane. This reduced the number of vehicles passing per time-step as desired, however it was still a higher value than expected, as can be seen in the results presented in figure 5.17.

![Figure 5.17: Results of 1st incident scenario test using vehicles stopping](image)

(a) A3 detector - C direction (origin)  
(b) Paranhos detector - C direction (destination)

Observing the results, it is evident that using only the method of stopping the vehicles in the road segment was not enough to fully emulate the traffic state during a real incident scenario. A reason for it is that in real-life when there is an accident or a lane blocked in the middle of the

²http://sumo-sim.org/userdoc/FAQ.html#How_to_simulate_an_accident
Tests and Evaluation

Figure 5.18: Map of accident area, with the defined speed limits

road, the cars in the adjacent lanes reduce their velocity, normally provoking a traffic congestion. Albeit, in the simulator this situation is not depicted at all since the cars in the neighbour lanes keep moving with the same speed, not concerning about the existing blockade in the next lane.

For that reason in the second experiment it was decided to set a speed limit on the lanes near the area of the accident besides the use of two vehicles to block the traffic. A description about the implementation of this speed limits can be found in section 4.2. The defined limits of velocity were: 30 km/h on the lane where the accident happened, 40 km/h on the middle lane and 55 km/h on the left-most lane as well as in the road segments before the accident area. In figure 5.18 are presented in detail the speed limits for each lane.

Hence, the simulation was executed again, this time with two vehicles halted in the right-most lane and with speed limits on the lanes located in the surrounding area of the accident. A tiny improvement was achieved with the additional methods used, having a small variation in the vehicles counted in the source detector A3, and there was in the sink detector Paranhos a reduction in the traffic flow per each measurement interval. The detailed results can be seen in the graphics of figure 5.19.

The error rates obtained by each experiment are presented in table 5.4, and observing it we can state that both tests got similar error rates, with a meaningless difference of 0.2% at the origin points. Thus, two conclusions can be taken: one is that the obtained error rates are still relatively high to consider the model valid, and which leads to the second conclusion that to modelling incidents using SUMO might not be so easy and simple to accomplish by just stopping cars and
Tests and Evaluation

<table>
<thead>
<tr>
<th>Experiment</th>
<th>A3 (C)</th>
<th>Paranhos (C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1 (using two vehicles stopped)</td>
<td>10.5%</td>
<td>9.9%</td>
</tr>
<tr>
<td>#2 (using variable speed sign)</td>
<td>10.3%</td>
<td>9.9%</td>
</tr>
</tbody>
</table>

Table 5.4: Error rates obtained in the accident experiment with OD matrices

reduce the speed on the lanes. Indeed, to depict the traffic pattern of an accident using simulation is necessary to combine the stopping of vehicles and the use of variable speed signs, by varying the values of those parameters through the simulation time in order to reproduce the velocity variations caused by a block on a lane.

5.2 Data-driven Model

The tests made with the developed neural networks focused on the use of different activation functions: Sigmoid, Tanh and Log, and their formulas can be seen in table 5.5. In addition, two network structures were tested: one hidden layer with 15 neurons; and two hidden layers, one with 10 neurons and the other with 5 neurons.

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sigmoid</td>
<td>[ f(x) = \frac{1}{1+e^{-x}} ]</td>
</tr>
<tr>
<td>Tanh</td>
<td>[ f(x) = \frac{e^x - 1}{e^x + 1} ]</td>
</tr>
<tr>
<td>Log</td>
<td>[ f(x) = \begin{cases} \log(1+x) &amp; , x \geq 0 \ -\log(1-x) &amp; , \text{otherwise} \end{cases} ]</td>
</tr>
</tbody>
</table>

Table 5.5: Description of activation functions tested

As said in section 3.3.2 the dataset used for testing was composed by a complete week of measures (from Monday until Sunday) without the occurrence of incidents, collected in May 2013. The measurement unit used was the average speed registered by the traffic detectors within a time interval of 5 minutes, and it was used a road section of the motorway way from East to West, located in Paranhos junction node. In the initial tests were only used the values from one of the two existing detectors in that node, specifically the "A3" detector (see map of the node in figure 4.2). Also, like it was done in the test of the simulation model, it was used a formula to calculate the error rate of predictions, where the unit used was the predicted average speed. Before testing the performance of the networks to predict traffic, it was made an evaluation of the time spent by each network during the training stage.

Modelling traffic with normal ANN

The tests started with the normal neural network model, and the results obtained as well a comparison with the real data are presented in this section using graphics, where the y axis is about the
Tests and Evaluation

average speed and the x axis is about the time of the day along the week of test. Firstly it was tested the use of the Sigmoid function, and during the learning phase the one hidden layer network took 44 epochs to reach an error rate lower than 0.1% and finalize the training, spending 1.33 seconds to do it. In comparison, the network with two hidden layers spent two times more (3.91 seconds) to conclude the training than the one hidden layer and 115 epochs, as it is shown in the table 5.6.

<table>
<thead>
<tr>
<th>Network</th>
<th>Time (seconds)</th>
<th>Epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>One hidden layer (15 neurons)</td>
<td>1.33</td>
<td>44</td>
</tr>
<tr>
<td>Two hidden layers (10+5 neurons)</td>
<td>3.91</td>
<td>115</td>
</tr>
</tbody>
</table>

Table 5.6: Epochs and time spent per each ANN to finish the training - Sigmoid function

About the results obtained, the ANN with one hidden layer predicted a range of values close to the actual values, revealing however a difficulty to make strong variations of values between adjacent time steps. Even using data from normal days to train and test, this network tends to normalize the values along the time. This characteristic was noted even more in the rush-hour intervals where the network followed the tendency to decrease the value of average speed but not reaching close to the real data. In figure 5.20 is presented the comparison between the results and the real data using the one layer network.

Figure 5.20: Results of the test of normal ANN with Sigmoid function - one hidden layer

With the two layers network the results achieved were mostly similar, with only a slight improvement, where it showed to be more flexible to vary the lowest values between measurements. In the rush-hour periods this network got the predictions closer to the actual data. The forecasting results of this network are presented in figure 5.21. Summing up, since both networks had identical results, it is not possible to clearly define which of the two structures was better for traffic forecasting.

Afterwards it was tested the forecasting using the Tanh activation function. About the learning phase, the network with one hidden layer took 2.20 seconds to complete the training, with a total
Tests and Evaluation

Figure 5.21: Results of the test of normal ANN with Sigmoid function - two hidden layers

of 70 epochs, while the ANN with two hidden layers spent less time and epochs to get trained, taking 1.55 seconds. These execution times can be seen in table 5.7.

Concerning the results, both networks showed the tendency, in general, to predict values which are higher than the expected data, specially the network with one hidden layer as can be found in the graphic of figure 5.22. This network learnt the rush-hour traffic patterns at a fair level, however there was an average speed difference between the predictions and the real values of around 15 \( km/h \) more for Monday, Tuesday and Thursday; and near 20 \( km/h \) more than actual data on Wednesday and Friday.

Figure 5.22: Results of the test of normal ANN with Tanh function - one hidden layer

<table>
<thead>
<tr>
<th>Network</th>
<th>Time (seconds)</th>
<th>Epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>One hidden layer (15 neurons)</td>
<td>2.20</td>
<td>70</td>
</tr>
<tr>
<td>Two hidden layers (10+5 neurons)</td>
<td>1.55</td>
<td>47</td>
</tr>
</tbody>
</table>

Table 5.7: Epochs and time spent per each ANN to finish the training - Tanh function
Tests and Evaluation

The network with two hidden layers got lower results than the first one, essentially in the peak hours as can be seen in figure 5.23. In fact, this tendency to predict lower values showed that it was better to forecast the traffic state of rush-hours, having only a difference of 10 km/h between its forecast and the actual values on Monday, Tuesday and Thursday, and a 15 km/h more than expected on Wednesday and Friday.

![Figure 5.23: Results of the test of normal ANN with Tanh function - two hidden layers](image)

Additionally, it was made an experiment with a combination of two different activation functions for the network with two hidden layers, with the goal to research if the use of different functions could improve the results. Thus, the structure used was: one of the layers with the Sigmoid function and the other using the Tanh activation function. Indeed, this modification caused a significant variation in the predictions, with the forecasted traffic being underestimated with lower values than the real data, as it is presented in figure 5.24. This variation is more evident in the forecasts for the rush-hour period where the results were either similar to the actual data (in the measurements for Wednesday, Thursday and Friday) or even lower than expected (in the measurements for Monday and Tuesday).

To conclude, when using the Tanh activation function the best of the three models tested was the second network (two hidden layers using only Tanh function) since it produced balanced results in the normal traffic periods and during rush-hour. A downside of the last model developed (with the combination of Tanh and Sigmoid function) is that it provokes big variations of traffic which is not exactly what happens in a normal traffic state.

Finally, it was tested a third activation function, the Log function, and the training time for the ANN with one hidden layer was of 3.42 seconds and 159 epochs. In contrast, the learning stage of two layers network took only 1.03 seconds to conclude and 47 epochs, as it is presented in table 5.8.

In general both networks got good results, having the ANN with one hidden layer forecasted the traffic for the normal hours periods (i.e. non rush-hour) with a small difference between the predicted values and the real data. For rush-hour periods where there was great speed variations in a short time interval (Wednesday and Friday), its results were not so accurate, having a difference...
Tests and Evaluation

Figure 5.24: Results of the test of normal ANN using Tanh and Sigmoid function - two hidden layers

of around 10 km/h more than it was expected. In the other periods of rush-hour where the velocity variation was not so deep, the network predicted with an average error of 5 km/h as can be seen in the graphic presented in figure 5.25.

Figure 5.25: Results of the test of normal ANN using Log function - one hidden layer

By observing the results graphic of the network with two hidden layers (see figure 5.26), we can see that it showed a trend to have a higher variation of values during the hours with normal traffic flow than the ANN with one layer, predicting sometimes above the real value and other times below the actual value. About the periods of rush-hour it had worse results when compared to the first network, with an average variation of 10 km/h except in the measurements for Friday where the variation was of around 15 km/h.

Thus, the best model using the Log activation function was the model with one hidden layer, which proved to better depict the traffic patterns, including during the rush-hour period where it got close predictions to real values.
Tests and Evaluation

<table>
<thead>
<tr>
<th>Network</th>
<th>Time (seconds)</th>
<th>Epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>One hidden layer (15 neurons)</td>
<td>3.42</td>
<td>159</td>
</tr>
<tr>
<td>Two hidden layers (10+5 neurons)</td>
<td>1.03</td>
<td>47</td>
</tr>
</tbody>
</table>

Table 5.8: Epochs and time spent per each ANN to finish the training - Log function

After finishing the test of the normal neural network, the conclusions that can be taken are that the most accurate forecasts were made by the networks using the Log activation function since both of the structures implemented acquired good results. In fact, the implemented network with one hidden layer using the Log activation function got the best results, predicting the traffic along the normal hours as well as during the rush-hour periods with values very close to the real data. Also, the other ANN implemented with Log function, with two hidden layers, got a fair set of forecasts, only varying sometimes more than expected, when compared to the network version of one hidden layer. In contrast, the networks which got worst results were the network using Tanh function with one hidden layer, and the ANN of two hidden layers with the combination of Sigmoid and Log functions. The first one predicted values higher than expected, while the second got forecasts with extreme values and a big variation than the real data.

In table 5.9 are presented the error rates obtained by each network and observing it we can conclude that the ANNs using the Log activation function got the best predictions with a error rate of 2.7%, where the networks using Sigmoid got the higher rate (around 3.5%).

<table>
<thead>
<tr>
<th>Hidden layer structure</th>
<th>Sigmoid</th>
<th>Tanh</th>
<th>Log</th>
</tr>
</thead>
<tbody>
<tr>
<td>One layer (15 neurons)</td>
<td>3.4%</td>
<td>3.2%</td>
<td>2.7%</td>
</tr>
<tr>
<td>Two layers (10+5 neurons)</td>
<td>3.6%</td>
<td>2.9%</td>
<td>2.7%</td>
</tr>
<tr>
<td>Two layers (Sigmoid + Tanh)</td>
<td>-</td>
<td>3.0%</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5.9: Error rates obtained in the tests with normal ANN

Modelling traffic with normal ANN - using an incident scenario

Like it was done in the tests made with the simulation model, the basic model of the normal neural network was adapted to create an ANN that could work with incident scenarios. In fact, the main changes were increasing the number of inputs and outputs in order to use data from two detectors instead of the first version which only took in account the values of one traffic detector. This approach was intended to use the measures of two adjacent detectors and learn the pattern when an incident occurs by using the gap of the number of vehicles between both points. In terms of function, this network model uses the values of both traffic detectors to detect the traffic at each one.

In figure 5.27 are presented the predictions obtained for each point and we can see that the forecasts for the first detector (5.27a) were reasonably accurate for the periods where traffic was normal as well as for the rush hours periods having a tiny difference of 5 km/h more on Monday.
Tests and Evaluation

![Average speed - Paranhos node (from Monday to Sunday)](image)

Figure 5.26: Results of the test of normal ANN using Log function - two hidden layers

and Tuesday, and of 10 \( \text{km/h} \) on Wednesday and Friday. For the second detector (5.27b) the network got good forecasts during the normal traffic period, however its performance was worse when it forecasted for the peak hours. It got predictions in average 10 \( \text{km/h} \) higher than expected (on Monday, Wednesday and Friday) and 20 \( \text{km/h} \) higher than the real data for Tuesday and Thursday.

![A3 detector (1st detector)](image)

(a) A3 detector (1st detector)

![Paranhos detector (2nd detector)](image)

(b) Paranhos detector (2nd detector)

Figure 5.27: Results of the test of ANN using an incident scenario

Concerning the error rates obtained by the network when testing an incident scenario, it got a better rate on the predictions for first detector (A3 detector), which was almost 1% better than the rate achieved in the forecasts for the second detector (Paranhos detector) as can be seen in table 5.10. Comparing this rates with the error ratio obtained in the previous tests (see table 5.9) the values are similar, having the first detector obtained a rate (2.8%) identical to the rate of the best models, which used \( \text{Log} \) as activation function, and the second detector a rate (3.6%) equal to the rate achieved by the networks using \( \text{Sigmoid} \) function.

To conclude, we can see from the similarity of results with the normal network, that this neural network can provide traffic forecasts with the same accuracy as the normal model with the benefit of being able to better detect the influence of traffic flow over two adjacent detectors.
Tests and Evaluation

<table>
<thead>
<tr>
<th>Detector</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>A3 detector</td>
<td>2.8%</td>
</tr>
<tr>
<td>Paranhos detector</td>
<td>3.6%</td>
</tr>
</tbody>
</table>

Table 5.10: Error rates obtained in the tests of an incident scenario with normal ANN

Modelling traffic with fuzzy ANN

After testing the normal ANN model, it was time to test the fuzzy neural network with different activation functions. In the tests made the measurement unit used was the average speed, but this time using the fuzzy scale defined on chapter 3 (see table 3.2) instead of defining it in $km/h$.

As it was done while testing the normal network, the first tested activation function was the **Sigmoid** function and during the learning stage there was a considerable differences between the time spent by both types of network. The first network, with one hidden layer, took 62.23 seconds and 2076 epochs to conclude the training, while the ANN with two hidden layers finished its training after 142.73 seconds and 3270 epochs. In table 5.11 are presented the execution time values. The reason for this great divergence is the fact that during the deployment of the network with fuzzy logic, it occasionally did not achieve the desired error rate of 0.1% and thus did not complete the training stage, entering in an endless cycle. Due to this reason, it was decided to force the stop of the learning process if after a certain time period there was no improvements on the training error rate. Therefore, the rule to stop the learning stage before the time was: if the error rate increased during ten consecutive measurements, instead of decrease, after executing more than 300 epochs then the training should finish.

<table>
<thead>
<tr>
<th>Network</th>
<th>Time (seconds)</th>
<th>Epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>One hidden layer (15 neurons)</td>
<td>62.23 *finished before time</td>
<td>2076</td>
</tr>
<tr>
<td>Two hidden layers (10+5 neurons)</td>
<td>142.73</td>
<td>3270</td>
</tr>
</tbody>
</table>

Table 5.11: Epochs and time spent per each fuzzy ANN to finish the training - Sigmoid function

Concerning the results, the network with one hidden layer predicted well the traffic for the non rush-hour periods, forecasting the speed value within the expected range (level four - normal traffic state). For the rush-hour periods the results obtained were not so satisfactory, with the ANN having some difficult to learn and depict deep variations of speed (i.e. from speed level 4 to level 2), predicting the speed within a range higher than the expected values as can be seen figure 5.28.

The ANN with two hidden layers had similar results with regards to the normal traffic state period, predicting the values as expected. An improvement from the results of the first network was achieved in the prediction for the traffic during the peak hours, with the forecasted speed range close to the real values. The graphic with the results of the second network is presented in figure 5.29.

In conclusion, the results obtained by the network with two hidden layers were more uniform in the rush-hour time, being this structure better when using the **Sigmoid** function than the model.
Figure 5.28: Results of the test of fuzzy ANN using Sigmoid function - one hidden layer

Once tested the use of Sigmoid function, the next function used was the Tanh. During the training stage, and as it happened when using the Sigmoid function, the two networks took different amounts of time to perform their learning process. The one with hidden layer took 81.87 seconds and 3029 epochs and finished the training stage before the time, from the fact that it was unable to achieve the desired error rate. Thus, the acquired error rate was of 0.17%, a value relatively close to the desired error rate (0.1%). In opposition to it, the ANN with two hidden layers took 625.86 seconds and 19321 epochs to conclude the learning, however its training lasted until a error rate below 0.1% was achieved. In table 5.12 are presented the execution times and epochs taken by both networks.

About the results, both networks got good results when predicting for the hours when the traffic is in a normal state (i.e. at level four). For forecasting in the rush-hour periods, the network with one hidden layers got fair results by predicting values between the range of levels two and three ("Stop-and-go" and "Slow" states) as can be seen in figure 5.30. In comparison, the network
Tests and Evaluation

<table>
<thead>
<tr>
<th>Network</th>
<th>Time (seconds)</th>
<th>Epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>One hidden layer (15 neurons)</td>
<td>81.87 *finished before time</td>
<td>3029</td>
</tr>
<tr>
<td>Two hidden layers (10+5 neurons)</td>
<td>625.86</td>
<td>19321</td>
</tr>
</tbody>
</table>

Table 5.12: Epochs and time spent per each fuzzy ANN to finish the training - Tanh function

with two hidden layers showed to have more flexibility to deal with speed variations, and to have a better ability to learn, and consequently depicts, the rush-hour patterns as can be observed in the graphic of figure 5.31. Its forecasts were close to the real values, especially the ones obtained for Wednesday and Friday.

![Figure 5.30: Results of the test of fuzzy ANN with Tanh function - one hidden layer](image1)

![Figure 5.31: Results of the test of fuzzy ANN with Tanh function - two hidden layers](image2)

In sum, as referred before both networks got good results however the model which got more acceptable results was the first network developed with one hidden layer, since it provided a set of forecasts which were more linear and close to the actual values, while the ANN with two layers got an irregular collection of predictions which were sometimes lower and other times higher than
expected.

For last, the two networks were tested using the Log function as activation function, and in table 5.13 are presented the execution times each one took. By observation, it can be seen that the network with one hidden layer spent 155.24 seconds and 7555 epochs, finishing the training before expected and obtaining an error rate of 0.16%. The ANN with two hidden layers also finished its learning before reaching the desired error rate and needed even more time by taking 828.80 seconds and 37291 epochs. Despite its training process was forced to finished before reaching the base error rate, the error percentage obtained was of 0.11%, which is quite close to the defined value.

<table>
<thead>
<tr>
<th>Network</th>
<th>Time (seconds)</th>
<th>Epochs</th>
</tr>
</thead>
<tbody>
<tr>
<td>One hidden layer (15 neurons)</td>
<td>155.24 *finished before time</td>
<td>7555</td>
</tr>
<tr>
<td>Two hidden layers (10+5 neurons)</td>
<td>828.80 *finished before time</td>
<td>37291</td>
</tr>
</tbody>
</table>

Table 5.13: Epochs and time spent per each fuzzy ANN to finish the training - Log function

Regarding the results, the one hidden layer network behaved well to forecast in the non rush-hour periods, having in contrast some non-linear results when predicting for the rush-hour time. As can be seen in the graphic of figure 5.32, this network predicted in some days a speed value with a higher velocity range than expected (on Monday and Friday), and on other days achieved results which are really close to the expected values (2nd rush-hour of Tuesday and on Wednesday). The second network, with two hidden layers, has also good results when predicting for the normal traffic state hours, and got as well a set of irregular patterns for the rush-hour periods. This network proved to be highly sensitive to relatively high variations as can be observed in the results presented in figure 5.33. There was two days (on Monday and Friday) where the predictions were in a lower speed range than what was expected, and in other days where the forecasted values were higher than expected.

![Figure 5.32: Results of the test of fuzzy ANN with Log function - one hidden layer](image)
Tests and Evaluation

Thus, with regards to the models using Log function, it was considered that the network with one hidden layer was better than the two layers network, since it forecasted values in a more linear way. Also, the propensity of the ANN with two layers to have big variations and non-linear values in the forecasts for the rush-hours made it a worse option when compared to the first network developed.

![Figure 5.33: Results of the test of fuzzy ANN with Log function - two hidden layers](image)

To conclude, in what concerns the neural network models developed with fuzzy logic, the network with one hidden layer using Tanh as activation function got the best results, predicting traffic for the normal hours as well as for the rush-hour periods with a satisfying accuracy rate. Another characteristic of this ANN was the fact that the results were linear, without deep variations when compared to the real data. Also, the network using Sigmoid function with two hidden layers got similar results, proving to be an alternative but still a good option to forecasting. On the other hand, the network implemented with two hidden layers and using Log function got the worst results due to its highly irregular set of predictions made for the peak hours, leading to infer that it might not be a reliable method to be used when the accuracy is a must.

### 5.3 Summary

Once tested all the developed models, it can be stated that concerning the forecasting through the use of simulation, the model which got best results was the one using OD matrices, since it allowed to obtain a simulation model duly calibrated and able to depict the traffic behaviour in the test-bed used. Regarding the forecasting using artificial neural networks, it were obtained better results with the normal ANN than with the fuzzy network, specifically the normal ANN model using Log as activation function which got the best predictions.

The tests carried out, both with the simulation and the neural networks, were made using a computer with average characteristics and whose specifications are described in table 5.14.
Tests and Evaluation

<table>
<thead>
<tr>
<th>Computer parts</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor (CPU)</td>
<td>Intel Core 2 Duo P8600</td>
</tr>
<tr>
<td>Clock speed</td>
<td>2.4 GHz</td>
</tr>
<tr>
<td>RAM memory</td>
<td>4GB</td>
</tr>
<tr>
<td>Operating System</td>
<td>Windows 7</td>
</tr>
<tr>
<td>System Type</td>
<td>64-bit version</td>
</tr>
<tr>
<td>Graphics Card</td>
<td>Nvidia GeForce 9300M GS</td>
</tr>
</tbody>
</table>

Table 5.14: Specifications of computer used to test the simulation and the neural networks models
Chapter 6

Conclusions and Future Work

After presenting the methodologies used, the implementation steps taken and the tests made, in this chapter are described the overall conclusions about the goals achieved and the contributions of this thesis, as well as some final remarks such as possible improvements that might be made as a future work.

6.1 Goals achieved and final remarks

Once finished the thesis practical phase it was concluded that the defined goals were accomplished, have being developed model- and data-driven techniques to forecast traffic on road networks, using the two defined approaches, simulation and artificial neural networks.

In sum, after evaluating the results of the implemented models, with regards to the simulation techniques the model developed using the application dfrouter revealed to not be the best option for the objectives of this thesis, due to its difficulty in emulate the actual routes made by the vehicles in real-life. On the other hand, the alternative model which made use of OD matrices, allowed to have a properly calibrated simulation model which was capable to depict the traffic under normal characteristics, i. e. without any blocking or existing incident, in a realistic way.

In contrast, for an incident scenario the performance of the OD matrices model was not so satisfactory as expected, thus it was not possible to obtain a duly calibrated model to simulate the traffic behaviour under an incident scenario.

Concerning the data-driven models, the normal neural network got the best results, especially on the predictions for the rush-hours, showing a better ability to forecast and handle with high variations in the average speed values than the fuzzy networks. Therefore, a model to possible explore and to deepen in the future would be the normal ANN model developed, using the average speed in their normal scale as input. About the ANN implemented with fuzzy logic, it may have some potential and advantages to be used in another contexts and with more tests, however during the tests performed its performance was average and lower than the normal ANN. A solution to
Conclusions and Future Work

6.2 Future Work

In terms of possible future developments to continue the research covered by this thesis, one possible improvement was to expand the tests of both models (simulation and neural network) to the whole VCI motorway by departing from the work developed with the test-bed of Paranhos junction node, since both models can be adapted to work with the rest of the motorway. Additionally, the domain of the tests performed with the neural network could be expanded to cover all the existing traffic scenarios, from normal days to holidays season, as well as increase the number of inputs, by taking in account for instance the weather conditions related to each day. About the traffic data used by both models, it could be endowed of more precision by considering the GPS data related to the vehicles’ routes information [FCR10]. In addition, the GPS data can be used to improve the map behind the simulation model by processing it [FCR09].

Also, there are some potential applications that can be taken from the contributions made by this thesis such as, using the developed simulation model by artificial transportation systems applied in serious games scenario [RAKG13], for artificial traffic control systems [RFBO08], or even for simulation frameworks [FERO08] and decision-support tools related to transportation domain [ROB07].

Another enhancement could be the combination of both approaches, simulation with artificial neural network, by having the traffic data forecasted by the ANN and then use those forecasts in the simulation software. Basically, the neural network would be responsible for making traffic forecasts using the historical data; and then those predicted data would be used to feed the simulator to forecast how the traffic behaves with the received data. This hybrid model would lead to have a simulation model combined with artificial neural network, where the traffic forecasting was made by data-driven approach through the analysis of traffic patterns and then depicted in a real-life scenario by simulation.
References


REFERENCES


REFERENCES


REFERENCES


REFERENCES
Appendix A

Data and configuration files used for testing the models

In this appendix are presented the configuration files as well the data used in the tests executed.

A.1 Data used for modelling traffic with DFRouter

Table A.1: Data values used in the tests with dfrouter

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>A3 (C)</th>
<th>Paranhos (C)</th>
<th>Paranhos (D)</th>
<th>A3(D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>73</td>
<td>78.95</td>
<td>106</td>
<td>83.65</td>
</tr>
<tr>
<td>5</td>
<td>86</td>
<td>81.01</td>
<td>109</td>
<td>86.49</td>
</tr>
<tr>
<td>10</td>
<td>84</td>
<td>80.61</td>
<td>103</td>
<td>88.09</td>
</tr>
<tr>
<td>15</td>
<td>84</td>
<td>79.74</td>
<td>98</td>
<td>84.26</td>
</tr>
<tr>
<td>20</td>
<td>72</td>
<td>80.01</td>
<td>87</td>
<td>88.23</td>
</tr>
<tr>
<td>25</td>
<td>66</td>
<td>74.35</td>
<td>85</td>
<td>83.05</td>
</tr>
<tr>
<td>30</td>
<td>65</td>
<td>81.37</td>
<td>78</td>
<td>89.56</td>
</tr>
<tr>
<td>35</td>
<td>71</td>
<td>79.87</td>
<td>84</td>
<td>87.31</td>
</tr>
<tr>
<td>40</td>
<td>48</td>
<td>81.98</td>
<td>58</td>
<td>91.4</td>
</tr>
<tr>
<td>45</td>
<td>53</td>
<td>79.04</td>
<td>64</td>
<td>89.23</td>
</tr>
<tr>
<td>50</td>
<td>63</td>
<td>81.3</td>
<td>59</td>
<td>94.46</td>
</tr>
<tr>
<td>55</td>
<td>49</td>
<td>79.18</td>
<td>66</td>
<td>87.65</td>
</tr>
<tr>
<td>60</td>
<td>38</td>
<td>80.63</td>
<td>49</td>
<td>88.24</td>
</tr>
<tr>
<td>65</td>
<td>45</td>
<td>78.42</td>
<td>46</td>
<td>85.52</td>
</tr>
<tr>
<td>70</td>
<td>27</td>
<td>76.11</td>
<td>39</td>
<td>88.05</td>
</tr>
</tbody>
</table>

Continued on next page
Table A.1 – continued from previous page

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Vehicles (qpkw)</th>
<th>Avg. speed (vpkw)</th>
<th>Vehicles (qpkw)</th>
<th>Avg. speed (vpkw)</th>
<th>Vehicles (qpkw)</th>
<th>Avg. speed (vpkw)</th>
<th>Vehicles (qpkw)</th>
<th>Avg. speed (vpkw)</th>
</tr>
</thead>
<tbody>
<tr>
<td>75</td>
<td>33</td>
<td>78.03</td>
<td>42</td>
<td>86.02</td>
<td>50</td>
<td>83.78</td>
<td>44</td>
<td>80.16</td>
</tr>
<tr>
<td>80</td>
<td>29</td>
<td>78.55</td>
<td>41</td>
<td>86.59</td>
<td>53</td>
<td>88.6</td>
<td>45</td>
<td>86.84</td>
</tr>
<tr>
<td>85</td>
<td>29</td>
<td>81.59</td>
<td>47</td>
<td>91.96</td>
<td>31</td>
<td>86.45</td>
<td>28</td>
<td>80.11</td>
</tr>
<tr>
<td>90</td>
<td>20</td>
<td>81.25</td>
<td>29</td>
<td>86.93</td>
<td>41</td>
<td>88.34</td>
<td>31</td>
<td>81.65</td>
</tr>
<tr>
<td>95</td>
<td>22</td>
<td>79.36</td>
<td>32</td>
<td>89.97</td>
<td>52</td>
<td>85.13</td>
<td>41</td>
<td>77.85</td>
</tr>
<tr>
<td>100</td>
<td>19</td>
<td>83.74</td>
<td>26</td>
<td>95.12</td>
<td>33</td>
<td>79.82</td>
<td>28</td>
<td>77.75</td>
</tr>
<tr>
<td>105</td>
<td>35</td>
<td>73.34</td>
<td>26</td>
<td>88.85</td>
<td>46</td>
<td>87.37</td>
<td>33</td>
<td>81.15</td>
</tr>
<tr>
<td>110</td>
<td>33</td>
<td>83.39</td>
<td>37</td>
<td>93.05</td>
<td>31</td>
<td>87.06</td>
<td>31</td>
<td>78.42</td>
</tr>
<tr>
<td>115</td>
<td>22</td>
<td>77.73</td>
<td>26</td>
<td>91.5</td>
<td>31</td>
<td>77.26</td>
<td>27</td>
<td>77.7</td>
</tr>
<tr>
<td>120</td>
<td>21</td>
<td>75.05</td>
<td>29</td>
<td>87.69</td>
<td>44</td>
<td>86.16</td>
<td>32</td>
<td>85.88</td>
</tr>
<tr>
<td>125</td>
<td>26</td>
<td>81.5</td>
<td>28</td>
<td>93.71</td>
<td>32</td>
<td>84.88</td>
<td>19</td>
<td>82.95</td>
</tr>
<tr>
<td>130</td>
<td>13</td>
<td>84.54</td>
<td>14</td>
<td>96.71</td>
<td>45</td>
<td>81.33</td>
<td>35</td>
<td>82.34</td>
</tr>
<tr>
<td>135</td>
<td>19</td>
<td>82.89</td>
<td>26</td>
<td>91.65</td>
<td>36</td>
<td>86.78</td>
<td>35</td>
<td>76.83</td>
</tr>
<tr>
<td>140</td>
<td>21</td>
<td>80.24</td>
<td>27</td>
<td>88.11</td>
<td>29</td>
<td>83.21</td>
<td>19</td>
<td>71.63</td>
</tr>
<tr>
<td>145</td>
<td>19</td>
<td>87.79</td>
<td>28</td>
<td>91.71</td>
<td>39</td>
<td>88.51</td>
<td>24</td>
<td>82.54</td>
</tr>
<tr>
<td>150</td>
<td>19</td>
<td>79.84</td>
<td>24</td>
<td>89.21</td>
<td>25</td>
<td>76.48</td>
<td>27</td>
<td>75.33</td>
</tr>
<tr>
<td>155</td>
<td>16</td>
<td>82.25</td>
<td>13</td>
<td>86.08</td>
<td>32</td>
<td>83</td>
<td>20</td>
<td>80.35</td>
</tr>
<tr>
<td>160</td>
<td>23</td>
<td>77.17</td>
<td>26</td>
<td>91.62</td>
<td>16</td>
<td>87.94</td>
<td>12</td>
<td>85.83</td>
</tr>
<tr>
<td>165</td>
<td>15</td>
<td>70.53</td>
<td>15</td>
<td>83.27</td>
<td>26</td>
<td>87.23</td>
<td>15</td>
<td>81</td>
</tr>
<tr>
<td>170</td>
<td>9</td>
<td>67.22</td>
<td>11</td>
<td>91.82</td>
<td>23</td>
<td>86.26</td>
<td>24</td>
<td>81.38</td>
</tr>
<tr>
<td>175</td>
<td>22</td>
<td>82.64</td>
<td>23</td>
<td>94.17</td>
<td>27</td>
<td>85.11</td>
<td>24</td>
<td>78.42</td>
</tr>
<tr>
<td>180</td>
<td>23</td>
<td>78.61</td>
<td>22</td>
<td>93.5</td>
<td>22</td>
<td>87.5</td>
<td>16</td>
<td>79.88</td>
</tr>
<tr>
<td>185</td>
<td>11</td>
<td>75.55</td>
<td>16</td>
<td>87.25</td>
<td>30</td>
<td>80.3</td>
<td>26</td>
<td>79.42</td>
</tr>
<tr>
<td>190</td>
<td>14</td>
<td>72.64</td>
<td>16</td>
<td>88.38</td>
<td>19</td>
<td>79.21</td>
<td>14</td>
<td>75.79</td>
</tr>
<tr>
<td>195</td>
<td>9</td>
<td>80.89</td>
<td>12</td>
<td>91.42</td>
<td>18</td>
<td>85</td>
<td>13</td>
<td>82.62</td>
</tr>
<tr>
<td>200</td>
<td>13</td>
<td>69.54</td>
<td>15</td>
<td>87.07</td>
<td>13</td>
<td>81.69</td>
<td>12</td>
<td>80.08</td>
</tr>
<tr>
<td>205</td>
<td>23</td>
<td>81.61</td>
<td>20</td>
<td>94.3</td>
<td>19</td>
<td>82</td>
<td>16</td>
<td>77.5</td>
</tr>
<tr>
<td>210</td>
<td>16</td>
<td>83.56</td>
<td>21</td>
<td>95.14</td>
<td>13</td>
<td>86.77</td>
<td>9</td>
<td>79</td>
</tr>
<tr>
<td>215</td>
<td>20</td>
<td>77.15</td>
<td>23</td>
<td>92.35</td>
<td>10</td>
<td>78.3</td>
<td>5</td>
<td>77.6</td>
</tr>
<tr>
<td>220</td>
<td>18</td>
<td>78.56</td>
<td>24</td>
<td>91.62</td>
<td>14</td>
<td>85.64</td>
<td>10</td>
<td>77.2</td>
</tr>
<tr>
<td>225</td>
<td>12</td>
<td>83.83</td>
<td>13</td>
<td>91.54</td>
<td>25</td>
<td>88.68</td>
<td>24</td>
<td>84.96</td>
</tr>
<tr>
<td>230</td>
<td>11</td>
<td>84.82</td>
<td>17</td>
<td>89.06</td>
<td>14</td>
<td>79.71</td>
<td>9</td>
<td>79.11</td>
</tr>
<tr>
<td>235</td>
<td>11</td>
<td>83.73</td>
<td>19</td>
<td>93.21</td>
<td>16</td>
<td>83.12</td>
<td>12</td>
<td>79.17</td>
</tr>
<tr>
<td>240</td>
<td>8</td>
<td>72.88</td>
<td>16</td>
<td>86.19</td>
<td>21</td>
<td>82.81</td>
<td>18</td>
<td>77.39</td>
</tr>
<tr>
<td>245</td>
<td>16</td>
<td>75.94</td>
<td>18</td>
<td>90.06</td>
<td>14</td>
<td>93.36</td>
<td>13</td>
<td>84.23</td>
</tr>
</tbody>
</table>

Continued on next page
Table A.1 – continued from previous page

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Vehicles (qpkw)</th>
<th>Avg. speed (vpkw)</th>
<th>Vehicles (qpkw)</th>
<th>Avg. speed (vpkw)</th>
<th>Vehicles (qpkw)</th>
<th>Avg. speed (vpkw)</th>
<th>Vehicles (qpkw)</th>
<th>Avg. speed (vpkw)</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>13</td>
<td>80.85</td>
<td>20</td>
<td>90.65</td>
<td>15</td>
<td>80.93</td>
<td>13</td>
<td>71.62</td>
</tr>
<tr>
<td>255</td>
<td>25</td>
<td>78.76</td>
<td>34</td>
<td>91.5</td>
<td>22</td>
<td>85.77</td>
<td>15</td>
<td>81.47</td>
</tr>
<tr>
<td>260</td>
<td>12</td>
<td>76.5</td>
<td>15</td>
<td>83.13</td>
<td>16</td>
<td>72.19</td>
<td>10</td>
<td>84.7</td>
</tr>
<tr>
<td>265</td>
<td>22</td>
<td>81.86</td>
<td>23</td>
<td>91.26</td>
<td>25</td>
<td>75.04</td>
<td>21</td>
<td>77</td>
</tr>
<tr>
<td>270</td>
<td>24</td>
<td>78.08</td>
<td>27</td>
<td>89.89</td>
<td>20</td>
<td>84.1</td>
<td>18</td>
<td>82.67</td>
</tr>
<tr>
<td>275</td>
<td>15</td>
<td>70.67</td>
<td>19</td>
<td>89</td>
<td>21</td>
<td>87.43</td>
<td>14</td>
<td>82.57</td>
</tr>
<tr>
<td>280</td>
<td>15</td>
<td>76.53</td>
<td>18</td>
<td>85.33</td>
<td>21</td>
<td>82.86</td>
<td>17</td>
<td>80.94</td>
</tr>
<tr>
<td>285</td>
<td>28</td>
<td>77.14</td>
<td>31</td>
<td>91.48</td>
<td>22</td>
<td>81.14</td>
<td>13</td>
<td>82.85</td>
</tr>
<tr>
<td>290</td>
<td>23</td>
<td>79</td>
<td>26</td>
<td>88.54</td>
<td>19</td>
<td>89.21</td>
<td>16</td>
<td>80.5</td>
</tr>
<tr>
<td>295</td>
<td>15</td>
<td>75.33</td>
<td>16</td>
<td>87.75</td>
<td>23</td>
<td>79.26</td>
<td>18</td>
<td>77.61</td>
</tr>
<tr>
<td>300</td>
<td>22</td>
<td>71.82</td>
<td>28</td>
<td>89.89</td>
<td>13</td>
<td>84.31</td>
<td>11</td>
<td>77.36</td>
</tr>
<tr>
<td>305</td>
<td>22</td>
<td>80.18</td>
<td>21</td>
<td>91.24</td>
<td>23</td>
<td>72.13</td>
<td>24</td>
<td>78.25</td>
</tr>
<tr>
<td>310</td>
<td>25</td>
<td>81.44</td>
<td>35</td>
<td>90.14</td>
<td>20</td>
<td>84.05</td>
<td>23</td>
<td>74.65</td>
</tr>
<tr>
<td>315</td>
<td>43</td>
<td>75</td>
<td>47</td>
<td>88.66</td>
<td>26</td>
<td>83.77</td>
<td>19</td>
<td>78.84</td>
</tr>
<tr>
<td>320</td>
<td>29</td>
<td>76.76</td>
<td>39</td>
<td>89.72</td>
<td>33</td>
<td>80.55</td>
<td>36</td>
<td>81.28</td>
</tr>
<tr>
<td>325</td>
<td>33</td>
<td>75.67</td>
<td>34</td>
<td>88.21</td>
<td>30</td>
<td>87.2</td>
<td>20</td>
<td>88.35</td>
</tr>
<tr>
<td>330</td>
<td>28</td>
<td>80.64</td>
<td>29</td>
<td>91.79</td>
<td>24</td>
<td>81.33</td>
<td>21</td>
<td>85.48</td>
</tr>
<tr>
<td>335</td>
<td>41</td>
<td>77.22</td>
<td>43</td>
<td>88.6</td>
<td>32</td>
<td>81.88</td>
<td>24</td>
<td>83.71</td>
</tr>
<tr>
<td>340</td>
<td>38</td>
<td>76.87</td>
<td>50</td>
<td>88.08</td>
<td>32</td>
<td>78.78</td>
<td>29</td>
<td>82.66</td>
</tr>
<tr>
<td>345</td>
<td>42</td>
<td>77.9</td>
<td>50</td>
<td>87.74</td>
<td>52</td>
<td>77.21</td>
<td>38</td>
<td>76.32</td>
</tr>
<tr>
<td>350</td>
<td>46</td>
<td>72.07</td>
<td>56</td>
<td>88.68</td>
<td>34</td>
<td>82.18</td>
<td>33</td>
<td>79.88</td>
</tr>
<tr>
<td>355</td>
<td>47</td>
<td>71.36</td>
<td>49</td>
<td>88.1</td>
<td>39</td>
<td>83.56</td>
<td>31</td>
<td>79.16</td>
</tr>
<tr>
<td>360</td>
<td>33</td>
<td>74.94</td>
<td>42</td>
<td>85.81</td>
<td>51</td>
<td>86.18</td>
<td>47</td>
<td>80.77</td>
</tr>
</tbody>
</table>
Data and configuration files used for testing the models

Listing A.1: Configuration file used in the tests performed out with dfrouter

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
  <input>
    <net-file value="paranhos_ramps.net.xml"/>
    <additional-files value="rotas.rou.xml taz.poi.xml sumo_detectors.add.xml"/>
  </input>
  <time>
    <begin value="0"/>
    <end value="8400"/> <!-- simulation time for +/- 6 hours-->
  </time>
  <time-to-teleport value="-1"/>
</configuration>
```

A.2 Data used for modelling traffic with OD matrices

Table A.2: OD matrix used in the experiments with OD2trips

<table>
<thead>
<tr>
<th>Origin/Destination</th>
<th>PteArrabida</th>
<th>Paranhos</th>
<th>FGuimaraes</th>
<th>CLindo</th>
<th>PteFreixo</th>
<th>Total vehicles</th>
</tr>
</thead>
<tbody>
<tr>
<td>PteArrabida</td>
<td>-</td>
<td>188</td>
<td>142</td>
<td>632</td>
<td>4032</td>
<td>4994</td>
</tr>
<tr>
<td>Paranhos</td>
<td>118</td>
<td>-</td>
<td>200</td>
<td>203</td>
<td>217</td>
<td>738</td>
</tr>
<tr>
<td>FGuimaraes</td>
<td>245</td>
<td>257</td>
<td>-</td>
<td>-</td>
<td>649</td>
<td>1151</td>
</tr>
<tr>
<td>PteFreixo</td>
<td>5266</td>
<td>495</td>
<td>1259</td>
<td>-</td>
<td>-</td>
<td>7020</td>
</tr>
<tr>
<td>Total vehicles</td>
<td>5629</td>
<td>940</td>
<td>1601</td>
<td>835</td>
<td>4898</td>
<td>13903</td>
</tr>
</tbody>
</table>

Listing A.2: Configuration file used in the tests performed out with OD2trips

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
  <input>
    <net-file value="paranhos_dummy.net.xml"/>
    <additional-files value="rotas.rou.xml taz.poi.xml vss_additional.add.xml vss_def.def.xml sumo_detectors.add.xml"/>
  </input>
  <time>
    <begin value="0"/>
    <end value="8400"/> <!-- simulation time for +/- 6 hours-->
  </time>
  <time-to-teleport value="-1"/>
</configuration>
```

A.3 Data used for incident scenario using OD matrices

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
  <input>
    <net-file value="paranhos_dummy.net.xml"/>
    <additional-files value="rotas.rou.xml taz.poi.xml vss_additional.add.xml vss_def.def.xml sumo_detectors.add.xml"/>
  </input>
  <time>
    <begin value="0"/>
    <end value="8400"/> <!-- simulation time for +/- 6 hours-->
  </time>
  <time-to-teleport value="-1"/>
</configuration>
```
Data and configuration files used for testing the models

Listing A.3: Configuration file used in the incident tests performed out with *OD2trips*

```xml
</input>
<time>
  <begin value="0"/>
  <end value="9300"/> <!-- simulation time for 10 hours-->
  <step-length value="0.5"/>
</time>
</configuration>
```

**Table A.3: OD matrix used in the experiments of incident scenario with *OD2trips***

<table>
<thead>
<tr>
<th>Origin/Destination</th>
<th>PteArrabida</th>
<th>Paranhos</th>
<th>FGuimaraes</th>
<th>Total vehicles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paranhos</td>
<td>66</td>
<td>-</td>
<td>-</td>
<td>66</td>
</tr>
<tr>
<td>FGuimaraes</td>
<td>65</td>
<td>-</td>
<td>-</td>
<td>65</td>
</tr>
<tr>
<td>PteFreixo</td>
<td>8727</td>
<td>232</td>
<td>233</td>
<td>9192</td>
</tr>
<tr>
<td>Total vehicles</td>
<td>8858</td>
<td>232</td>
<td>233</td>
<td>9323</td>
</tr>
</tbody>
</table>