InterSCSimulator: A Scalable, Open Source, Smart City Simulator

Eduardo Felipe Zambom Santana

Thesis presented to the
Institute of Mathematics and Statistics
of the University of São Paulo
in partial fulfillment
of the requirements
for the degree of
Doctor of Science

Program: Computer Science
Advisor: Prof. Dr. Fabio Kon

São Paulo
February 10th, 2019

InterSCSimulator: A Scalable, Open Source, Smart City Simulator

Eduardo Felipe Zambom Santana

This version of the thesis includes the corrections and modifications suggested by the Examining Committee during the defense of the original version of the work, which took place on February 10th, 2019.

A copy of the original version is available at the Institute of Mathematics and Statistics of the University of São Paulo.

Examining Committee:

- Prof. Dr. Fabio Kon - IME-USP
- Prof. Dr. Mariana Abrantes Giannotti - EP-USP
- Prof. Dr. Francisco José da Silva e Silva - UFMA
- Leandro Aparecido Villas - UNICAMP
- Daniel Macedo Batista - IME-USP

Acknowledgements

I thank my parents Luiz and Sonia for their support in all my life, and I thank my brother Luiz Henrique who opened the way for us to get here and also my sister Bruna for her support. I also acknowledge my girlfriend Brianda who has accompanied me throughout my career.

I thank Professor Fabio Kon for his guidance in the development of this work and for the many discussions over the last four years that have made me learn as much about Computer Science as about scientific research. I thank Professors Marco Aurélio Gerosa, Alfredo Goldman, Daniel Batista and Kelly Braghetto for their contributions and critics in the various works carried out during my PhD I thank the teachers Claudio Marte and Mariana Gianotti of the Polytechnic School who helped me in the concepts of traffic management that were very important for the development of this thesis. Finally, I thank Professor Antonio Francisco do Prado for his guidance in my master, which certainly made my PhD a lot easier.

I thank the researchers who contributed to the development of this work: Nelson Lago, Arthur Del Esposte, Lucas Kanashiro, Dejan Milojicic, Gustavo Covas, and Diego Tomasiello.

Resumo

Eduardo Felipe Zambom Santana. **InterSCSimulator: A Scalable, Open Source, Smart City Simulator:** . Tese (Doutorado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2019.

Grandes cidades ao redor do mundo enfrentam grandes desafios para garantir boas condições de vida para seus cidadãos. Uma abordagem para responder aos problemas das cidades é a ideia de Cidades Inteligentes, a qual tem como principal característica o uso de Tecnologias de Telecomunicações e Informação (TIC) para melhorar os serviços da cidade. Simular cenários de Cidades Inteligentes pode beneficiar bastante essa área de pesquisa e também gestores de cidades. Um simulador desse tipo precisa representar diversos tipos de agentes como carros, hospitais e a infraestrutura da cidade. Uma possível implementação desse simulador pode usar o modelo de atores como paradigma de programação, implementando cada agente como um ator. O Erlang é uma das linguagens de programação baseada no modelo de atores mais utilizadas para o desenvolvimento de aplicações de larga escala. Esta tese apresenta a primeira versão do InterSCSimulator, um simulador de Cidades Inteligentes de código aberto, extensível e de larga escala desenvolvido em Erlang. Experimentos mostraram que o simulador é capaz de simular todo o trânsito de uma metrópole como São Paulo. Adicionalmente, são apresentados diversos casos de usos demonstrando como o simulador pode ser utilizado em trabalhos sobre Cidades Inteligentes como pesquisas sobre novos modos de transportes, redes veiculares e aplicações de Cidades Inteligentes.

Palavras-chave: Cidades Inteligentes. Simulação. Escalabilidade.

Abstract

Eduardo Felipe Zambom Santana. **InterSCSimulator: A Scalable, Open Source, Smart City Simulator:** . Thesis (Doctorate). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2019.

Large cities around the world face numerous challenges to guarantee the quality of life of its citizens. A promising approach to cope with these problems is the concept of Smart Cities, of which the main idea is the use of Information and Communication Technologies to improve city services and infrastructure. Being able to simulate the execution of Smart Cities scenarios would be extremely beneficial for the advancement of the field and for governments. Such a simulator would need to represent a large number of agents such as cars, hospitals, and gas pipelines. One possible approach for doing this in a computer system is to use the actor model as a programming paradigm so that each agent corresponds to an actor. The Erlang programming language is based on the actor model and is the most commonly used implementation of it. In this thesis, we present the first version of InterSCSimulator, an open-source, extensible, large-scale traffic Simulator for Smart Cities developed in Erlang. Experiments showed that the simulator is capable of simulating millions of agents using a real map of a large city. We also present study cases which demonstrate the possible uses of the simulator such as tests new urban infrastructure and test the viability of future transportation modes.

Keywords: Smart Cities. Simulation. Scalability.

List of Abbreviations

AV	Autonomous vehicles
DEUS	Discrete-Event Universal Simulator
ETS	Erlang Term Storage
GPU	Graphical Processing Unit
IDE	Integrated Development Environment
MATSim	Multi-Agent Transport Simulation
MDE	Model-Driven Engineering
OD	Origin-Destination
OSM	Open Street Maps
PoI	Point of Interest
PLC	Power Lines Communication
PTV	Planung Transport A Verkehr AG
SUMO	Simulation of Urban Mobility
VANET	Vehicular Ad-Hoc Network

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Since 2007, most of the world population lives in cities (United Nations, 2009) and the currently available and resources are not enough to cope with the crescent demand caused by the population growth and the concentration (Caragliu et al., 2011). Making the cities smart can optimize the usage of resources and infrastructure affordably and sustainably. However, creating and deploying the infrastructure and develop the Smart City services and applications is a complex challenge due to a plethora of problems such as costs, risks, and political issues.

Currently, there are several experiments and infrastructure to test Smart Cities systems and initiatives, both from the academia and from the industry. For example, the Smart Santander project (Sanchez et al., 2014), with more than 20 thousand sensors deployed in Santander, and the Cisco company with projects in different cities such as Kansas City, Copenhagen, and Manchester. Most of these projects have a sensing infrastructure with temperature, noise, and rain sensors and a software platform to collect and analyze the data from the city. However, the vision of Smart Cities has different components not yet integrated on these projects such as vehicles and buildings, and some technologies not yet available such as autonomous cars. Another problem is most of the Smart Cities experiments were deployed in small or medium cities, which do not have complex challenges as the big metropolis such as São Paulo, Mexico City, and Tokyo.

An alternative to facilitate tests and experiments of Smart Cities applications and infrastructure is the use of simulators. Such simulators must be able to represent different aspects of the city such as traffic, resources distribution, and sensor networks. However, there is a large number of agents in a Smart City environment that require a very scalable simulator. For example, to reproduce the entire mobility pattern of the city of São Paulo, it is necessary to simulate approximately 15 thousand buses, 5 million cars, and 11 million people. Besides, the simulator must allow the straightforward definition of scenarios using real data from the cities, because most of Smart City simulators users are domain specialists and non-computer scientists. Another problem is that most of the simulators implement just a city domain such as traffic, public transportation, or energy distribution.

To allow the easy simulation of large-scale, Smart Cities scenarios we developed

InterSCSimulator, an open-source, scalable, Smart City simulator[1]. This simulator enables the definition of scenarios with high-level data such as maps, origin-destination surveys, and description of the bus system and the execution of them with millions of actors running simultaneously. InterSCSimulator aims to be useful to many stakeholders such as city administrators, researchers, and application and platform developers.

To develop the InterSCSimulator we used Erlang, a functional programming language used in the implementation of large-scale applications in many domains such as Internet instant message, databases, and telecommunications network. Erlang allows the straightforward development of massively parallel and distributed applications using the actor model (Agha, 1985). In this model, a program can create millions of parallel execution lines, called actors. The communication among them is through asynchronous messages, and the use of shared memory is not allowed, avoiding many concurrent problems such as deadlocks and busy wait.

Experiments showed that InterSCSimulator is capable of scaling to more than 20 million actors, enabling the simulation of big metropolises such as São Paulo and New York. The simulator also runs faster than real-time depending on the hardware infrastructure. To verify and validate the InterSCSimulator four different groups related to the InterSCity project already used the simulator in different scenarios: 1) Comparing different possibilities in the use of city subway infrastructure, 2) Generating workload to experiments in a Smart City platform, 3) Allowing analyses of a bus movement model, and 4) Creating a mobility trace of the city of São Paulo.

InterSCSimulator is in continuous development, and different research projects currently use it as a support tool or as the main topic. In this thesis, we present the general architecture of the simulator and the development of the mobility domain, including vehicle simulation and public transportation (subway and buses). We also comment on the integration of the simulator with InterSCity, a Smart City software platform, which allowed the test of traffic applications.

1.1 Motivation

Creating smarter cities can help in the improvement of the quality of life of billions of people around the world. The impact in big cities in poor and developing countries can be even more significant because of the lack of necessary infrastructure and services in these cities. The primary motivation of this work is to provide a tool that can facilitate the development of smart services to city managers, software developers, and researchers.

Also, Smart Cities research is gaining a lot of attention in the last years, as showed in Figure 1.1, the interest in the term increased a lot between 2014 and 2016 and since then maintain a constant amount of searches in Google[2]. However, there are several challenges in testing Smart City applications and initiatives due to costs, lack of infras-

[1]InterSCSimulator - github.com/ezambomsantana/smart_city_model
[2]Google Trends - trends.google.com/trends

tructure, and governmental issues. Moreover, a lot of the planned Smart Cities ideas are of technologies that are not available yet such as vehicular networks. Therefore, the use of large-scale simulators can enable the tests of such initiatives without building or changing the infrastructure of a city.

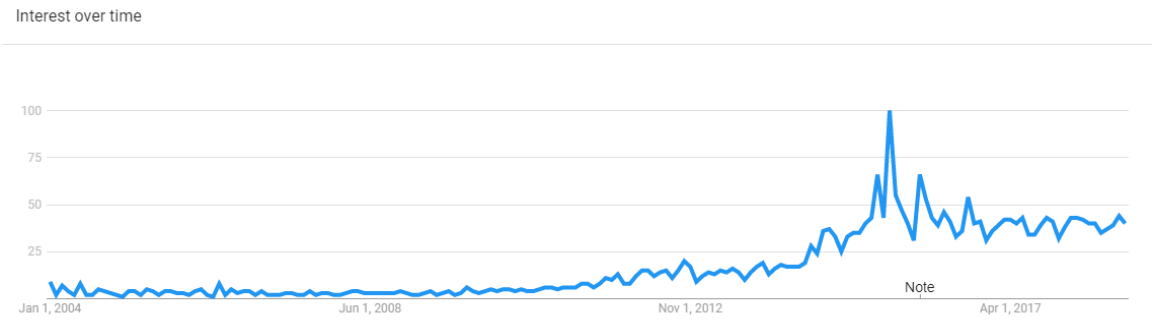

Figure 1.1: *Evolution in the number of searches about the term Smart City in Google from 2004 to 2018*

Besides tests and experiments of Smart Cities solutions, the simulator can also help city governments verifying the impacts of a myriad of actions in the city such as the building of a new bridge, of a new subway line, and changes in the bus lines itinerary. To evaluate the impact of these actions in entire cities, it is essential to provide a very-scalable simulator able to simulate the whole transportation infrastructure, including the road network, subway network, and bus system. Most of the current open source simulators can represent just a small area and with a limited number of agents and the proprietary options are not extensible, what prevents the creation of new simulation scenarios.

1.2 Objectives and Challenges

The primary objective of this work is design, implement, and evaluate InterSCSimulator, a large-scale Smart City simulator. The simulator must enable the simulation of a city with millions of actors in an extensive area such as the city of São Paulo with more than 11 million inhabitants, 4 million cars, 15 thousand buses, and 48 thousand streets in an area of more than 1,000 km^2. It is also objective of this work provide tools to facilitate the development of simulation scenarios using real data such as the map of the city, origin-destination surveys, and buses time-tables. The simulator also must generate output data that are easy to analyze and visualize.

A secondary objective is to show that the simulator is useful for different activities in Smart Cities projects. For example, the test and experiments of Smart City applications and platforms, to evaluate the infrastructure of a city, and as a testbed for new city simulation models.

To execute a simulation with the desired scalability, we had to solve many computational challenges. The first challenge was to select the right language and tools that enable the implementation of highly scalable software. Also, we had to test many data

structures and algorithms to support the simulation scale. This research was necessary because most of the current simulators enable the simulation of just hundreds or thousands of simultaneous objects because of problems in their architecture. To simulate a city with the size of São Paulo in a reasonable time it is required a highly scalable and efficient implementation.

To achieve the necessary scalability, we developed a simulator able to execute parallel and distributed simulations. It adds yet more complexity to the implementation of the simulator because some of the models used in the simulator were never used in parallel implementations. To facilitate this task, we used Erlang, a language that promotes the development of scalable parallel and distributed applications.

Finally, developing simulation models consistent with reality is also a challenge. To solve this problem, we made extensive research of different models and algorithms used to simulate cities such as traffic and public transportation models. We also used a lot of real-data such as maps and origin-destination surveys to further the realism of the simulations.

1.3 Research Questions

Our work involves answering the following two research questions:

RQ1: "What are the requirements to develop a general purpose, scalable Smart City simulator?"

RQ2: "What is the most suitable programming model for the development of a large-scale Smart City simulator?"

To answer RQ1, we conducted a literature review of Smart Cities and related simulators. With this review, we identified the functional and non-functional requirements that a Smart City simulator should handle. For example, the simulator must represent the city map in an efficient data structure and allow a straightforward definition of the trips that must be simulated.

To answer RQ2, we developed a Smart City simulator using Erlang, a language that implements the actor model and performed experiments to evaluate the simulator scalability. The evaluation shows that the simulator handles more than 20 millions actors efficiently in a single simulation executing faster than real-time.

Finally, to discuss the possible uses of the simulator, we present the contexts that InterSCSimulator were already employed. For example, the simulator was used to analyze the scalability of a Smart City platform and to compare mobility scenarios in the city of são Paulo.

1.4 Thesis Organization

This thesis is organized as follows. Chapter 2 presents the fundamental concepts used in this thesis which are Smart Cities, Traffic Simulation, and the Actor Model. Chapter 3 cites related work including simulators of different domains of Smart Cities. Chapter 4 introduces the development of InterSCSimulator, showing the functional and non-functional requirements, architecture, components, and implementation evolution. Chapter 5 presents the simulation of the city of São Paulo, with more than 20 million trips, which served as the base for the scalability evaluation of the simulator. Chapter 6 presents the scalability evaluation of the simulator, showing experiments regarding its execution time and resources usage. Chapter 6.4 compares the scalability of InterSCSimulator and SUMO. Chapter 7 presents examples of research already supported by the InterSCSimulator. Finally, Chapter 8 discusses the results of this thesis and future work.

Chapter 2

Fundamental Concepts

This chapter presents the main base concepts used in the development of this work. Section 2.1 presents definitions and dimensions of Smart Cities and possible simulation scenarios. Section 2.2 describes the fundamentals of traffic simulation, highlighting the types of traffic simulations and examples of Smart Cities scenarios already simulated. Section 2.3 presents the actor model and the Erlang language which we used in the implementation of the InterSCSimulator. Finally, Section 2.4 concludes this chapter, relating the presented concepts and the research presented in this thesis.

2.1 Smart Cities

Most of the Smart Cities definitions highlights the expected impacts of innovative services and applications in the city population and the environment such as citizens empowerment, better quality of life, and sustainability. Other, focus on the idea of using ICT tools to create and improve the cities infrastructure and services. Table 2.1 presents Smart Cities definitions that we found in the literature. Most of these definitions cite that the primary objective of a Smart City is improving the citizen quality of life. Some definitions (Giffinger et al., 2007; Guan, 2012) do not establish the mean to achieve this objective, while others define that this objective will be achieved using a technological infrastructure to improve the infrastructure and services of the city (Caragliu et al., 2011; Dameri, 2013; Harrison et al., 2010).

Most of the Smart Cities definitions cite the necessity of using Information and Communication Technologies (ICT) to improve the use of the city infrastructure, the resources management, and the city services (Harrison et al., 2010; Washburn et al., 2009). Some definitions also cite the importance of sustainability in the city, making more efficient use of resources such as water and electricity (Caragliu et al., 2011; Dameri, 2013).

Another important discussion is the necessity of leveraging the economic development of the cities (Dameri, 2013), facilitating the integration and participation of the whole city population. Two definitions (Dameri, 2013; Giffinger et al., 2007) cite the importance of participatory governments allowing the citizens define the cities priorities.

Table 2.1: *Smart Cities definitions found in the literature*

Definition	Author
"A Smart City is a city well performing built on the 'smart' combination of endowments and activities of self-decisive, independent and aware citizens"	(Giffinger et al., 2007)
"A city to be smart when investments in human and social capital and traditional (transport) and modern (ICT) communication infrastructure fuel sustainable economic growth and a high quality of life, with a wise management of natural resources, through participatory governance"	(Caragliu et al., 2011)
"A smart city is a well-defined geographical area, in which high technologies such as ICT, logistic, energy production, and so on, cooperate to create benefits for citizens in terms of well-being, inclusion and participation, environmental quality, intelligent development; it is governed by a well-defined pool of subjects, able to state the rules and policy for the city government and development"	(Dameri, 2013)
"A city that monitors and integrates conditions of all of its critical infrastructures, including roads, bridges, tunnels, rails, subways, airports, seaports, communications, water, power, even major buildings, can better optimize its resources, plan its preventive maintenance activities, and monitor security aspects while maximizing services to its citizens"	(Hall et al., 2000)
"A city connecting the physical infrastructure, the IT infrastructure, the social infrastructure, and the business infrastructure to leverage the collective intelligence of the city"	(Harrison et al., 2010)
"A smart city, according to ICLEI, is a city that is prepared to provide conditions for a healthy and happy community under the challenging conditions that global, environmental, economic and social trends may bring."	(Guan, 2012)
"The use of Smart Computing technologies to make the critical infrastructure components and services of city which include city administration, education, health-care, public safety, real estate, transportation, and utilities more intelligent, interconnected, and efficient"	(Washburn et al., 2009)

Also regarding ICT, some definitions cite that Smart Cities applications and services should monitor the cities infrastructure such as streets, bridges, train lines and stations, and public buildings (Hall et al., 2000). Also, they must monitor and control their resources such as water and electricity (Hall et al., 2000). Finally, the data collected in the city monitoring must be available and integrated to facilitate the creation of Smart Cities applications and services (Harrison et al., 2010; Washburn et al., 2009).

Besides the Smart Cities definitions, Giffinger et al. (Giffinger et al., 2007) describe six dimensions to measure the smartness of a city. The dimensions are: *Smart Economy*, *Smart People*, *Smart Governance*, *Smart Mobility*, *Smart Environment* and *Smart Living*. Many authors already use this classification (Hernández-Muñoz et al., 2011; Papa et al., 2013) and there is a benchmark used to rank the smartest cities in Europe using these dimensions.[1] The definition of each dimension is in the following:

- **Smart Economy** measure the economic development of the city through parameters such as the quality of the enterprises in the city and the city entrepreneurship ecosystem. Examples of initiatives related to this dimension are incentives to companies for the development of technological solutions for the city and the improvement of the business environment with adequate legislation and business infrastructure.
- **Smart People** is related to the development of the city's population using parameters such as education, employment rate, and income. Some actions related to this dimension are projects for digital inclusion of citizens and programs for scientific and technological education.
- **Smart Governance** access the quality and transparency of municipal public agencies with parameters such as ease of use of public services, investments in technology, and transparency in the public data and the use of city resources. Some actions related to this dimension are the creation of participatory governments and the dissemination of information about the city in transparency and open data portals.
- **Smart Mobility** measures the ease of mobility in the city in the various modes of transportation such as bus, subway, car, and bicycle. It uses parameters such as kilometers of congestion, subway network size and the number of people using public or non-polluting transportation. Some actions related to this dimension are real-time monitoring of streets flow, the use of sensors to indicate free parking spots and applications to facilitate and encourage the use of public and sustainable transport, such as bicycles and electric vehicles.
- **Smart Environment** check the sustainability of the city using parameters such as environmental pollution, efficiency in the use of city resources such as water and electricity, and the amount of recycled waste. Some actions related to this dimension are the measurement of the city's air and water quality, the use of renewable energy sources and the real-time measurement of the resources used in buildings.
- **Smart Living** evaluates the quality of life of the population using parameters such as entertainment, security, and culture. For example, counting the number of green areas, the number of libraries and homicide rate of the city. Some actions related

[1] Smarts Cities in Europe - www.smart-cities.eu

to this dimension are the use of elderly health monitoring applications, automatic image processing of security cameras and applications that show the cultural events programmed in the city.

There are a plenty of Smart Cities initiatives around the world, most in Europe (Caragliu et al., 2011; Manville et al., 2014), several in the United States [2], Japan and China (Liu and Peng, 2013) and some other project in other countries such as Brazil (Fortes et al., 2014), United Arab Emirates (Janajreh et al., 2013) and South Korea (Kshetri et al., 2014). These data show that the vast majority of projects are concentrated in developed countries, there are a few projects in developing countries. In Brazil, there are already several initiatives such as in São Paulo, Búzios, Recife, and Joinville. We did not find any project in the poorest countries of the globe. Figure 2.1 shows a map of the initiatives found in the literature or pages of the projects.

Figure 2.1: *Smart Cities initiatives around the world*

Some examples of very advanced initiatives in Smart Cities are Santander, Spain, which through the SmartSantander project has already deployed an extensive sensor network in the city to collect data such as temperature, vacant parking spot and noise levels in the city streets. Amsterdam, which has a variety of Smart Cities projects such as encouraging the use of electric cars, bicycles and public transport, automatic monitoring of the city conditions and the use of a smart electricity distribution network. Barcelona, Spain, which has several projects to increase citizens' participation in city decision-making and to make data available on public administration openly.

The next sections present Smart Cities initiatives, applications, and services already implemented in cities around the world. Based on these initiatives, we derived possible scenarios that are possible to simulate in a Smart City simulator.

[2] 10 Smartest Cities in USA - www.fastcoexist.com/3021592/the-10-smartest-cities-in-north-america

2.1.1 Smart Economy

This subsection presents projects related to the economic growth of the city, attracting investments and creating more and better jobs. Examples of services and applications of this field are associated with tourism, efficient usage of city resources and the attraction of enterprises and startups to the city.

In Santander, researchers developed an augmented reality application to smartphones which contain plenty of information of more than 2700 points of interest in the city such as museums, bookstores, buses stop, tourism office, and bicycle rental stations. The application also shows the position of buses and taxis in real-time (Sanchez et al., 2014). Figure 2.2 presents the main screen of the applications, which shows the information of a bus line and a point of interest of the city.

Figure 2.2: *SmartSantander augmented reality application*

In Cagliari, Italy, the municipal government has developed a platform based on the Internet of Things and Cloud Computing to collect data and create services for tourists in the city (Nitti et al., 2017). To test the platform, they developed a case study in which a tourist selects a list of Points of Interest (PoIs) that they want to visit in the city. Each PoI has a variety of static data, such as address and hours of operation, and data captured in real time, such as the size of the entrance queue and the current number of visitors. With this data, the application calculates the best sequence of PoI that the tourist should visit. The purpose of the application is to optimize the time of the tourist, allowing him to be able to visit as many attractions as possible in the time that he is in the city.

Also related to tourism in Amsterdam, the government has adopted the CitySDK

Tourism API (Pereira et al., 2015), a tool that allows the development of applications to help tourists visiting the city. This tool collects data from the city's open data portal, which is in CSV, XLS, and text files and makes them available in an easy-to-access and processing API for third-party applications. Some of the shared data are city landmarks such as museums, parks and historic buildings, events that are happening in the city and tourist itineraries.

Búzios is one of the first cities in Brazil to start a project for the implementation of a Smart City infrastructure (Fortes et al., 2014). The main purpose of the project is to make the city more sustainable, using resources rationally and efficiently. Among the main actions carried out in the city is the implantation of an intelligent electric energy network, the creation of intelligent buildings and the improvement of the communication systems of the city using technologies such as Wi-Fi, Mesh networks, and Power Lines Communication (PLC).

Also in Amsterdam, the government deployed the first Smart Electricity Grid in a region of the city with approximately 10,000 houses. In this network, it is possible for c to consume and produce energy and to monitor energy usage in their homes in real time. Also, this project also facilitates the monitoring and maintenance of the network by city authorities.

In this application domain, researchers can develop simulations of a Smart Grid, comparing the cost and the emission of pollutants in the use of diverse sources of electrical energy and simulating the network of distribution, production, and consumption of energy. They can also simulate buildings that attract large numbers of people, such as sights and places of big events, making it possible to understand their impacts on the city's traffic.

2.1.2 Smart Population

This subsection presents projects aimed at improving social parameters related to the population of the city such as education, employment, and income. Moreover, some researchers discuss the empowerment of citizens with data that allow them to make better political choices. Examples of services and applications in this area are related to education such as initiatives and applications for improving education and facilitating the digital inclusion of city citizens and improving the city's business environment by increasing the quantity and quality of jobs.

An initiative in England teaches students to work with data sets related to the city (Wolff et al., 2015). The idea is to enable citizens to know the tools to analyze the data independently of the will of companies or city rulers. The purpose of this work is to extend class activities with activities such as Hackathons aimed at the development of applications and services for the city. The researchers already did tests using datasets on the use of electricity in the city.

In Barcelona, the municipality created a laboratory (Barcelona Urban Innovation Lab Dev), which investigates various urban problems and fosters the participation of the private sector in the development of products and services related to the improvement of urban

space (Bakici et al., 2013). The laboratory provides human resources and tools to support the development of these solutions. The objective of this laboratory is to attract companies to develop tools for the city, creating jobs that need high qualification and also creating solutions to improve quality of life in the city.

Scenarios linked to education can be simulated in this area such as schools and universities, and from which part of the city these institutions attract the population. So governors and private schools can choose the best area for the creation of new schools or university campuses and also measure the impact on the city traffic.

2.1.3 Smart Governance

This subsection presents projects related to the governance of the city. The main objectives of this type of projects are facilitating the city administration and enable the citizen participation in the city decision-making. Examples of services and applications in this field are platforms to city monitoring, open data portals, and the encouragement of the citizens to participate in the city decisions.

Seattle is considered by some rankings the smartest city in the United States [3]. There, researchers performed a survey (AlAwadhi and Scholl, 2013) with citizens and public agents asking what the main services, applications, and initiatives that the city are developing to improve the citizens' quality of life are. Among the cited projects are the open data portal [4], the infrastructure to support the use of electric vehicles, and the use of Customer Relationship Management (CRM) to control the communication between the city government and the citizens. According to the survey, the benefits of these actions are the improvements in the city services, the reduction of the city expenses, and electrical energy saving.

In Chicago, the municipality developed the platform WindyGrid (Thornton, 2013), which collects, stores, and process the data of the city. The objective of this platform is providing a unified platform to city operators visualize the city operation in real-time. Examples of the collected data are calls to the emergency service (911), events in the city traffic, publications about the city in social networks, and data regarding the public buildings. The WindyGrid provides three main functional requirements to the city managers: incidents monitoring though emergency calls and social networks mining, historical data visualization, and real-time analyses of events in the city.

In Amsterdam, there are many projects to leverage the city management transparency, especially the city expense and the decisions of the city government. For example, the Budget Monitoring allows citizen and NGOs access and suggest changes to the city budget and the Smart City SDK, which provides access to the real-time city data to application developers, including data about the city traffic, airports arrivals and departures, and the climate. Finally, to facilitate the participatory government, the city developed the AmstermOpent, a platform that allows the citizens to suggest actions and works in the city.

[3] https://goo.gl/5xAhh9
[4] Seattle Open Data - data.seattle.gov

Many cities around the world already provide access to the city documents through open data portals allowing citizens to supervise the government expense and actions. For example, Barcelona[5] make available a big number of data about the city administration such as the city budget and expenses, municipal services stats, and data about the city population. Dublin, in Ireland, also has a completely open data platform called Dublinked (Stephenson et al., 2012) which provide access to more than 200 datasets offering historical and real-time data. Moreover, using Web Semantic tools, it is possible to link the data from the platform automatically.

São Paulo has many open data projects such as the open data portal of the city [6], and GeoSampa[7] which provides different georeferenced datasets such as the location of public equipment, buses stops, and flood points in the city. Another example is the Olho Vivo API [8], which allows the monitoring in real-time of all the buses in the city. This API allowed the development of many applications to facilitate the mobility in the city by buses such as Moovit[9] and Coletivo[10].

Another tool extensively used by the cities to share their data with the citizens is the dashboards. Usually, these tools present real-time data in a city map with various information about the city such as climatic conditions, air quality, traffic conditions, and state of the public equipment. One dashboard example is the Dublin dashboards[11], which provides different data about the city such as temperature, air quality, noise levels, and level of the rivers in the city. Figure 2.3 presents an example of the city dashboard displaying data of the city traffic.

This Smart City domain is linked to tools for monitoring the city, and we did not think in any scenario that can be simulated. However, a Smart City simulator can assist in tests and experiments of applications cited in this subsection, enabling the generation of workloads such as sensor and traffic simulation for the creation of monitoring applications.

2.1.4 Smart Mobility

This subsection presents projects related to mobility, which has the main objective of improving the people flow in the city and monitor the mobility infrastructure of the city such as roads and subway stations. There are plenty of examples of Smart Mobility services and applications such as traffic monitoring through security cameras, best route services, smart parking applications, and applications to show the best route using public transportation.

With the SmartSantander platform, a research project developed an infrastructure to show the free parking spots in the city. Moreover, it creates a service to predict the use of the spots during events in the city (Vlahogianni et al., 2014). This service has the

[5] OpenData BCN - opendata.bcn.cat/opendata/ca
[6] Dados Abertos São Paulo - saopauloaberta.prefeitura.sp.gov.br
[7] GeoSampa - geosampa.prefeitura.sp.gov.br
[8] API Olho Vivo - www.sptrans.com.br/desenvolvedores/APIOlhoVivo.aspx
[9] Moovit - https://goo.gl/FMYk8u
[10] Coletivo - https://goo.gl/QBvoNc
[11] Dublin Dashboards - www.dublindashboard.ie

Figure 2.3: *Dublin Dashboard showing available park spots, available bikes in bikes stations, and the traffic situation in avenues in the city*

objective of avoiding that drivers lose time searching for a parking spot in the city, what increases the traffic in the city and the emission of polluting gases. Figure 2.4 presents the monitored parking spots in the city map.

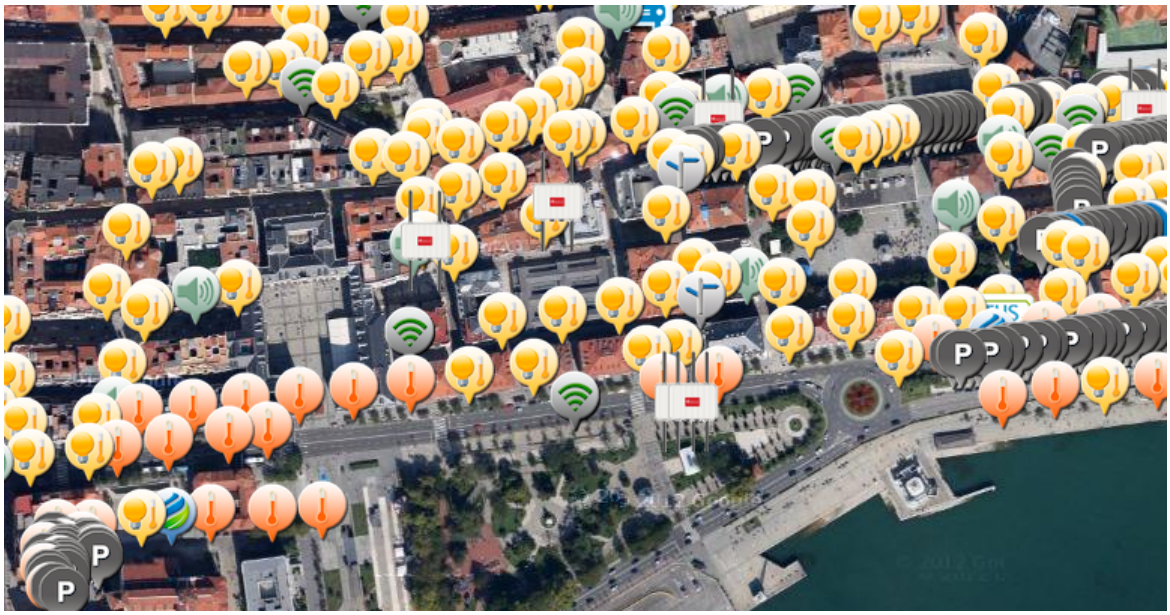

Figure 2.4: *Map with the monitored parking spots in the city of Santander*

The city of Barcelona also has a project to encourage the use of sustainable transportation modes. For example, the city developed a large infrastructure to facilitate the use of electric cars with more than 300 recharge points deployed in the city. Moreover, the city is promoting the use of shared bicycles and provided more than 400 loan stations to the citizens.

Amsterdam is also implementing several solutions in the area of traffic control and monitoring. For example, projects under development in the city are: encouraging the use of electric cars, providing battery recharging stations in various parts of the city, monitoring the main city roads for the rapid attendance of traffic problems, reservation of parking spaces in the city, avoiding the search for a vacancy, reducing the emission of CO_2, and the incentive to use bicycles.

In Madrid, researchers developed other two projects. The first was an application to smartphones to facilitate the commuters to find the best route using public transportation in the city. The applications also used contextual information to find best routes such as the user location and the city weather. The second application uses the user's smartphones to estimate the number of people inside a bus. People expecting for a bus can use this information to decide if it worth to take the bus or wait until the next one.

In São Paulo, the startup Scipopulis [12] developed a Bus Panel used by the São Paulo Transport Secretariat (SPTrans) and the Company of Traffic Engineering (CET). This application monitors the more than 14,000 city buses and shows the speed of buses in real time on all the streets and integrates data from various sources (GPS positions of buses, segregation of the road, accidents, etc.). This information is contextualized regarding the time of day, type of road (corridor, track or shared road), accidents in the region and amount of bus passing through that route. The operator can monitor entire bus lines or sections of a line, and view the history of speeds for each street segment. The city transport network planning, operation and management teams use the panel to identify chronic bottlenecks, contingency problems, identify ways to implement buses single lanes and corridors, and the times at which the buses should operate. Figure 2.5 shows a screen of the Bus Panel application.

In the dimension of Smart Mobility, it is possible to simulate a large number of scenarios such as the movement of vehicles in the city, compare various parameters in the road network such as increasing or decreasing street speed, accidents, and problems in roads such as floods or protests. It is also possible to simulate the public transportation of the city such as buses and subways, comparing the city traffic by increasing or decreasing the number of users on public transport, searching for the best route to a new bus line and analyzing the impact of new subway lines.

2.1.5 Smart Environment

This subsection presents projects related to the environment. Most of them have the objective of making the city more sustainable improving services such as garbage collection and recycling, efficient distribution of resources such as water and electricity, and reducing air and water pollution in the city.

Masdar is a neighborhood in the city of Abu Dhabi in the United Arab Emirates built with the objective of testing several initiatives of Smart Cities. The areas tackled in the project are the use of renewable energy sources, the conscious use of water and the reduction of the amount of garbage generated. Also, the city was planned with an

[12] Scipopulis - www.scipopulis.com

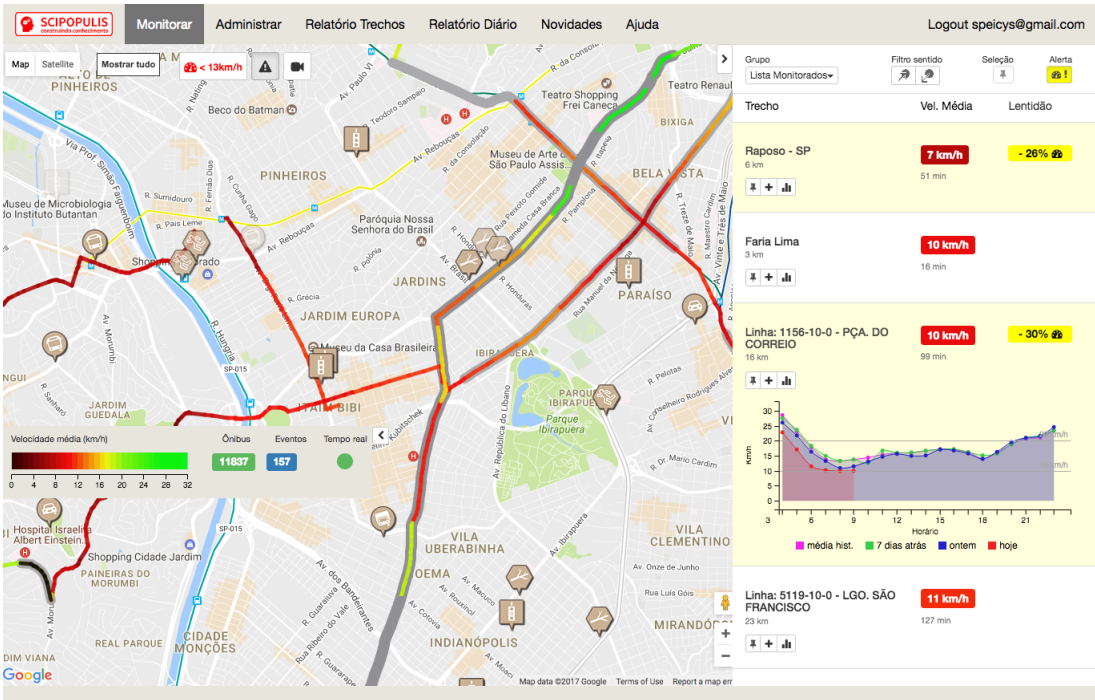

Figure 2.5: *São Paulo bus dashboard showing the speed of the buses in the main avenues and bus corridors in the city*

intelligent transportation network to reduce the need for the use of individual vehicles, reducing the emission of pollutants. In this neighborhood, all buildings are designed in a way that saves resources and produces their energy with the use of solar panels.

In Manchester, the city is developing a project to build smart houses. In this houses, the citizen can monitor in real-time the use of resources such as electric energy and water. The objective of this project is to decrease the pollution emission and save the natural resources of the city (Manville et al., 2014).

The city of Santander has implemented a project to manage garbage collection in the city(Díaz-Díaz et al., 2017). This project uses data from more than 1000 sensors that monitor how full the city dumps are. Also, it is possible to monitor garbage trucks and garbage dumps. This system allows garbage trucks to visit only dumps that are full, reducing the distance that the trucks must travel. Three significant benefits of this service are the decrease in the truck expenses, the reduction of emissions of pollutants by the vehicles and the better management of the garbage collected in the city. Barcelona also has a similar project in which it is possible to monitor the current state of the garbage dumps in the city. Figure 2.6 shows the city map with the sensors in the dumps.

Also in Santander, the city is deploying a Smart Light project, which installed more than 20.000 LED lamps with a movement detection system. This system turns on the lamp only when it detects a person is moving close from the lamppost. Also, along with the system, the smart lamps also allow the monitoring of the state of the lamp, facilitating the detection of problems in the system. With this project, the city plans to reduce 80% the electric power consumption with the public lighting. Also, the city expects to reduce the expenses with the maintenance of the system, because currently the maintenance is made

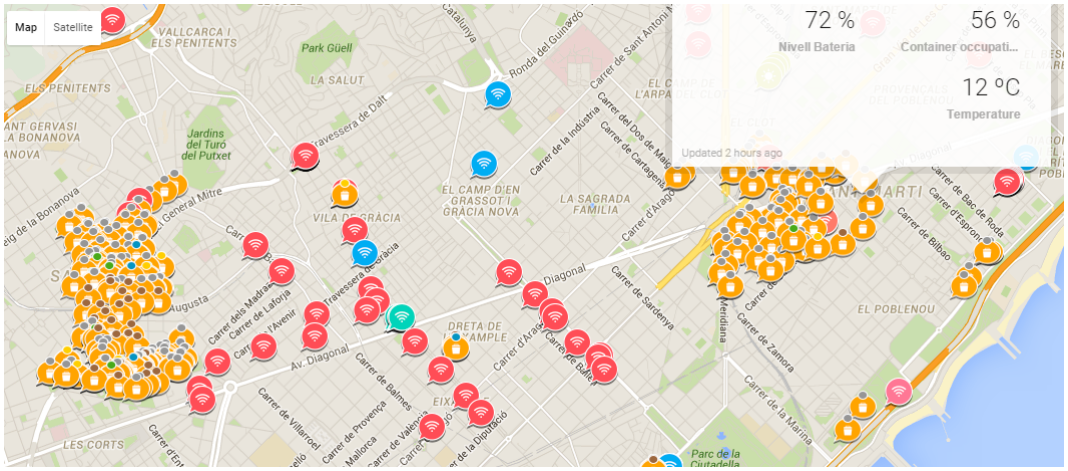

Figure 2.6: *Sensors indicating the amount of garbage in the city dumps in the Sentilo platform*

through rounds across the city.

A district of Barcelona is already operating a system for heating and cooling buildings (March and Ribera-Fumaz, 2016). The system works with a water network that passes through several district buildings, mainly public building and using the energy generated by the city's waste incinerators heat or cool the water in the pipeline. The city estimated that this system uses 35% less power and reduces pollutant emissions by 50% than conventional systems heating systems.

In the dimension of Smart Environment, it is possible to simulate the impact of different initiatives in the environment. For example, the emission of pollutants by the car fleet of the city, the variation in the emission if more people use public transportation, and the impact of using electric vehicles. Also, it is possible to simulate the environmental impact of the use of different electric power sources such as hydroelectric, wind, and solar.

2.1.6 Smart Living

This subsection presents projects related to the improvement of citizens' quality of life. These project aims to improve services that are directly related to citizens' routine, such as security, cultural activities and sports activities. Examples of services in this dimension are applications that warn of events taking place in the city, for monitoring crowded areas, and for reporting problems in public places such as parks and government buildings.

In Santander, researchers installed a network of more than 20 thousand sensors and actuators in the city. This sensor network collects a vast amount of data in various regions of the city such as temperature, free parking spaces, points of interest and luminosity. The data collected is used for the development of applications and services that improve the quality of life of the citizens. Figure 2.4 shows a map in which each point is a device deployed in the city that sends data to the platform. An example of an application developed with the data of the sensor network is one that reminds the city population of events that will occur in the city. Besides the event, the application also sends information about the region of the event to the users such as parking spots, noise levels, and temperature.

In Dublin, in the same dashboard presented in Section 2.1.3 the city provides in real-time the stream of surveillance cameras in different areas in the city. These videos allow the monitoring of various problems that can occur in the city such as traffic accidents, crimes, and health issues. Also, developers can use these video stream to the development of applications in the city to automatically monitor the city conditions such as level of the rivers and traffic problems.

The data used in the applications and services presented in this section are about activities that will occur and streams of surveillance cameras captured in real-time in the city. In this domain, it is possible to simulate the deployment of a sensor network in the city, allowing the comparison of the costs and coverage area of the network depending on the number and type of the sensors. This simulation can also be very useful to the tests and experiments for Smart City applications and services.

2.2 Traffic Simulation

One of the main components of a Smart City simulator is the traffic model because it controls the way the city road infrastructure is implemented. For example, it is possible to model the city as a matrix or using a digraph. This section presents concepts of traffic simulations including the city, vehicle, and people representation. Moreover, we describe the most common traffic models used in the literature: microscopic, mesoscopic, and macroscopic. These three types of traffic simulation differ mainly in the detail level of the simulation (Barceló et al., 2010). The main characteristics of each type of traffic model are:

Microscopic: Microscopic simulators model each vehicle individually and to calculate the speed of the vehicle they used mathematical models that consider the behavior of each vehicle about the other vehicles. In the literature, we found two most common models, the **Car Following Model**, which calculate the speed of the vehicle according to the speed of the vehicles in front of it and the **Lane Change Model**, which models when a vehicles change the lane in the street. This type of simulation is usually used to understand the behavior of small areas in a city such as intersections and rotations.

Mesoscopic: Typically, the mesoscopic models also model each vehicle individually. However, the speed of the vehicle in a particular path is calculated by a density function that usually considers the length and number of lanes of the road to calculate its capacity. Researchers normally use macroscopic models for simulating large areas such as a neighborhood and even whole cities.

Macroscopic: Macroscopic models model the transit of a region as flows in a road network. This type of simulation does not consider individual vehicles and speeds are also calculated by functions that analyze the size of the vehicles flow on the road over a given period. There are macroscopic simulators capable of simulating a road network of an entire country.

Among the types of traffic simulation presented, the mesoscopic models are the more

suitable for this work. For a Smart Cities simulation, it is necessary to model individual vehicles for analysis and model scenarios for application and platform testing. However, there is no need to model in detail the interaction between the vehicles, as the microscopic models do. Another significant advantage of mesoscopic models over microscopic models is that in mesoscopic models it is possible to implement simulators capable of modeling large urban areas such as the metropolitan region of a large city such as São Paulo and New York.

Examples of Smart Cities scenarios already implemented in mesoscopic traffic simulators are:

Electric Vehicles: Many traffics simulators were used to simulate electric vehicles and the required changes in the city infrastructure to support the growth in the energy demand. In these simulations, the researchers modified traffic models to simulate the energy consumption of the vehicles and implemented models to simulate the city infrastructure such as recharge points and renewable energy sources (Allan and Farid, 2015; Geske et al., 2010).

Emission of Pollutants: Some traffic simulators include a pollutant emission model, calculated using the type of the vehicles, the traffic, and the traveled distance in each simulated travel (Xia and Shao, 2005; Hülsmann et al., 2014; Krajzewicz et al., 2012; Zhou, Tanvir, et al., 2015). Usually, a mathematical formula is used to calculate the amount of CO_2 and other pollutants emitted by each travel. Different researchers can be performed using this model. For example, researchers can measure the impacts in the environment of the increase in the use of the public transportation and bicycles.

Vehicular Networks: There are many simulators used to the simulation of vehicular networks, simulating Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure - (V2I) communications. For example, the RedSwarm project (Stolfi and Alba, 2014) has the objective of developing new routing algorithms using real-time data to reduce the vehicles travel time. In this project, many devices deployed in different regions of the city measure the traffic conditions and communicate with the vehicles indicating their best route. Besides, V2V communications are used for the propagation of messages about problems in the roads. To test the algorithms developed in the project, the researchers used the traffic simulator SUMO (Behrisch et al., 2011).

2.3 Actor Model and Erlang

The Actor Model (Camargo Francesquini, 2014) is a robust model for the development of highly concurrent and distributed applications. In this model, each actor is an independent processing unit that has its memory area. Actors can communicate only through asynchronous messages. Each actor has a message box in which all the messages that the actor receives are stored until the actor processes it. After receiving a message, an actor can change his internal state, reply it, or create new actors.

This model minimizes two significant problems of concurrent systems: **Race Condition**, because the actors do not share state or resources, so there is no need for synchroniz-

ing mechanisms such as traffic lights or monitors. **Busy Waits**, because all communication between the actors is through asynchronous messages. Although the actor model is an old idea (defined in 1985) (Agha, 1985), this model has been gaining popularity in recent years due to multi-core architectures and ease of development of distributed applications.

As well as the implementation of competing applications, the Erlang language facilitates the development of distributed applications. It is possible because in the Actor Model there are no differences if the actors are running on the same or different machines. The only requirement for distributing the application is that languages based on the actor model should allow the exchange of messages from actors who are running on different machines transparently. Currently, several languages are based on the model of actors like Erlang and Scala (Tasharofi et al., 2013), and many others have an implementation for the actor model such as Ruby [13], and Java [14].

2.3.1 Erlang

Erlang is a functional programming language based on the actor model. It was developed meanly to facilitate the implementation of parallel, distributed, large-scale software. The language was created by Ericsson[15] in the 80's to the development of telecommunication applications. Currently, Erlang is used in several application domains such as Internet chat [16], databases (J. C. Anderson et al., 2010), and simulators (T. Song et al., 2011; Toscano et al., 2012).

The main characteristics of Erlang, inherited from the Model of Actors, are quite adequate for the development of large-scale simulators:

Parallelism: The Erlang virtual machine allows the creation of a large number of application threads. In the Erlang programming model, each actor created in the application is mapped to an application thread that can perform actions independently of the other actors. Erlang also creates a system thread for each CPU of the computer running the application and balances the actors between those threads to try to maximize CPU utilization.

Distribution: In the Actor Model, there is no difference whether the actors are running on the same machine or distributed machines making distributed application development transparent to programmers. The Erlang Virtual Machine provides the entire implementation of the exchange of messages among different machines. The only requirement for an Erlang programmer is to create a text file that has the address of all machines on which Erlang actors will run.

Fault Tolerance: As previously mentioned, each actor Erlang is independent of the other actors in the application. So an error occurred in one actor does not propagate to the rest of the application.

[13] Celluloid - celluloid.io
[14] Reactors.io - reactors.io
[15] Ericsson - www.ericsson.com
[16] WhatsApp - goo.gl/If6k3d

Communication Protocol: Erlang actors, as previously mentioned, can only communicate through message exchange, which minimizes the need for mutual exclusion algorithms and tools. The entire communication mechanism, local or distributed, is implemented by the virtual machine, which makes the use of these functionalities transparent to the programmer.

There are some of the disadvantages of using Erlang in the simulator implementation. For example, we must implement the synchronization of the actors, because in Erlang each actor is independent of the other and it is impossible to know the order of execution of threads that execute in parallel. Another problem is the lack of tools for the development of Erlang applications such as Integrated Development Environment (IDE) and test tools.

2.3.2 Sim-Diasca

In the InterSCSimulator development, we used Sim-Diasca (Simulation of Discrete Systems of All Scales) (T. Song et al., 2011), a general purpose simulator that aims to allow the simulation of large scenarios implemented by EDF[17]. This simulator is implemented in Erlang, supporting the execution of distributed and parallel simulations.

Sim-Diasca handles all the essential requirements for the execution of discrete-event simulations such as simulation time management, random number generation, events ordering and reproducibility of the results. Besides, it also manages the distribution and parallelism of the simulation actors making loading balancing, creating actors in distributed machines, and handling faults in a distributed environment.

2.4 Discussion

This chapter presented the most important concepts used in the development of this work. First, we introduced a literature review about Smart Cities, what was necessary to understand the domains and dimensions of Smart Cities and what is possible and useful to simulate. Then, we presented a review of traffic simulations and the types of simulations that we found in the literature, it was important because the city traffic has an important role in the entire city working. Finally, we presented the actor model and the Erlang language, which were used in the implementation of the simulator, mainly because this language can support the development of large-scale parallel and distributed systems.

Our literature review showed that it is possible to simulate various Smart City scenarios, and these simulations can facilitate the tests and experiments of Smart City solutions. We also discussed why Erlang is a reliable choice for the implementation of the simulator and why mesoscopic simulation is a good alternative to met our objectives. In the next chapter, we will discuss the related work, mainly showing simulators that already provides one or more Smart Cities scenarios.

[17]EDF — www.edf.fr/content/sim-diasca

Chapter 3

Related Work

In our literature review and web searches, we found simulators of different city domains such as traffic, smart grids, and soil occupation. Some of them are extensible to simulate complex cities scenarios such as the effect of electric vehicles in smart grids, the use of public transportation in the urban mobility, and the influence of shared cars in the city traffic. This chapter presents simulators that implement at least one smart city domain. We defined the requirements for the development of InterSCSimulator based on the simulators described in this section.

3.1 Traffic Simulators

As the main use of the current version of InterSCSimulator is for traffic simulation, we will first present the traffic simulators found in the literature. There are two types of traffic simulators: the microscopic, which aims to simulate the individual behavior of drivers and the macroscopic, which can use mesoscopic or macroscopic models and intent to reproduce the entire mobility system of a large area. On the one hand, a microscopic simulator considers just a small number of city blocks and intends to evaluate the impact of an intervention in the city infrastructure such as a new lane in a street or a new group of semaphores in a neighborhood. On the other hand, a macroscopic simulator examines a broader area and aims to study impacts of extensive changes in the city infrastructure such as a new subway line, a new bus corridor, or the effect on the mobility of the city with a new financial district in the city. We will detail four simulators that are more related to our work and then present a list of other projects.

3.1.1 MATSim

MATSim is an open source, agent-based, mesoscopic, traffic simulator (Horni et al., 2016) developed by the Transport Systems Planning and Transport Telematics group from the Technische Universität Berlin (TU-Berlin). In this simulator, each person is modeled as a software agent with individual settings, and the sum of all agents actions represent the

demographic information of a region (Balmer et al., 2008). Each agent has a plan, which contains all the daily activities from the simulated person such as work, study, and leisure. Besides the agents' schedule, MATSim also requires a graph that represents the city road network.

After the creation of the scenario, the simulator executes all the displacements of the agents in the scheduled time using the city road network. The core of the simulator use only pedestrian and car models. However, several works are extending MATSim to the use of buses (Fourie, 2014), autonomous taxis (Bischoff and Maciejewski, 2016), and shared vehicles (Balać et al., 2018).

MATSim uses a queue-based algorithm to simulate the movement and calculate the speed of the vehicles during the simulation. In this model, all the streets are queues in which the cars have to wait a determined time to go to the next street. The flow and storage of the links are limited. If both are free, the vehicles travel in the free-flow speed. Otherwise, their speed is calculated using a density formula. The simulator generates a text file with all the events occurred during a simulation to save the results. This file is used for statistical analyses or visualization of the simulation.

A significant advantage of MATSim is the offer of several tools to facilitates the creation of simulation scenarios such as a parser of the OpenStreetMap format, a coordinate system converter, a map editor, and a visualization tool.

Regarding scalability, MATSim is capable of simulating large areas such as cities and metropolitan regions. For example, Balmer et al. (2008) present a simulation with more than 200 thousand simultaneous actors in the region of Zurich and Kickhofer et al. (2016) present the development of the scenario of Santiago, Chile with a population of almost 50 thousand agents. However, due to its not parallel and distributed architecture and the usage of the Java language, MATSim is not capable of simulating an entire population of a large city such as São Paulo or Tokyo.

3.1.2 SUMO

SUMO is a microscopic, open source traffic simulator developed by German Aerospace Center (Behrisch et al., 2011). This simulator uses a Car-Following-Model and a Lane-Changing model. The first simulates the speed of a car based on the vehicles that are in front of it, and the former calculates the probability of a car change its lane during a trip. As MATSim, a digraph represents the city road network with the edges defining the stretches of the roads and the nodes creating the intersections.

SUMO provides many auxiliary tools to facilitate the development of traffic scenarios. For example, a network converter suited for reading data from the OpenStreetMap and from other simulators such as MATSim and VISSIM, a demand modeler capable of reading origin-destination data and create the trips to simulate, and a dynamic router generator, responsible for creating routes in the city graph for each simulated trip.

Many academic and commercial projects use SUMO as an auxiliary tool. For example, VEINS (Riebl et al., 2015) combines SUMO and OMNET++, an open source network

simulator, to simulate VANETs in different projects (Buse et al., 2018; Aslam et al., 2018). Other examples of the use of SUMO are to simulate smart traffic lights algorithms (Azevedo et al., 2016), generate mobility traces (Codeca et al., 2017; Uppoor et al., 2014), evaluate the required infrastructure and the use of electric vehicles (Sagaama et al., 2018), and analyze the impact of autonomous vehicles (Tettamanti et al., 2018; Garzón and Spalanzani, 2018).

Due to its model, SUMO is not very scalable, mainly because microscopic models are very CPU intensive and not easy to parallelize and distribute. The largest simulations we found using SUMO are the Luxembourg simulation with 138,361 vehicles during an entire day (Codeca et al., 2017) and the Cologne scenario with approximately 200 thousand vehicles during the peak hour in the city, from 6 am to 8 am (Uppoor et al., 2014).

3.1.3 GPU Mesoscopic Traffic Simulation

X. Song et al. (2017) present a mesoscopic traffic simulator framework using GPUs (Graphical Processing Units). This work aims to use the excellent processing power of GPUs for the fast execution of large-scale, traffic scenarios. The framework employs CPUs to create the agents' trips and GPUs to simulate the trips in the city graph. The simulator uses an asynchronous simulation step strategy which allows the execution of parallel algorithms.

As MATSim, this simulator uses a queue-based algorithm with a speed-density relationship on the links. We used this model in the InterSCSimulator traffic simulation, and we describe it better in Section 4.3. This simulator was tested using a real scenario of Singapore with a traffic network with 3179 nodes and 9419 links and simulating 100 thousand agents in the peak-hour of the city (7 am to 8 am).

The simulator experiments showed a two-times improvement in the execution time of the simulators using the CUDA library compared to a reference implementation written in C++. However, the authors describe two possible problems: the communication of the CPU and GPU is a bottleneck to the simulator, and the limited memory available in the GPUs that limits the size of the city network and the number of simultaneous agents.

3.1.4 PTV - Planung Transport A Verkehr AG

PTV[1] is a German company that develops many products for traffic planning and management. Among these products, there are two traffic simulators, Vissim and Visum, the former microscopic and the latter macroscopic. Vissim uses two main microscopic models, a Psycho-Physical Car Following model to define the speed and behavior of the drivers and a Lane Selection model for drivers to change the lane during their trip. This simulator also allows the simulation of different vehicles such as cars, buses, trucks, and taxis.

[1]PTV Group – http://www.ptvgroup.com/en/

Visum uses a macroscopic model to simulate a broader area such as an entire city or a metropolitan region. It does not simulate individual vehicles but flows moving in the city infrastructure. The advantages of these simulators are the straightforward definition of simulation scenarios, the company support to their customers, and a great variety of tools to make analyses of the simulation results. However, these simulators are not open source, not extensible, and have a high cost.

Vissim and Visum are used in many cities around the world, both by city managers and researchers for different purposes. For example, traffic modelers use Vissim to evaluate road intersections (Kumar et al., 2017; Du and Sun, 2018) and to analyze different algorithms to traffic signal coordination (Bai et al., 2018; Yan et al., 2018). Visum was used to evaluate vehicular emissions in a large metropolitan area (Sider et al., 2014) and to understand cities urban mobility (Montero Mercadé et al., 2018) .

3.1.5 Other Simulators

DEUS (Discrete-Event Universal Simulator) is a discrete-event, general purpose simulator that was used to simulate Vehicular Ad-Hoc Networks (VANET) (Picone et al., 2012). DEUS is open-source and developed in Java. Its programming model is simple, containing two main classes: Node and Event. The Node class represents the possible agents of the simulation scenario and the class Event the possible events that occur in the simulation, always related to one or more node. The traffic scenario uses a mesoscopic traffic model which calculates the speed of the cars accordingly with the number of vehicles in each street. However, due to its architecture, which is not parallel and not distributed, DEUS cannot simulate an entire metropolitan region. We performed tests with this simulator and it could simulate at most 10 thousand vehicles in the city of Parma, Italy.

Siafu is an open-source, agent-based urban simulator developed in Java (Nazário et al., 2014). This simulator aims to simulate different scenarios in a city such as traffic and buildings. This simulator has a rich graphical interface that allows real-time visualizations and modification of the simulated scenario. Indeed, the simulator provides a tool to export all the results of the simulation to a CSV file. The data generated in the simulator is prepared to be analyzed by machine learning tools. In this simulator, the creation of the simulation scenario is manual using a Graphical User Interface (GUI). Therefore, this simulator is not suitable for large scenarios simulations.

Mezzo is a mesoscopic simulation model suitable for the integration of meso-micro models (Burghout et al., 2006). The output of the model uses a format that facilitates the development of microscopic models based on the data of Mezzo. Besides the model, Burghout et al. (Burghout et al., 2006) present simulations using the real road network of Stockholm, Sweden, and travels based on an origin-destination survey. However, they do not discuss the scalability of the model, and we can not find details about the model implementation.

Fernández-Isabel and Fuentes-Fernández (2015) propose the use of Model-Driven Engineering (MDE) for the development of traffic simulations. Following their proposal,

the development of a traffic simulation has two phases. First, traffic engineers model the traffic components and define the input data of the simulation. Then, software developers implement the required tools to the definition and execution of the simulation. The objective of using MDE is to avoid misunderstanding among the multidisciplinary team necessary to create a traffic simulator. The paper also presents a study case of a simulation using a synthetic road network and travels.

DTALite (Zhou and Taylor, 2014) is a mesoscopic traffic simulator that besides the density formula used to calculate the vehicles speed, uses a queue model to control the vehicle flow in the links of the city graph. This model verifies the flow of vehicles that are trying to enter in an edge in a time instant and limits this flow to a maximum input flow of each link of the graph. The simulator developers tested it with the road network of the city of Raleigh in the United States with 2,389 vertex and 20,259 links. An experiment presents a simulation with approximately 1 million travels between 6 am, and 10 am.

The SimMobilityMT (Lu et al., 2015) is a traffic simulator developed by the Intelligent Transportation Systems Lab of the Massachusetts Institute of Technology with three complementary models. The Pre-Day model defines the synthetic population that will be simulated and what are their activities during the simulation period. The Within-Day model simulates the travels of all agents created by the previous models. The traffic conditions can modify the routes of the agents during the simulation. The Supply model updates the data of the road network of the city during the simulation. The simulator was tested in a large scenario using the traffic network of Singapore with 340 thousand travels during the peak hour of the city, from 08:30 am until 09:30 am.

None of the simulators cited above scale to an entire metropolitan area using a map with thousands of streets and millions of vehicles moving simultaneously in the city. All the open source simulators use Java or C++ programming, languages which the development of parallel and distributed applications are not transparent. Just one simulator (X. Song et al., 2017) uses GPUs which can leverage the use of parallel algorithms. Therefore, the use of a language that simplifies the development of parallel and distributed applications can facilitate the implementation of a scalable Smart City simulator. Table 3.1 presents a comparison of all the simulators presented in this section and the InterSCSimulator.

	Language	Model	Large-Scale	Usability	Parallel	Open-Source
DEUS	Java	Meso	No	Yes	No	Yes
SUMO	C++	Micro	No	No	Yes	Yes
Siafu	Java	Meso	No	Yes	No	Yes
MATSim	Java	Meso	Yes	Yes	Yes	Yes
Mezzo	?	Meso	No	No	No	?
GPU Simulation	C++/CUDA	Meso	Yes	No	Yes	Yes
MDE Model	Java	Meso	Yes	No	No	?
DTALite	C++	Meso	Yes	No	Yes	Yes
Vissim		Micro	No	Yes	No	No
Visum		Macro	Yes	Yes	No	No
SimMobilityMT	C++	Meso	Yes	No	Yes	Yes
InterSCSimulator	Erlang	Meso	Yes	Yes	Yes	Yes

Table 3.1: *Comparison among the analyzed simulators and the InterSCSimulator*

3.2 Other Domain Simulators

The work presented in this thesis focus on the simulation of traffic scenarios. However, the InterSCSimulator aims to simulate comprehensive Smart Cities scenarios. For example, the simulator already simulates buildings and sensors. Therefore, we also analyzed simulators related to other Smart Cities scenarios such as Smart Grids, Sensors Networks, and soil usage.

Ursachi and Bordeassa (Ursachi and Bordeasu, 2014) present a Smart Grid simulator. A Smart Grid is an intelligent electricity system, and the simulator has the objective of finding an optimized electricity distribution model. Their algorithm combines the cost of producing energy from different sources such as solar, wind, and hydroelectric, the distance from the customers to the energy producers, and the climatic conditions. The simulator always tries to use the electricity source with the minor cost and environmental impact.

Karnouskos and de Holanda (Karnouskos and De Holanda, 2009) present an agent-based Smart Grid simulator. This simulator aims to create the dynamic environment of a Smart City. The simulator provides many entities which consume or produce electricity such as power plants, electric vehicles, houses with different energy consumption levels, and public lights. The simulation input is the description of the simulated entities and the simulation time, the output is a file with all the energy generation and consumption events which allow the creation of charts and animations of the simulation results. The scalability evaluation shows that the simulator supports more than 5000 objects and 70 million events.

UrbanSim (Waddell et al., 2003) is a simulator with the objective of modeling the

impact of changes in the use of the soil of a city in many areas such as mobility and real estate. The system has a set of models and agents that represent the main objects an urban environment such as houses, buildings, and places with great people concentration. Also, it is possible to simulate events that change the average price of buildings in a region such as the constructions of a new park or a new shopping center. The impacts in traffic are caused by changing the number of people moving in the city and the origin-destination of a set of actors. However, UrbamSim does not simulate the agents individually, it calculates through models the modifications and generates tables with the data about the city.

A sensor network is an essential component of Smart Cities. There are many simulators capable of simulating a city sensor network. Among these simulators, there are ones that reproduce the low level of a computer network such as SENSE (Chen et al., 2005) and ATEMU (Polley et al., 2004), what is not the focus of this thesis. Otherwise, there is the simulators that model applications of sensor networks or the data generated by the sensors such as the CupCarbon (Mehdi et al., 2014) and SenSE (Zyrianoff et al., 2017), this type of the simulator has the same objective of this thesis because many Smart Cities scenarios will depend on a city sensor network. For example, a smart parking scenario can have many sensors to indicate if a parking spot is free or occupied, and a city sensing scenarios can collect data from an extensive network to analyze the city temperature, pollution, and noise.

3.3 Discussion

This chapter presented traffic and other Smart City domain simulators. Our literature review showed that there are many projects to simulate the road network of a city and the trips of the citizens during a day in the city. However, most of them have significant limitations such as the scalability, the simulation of only individual cars and not of different mobility modes, and the difficult to adapt the simulator for new traffic models. The review also gathered the main requirements for the implementation of a new smart city simulator that will be presented in Section **??** .

The next chapter presents the implementation of InterSCSimulator, a scalable, open-source Smart City simulator. To develop our simulator, we considered the strength of all the studied simulators such as providing tools to create simulation scenarios, the use of open traffic models, and the integration of different models. We also examined the implemented architecture of the simulators and decided to use the Erlang language, which is appropriate for the development of large-scale, parallel and distributed applications. Finally, whenever possible, we tried to reuse many tools from other simulators such as the visualization tool from MATSim and the network generator from SUMO.

Chapter 4

InterSCSimulator

In this chapter, we will present InterSCSimulator, an open-source, large-scale Smart City simulator. The goal of this simulator is to simulate complex Smart Cities scenarios with millions of agents. The current version of the software allows the simulation of different traffic scenarios. Section 4.1 describes the functional and non-functional requirements handled in the simulator implementation. Section 4.2 presents the architecture of the simulator. Section 4.3 describes the implementation of the traffic model used in the simulator. Section 4.4 presents the inputs, outputs, and components of the simulator. Section 4.5 lists extensions to the simulator to allow the simulation of other Smart City applications domains, some of them already implemented. Finally, Section 4.6 describes the evolution and challenges faced during the simulator implementation.

4.1 InterSCSimulator Requirements

In this section, we will describe all the requirements handled in the InterSCSimulator implementation.

4.1.1 Functional Requirements

To identify the functional requirements of InterSCSimulator, we studied the simulators presented in Chapter 3 and the Smart City domains presented in Chapter 2. In this work, we decided to focus on the mobility domain because it has important challenges and is related to many other city domains such as health and garbage collection.

The main functional requirements are the representation of the road network of the city, the definition of simulated travels and a model to allow realistic movement in the city. The list of the requirements is in the following:

- **Road Network Representation:** In a Smart City simulator, it is necessary to represent a real city network to allow the simulation of the movement of vehicles and people in the city. This network must be described in a data structure easy

to manipulate and execute algorithms such as to find the best path and calculate vehicles speed. The approach used by most simulators is creating a digraph of the city using a map from a map service such as Google Maps[1] or Open Street Maps[2].

- **Travels Definition:** The simulator must receive a list of travels that it must simulate with parameters such as origin, destination, and travel time. There are many possibilities to create the travels such as an Origin-Destination (OD) survey (Khan and M. Anderson, 2016), real mobility traces from smartphones (Jamil et al., 2014), and random travels. In this work, we created tools to convert the OD created by the municipality of São Paulo to the format of our simulator.
- **Vehicle Simulation:** It is necessary to define a mathematical model to calculate the flow and speed of the vehicles. There are many models available in the literature, from a very complex micro-model to a very simple free-flow model. This work uses a mesoscopic model described by Song. et al. (X. Song et al., 2017) and will be detailed in Section 4.3.
- **Pedestrian Simulation:** Besides the vehicles, it is necessary to model pedestrian movements to allow movements to subway stations and bus stops and also trips made entirely on foot.
- **Bus System:** It is necessary to model bus stops and bus trips, allowing passengers to enter and leave the vehicles. The buses must move on the city using the same traffic model used by the car simulation.
- **Subway:** To simulate big cities, it is also necessary to model the subway network of the town, allowing commuters to arrive at the stations by all the other travel modes.
- **Simulation Output:** The simulator must generate outputs to allow data analyses and visualization of the simulation. Typically, the simulators create a trace file with all the events occurred during the simulation. Our simulator also saves a trace file in XML or CVS format.
- **Data Analyses:** The simulator output must facilitate the data analyzes using statistical tools such as R and Python. Hence, all the output must have all the crucial events of the simulation.
- **Simulation Visualization:** The simulator also must allow an animated visualization of the simulation. Many simulators already have an integrated tool to visualize the simulation during execution time, and other have a tool that analyzes the simulation output and generates the visualization.

The requirements show that the simulator must receive as inputs the road network of a city and the list of travels that will be simulated. Other possible data is information about public transportation and pedestrians. With this data, the simulator must calculate the agents travel using different mobility models. Finally, the simulator must save the events to allow the data visualization and analyses.

[1]Google Maps - googlemaps.com
[2]Open Street Maps - www.openstreetmap.org

4.1.2 Non-Functional Requirements

Besides the functional requirements, a Smart City simulator must also (1) allow the easy definition of the simulation scenarios, (2) allow the execution of scenarios with millions of simultaneous actors, and (3) allow the easy extension of the simulator. Therefore, the non-functional requirements handled by the simulator are:

- **Scalability:** To simulate an entire city, it is necessary to manage millions of simultaneous actors such as cars, people, sensors, and building. Therefore, the simulator must scale from hundreds to millions of actors. To achieve this objective, it is required the use of efficient algorithms and data structures and the development of a massively parallel and distributed simulator.
- **Usability:** The definition of a simulation scenario must be simple, allowing people that are not computer specialist use the simulator without knowing its internal implementation. Most of the analyzed simulators use XML files to the definition of the scenarios, many of them provide tools to convert data from different sources to the format of the simulator such as maps, origin-destination matrix, and bus lines.
- **Extensibility:** It is highly unlikely that a simulator provides all the models and abstractions required to create all the Smart City simulations. Therefore, the extension of the simulator must be straightforward, offering simple mechanisms to allow the implementation of new actors and the extension of the existing ones. It is an excellent vantage of open source simulators that has a well-documented architecture and code.

4.2 Simulator Architecture

InterSCSimulator has a three-layer architecture. Figure 4.1 presents the layers: **Sim-Diasca** is a generic, discrete-event simulator that offers the necessary tools to develop simulator models, **City Model** which implements the behavior of the city actors and is the central part of this work and **Simulation Scenarios** which are the possible simulation scenarios developed with the city model.

Sim-Diasca main requirements are the simulation time management, the random number generation (allowing reproducibility), the synchronization of the actors to guarantee the simulation consistency and a basic actor model to facilitate the development of the simulation model. Besides, using the Erlang language, Sim-Diasca facilitates the communication among the actors, the parallelism, and distribution of the simulator, and fault tolerance.

The City Model is composed of the actors that are simulated in the Smart City scenarios. This layer is the central development of this research. The current version of the simulator provides the following actors: **Person** that represents a person traveling in the city, **Car**, when a person travels in a car, **Bus**, that represents the buses moving in the city and allow the boarding of passengers, **Subway** that simulates the subway network of the city, **Sensor** to reproduce Smart City sensors. Besides the actors, there are many other

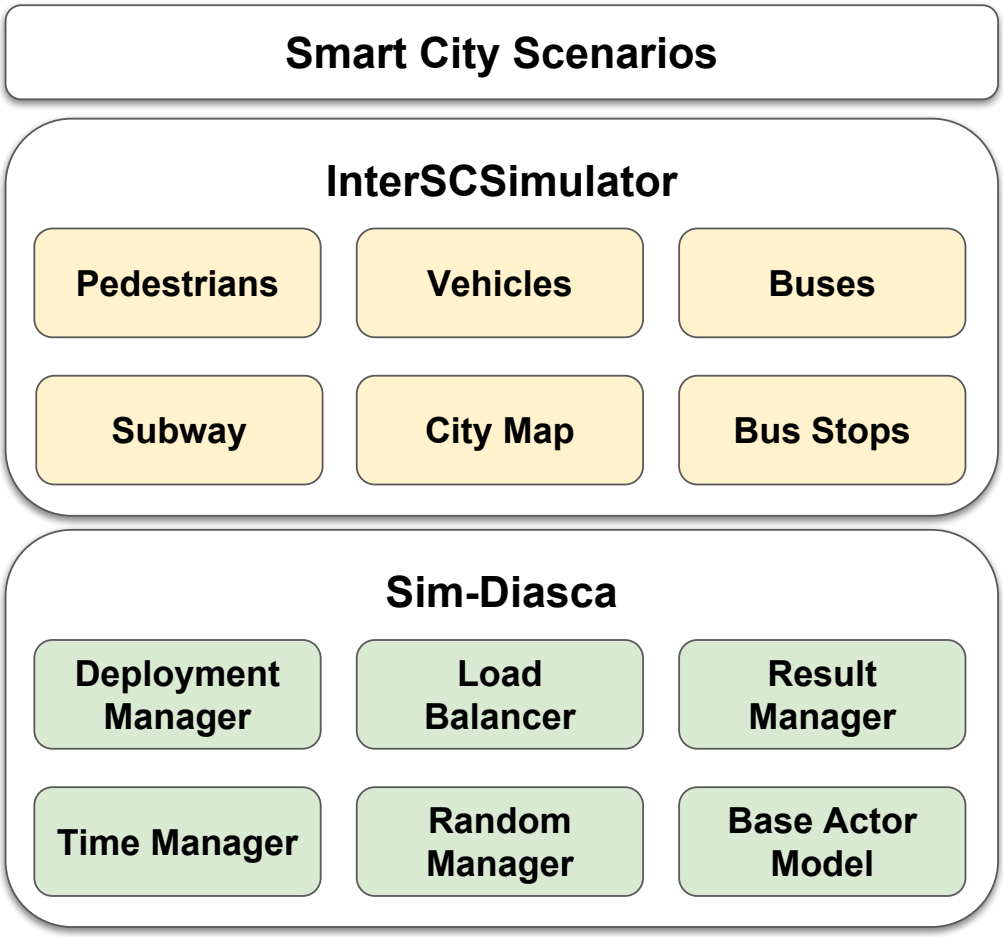

Figure 4.1: *InterSCSimulator Architecture*

structures in the model, such as the city graph representation, an API to retrieve real-time simulation data, and a structure to save the simulation event log.

4.3 Mobility Models

InterSCSimulator has many models to simulate the mobility of a city population, including a traffic model, which calculates the speed of the vehicles in the streets, a simple pedestrian model, and a model to simulates public transportation travels by subway or buses. A person can also make a trip using different travel modes, for example, the person starts walking to a subway station, then makes subway travel, and finally, get a bus to reach his destination. The next subsections describe the main models used by the simulator.

4.3.1 Vehicles Simulation

Car travels are composed of a sequence of links that the car will pass. To calculate the speed of the car in each link, the InterSCSimulator uses a mesoscopic traffic model.

This type of model simulates each car individually. However, the speed of these vehicles is calculated using a density function relating to the capacity of the street and the number of vehicles on the street (X. Song et al., 2017). The model implemented uses the following density function:

$$v = v0 * ((1 - (\frac{k}{k_{jam}})^{\beta}))^{\alpha}$$

Where:

- $v0$: the maximum speed of the street, used when the street is not congested.
- k: the current street density.
- kjam: the density when the street is congested.
- α e β: Configurable parameters that are defined by the calibration of the model. We used the values used in the original paper α=1.0 and β=0.05.

This function is always used when a vehicle enters in an edge of the graph. To calculates its speed, it is verified the current density in the edge and using the function, the speed is calculated. With the speed and the length of the edge, it is possible to calculate the time that the will need to go through the edge. The time is calculated using the following function:

$$time = \frac{edge\_length}{vehicle\_speed}$$

In the function, **time** is the seconds that the car takes to go through the edge, **vehicle_speed** is the value calculated by the traffic model and represents the speed (m/s) of the car in the street, and **edge_length** is the street length (m) that is available at OpenStreetMaps.

4.3.2 Pedestrian Simulation

Pedestrian simulation resembles the vehicle simulation. However, the speed calculus is straightforward. We used a normal distribution with mean 1.2 m/s, and a random speed calculated to the person. The function used to calculate the time that a person will take to go through an edge is the same used by the vehicles.

4.3.3 Subway Simulation

To simulate the subway system, InterSCSimulator has the subway graph of the simulated city, and it calculates the travel time of the subway travel by the number of stations that separate the origin and destination stations of the trip. Commonly, city documents with the subway system describe the travel duration between two following stations. Hence, we use a fixed time to calculate the travel time of the person in the subway system.

When an agent arrives at the origin subway stations, it sends a message to an Erlang process which simulates the subway system with its destination station. The Erlang process calculates the time that the agent will spend in the subway system, and then the person waits this time. After this interval, the agent is moved to the destination station and can finish its travel.

4.3.4 Bus Trip Simulation

The buses are simulated using the same traffic model used by the cars. The difference is that the buses pass in a sequence of links that represent bus stops. The people in the bus stop can board the bus if the line of the bus is the same that they are waiting.

The person can go walking or by car from its origin until a nearby bus stop. In the bus stop, the person waits for a bus of his desired line. When the bus arrives, the person boards the bus and moves using the same path and speed of the bus. When the bus comes at the destination bus stop, the person outboards the vehicle and the person finishes his travel.

The buses can have a maximum of 75 passengers, which is the limit of the city of São Paulo[3].

4.3.5 Multimodal Travels

The simulator allows a sequence of travels using the same or different travel modes. To this, each person can have a series of trips and the destination of each travel is the origin of the next one. Each travel does not have any relation to the previous one. However, the final attributes of the journey such as total time traveled distance, and travel cost is the sum of all travels.

4.4 Simulator Components

InterSCSimulator has three main components: **Scenario Definition** which read the input files and creates the simulation scenarios. **Simulator Engine** responsible for executing the algorithms and models of the simulation. **Events Manager** that receives all the events occurred in the simulation and saves in the trace file or send to the real-time API.

At the end of the simulation, there are other two components based on the result simulation: **Animated Visualization** which allows the graphic visualization of the events in the city graph and **Output Analyses** that using the output data perform statistical analyses using scripts R. Figure 4.2 presents the simulator components, the interactions among them, and the simulator input and output files.

[3]http://goo.gl/B7NJbj

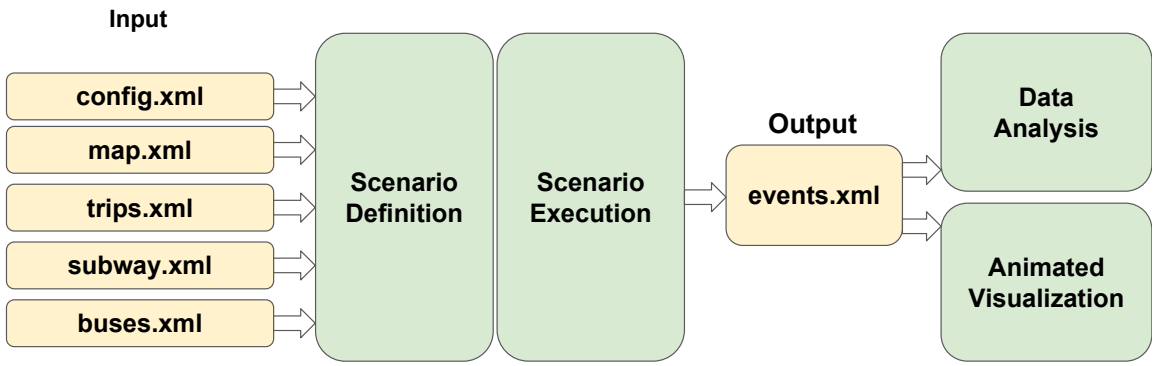

Figure 4.2: *InterSCSimulator Components*

The following sections will present each of the inputs, outputs, and components of the simulator.

4.4.1 Inputs

InterSCSimulator has three required XML files as input and other two optional files. The required files are:

- **config.xml** contains the path to the input and output files and the total simulation time.
- **map.xml** has the digraph representing the road network of the simulated city. We used the map on the OpenStreetMap format in which the junctions are the vertices, and the streets are the links of the graph.
- **trips.xml** defines the trips that must be simulated, each trip must determine its options such as origin, destination, start time, and travel mode.

The optional files are:

- **subway.xml** describes the city subway system as a graph which the stations are the vertices and the connection among the stations the edges.
- **buses.xml** contains the bus lines of the city, including its name, interval, and bus stops sequences.

The map file is transformed in a digraph using the Digraph API of the Erlang language[4]. Listing 1 shows an example of a map used in the simulator.

```
1   <network>
2    <nodes>
3     <node id ="1" x="-46.65805" y="-23.58162" />
4     <node id ="2" x="-46.65828" y="-23.58342" />
5     <node id ="3" x="-46.65228" y="-23.59341" />
6     <node id ="4" x="-46.63128" y="-23.51241" />
```

[4]Erlang Digraph API - erlang.org/doc/man/digraph.html

```
7      <node id ="5" x="-46.59228" y="-23.54341" />
8     </ nodes>
9     <links>
10     <link id="35985" from="1" to="2" length="100" freespeed="40"
           capacity="200" />
11     <link id="35986" from="2" to="3" length="200" freespeed="40"
           capacity="200" />
12     <link id="35987" from="3" to="1" length="80" freespeed="50"
           capacity="200" />
13     <link id="35988" from="4" to="1" length="180" freespeed="50"
           capacity="200" />
14     <link id="35989" from="1" to="5" length="30" freespeed="30"
           capacity="200" />
15     <link id="35990" from="6" to="2" length="40" freespeed="50"
           capacity="200" />
16     <link id="35991" from="7" to="2" length="50" freespeed="50"
           capacity="200" />
17     <link id="35992" from="1" to="7" length="60" freespeed="40"
           capacity="200" />
18     <link id="35988" from="3" to="6" length="180" freespeed="50"
           capacity="200" />
19    </ links>
20   </network>
```

Listing 1: *File describing the city road network*

The map file has two section, the first describes the vertex of the graph, that are intersections among the cities roads. The vertex has three attributes, an Id, the latitude and the longitude. The second section contains all the edges of the graph, which represents a street stretch. The properties of the edges are its length, maximum speed, and capacity.

The trip file contains all the travels that must be simulated. Each travel has an origin and destination, the start time, and the travel model. A best path algorithm is used to define the travel path in the city graph. Listing 2 presents an example of trips file with a set of travels.

```
1    <scsimulator_matrix>
2     <trip origin="24751" dest="60613" start="2880" mode="car" />
3     <trip origin="60613" dest="24791" start="6300" mode="car" />
4     <trip origin="45110" dest="21990" start="1620" mode="car" />
5     <trip origin="21090" dest="45110" start="6540" mode="car" />
6     <trip origin="24751" dest="24650" start="3420" mode="car" />
7     <trip origin="24650" dest="24751" start="5400" mode="car" />
8     <trip origin="24751" dest="24650" start="6660" mode="car" />
9     <trip origin="24650" dest="24751" start="8550" mode="walk" />
10    <trip origin="24751" dest="24751" start="2370" mode="car" />
11    <trip origin="24751" dest="24751" start="4440" mode="car" />
12    <trip origin="24751" dest="60658" start="2880" mode="car" />
13    <trip origin="60658" dest="24791" start="4320" mode="car" />
14    <trip origin="24751" dest="41594" start="6840" mode="walk" />
15    <trip origin="25822" dest="27925" start="7920" mode="car" />
16    <trip origin="27925" dest="66111" start="5760" mode="car" />
17    <trip origin="66111" dest="27925" start="6480" mode="car" />
18    <trip origin="24751" dest="33026" start="4230" mode="car" />
19   </scsimulator_matrix>
```

Listing 2: *File containing the trips that will be simulated*

It is also possible to simulate multimodal travels. For example, a person can walk until a bus stop, take a bus to a subway station, and then go walking from other metro station until its work. Listing 3 presents examples of multimodal travels.

```
1   <multi_trip name="1"  count="45" start="43201" mode="bus">
2     <trip origin="25298" dest="17409" mode="walk"/>
3     <trip origin="17409" dest="11072" line="8020-10-0" mode="bus"/>
4     <trip origin="11072" dest="30469" line="675N-10-0" mode="bus"/>
5     <trip origin="30469" dest="30469" mode="walk"/>
6   </multi_trip>
7   <multi_trip name="3" count="82" start="45001" mode="metro">
8     <trip origin="41972" dest="44116" mode="walk"/>
9     <trip origin="44116" dest="25781" mode="metro"/>
10    <trip origin="25781" dest="30165" mode="walk"/>
11  </multi_trip>
12  <multi_trip name="4" count="37" start="61201" mode="metro">
13    <trip origin="31468" dest="29513" mode="walk"/>
14    <trip origin="29513" dest="28267" mode="metro"/>
15    <trip origin="28267" dest="60642" mode="walk"/>
16  </multi_trip>
```

Listing 3: *Multi-Trip file*

The Config file defines some parameters to the simulator such as the total time of the simulation, the path to the input and output files and what analyses will be executed in the final of the simulation. Listing 4 presents an example of this file.

```
1   <scsimulator_config>
2       <config trip_file="path_to_trip_file" />
3       <config map_file="path_to_map_file" />
4       <config bus_file="path_to_bus_file" />
5       <config subway_file="path_to_subway_file" />
6       <config simulation_time="86400" />
7       <outputs>
8           <chart name="mostUsedRoads" />
9           <chart name="meanVelocityHour" />
10          <chart name="tripTimeByMode" />
11      </outputs>
12  </scsimulator_config>
```

Listing 4: *File with the simulator configurations*

The subway file defines the subway graph of the simulated city which has two sections. The first describes the subway stations and the second the connections among the stations. Listing 5 presents an example of the definition of a subway line in the input file.

```
1   <metro>
2     <stations>
3       <station name="Vila Prudente" idNode="252018921" />
4       <station name="Tamanduatei" idNode="674412889" />
5       <station name="Sacoma" idNode="2533391001" />
```

```
6         <station name="Alto Do Ipiranga" idNode="4443113954" />
7         <station name="Santos-imigrantes" idNode="1484829870" />
8         <station name="Chacara Klabin" idNode="922094745" />
9         <station name="Ana Rosa" idNode="467744160" />
10        <station name="Paraiso" idNode="459347687" />
11        <station name="Brigadeiro" idNode="856727276" />
12        <station name="Trianon-masp" idNode="2021000708" />
13        <station name="Consolacaoo" idNode="1819616337" />
14        <station name="Clinicas" idNode="1952545109" />
15        <station name="Sumare" idNode="466808387" />
16        <station name="Vila Madalena" idNode="1415720983" />
17      </stations>
18
19      <links>
20        <link idOrigin="674412889" idDestination="252018921" />
21        <link idOrigin="2533391001" idDestination="674412889" />
22        <link idOrigin="4443113954" idDestination="2533391001" />
23        <link idOrigin="1484829870" idDestination="4443113954" />
24        <link idOrigin="922094745" idDestination="1484829870" />
25        <link idOrigin="467744160" idDestination="922094745" />
26        <link idOrigin="459347687" idDestination="467744160" />
27        <link idOrigin="856727276" idDestination="459347687" />
28        <link idOrigin="2021000708" idDestination="856727276" />
29        <link idOrigin="1819616337" idDestination="2021000708" />
30        <link idOrigin="1952545109" idDestination="1819616337" />
31        <link idOrigin="466808387" idDestination="1952545109" />
32        <link idOrigin="1415720983" idDestination="466808387" />
33      </links>
34    </metro>
```

Listing 5: *File with the definition of the city subway graph*

The buses file defines the bus lines in the simulated city. Each bus line has the following attributes: **Id** is the name of the line; **interval** is the dispatch time interval of the buses of the line; **start_time** is the bus first dispatch time of the day; **stops** is the list of bus stops whose the bus must stop to load or unload passengers. Listing 6 presents an example of the definition of bus lines in the simulator.

```
1
2     <scsimulator_buses>
3         <bus id="1016-10-0" interval="1800" start_time="18000" stops=
              "507969889,2390204059,507969889,1400446171" />
4         <bus id="1016-10-1" interval="1800" start_time="18000" stops=
              "2396517544,163220296,2390192810,2390192798
5         " />
6         <bus id="1017-10-0" interval="1800" start_time="18000" stops=
              "832264854,1758396031,832264854,2448473366" />
7         <bus id="1017-10-1" interval="1800" start_time="18000" stops=
              "934050061,2109902387,2448473366,832264854" />
8         <bus id="1018-10-0" interval="1800" start_time="18000" stops=
              "2116463109,2390204059,507969889,2390204059" />
9         <bus id="1018-10-1" interval="1800" start_time="18000" stops=
              "2520196336,507969946,410243261,410243260" />
10        <bus id="1024-10-0" interval="1800" start_time="18000" stops=
              "934050061,1430544803,917144410,1430544803,917144410" />
```

```
11        <bus id="1025-10-0" interval="1800" start_time="18000" stops=
             "934050061,1430544803,917144410,1430544803" />
12        <bus id="106A-10-0" interval="1800" start_time="18000" stops=
             "2390339037,1709028082,185795118,929591881" />
13    </scsimulator_buses>
```

Listing 6: *Definition of the city buses*

4.4.2 Scenario Definition

The Scenario Definition component parses the input file and creates the simulation scenario. Additionally, it creates all the management Erlang actors and sets all the parameters of the simulation such as the simulation time and the output path. The most critical actors created in this phase are the **travel manager**, which generates the people actors during the simulation, **subway manager** which represents the subway system of the city, **city manager** which describes the road network of the town, and the **output manager** which saves all the simulation events to the log file.

In this phase are also calculated the best path to all the car and pedestrian travels. It is made before the simulation because it is a costly operation and if it was executed during the simulation could waste a lot of processing time.

4.4.3 Simulation Execution

In the execution of the simulator, the most important actions are the movement of people on foot, by car, or by public transportation. Each person of the simulation make an action (or event) and then schedules its next movement. The movement depends on the transportation modal of the person. In the following, we describe the main events that a person can execute during the simulation.

- **Start Travel:** When the simulation time is equal to the start time of the travel configuration, an agent is created to simulate that travel. Independently of the transportation mode. In this event, the agent goes to the first link of its path.
- **Move:** If an agent is moving on foot or car when it leaves a link and enters in another one, it is generated a move event. In this event, it is calculated the time that the agent will require to pass through the link. Walks and car travels are a succession of move events.
- **Move Bus:** When an agent arrives at a bus station, it waits until a bus of the line that it is waiting arrives. When the bus arrives, the agent enters the bus and generates a move bus event. After, the agent moves with the bus until its bus stop destination.
- **Move Subway:** If an agent will travel by subway when it arrives at the subway station, it informs the SubwayManager their origin and destination stations. The SubwayManager calculates the time in which the agent will spend in the subway system and notifies the agent.

- **Arrival:** When the agent arrives at its final destination, it creates an arrival event that saves the attributes of the travel in the output file such as the total time, distance, and cost. After this event, the agent is removed from the simulation.

The execution of the events is based on the models described in Section 4.3. All the events are saved on the simulator output file in chronological order and have common attributes such as simulation tick, link, type, and id of the agent which executed the event.

4.4.4 Outputs

As output, InterSCSimulator generates an XML or CSV file with all the events occurred during the simulation. The possible events are the ones described in the previous section. Listing 7 presents a stretch of the file with a simulation output.

```
192;arrival;8062_65;102388;126;616
200;arrival;8062_74;102388;126;616
228;arrival;8062_38;102388;126;616
235;arrival;8000_50;106030;186;1542
244;arrival;8062_37;102388;126;616
257;arrival;49_10;48743;157;807
257;arrival;49_17;48743;157;807
258;arrival;8062_20;102388;126;616
259;arrival;2280_15;13694;91;407
269;arrival;3241_72;42753;144;1668
```

Listing 7: *CSV output file*

In Listing 7 the first four elements are common to all events which are the simulation tick, the event, the agent id, and the link where the event occurred. Depending on the event, it could have more elements, such as the arrival event that also saves the total distance and the total time of the travel.

To generate the XML file, we used the MATSim format, allowing the usage of OTFVis[5], a tool to generate an animated visualization of the simulation. Listing 8 presents a stretch of the file with the XML output.

```
<events version="1.0">
  <event time="28" type="departure" person="p1" link="1425"  />
  <event time="28" type="entered link" person="p1" link="15"
      vehicle="car1" />
  <event time="32" type="entered link" person="p1" link="13"
      vehicle="car1" />
  <event time="54" type="entered link" person="p1" link="16"
      vehicle="car1" />
  <event time="62" type="entered link" person="p1" link="36"
      vehicle="car1" />
  <event time="78" type="entered link" person="p1" link="21"
      vehicle="car1" />
```

[5]OTFVis — matsim.org/docs/extensions/otfvis

```
8      <event time="96" type="entered link" person="p1" link="42"
          vehicle="car1" />
9      <event time="110" type="entered link" person="p1" link="57"
          vehicle="car1" />
10     <event time="114" type="entered link" person="p1" link="72"
          vehicle="car1" />
11     <event time="118" type="entered link" person="p1" link="67"
          vehicle="car1" />
12     <event time="125" type="arrival" person="p1" vehicle="car1"
          link="88" trip_time="191" distance="2634" />
13   </events>
```

Listing 8: *XML output file*

4.4.5 Simulation Visualization

Using the output data, it is possible to generate different visualizations of the simulation results. For example, it is possible to create an animated visualization of the simulation events. This visualization shows the city road network and the cars moving in the city. Figure 4.3 presents the visualization of one simulation in the city of São Paulo. Each green point in the figure is a car moving in the city graph.

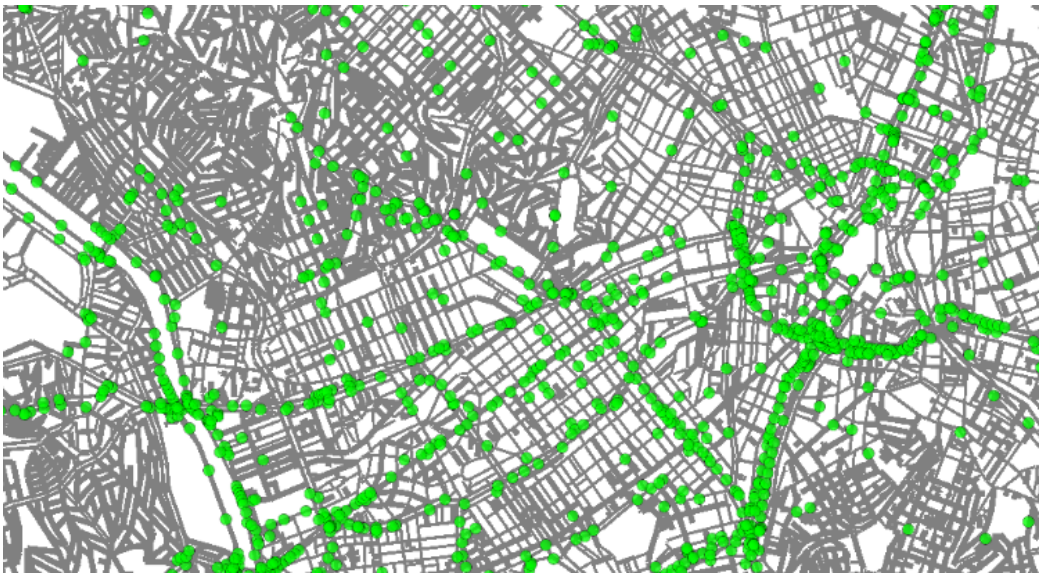

Figure 4.3: *São Paulo Simulation*

To show that InterSCSimulator works in any map generated from the OSM, we also made a simple simulation using the map of New York. Figure 4.4 presents the visualization of this simulation using OTFVis.

Despite that OTFVis is capable of simulating large simulations with more than 50 thousand actors at the same time, it does not have the same scalability as InterSCSimulator. Therefore, it is necessary more research allow the visualization of a very-large simulation in a large graph such as São Paulo road network.

Figure 4.4: *New York Simulation*

Besides the animated visualization, we also created a component that executes scripts written in R language to perform statistical analyses with the output data of the simulator. These scripts generate a series of charts such as the mean speed of the cars during the simulation time, the most used links in the simulation, and the mean time of the travels by the transportation mode. Figure 4.5 shows a chart generated by an R script which shows the 10 most used links during the simulation.

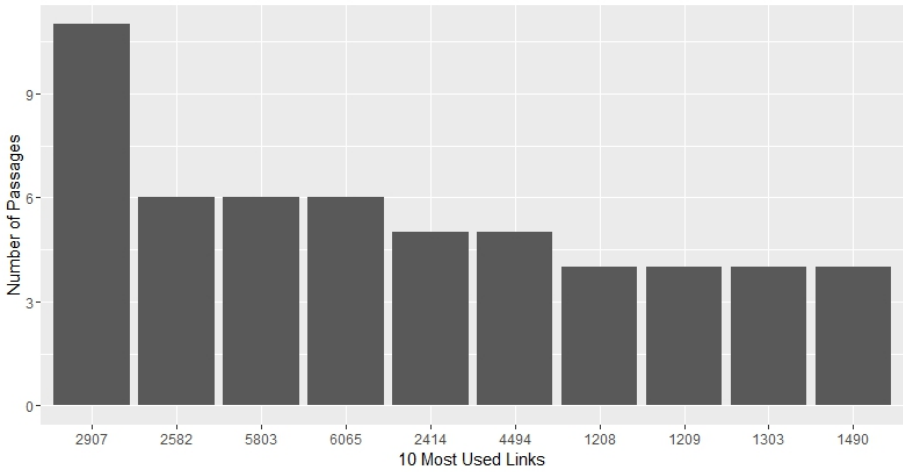

Figure 4.5: *Most used links in the simulation*

Figure 4.6 presents another example of a chart generated using the simulation output data. This chart presents the mean travel time by the different type of travels such as agents going to work, to home, or going to hospitals.

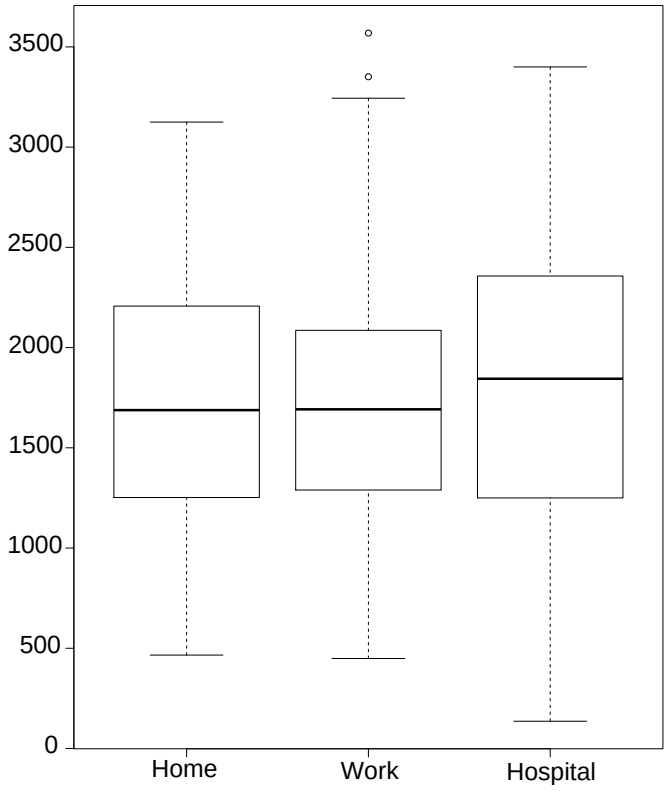

Figure 4.6: *Mean travel time by travel type*

4.5 Other Simulations

Besides the models described in this Chapter, we also implemented other components of Smart Cities applications such as:

- **Parking Spots:** We also developed the simulation of parking spots, which allow the configuration of a car travel that the person must find a free place to park the car. The idea of this simulation was to verify the impact in the city traffic of using a Smart Parking application, helping drivers to find the closest free parking spot.
- **Sensors:** The simulator enables the definition of a set of sensors which generates data in a time interval. These actors allow the simulation of different type of sensors such as air pollution, noise, traffic counters, and humidity. The simulation of sensors enables tests and experiments of Smart City applications and platforms.
- **Locations:** It is possible to simulate locations in the simulator. These locations can generate random travels in the city using a statistical distribution. The idea of this model is to reproduce places in the city that can spawn traffic such as stadiums, shopping centers, and large condominiums.
- **Events:** The simulator allows the simulation of events in the city streets such as an

accident that interferes in the flow of vehicles or streets closure in a weekend for leisure. The idea of simulating events is to evaluate how an event impacts the traffic of the city.

4.6 Simulator Architecture Evolution

Achieve the current scalability of the simulator was a big challenge. We had to develop different algorithms and use many data structures during the simulator implementation. This section presents the most significant modifications to the simulator architecture to improve the execution time and memory usage. The most significant improvements were in the city graph representation and the actor's creation.

This section presents a comparison of the simulation performed for the paper presented in the 2017 Multi-Agent Based Simulation Workshop. The simulations performed for this paper were re-executed with the current version of the simulator to show the improvements in the resource usage and the execution time. The complete scalability experiments with more than 15 million simulated agents are presented in Chapter 6.

4.6.1 City Graph Representation

To manipulate the city graph, we use the Erlang Digraph API[6] which facilitates a lot the creation of graphs and the execution of many algorithms such as the best path and minimal spanning tree.

In the first version of the simulator, we maintained the city graph in a unique Erlang actor. When a car entered a link, it must send a message to the graph actor and waits for the execution of the traffic model to get the time that it took to go through the street. This model worked but had a very high execution time, because this central actor was a bottleneck in the simulator architecture. We did not execute scalability tests in this version because we could simulate at most 50,000 agents in this version of the simulator.

The first solution to this problem was creating an independent actor to each vertex in the graph. In the second version, each actor had the id of the vertex and a dictionary[7] with all the links to the adjacent vertex. This dictionary stored the attributes of the link such as its length, capacity, and the current number of cars in the link. This version increased a lot the use of memory to store the city graph but lowered the execution time of the simulator. With this version, it was possible to execute simulations with more than 4 million actors in a single simulation. Figure 4.7 shows the execution time of this version of the simulator using a São Paulo scenario with 1, 2, 3, and 4 million cars.

Although the second version of the simulator improved a lot the simulator execution time, the architecture of the simulator with an Erlang actor to each vertex of the graph was using much memory and when the traffic was very congested, a number of vertex, mainly

[6]Digraph API – http://erlang.org/doc/man/digraph.html
[7]Erlang Dictionary – http://erlang.org/doc/man/dict.html

the ones that represented big avenues in the city, the execution time of the simulation increased a lot. To solve this problem, we used the Erlang ETS (Erlang Term Storage) Tables (Aronis and Sagonas, 2017).

The ETS Table is an in-memory store object available in the Erlang Virtual Machine. An ETS Table is capable of storing large amounts of objects with a constant time data access. All the Erlang actors can access and update an ETS table that is running in the Erlang VM. To avoid concurrency problems, the ETS API provides a series of atomic operations such as increment, decrements, exclusion of lists, and value update.

The third, and current, version of the simulator uses an ETS table to store the city graph. In this version, the digraph API is used only to calculate the travel paths. The models use the ETS that has all the vertex and links of the graph. Using the ETS there is no bottleneck in the simulator, the only problem that can occur is if two or more vehicles try to enter in the same link at the same moment. If this occurs, the cars will have to wait for the update in the ETS table from the other cars. The use of ETS table improved the execution time of the simulator almost three times. Figure 4.7 shows the execution time of this version of the simulator using the same scenarios of the previous Figure.

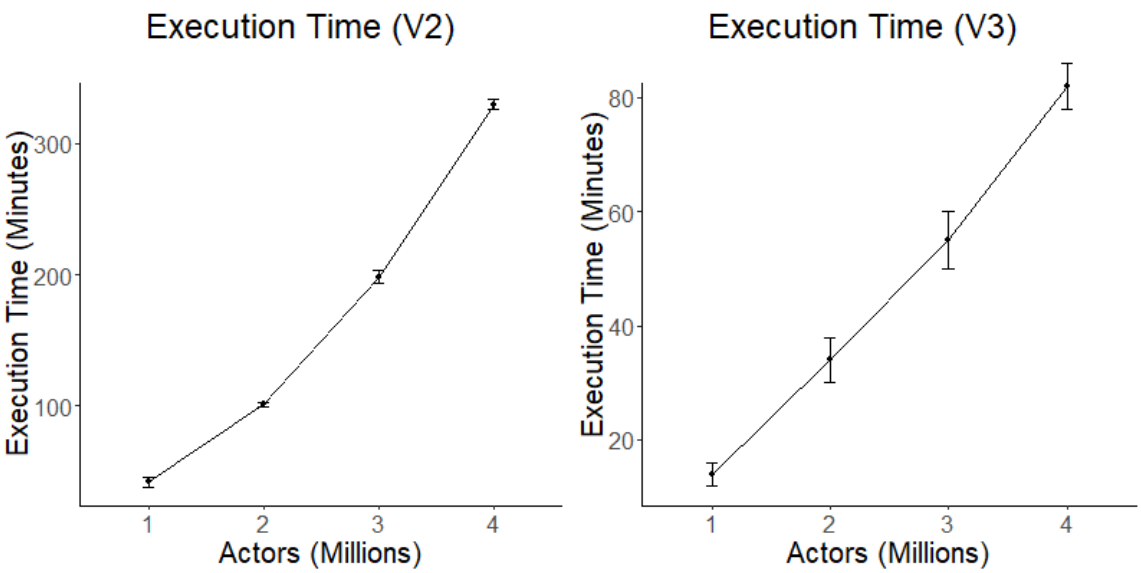

Figure 4.7: *Execution Time of InterSCSimulator V2 and V3*

4.6.2 Actors Creation

In the InterSCSimulator, an Erlang actor performs each simulated travel. Therefore, if a simulation has 10 million travels, 10 million Erlang actors are created during the simulation. In the first version of the simulator, all the actors were created during the simulation. It was not a problem because the simulator maximum scalability was 50 thousand actors.

However, in the second version of the simulator, the scale increased a lot and the creation of the actors was a bottleneck in the simulation and we found a bug in the Sim-Diasca simulator that made the creation of actor very slow. To minimize this problem,

we decided to create all the actors before the simulation start. This approach solved the scalability problem but increased the memory used by the simulator. Figure 4.8 shows the memory used to execute the simulations in this version of the simulator.

In the third version, the Sim-Diasca developers solved the problem in the actor creation and released a new version of the simulator. Therefore, we could return to the first approach and create the actors during the simulation. Along with other improvements in the simulator, including the ETS tables, the third version of the simulator uses almost five times less memory to execute. Figure 4.8 shows the memory used to execute the simulations in this version of the simulator.

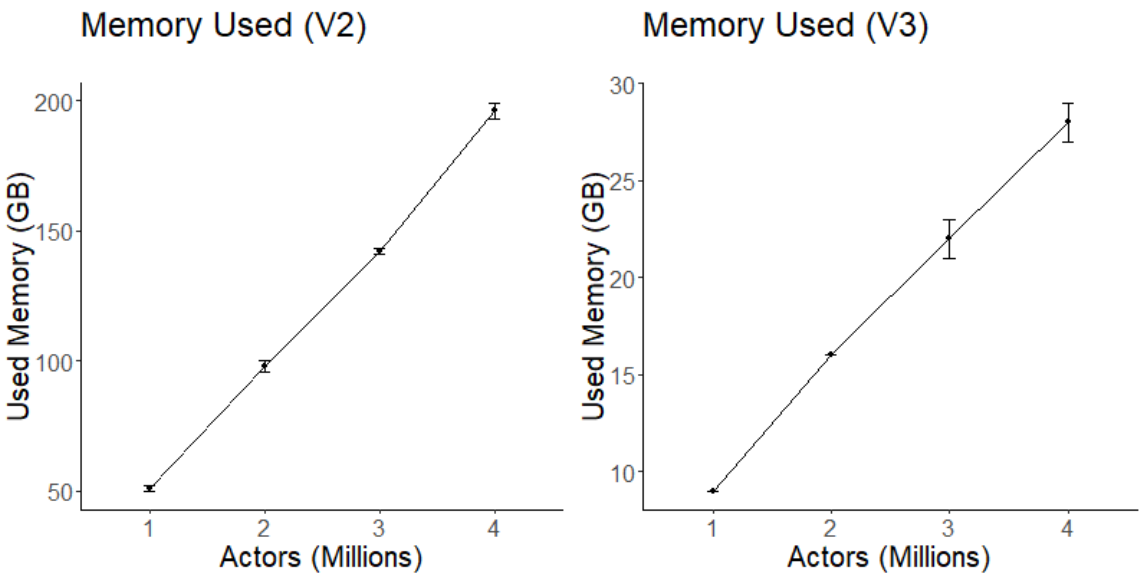

Figure 4.8: *Memory Used in InterSCSimulator V2 and V3*

Chapter 5

Case Study: São Paulo Simulation

This chapter presents the simulation of the city of São Paulo as a use case of InterSC-Simulator. Also, we validated and verified the models described in Chapter 4 using this scenario. The simulation validation and verification are critical to demonstrating that the models are working and producing useful results. This simulation is also the basis for the scalability analyzes presented in Chapter 6.

5.1 Input Data

We based our simulation in real data collected from different sources. The databases considered in this simulation are:

- Origin-Destination (OD) Matrix derived from a survey conducted by the city subway company for the year 2012[1]
- Map of the city based on OpenStreetMap [2]
- Bus lines and stops of the city provided by the Municipal Transportation Secretary
- Subway network of the city provided by the Metrô Company

5.1.1 Origin-Destination Survey

The OD Survey is a research conducted by the subway company of São Paulo. It is performed in intervals of 10 years and collects a large dataset of information about the city population. For example, the survey collects the origin and destination of all travels of a citizen, transportation mode, travel time, and estimated departure and arrival times. Besides the OD, the company also conducts, five years after the OD survey, a smaller research called Mobility Survey, aiming to updates the data from the original OD survey.

[1] Origin-Destination Survey - http://goo.gl/Te2SX7
[2] OpenStreetMap - https://www.openstreetmap.org

The 2012 survey of the city has more than 60 thousand records. Each record contains information about the trips of one person from an origin to a destination, which are different places in the city such as houses, workplaces, and schools. Besides the origin and destination, there are other valuable data about each trip such as the start time, expected arrival time, and transportation mode. The survey has information about bus, cars, rides, bicycle, suburban trains, and subway travels.

Other fundamental information to the simulator is an extrapolation factor which estimates the number of people that make similar trips to each OD record. We use this data to simulate the entire city population randomly positioning agents close to the original position of the OD record. With this information, there are more than 20 million trips and 11 million people in the OD, what is very close to the entire population of the city of São Paulo.

Figure 5.1 presents the number of trips in the OD survey. There are 5.058.002 trips by car, 4.029.546 trips by buses, 6.287.487 trips on foot, and 3.030.809 trips by subway or suburban train. The figure shows that most of the city population makes their trips walking or using public transportation, especially buses. However, there are also a huge number of cars in the city streets. The OD survey has data about other transportation modes such as car passengers, bicycles, taxis, and school buses. We did not consider these modes because some of them do not impact the traffic such as car passengers or we do not have real data to make useful simulation such as bicycles, taxis, and school buses.

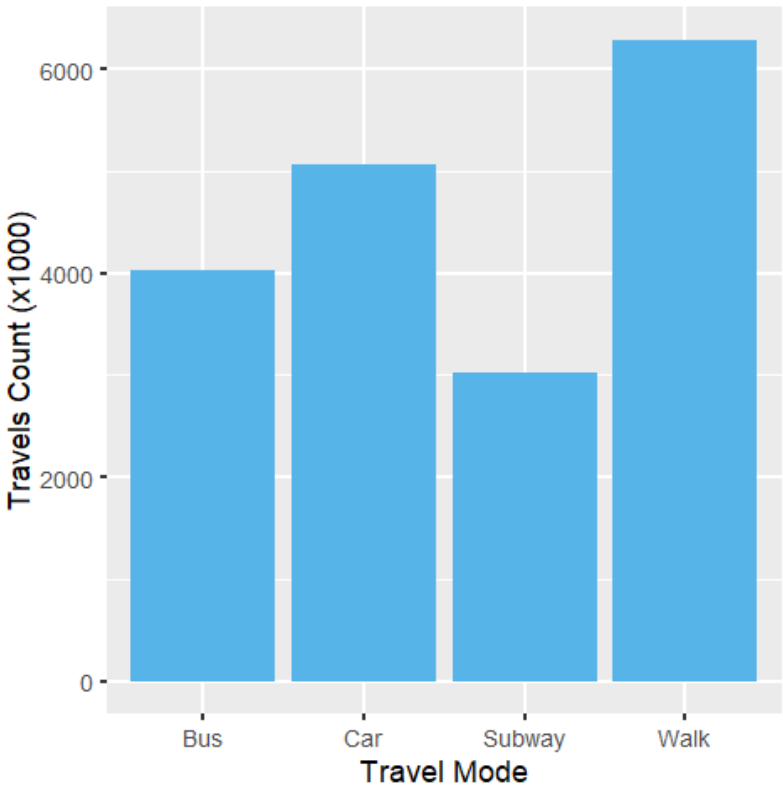

Figure 5.1: *Total Number of Trips by Transportation Mode*

Another essential information to the simulation is the time when each person starts

his travel. Figure 5.2 presents the tips grouped by the start time using 1-hour intervals. This chart shows that there is a considerable variation in the number of trips during the day. Most of the trips are during the peak hours (from 6 to 9 am and from 5 to 8 pm) and at the lunchtime. However, most of the trips during lunch time are on foot or by subway and are small. Probably they are performed by people going from their work to restaurants or home to lunch.

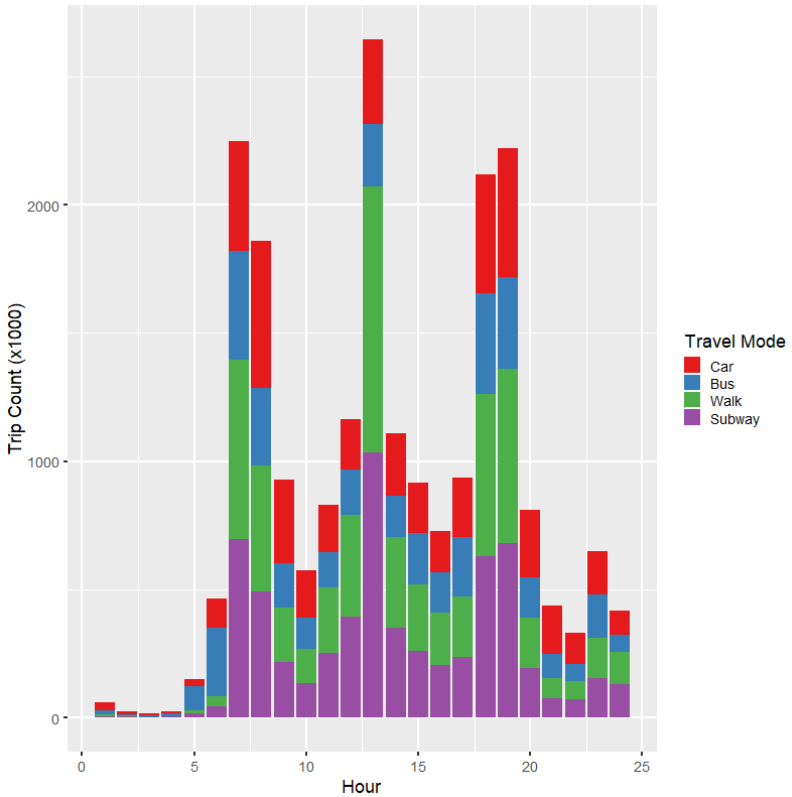

Figure 5.2: *Number of Trips by Transportation Mode and by Hour*

Another relevant information is that the OD survey has only the primary transportation mode of the commuters. Hence, it is possible that many people use more than one transportation mode during one travel. For example, in São Paulo it is common to have buses from a neighborhood that go to subway stations, where most of the people take the subway to arrive at the downtown. Finally, the OD has the expected travel time of the commuters. We used this information to verify if the used mobility models are generating reasonable results.

A problem with the OD survey is that it is updated every five years. Of course, there are many changes in the city mobility in this interval. However, even with this problem, the OD survey is the most representative data about São Paulo mobility patterns.

5.1.2 City Map

We used the OpenStreetMap to generate the graph of the city. This graph has approximately 50 thousand nodes and 120 thousand links and covers most of the streets

and roads of São Paulo and some parts of its metropolitan area. In the graph, the links represent stretches of the streets and the nodes the intersections. All the nodes in the graph have a unique id and their latitude/longitude. The links have the start and end node, the maximum speed, the length, the type, and the capacity of the street. With this graph, we can implement all the models described in Section 4.3.

5.1.3 Bus Lines and Stops

To simulate the bus lines and stops of São Paulo, we used data from the city transportation secretariat (SPTrans). They provided a spreadsheet with all the lines of the city with information about them such as the code, the stops, the start time, and the average interval of the buses along the day.

There are 2,347 lines in the city, and they have different characteristics. For example, some lines run only on weekends and there are others that work just for a few hours, mainly in the peak hour. The number of stops can also vary a lot from line to line: on average each bus makes 43 stops in its route. However, there are lines with just four stops and others with 144 stops. Regarding the stops, São Paulo has 19,144 bus stops. As with the buses, there is a significant variance in the stops. On average, five lines pass through each stop in the city. However, there are stops with just 1 line and others with 62 lines.

5.1.4 Subway and Suburban Train Network

To simulate the subway and train system of São Paulo we created a digraph using the data about the lines of the city. Although different companies manage the train and the subway system (CPTM and Metrô), both systems are integrated and have the same cost to the users. The subway system has six lines, 79 stations, and an extension of almost 90 kilometers. The train network has seven lines, 90 stations, and an extension of more than 270 kilometers. Together, the two systems have more than 4 million users per day.

5.2 Simulation Execution

Based on the data described in the last section, we created all the input files of InterSCSimulator and simulated an entire day in the city. The simulation was executed in a machine on the Google Cloud Environment with a memory of 160 GB and 16 cores. The simulation took approximately 7 hours to execute and at the peak used 138 GB of memory. The output file generated by the simulation has more than 17 million trips and occupy 2 GB of the disk.

5.3 Validation and Verification

According to Sargent (Sargent, 2013) to confirm if a simulator is generating useful results, it is essential to validate and verify the simulator output. The validation consists in analyzing if the models and data structures are working and generating the expected outputs. The verification compares the simulation results with the real system and the analyzed variables must be equal or very close in both cases. To validate the models we made several analyzes in the results of the simulator and to verify the simulation models we compared the real travel time of the OD survey with the simulated travel time.

5.3.1 Validation

The output of the simulator is a mobility trace with all the events that occurred in the simulation. To analyze the travel times, we filtered just the arrival events that occur when a commuter arrives at his destination. In this event, we save the total travel time and distance. With this data, we made analyzes about the mobility patterns in the city. For example, to show the concentration of people and vehicles, we generated a heat map with the agents in São Paulo downtown. Figure 5.3 shows the people and vehicles in the morning peak hour, between 8 and 8:10. The Figure presents a significant concentration of vehicles in important avenues of the city such as 23 de Maio and Radial Leste which are known for their great congestion.

Vehicle Trips

Figure 5.4 shows the analyzes pg car trips with the mean travel time, distance, and speed of the vehicles depending on their arrival time. The last chart shows the total number of vehicles that finished a trip for each hour of the simulation.

The charts show that, on average, the trips are much slower in the peak hours than during the rest of the day. It is mainly due to the decrease in the average speed of the vehicles caused by the massive amount of vehicles that are moving in the city. It is also possible to verify that the distance is not very important in the travel time, the average distance of the travels varies from 10 km to 14 km during the day.

Figure 5.5 presents the correlation of time, distance, the hour of the day, and the speed of car travels. The Figure shows that some variables are very correlated, for example time and distance and speed and time. On the contrary, there is no visible correlation between the distance and hour of the day and distance and speed.

Subway Trips

Figure 5.6 presents the correlation of time, hour of the day, and number of stations on subway trips. As the chart shows, the number of stations define the inferior limit of the travel time. However, a significant part of the subway commuters use another

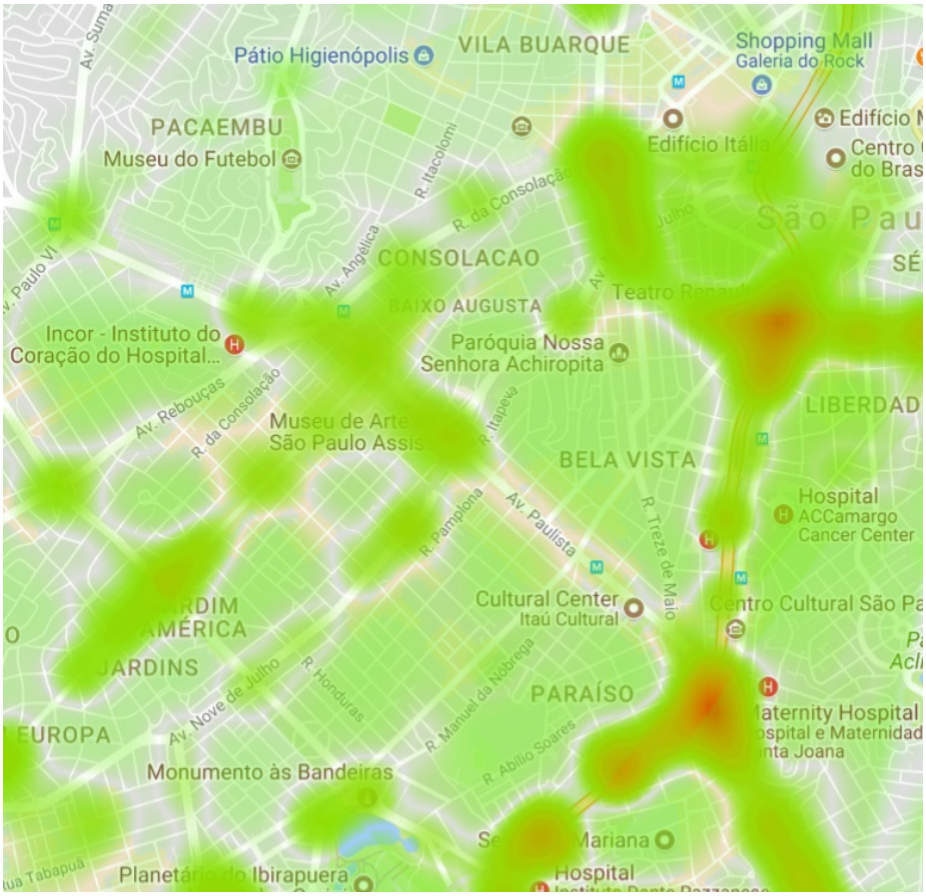

Figure 5.3: *Heat map of the simulated travels*

transportation mode such as buses or walking. The chart also shows that there is a higher number of trips during the peaks hours, in the morning, at lunchtime, and in the end of the afternoon.

Also, the chart shows that there are a small number of travels in the very beginning of the day because the subway system of São Paulo works until approximately 00:30. The last train of almost all lines depart at midnight, and each station closes after this train leaves. The system opens at 04:40 every day.

A limitation of our work is that in the peak hours it is probably that a user will take some time waiting for a train in a station, it occurs because the trains already arrive full or there is a massive queue to enters the train. We do not model this time wasted to get the next train yet. However, it is possible to create a model that calculates the time to enter a train based on the number of people that is in the stations.

Pedestrian Travels

Figure 5.7 presents the correlation of time, the hour of the day, and the distance of pedestrian travels. The Figure shows that there is a linear correlation between the time and distance variables, and there is no correlation between the hour and time and hour

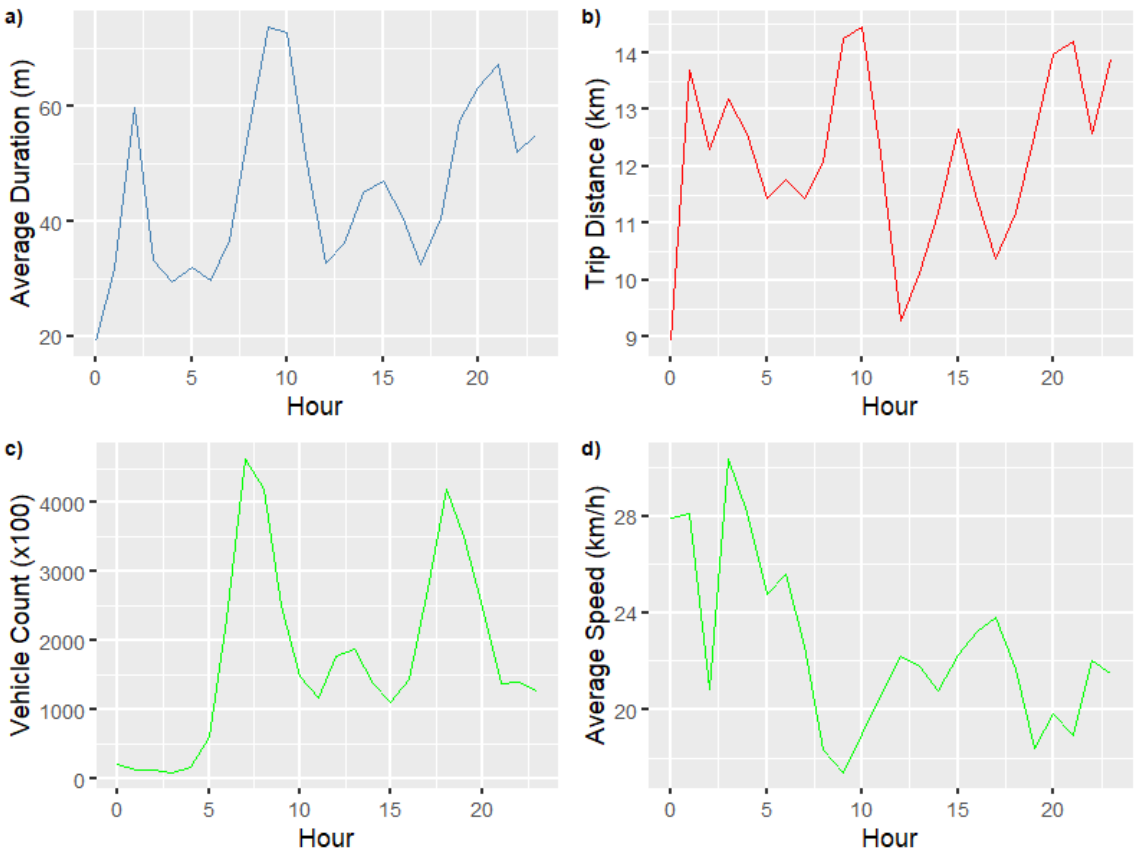

Figure 5.4: *Analyzes of the car travels*

and distance. The chart also shows that the travels are evenly distributed during the day, except during the dawn.

Our pedestrian model is simple, but it is possible to extend it adding information about the pedestrian such as age and mobility difficulties. We did not find any model in the literature that use personal information to model the pedestrian speed, but the information is available in the origin-destination survey.

Bus Travels

The bus travels also suffer the impact of the traffic in the city. Figure 5.8 shows the mean travel time of the buses ad the total number of arrivals per hour. In the first graph is possible to verify that the mean travel time of the buses in the peak hour are almost 15% bigger than the other hours. Also, as showed in the second chart, during the peak hours the number of buses moving in the city is almost the double of the rest of the day.

Regarding the people that travel by bus, Figure 5.9 presents the correlation of time and the hour of the day. The chart shows that exists many long travels with more than 3 hours and the bigger trips are during the morning peak and in the night starting in the night peak.

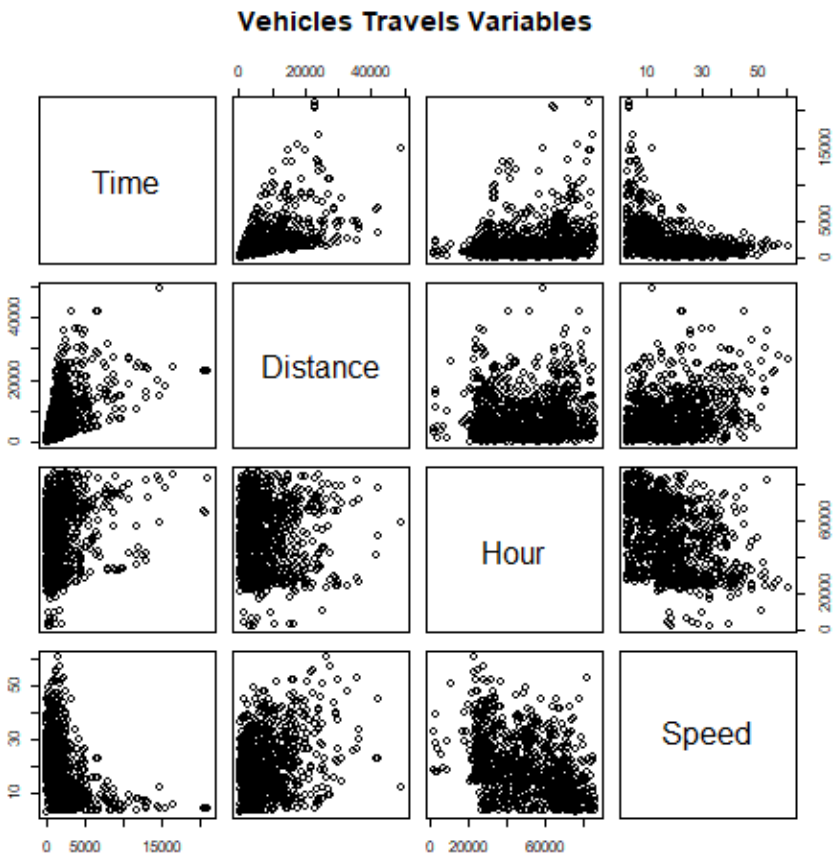

Figure 5.5: *Correlation among Car travels variables*

5.3.2 Verification

To verify the models, we compared the real travel time that is in the OD survey with the simulation travel time. This approach has some limitations, for example, most of the answers in the OD are rounded, because typically people approximate their travel time when answering the survey. Another problem is that the OD is from 2012 and we are using the mobility infrastructure of 2018. In the pedestrian and cars travels it will have a limited impact as the streets and roads did not have great changes. However, the bus and subway travels are significantly affected by this problem because the city now has new subway lines, and the bus lines are modified continuously.

Figure 5.10 presents the comparison of all travels. The chart shows that the travel time from simulated and real environments is between 300 and 6000 seconds. The median of the simulated travels are 1800 seconds and of the real travel are 1840 seconds, an error of only 2,5%. This difference is caused mostly by the bus travels as we will show in the next figures.

Figure 5.11 presents the comparison of car and pedestrian trips. Regarding the car trips, the chart shows that the travel time from simulated and real environments is between 300 and 5000 seconds. The median of the simulated travels is 1655 seconds and of the real

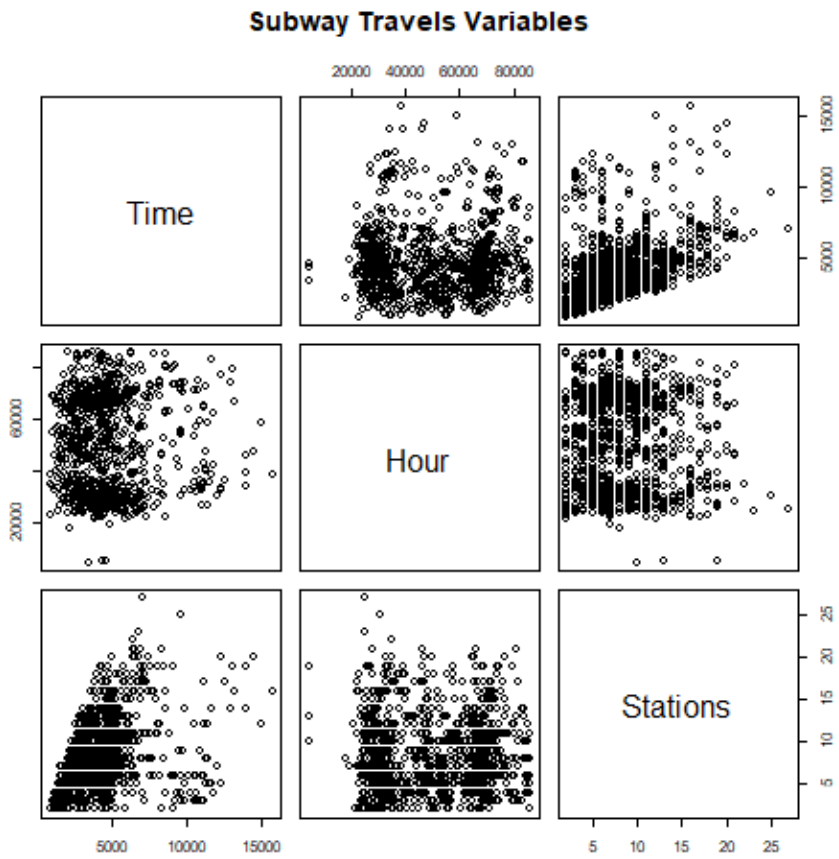

Figure 5.6: *Correlation between Subway trips variables*

travel are 1756 seconds, a difference of 6%. This small difference shows that the mesoscopic model used in the InterSCSimulator is reproducing well the city environment.

Regarding the pedestrian travels, the chart shows that the travel time from simulated and real environments is between 300 and 3000 seconds. The median of the simulated travels are 880 seconds and of the real travel are 868 seconds, a difference of 1%. This difference shows that the model used in the InterSCSimulator is reproducing well the city environment, mainly because simulating a mesoscopic pedestrian behavior is very straightforward.

Figure 5.11 presents the comparison of subway and bus travels. Regarding the subway travels, the chart shows that the travel time from simulated and real environments is between 300 and 7000 seconds. The median of the simulated travels are 4080 seconds and of the real travel are 3805 seconds, a difference of 7%. The subway model is also simple to simulate, most of the difference in this model is caused by the buses that are used by some commuters to arrive at a subway station.

Regarding the bus travels, the chart shows that the travel time from simulated and real environments is between 1000 and 6000 seconds. The median of the simulated travels is 3585 seconds and of the real travel are 3240 seconds, an error of 11%. This difference shows that the model used in the InterSCSimulator is not reproducing well the city environment.

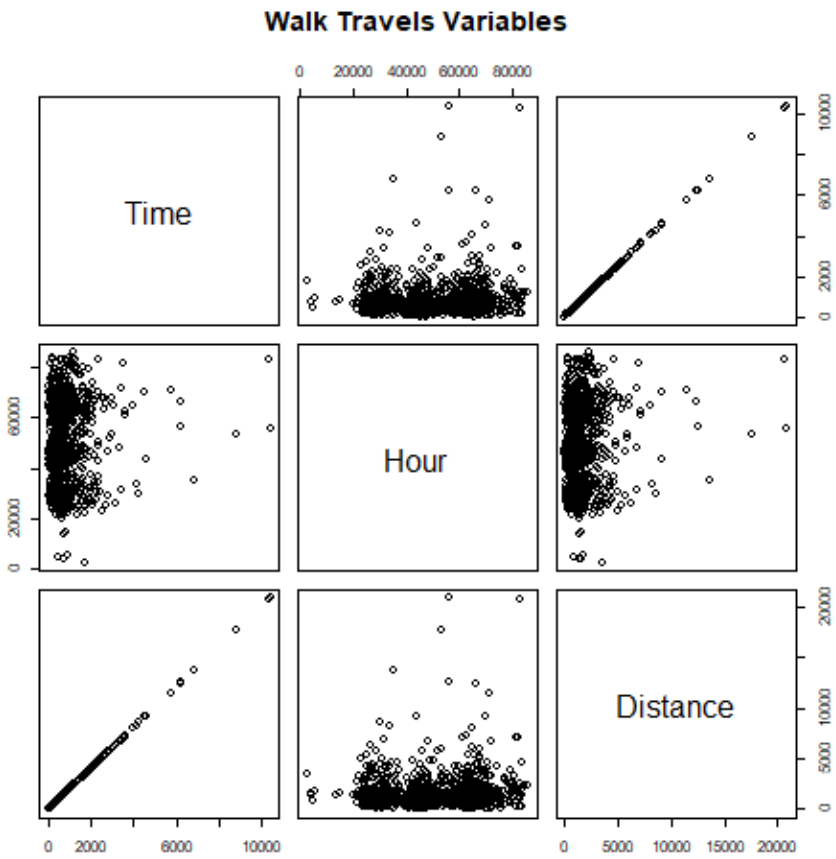

Figure 5.7: *Correlation between Pedestrian trips variables*

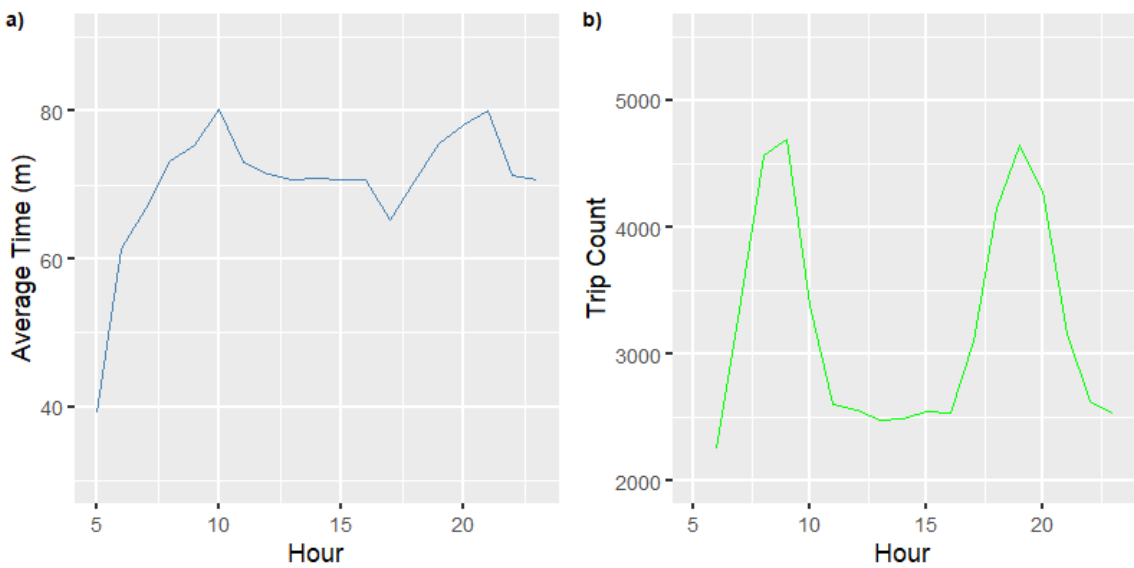

Figure 5.8: *Analyzes of bus trips*

The cause of this difference is that the bus travels are much harder to reproduce for different reasons such as the schedule of the buses varies a lot in São Paulo, the city has more than

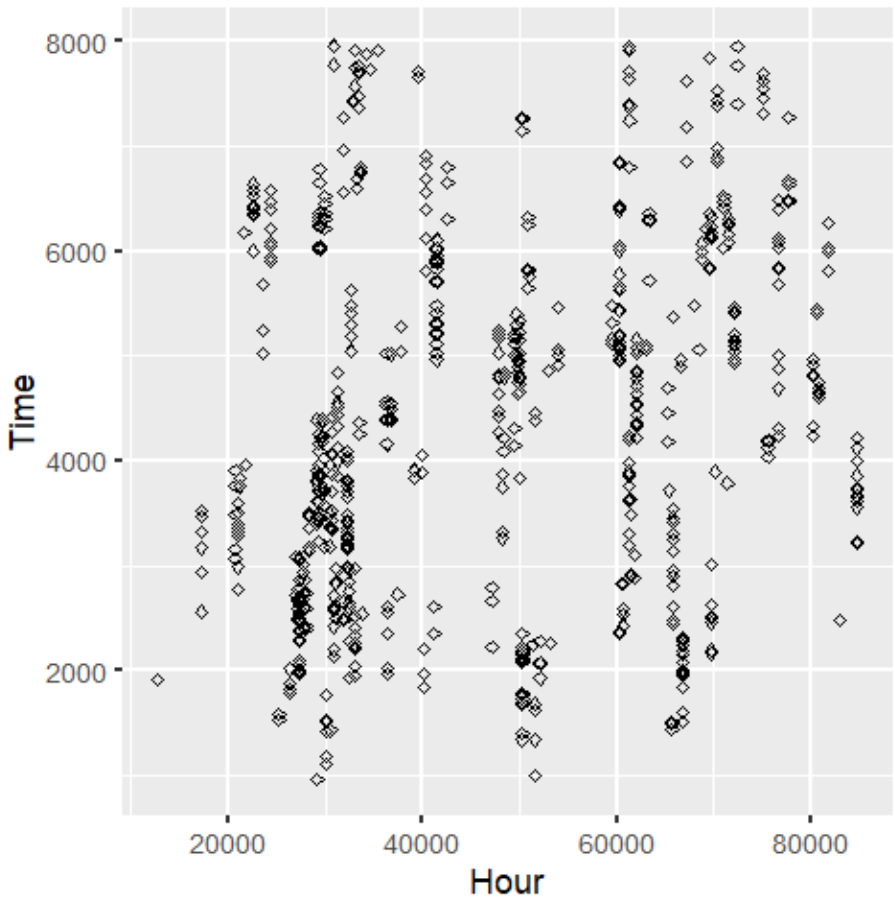

Figure 5.9: *Correlation between time and hour on Bus trips*

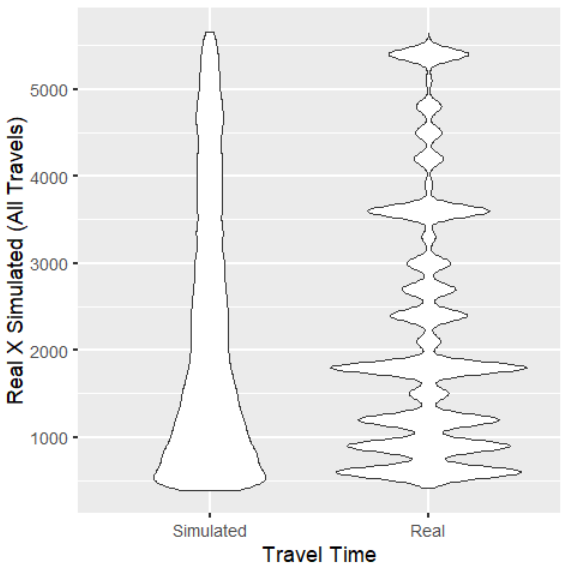

Figure 5.10: *Travel time in real and simulated environments*

2000 lines, and it is necessary to analyze one by one to reproduce the exact itinerary of the buses. Finally, for commuters that use more than one bus line, it is hard to reproduce

their exact itinerary.

A work based on the simulator developed in a Smart City course implemented a much better bus model to the simulator. However, we did not integrate this model with the original models of InterSCSimulator yet. We will present this model in Section 7.4.

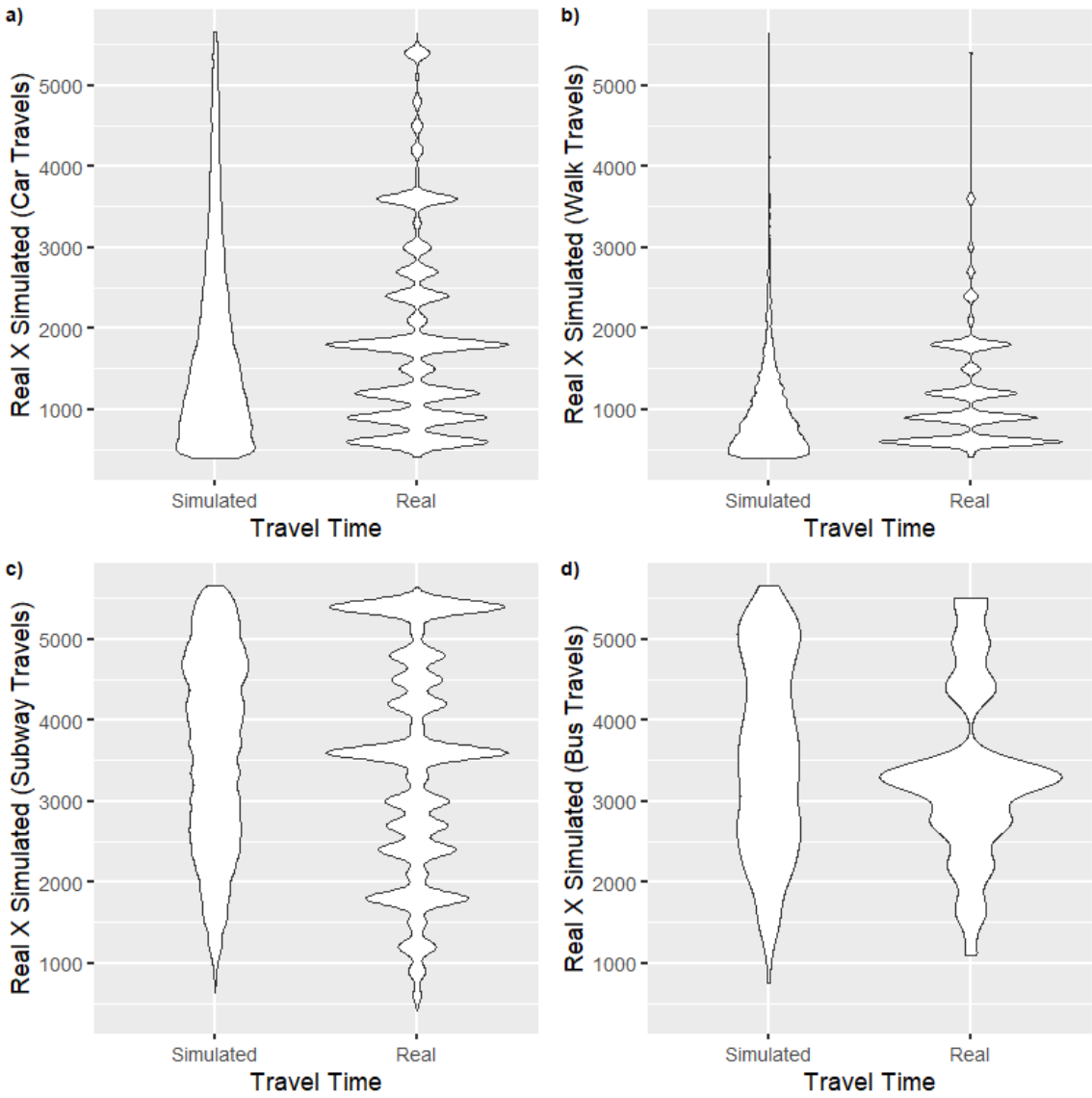

Figure 5.11: *Car, Pedestrian, Subway and Bus Travel time in real and simulated environments*

Chapter 6

Scalability Evaluation

This chapter presents the experiments to demonstrate InterSCSimulator scalability. To perform the experiments, we executed the simulator with different workloads, all based on the data presented in Chapter 5. We started the experiments using 20% of the OD population and increased the load until reaching the simulation of 100% of the population. These experiments allow us to verify the increase in the usage of the computational resources concerning the number of simulated agents, enabling the calculation of the resources required and the costs to run the InterSCSimulator. Section 6.1 presents the data used in all the executed experiments, Section 6.2 presents all the simulator executions and the use of computing resources such as CPU, memory, and disk, and Section 6.3 presents our findings from the experiments.

Also, in Section 6.4.2 we show a scalability comparison between SUMO and InterSC-Simulator to demonstrate that how each simulator handle a large number of vehicles and how the machine resources are used.

6.1 Experimental Data

To perform the experiments, we created subsets of the OD survey of São Paulo presented in Section 5.1. The first subset with 20% of the total trips and four other, each increasing 20% of the population until using the complete OD, all the experiments simulated an entire day, from 00:01 am to 11:59 pm. We executed the experiments in the Google Compute Engine (GCE), a cloud environment which provides virtual machines (VMs) with different capabilities. Table 6.1 summarizes the dataset and the VM memory used for each experiment. All the other characteristics of the VMs was the same to all the experiments: 16 CPUs with one core, Linux Debian Operating System, and 10 GB of disk.

All the experiments are available as Docker (Merkel, 2014) images to facilitate the execution and allow the reproducibility of the work[1].

[1]DockerHub - https://hub.docker.com/r/ezambomsantana/

	Total Trips	Car Trips	Pedestrian Trips	Subway Trips	Bus Trips	VM Memory
Experiment 20	3.681.168	1.011.600	1.257.497	606.161	805.909	32GB
Experiment 40	7.362.337	2.023.200	2.514.994	1.212.323	1.611.818	50GB
Experiment 60	11.043.506	3.034.801	3.772.492	1.818.485	2.417.727	70GB
Experiment 80	14.724.675	4.046.401	5.029.989	2.424.647	3.223.636	90GB
Experiment 100	18.405.844	5.058.002	6.287.487	3.030.809	4.029.546	104GB
Experiment 120	22.087.012	6.069.602	7.544.984	3.636.971	4.835.455	130GB

Table 6.1: *InterSCSimulator Scalability experiments*

6.2 Experiment Executions

All the experiments had a similar execution pattern. All start using almost 100% of CPU to create the scenario, the time to this activity depends on the size of the simulations, varying from 2 minutes in Experiment 20 to almost 40 minutes in Experiment 100. Then, the simulation starts using few resources until the start of the morning peak hour when the consumption of memory increases. The same occurs during the simulated afternoon. There are just small variations until the beginning of the night peak hour when the memory use increases again. The CPU usage does not have a pattern and varies during all the simulation. For example, Figure 6.1 presents the use of the resources during one execution of Experiment 100.

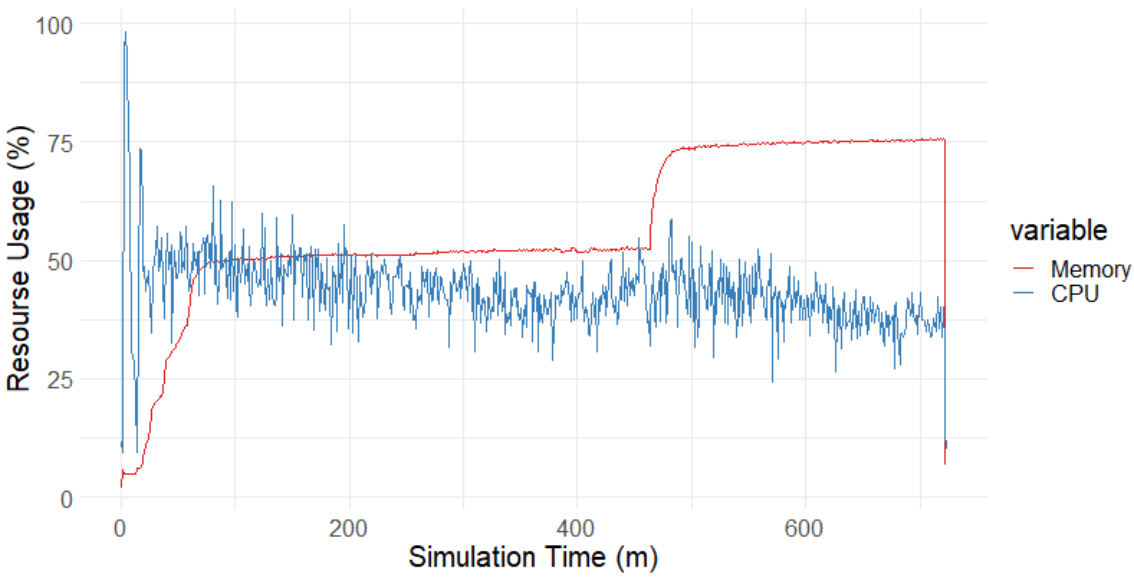

Figure 6.1: *Experiments with 100% trips of the OD Survey*

Table 6.2 summarizes the use of resources during the simulation executions. All the

simulations were executed five times, and the presented values are the mean of these executions.

	Memory	CPU	Events Hour
Experiment 20	68% (22GB)	70%	2.799.198
Experiment 40	70% (36GB)	65%	3.265.731
Experiment 60	71% (49GB)	55%	3.498.997
Experiment 80	73% (68GB)	52%	3.732.264
Experiment 100	76% (79GB)	49%	4.198.797
Experiment 120	76% (91GB)	47%	4.578.556

Table 6.2: *InterSCSimulator Scalability experiments*

The number of events during the simulation explains most of the variations in the use of computational resources. Figure 6.2 presents the number of events grouped by the simulation hour. The number of events during the two peak hours is almost five times larger than the rest of the simulation.

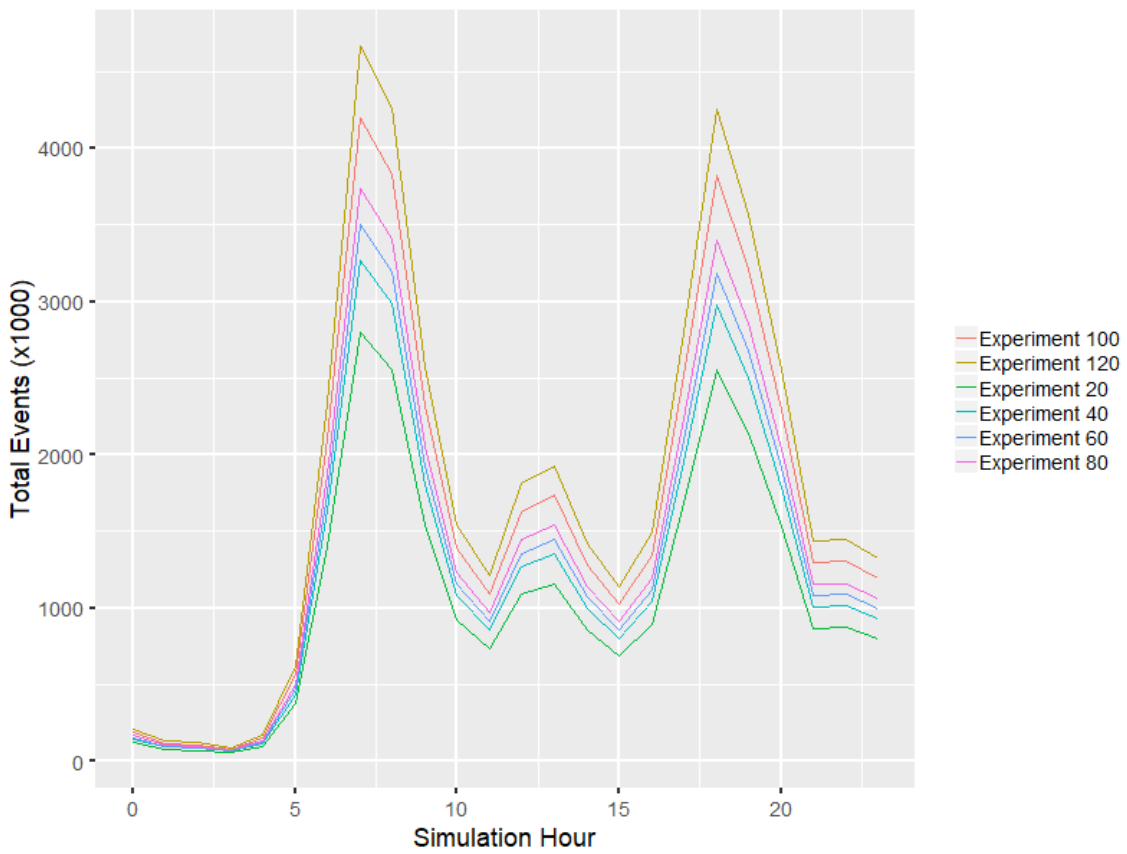

Figure 6.2: *Events during the simulation*

The explanation of the relation to the number of events and execution time of the simulation is explained by the increase in time to execute one simulated second. At the beginning of the simulation, when there are just a few events to process, the simulation time is almost 100x faster than real-time, then during the peak hour, this ratio decreases to 20x faster than real time in the first scenario and 0.8x in the last scenario. Figure 6.3 presents the ratio between the simulation time and the real time to all scenarios.

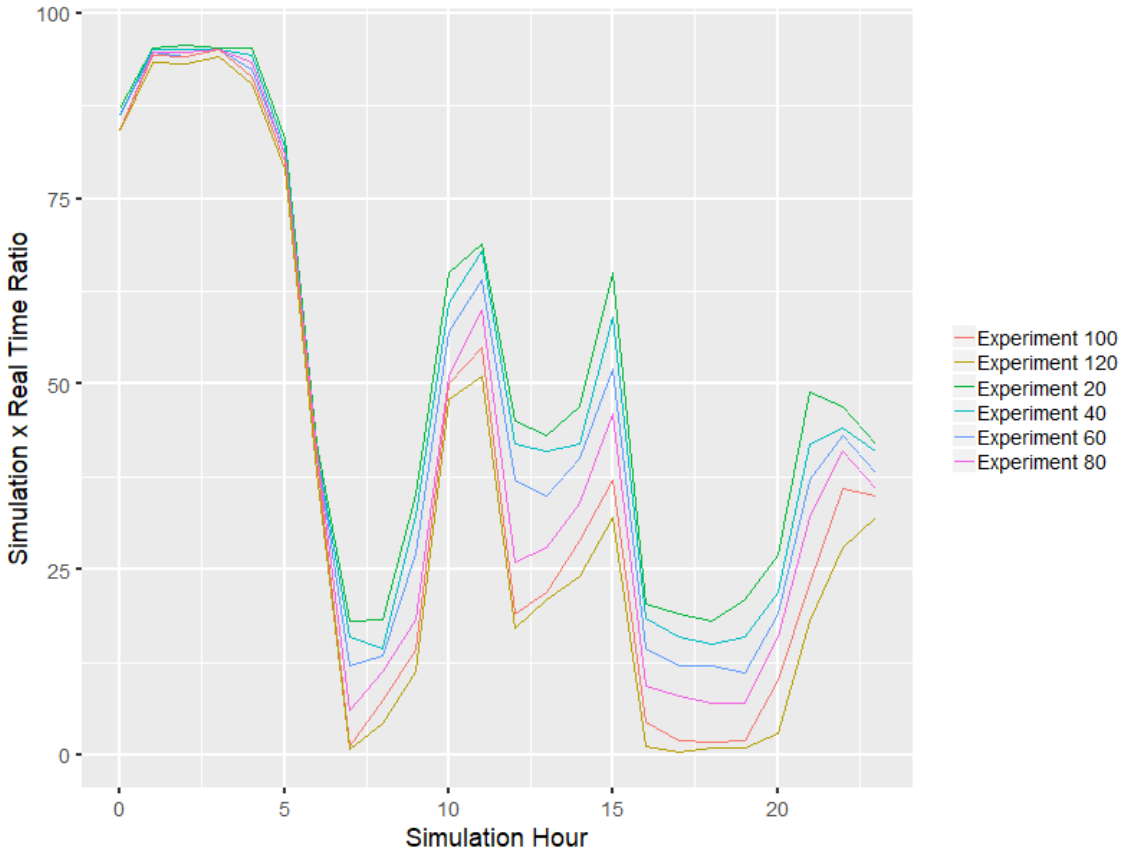

Figure 6.3: *Ratio between Simulation and Real Time*

6.3 Results and Discussion

The charts in Figure 6.4 summarize the results of the scalability experiments. Regarding machine resources, the Memory and Disk Space usage is proportional to the number of simulated trips, and the results showed that the simulator scales linearly to both. The CPU has a non-intuitive behavior: increasing the number of vehicles, and consequently the number of events, the mean usage of the CPU decreases. This happens because the Sim-Diasca mechanism to synchronize the simulation is a bottleneck, slowing down the simulation.

Regarding the costs, we used the Google Compute Engine to execute all the simulations. The prices started at 0.47 USD per hour for Experiment20 VM and increased to 0.82 USD for Experiment100. To calculate the total cost, we multiplied the VM cost by the experiment

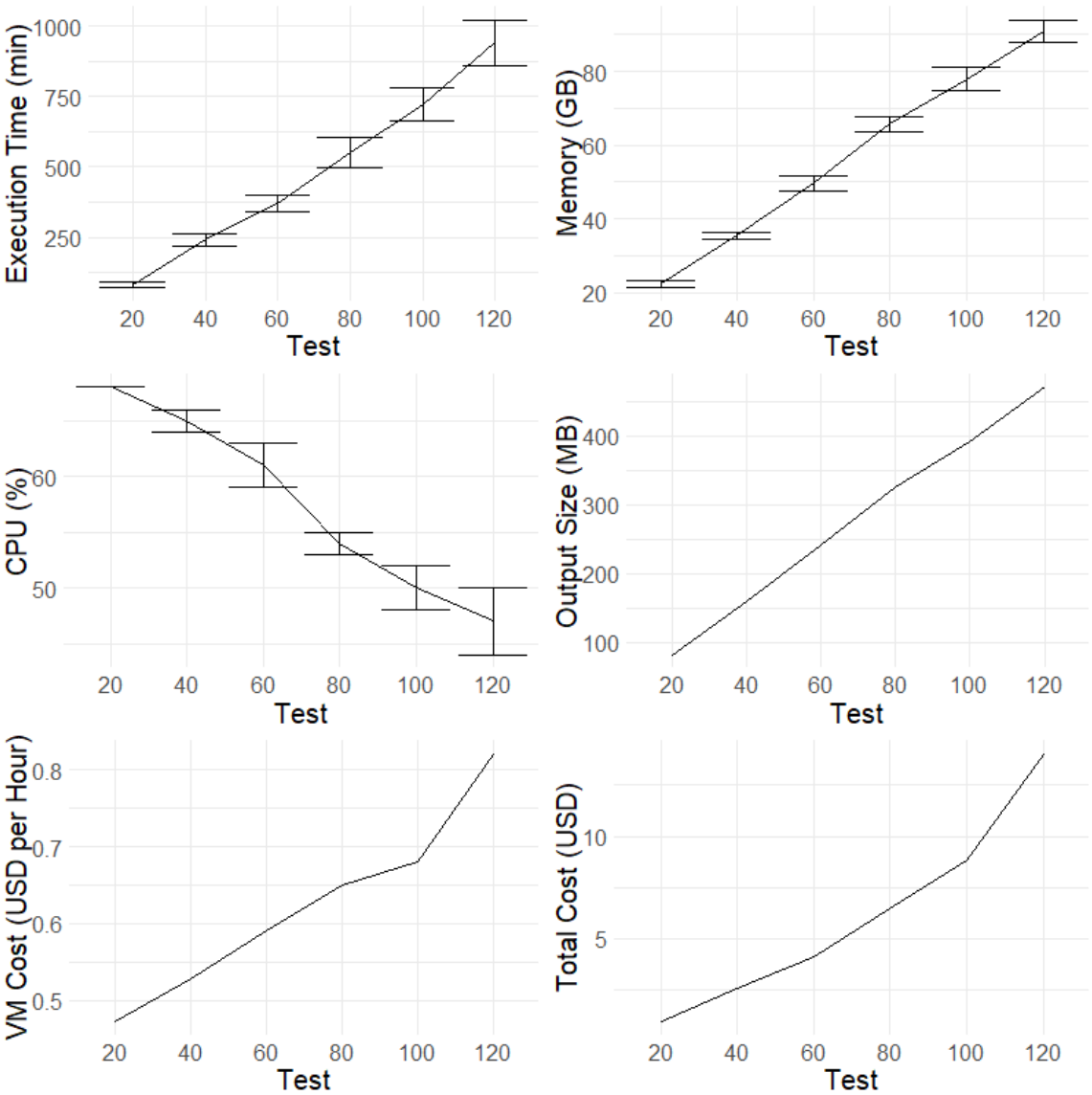

Figure 6.4: *Comparison among the five Experiments*

time. The results showed that the cost to execute a very large simulation is relatively low: with less than 10 dollars it is possible to simulate the one day traffic of the entire city of São Paulo.

Thus, we conclude that InterSCSimulator is scalable, using well the memory and the disk of the machine. Regarding the CPU and the execution time, there are many possible improvements in the simulator architecture to increase the CPU usage and decrease the simulation time in large simulations. The costs to run a large simulation in a powerful machine in the cloud is low, allowing the use of InterSCSimulator by any researcher and government.

6.4 Scalability Comparison

This section compares the scalability of SUMO and InterSCSimulator to show that the former can execute faster and using better machine resources. Of course, the simulation model of SUMO, which is a microscopic model, demands more resources than the mesoscopic model used in InterSCSimulator. However, other problems such as the single-core implementation of SUMO makes its execution time almost ten times bigger than InterSCSimulator.

6.4.1 Scenario

To compare the simulators we used as a base a SUMO simulation of Luxembourg City. This scenario simulates more than 210,000 vehicles in 24 hours. Both simulators used the map from OpenStreetMap to generate the road network, and we used the travels from SUMO original scenario to generate the InterSCSimulator travels. Figure 6.5 presents the simulated region.

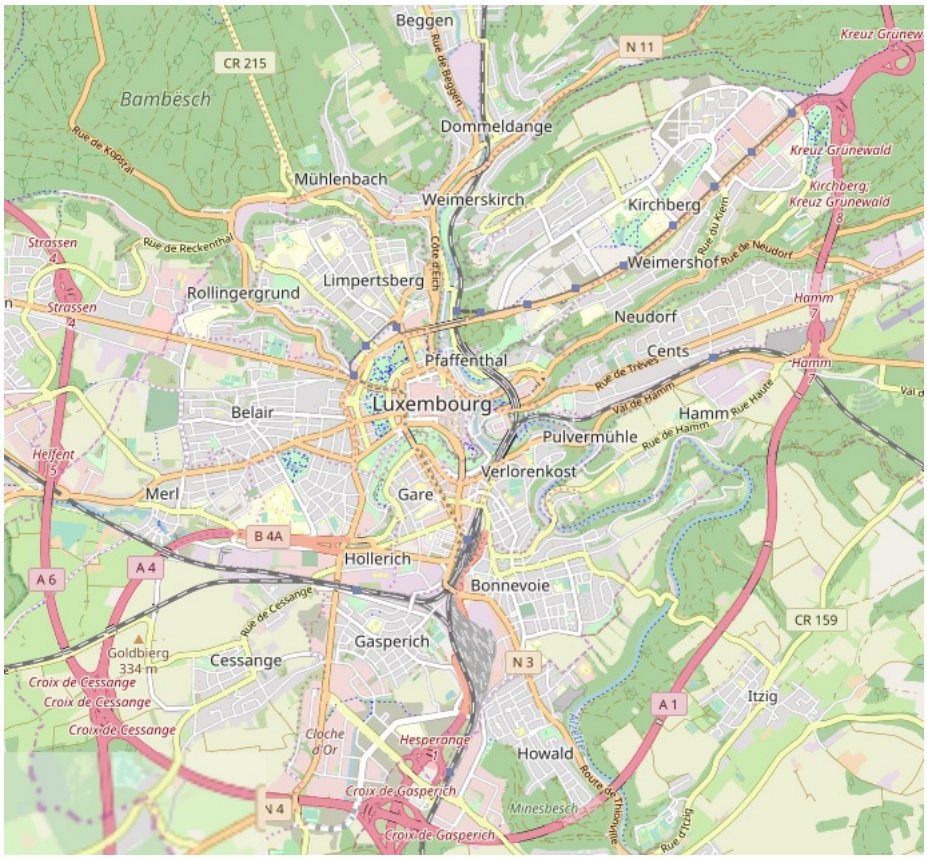

Figure 6.5: *Luxembourg area - The simulated region*

Table 6.3 summarizes the data used in the simulations. To understand the evolution of the resource usage in SUMO and InterSCSimulator, we executed three simulations, the first with 72,000 vehicles, and then multiplying it 2, 3, 4, and 5 times. Sumo has a

directive to multiply the number of vehicles automatically and for InterSCSimulator we created different input files. A better description and how to execute all the experiments are available in GitHub [2].

Map Nodes	2.376
Map Links	5.969
Vehicles	72,000 - 144,000 - 216,000 - 288,000 - 360,000

Table 6.3: *Simulation Scenarios*

6.4.2 Scalability Comparison

To compare the scalability of the simulators, we executed similar simulations in both simulators. The tests were executed in a machine with a Ryzen 5 2600X processor with 6 CPUs and 12 threads, 16 GB of DDR4 memory, and an SSD disk with 512GB with a Ubuntu 18.04. For both simulators, we collected the average CPU usage, the used memory, and the execution time. Table 6.4 summarizes the results of the execution on both simulators.

		Scenario 72	Scenario 144	Scenario 216	Scenario 288	Scenario 360
SUMO	Memory	8.1%	8.4%	9.2%	10%	10.9%
	CPU	9%	10%	10%	10%	10%
	Execution Time	12m 13s	22m 18s	32m 12s	120m	320m
InterSCSimulator	Memory	23.2%	23.3%	23.3%	23.5%	23.9%
	CPU	77%	79%	79%	79%	79%
	Execution Time	1m 25s	2m 28s	3m 49s	5m 15s	7m

Table 6.4: *SUMO and InterSCSimulator Scalability Comparison*

SUMO does not use the resources of the machine properly, mainly because the simulation engine is single threaded wasting most of the CPU time of the computer. As presented in the table, the CPU usage is around 10% in all simulations because it is using only one of the 12 threads of the computer. The used memory increased proportionally with the number of simulated cars, which was expected. The simulation execution time also increased and after the forth scenario the time started to increase very fast, what shows that SUMO can not handle a large number of vehicles.

InterSCSimulator used the CPU better than SUMO, using all the cores of the machine during all the simulation. InterSCSimulator used more memory than SUMO in the scenarios. It is caused mainly by the Erlang Virtual Machine, while SUMO is written in C, a language that does not need a VM. However, all the scenarios used almost the same amount of memory while in SUMO the memory usage increased with the number of

[2] GitHub Repository - https://github.com/ezambomsantana/sumo_x_interscsimulator

vehicles. The execution time of InterSCSimulator also increased proportionally to the number of simulated vehicles, but at a much lower pace than SUMO. Figure 6.6 presents the execution time comparison between the simulator.

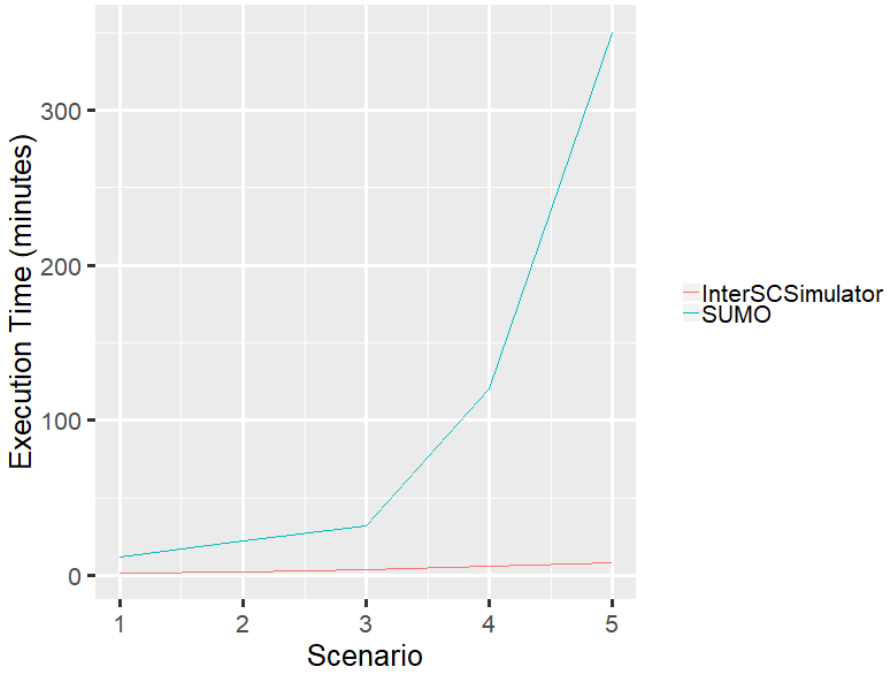

Figure 6.6: *Execution time of SUMO and InterSCSimulator*

6.4.3 Discussion

Both simulators have many uses for researchers and professionals. For example, the detailed model of SUMO is better to understand the impact of different semaphore position and times. However, if it is necessary to simulate an entire city to generate data to test a Smart City application, the InterSCSimulator is more suitable, because it can simulate in real-time a vast number of actors. A possible future work is to implement a microscopic model in InterSCSimulator and verify if its architecture, based on the Erlang language and in the actor model, can still having a better execution time compared to SUMO.

Chapter 7

Simulator Use Cases

This chapter presents the utilization of the InterSCSimulator to support other research during the last two years. Section 7.1.1 presents the utilization of the simulator to perform experiments on InterSCity Platform, a Smart City software platform. Section 7.2 describes the simulation of different mobility scenarios in the Paraísopolis community in São Paulo. Section 7.3 presents the simulation of Vehicular Ah-Doc Networks (VANETs) aided by mobility traces generated from InterSCSimulator. Finally, Section 7.4

7.1 InterSCity Platform Integration

To test Smart City services and applications, we integrated the InterSCSimulator and InterSCity Platform. The InterSCity platform is an open source micro-services-based platform to enable collaborative research, development, and experiments in smart cities. The platform handles the mandatory requirements to support integrated smart city services and applications in different domains such as transportation, health-care, and environmental monitoring. The integration is performed in two ways:

- **Application Request:** The platform simulates the access of a person to a Smart City application deployed in the platform. To simulate it, the simulator makes an HTTP request to a platform service that sends a response with the data requested. The simulator must handle the response and change the behavior of the actor that sends the request.

- **City Infrastructure:** The simulator is also capable of simulating city sensors which generate data and send to the platform. This data is sent to the platform using a queue system deployed in the platform.

Figure 7.1 presents the integration of the platform and the simulator. The application request integration is at the top of the figure, showing that the simulator makes a request and receive a response from the platform. The City Infrastructure integration is at the bottom of the figure, showing that the simulator sends sensor data to the platform.

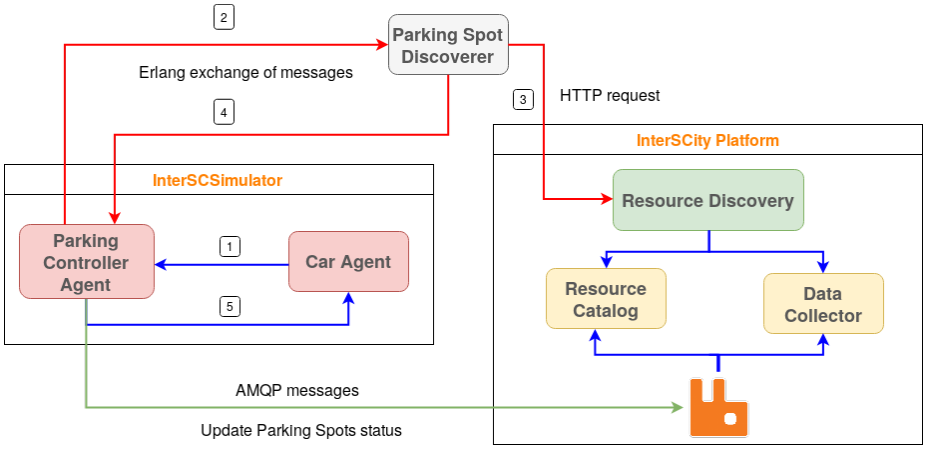

Figure 7.1: *Platform Response Time*

The InterSCity research group tested a Smart Parking application using the simulator-platform integration. The scenario was: (1) The car agent sends a message to the Parking Controller Agent asking for a parking spot; (2) the agent makes an HTTP request to the platform seeking a parking spot; (3) the platform seeks for a parking spot, and (4) the platform sends a parking spot and its location; (5) the driver changes its route to go to the parking spot. When the driver stops the car in the parking spots, update the state of the spot sensor in the platform.

7.1.1 InterSCity Platform Scalability Experiments

The focus of InterSCity platform is to handle the required scalability of a Smart City platform. It is necessary because a Smart City will have thousands of applications and millions of users. The platform has to deal with a considerable variation in the load during the day. For example, a traffic application will be hugely used during the peak hour and normal use in the other hours.

As perform scalability experiments in real environments is not easy, the developers of the platform used the simulator integration to generate a massive workload to the platform. The simulated scenario was the Smart Parking application in the morning peak hour in the city of São Paulo. The simulation had more than 400 thousand simulated cars, and each car made one or more requests to the platform searching for a parking spot.

Figure 7.2 presents the workload generated in the simulator. The number of requests increase with the passage of time and allow the verification of the behavior of the platform with the variation of the load. The figure presents the workload mean of 15 executions of the platform experiments.

The simulator developers executed more than 50 experiments using the simulator, aiding them to find many problems in the platform implementation and achieve the desired scalability. Figure 7.3 presents the response time of the platform using the load of Figure 7.2.

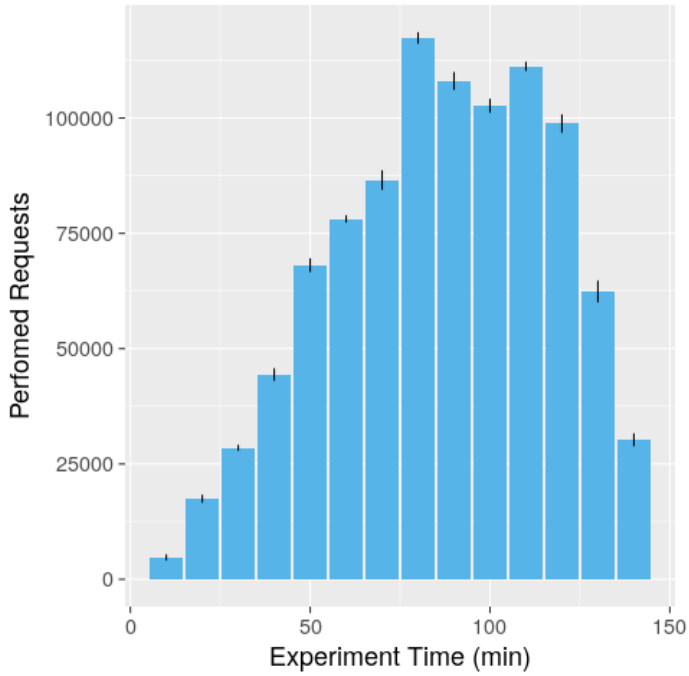

Figure 7.2: *Workload generated in the simulation*

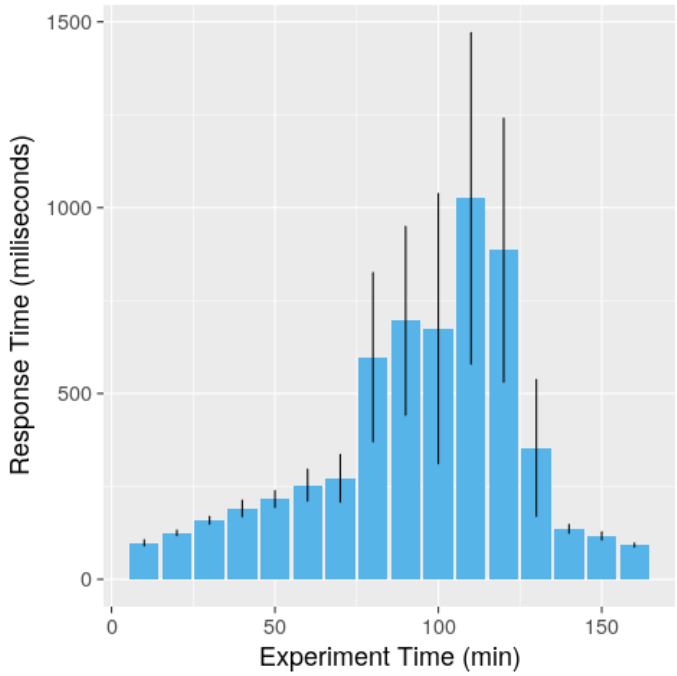

Figure 7.3: *Platform Response Time*

7.2 Scenarios in Paraisópolis

InterSCSimulator is useful to compare large-scale mobility scenarios. The Traffic Engineering Group from the Polytechnic School from the University of São Paulo used

the simulator to study the impact of a subway line under construction in the city of São Paulo, especially in the Paraisópilis community, one of the largest poor neighborhoods of the city. This subway line will have two stations in the community, and it will have a significant impact on the population access to quality transportation.

In their work, they examined four simulated scenarios based on a city origin-destination survey and compared their travel time, financial cost, and carbon footprint of the simulated population. They based all the scenarios on realistic changes that might occur with the new subway line. Besides the OD, they also used data from the city buses and metro lines. They simulated the entire community population (approximately 44 thousand people) and other cars from the city to generate car traffic.

The simulation showed that the users of buses could benefit from a decrease in their trip time, and car uses can have economic benefits with the new subway line. For example, of the population that used cars in the original scenario, approximately 1,500 had their travel times decreased, and 4,000 had their travel times increased. The people that had their travel time increased by more than 30 minutes (around 2,000 people) are unlikely to change their travel mode. However, since the cost reduction can be very substantial (mainly when taking into consideration parking fees), even some of them might prefer the subway.

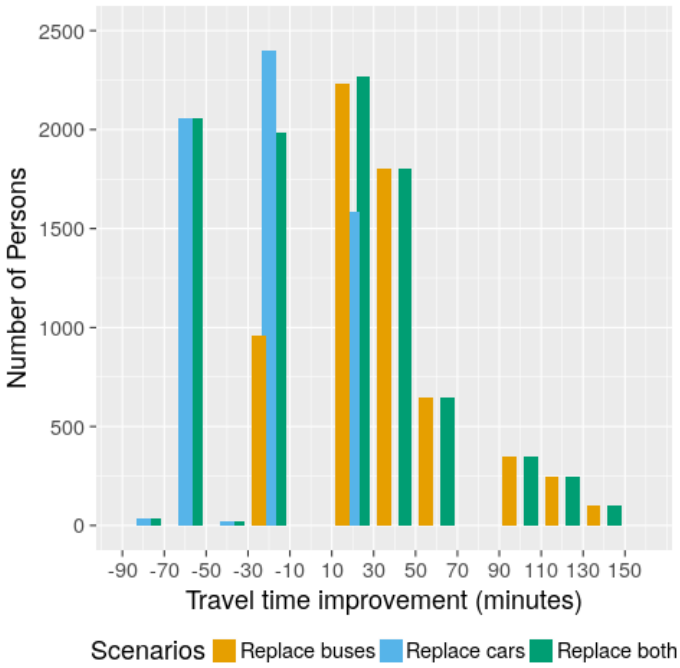

Figure 7.4: *Travel time improvement*

7.3 VANETs Simulation

Using the São Paulo simulation presented in Chapter 5 we generated mobility trace to enable tests and experiments of Vehicular Ah-Doc Networks (VANETs). The trace contains

the events of all the cars of the simulation during an entire day. The advantage of this trace comparing with other traces available in the literature is the enormous amount of vehicles. While the trace generated in this research contains more than 4 million travels of buses and cars, the other traces contains less than 700 thousand travels.

To show the use of this trace, we converted the output of InterSCSimulator to the input format of NS-3[1], a popular network simulator. Listing 1 presents an example of a NS-3 input file.

```
$node_(0) set X_-23.55084
$node_(0) set Y_-46.62869
$node_(0) set Z_ 0
$ns_ at 15451 "$node_(0) setdest -23.55084 -46.62869 0"
$node_(1) set X_-23.55084
$node_(1) set Y_-46.62869
$node_(1) set Z_ 0
$ns_ at 15469 "$node_(1) setdest -23.55084 -46.62869 0"
```

Listing 1: *NS-3 input file*

In the file, the lines starting with a $node, define the creation of network node in the NS-3. The lines starting with $ns represent the movements of the vehicles and occurs many times until the vehicle arrives at its destination. We used this file in an NS-3 simulation which creates a vehicular network based on the distance of the vehicles. The simulation generates a MAC address to each vehicle, and at each simulation steps it connects or disconnects the vehicles. Listing 2 presents an example of the output of an NS-3 simulation.

```
15469 0 -23.5508 -46.6287 00:00:00:00:00:01 0
15469 1 -23.5508 -46.6287 00:00:00:00:00:02 0
15470 0 -23.5508 -46.6287 00:00:00:00:00:01 1
00:00:00:00:00:02
15484 2 -23.5508 -46.6287 00:00:00:00:00:03 0
15485 0 -23.5517 -46.6279 00:00:00:00:00:01 2
00:00:00:00:00:02 00:00:00:00:00:03
```

Listing 2: *NS-3 Output*

Each line in the file corresponds to a vehicle in an instant of the simulation. The attributes are the simulation time, the latitude and longitude, the numbers of cars connected to the vehicle, and the MAC address of all connected cars. From this output files, it is possible to perform different analyses such as the number of connections and disconnections per second and the mean number of connections to the simulated vehicles. For example, Figure 7.5 presents a graph with the mean number of connections per vehicle during 07:00 and 07:02 in the whole city.

[1]NS-3 – https://www.nsnam.org/

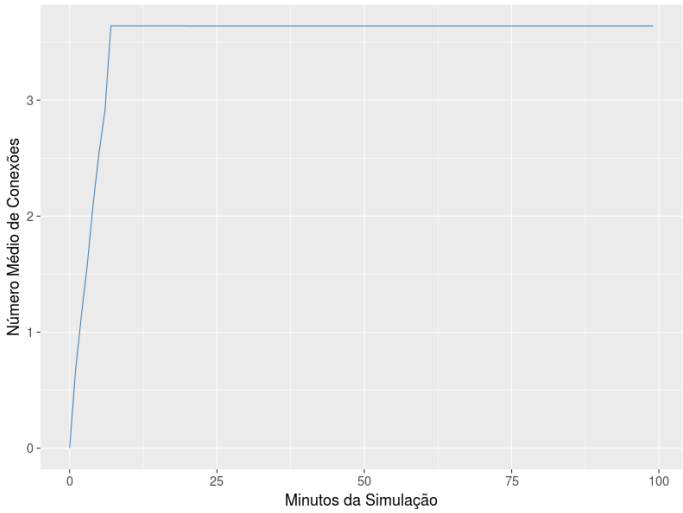

Figure 7.5: *Mean number of connections during the simulation*

7.4 Bus Model Experiments

The InterSCSimulator was used by a study group of a Smart City course to make experiments of a bus mobility model. They created the model based on the real traces of the movement of the buses of São Paulo provided by the transportation secretariat of the city (SpTrans). The original bus mobility model from InterSCSimulator used the planned intervals to create the buses on the simulation and compared the speed of the buses with the general traffic in the simulation.

However, using the real data from the city, collected by the Automatic Vehicle Location (AVL), the group showed that there are great differences in the planned and real intervals of the city buses. With this data, the group extended the bus mobility model with more accurate data, improving the results of the simulator. Moreover, the students compared the simulated and the real travels times and showed that the results were very close to the real system. Figure 7.6 compares the simulated and the real times.

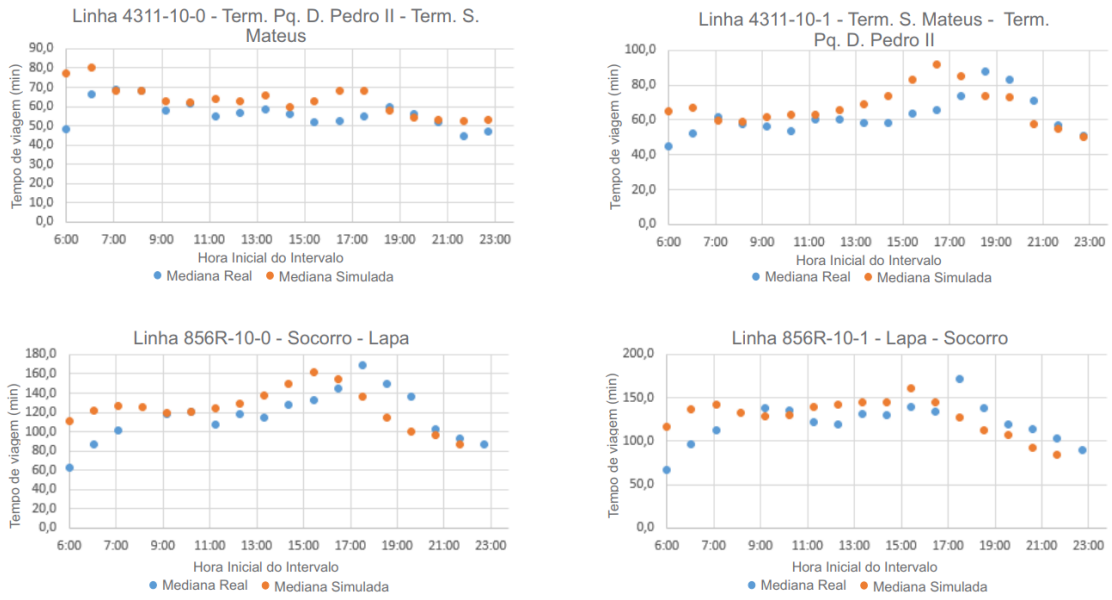

Figure 7.6: *Simulation and Real Time Comparison*

7.5 Digital Rails

Autonomous vehicles (AVs) technology brings new solutions and challenges for urban mobility. The development and adoption of AVs has the potential to reduce traffic jams and increase traffic safety. However, despite the advancement of automation technology in both research and commercial environments, full autonomous vehicles requiring no human intervention are not expected to be available in the short-term.

A work investigated the Digital Rails (DR), a proposal to allow AVs to share the roads with regular vehicles, with minimal changes to the current cities infra-structure. DR consists on dedicated lanes for AVs that allows AV platoons to traverse arterial roads at high speeds and synchronized traffic signals coordinates the traffic with regular vehicles. On roads with DR lanes, traffic signals on successive intersections should be synchronized to allow the platoons to travel without stops.The proposal for Digital Rails was first elaborated by designers at a design consultancy firm called Questtonó[2].

The analyses evaluated the impact that such system could have in traffic using simulations based on the city of São Paulo. The DR simulations expanded the InterSCSimulator implementing a couple of features on the simulator such as the synchronized traffic signals, the exclusive AV lanes, and the AV movement model. Figure 7.7 presents the DR network simulated in the city of São Paulo.

To simulate the impact of the DR in the city, it was created a scenario based on the simulation presented in Chapter 5. Four simulations were executed, the first with 0% of DR vehicles, the second with 25% and the last with 75%. As expected, the average travel time increased when the ratio is 0, because the assignment of a lane for DR decreased the road capacity on the selected arterial ways. With 25% of vehicles able to use DR, the average

[2] https://www.questtono.com/en/

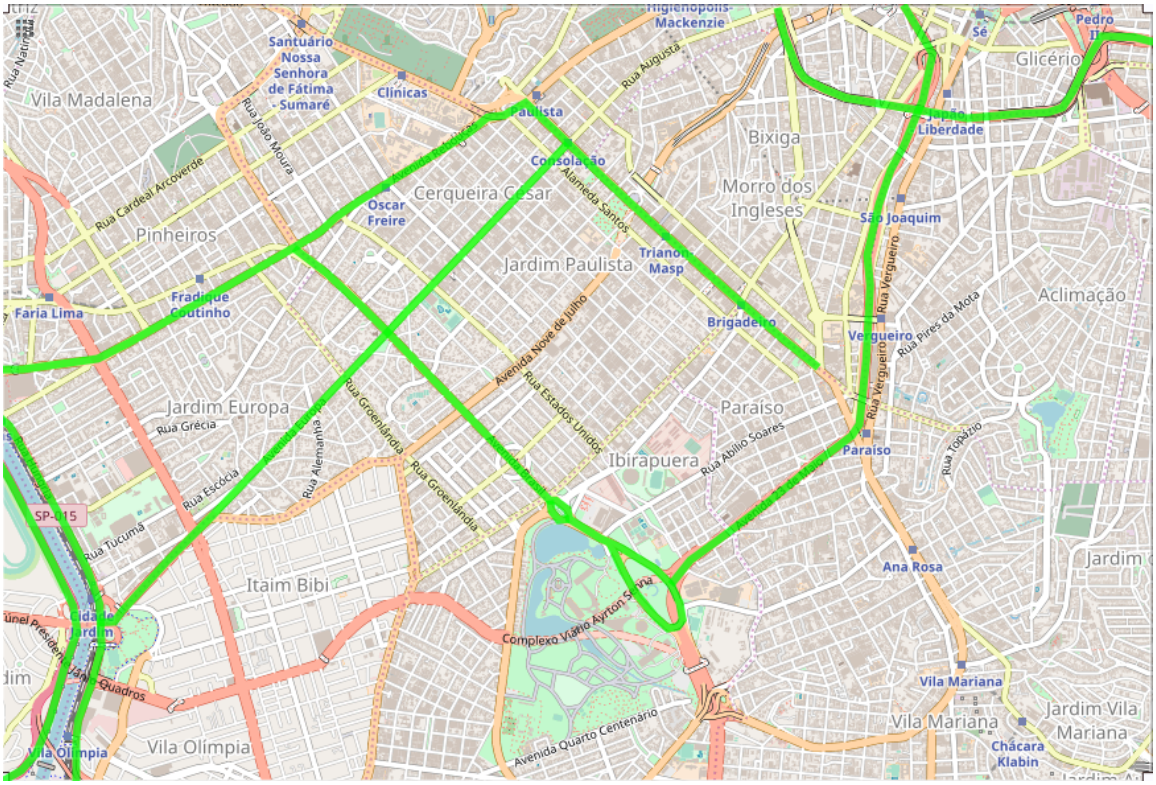

Figure 7.7: *DR network simulated in the city of São Paulo*

travel time were lower or very similar to the benchmark scenario. For ratios greater than 50% of vehicles able to use DR, all average times were lower than the benchmark. With 100% of vehicles able to use DR, the travel times were about 65% of the benchmark.

We also analyze travel times considering only vehicles that are not able to use DR. Figure **??** shows the evolution of travel times for them. With a ratio of 25% of vehicles able to use DR, the average travel time is equal or lower than in the benchmark scenario. For ratios higher than 50%, the average travel time is smaller than in the benchmark scenario. Finally, for 75% of vehicles able to use DR, the average travel time is between 67% and 79% of the benchmark scenario.

Chapter 8

Conclusions, Contributions, Limitations, and Future Work

A Smart City simulator, such as the InterSCSimulator, can be very useful to different city stakeholders. For example, city managers can design and evaluate changes in the city infrastructure and public policies, startups can generate data to test new Smart City applications, and researchers can perform experiments and test hypothesis designing different city scenarios. However, there are still many challenges to the effective use of simulators for Smart Cities such as scalability, lack of appropriate models, and difficult to use state-of-the-art tools.

The research presented in this thesis aimed to provide tools to face some of the challenges of developing a city simulator able to execute large-scale scenarios. The main result of this work is the development of InterSCSimulator, an open-source, scalable, Smart City simulator. This simulator is capable of simulating an entire metropolis such as São Paulo with more than 15 million trips in super real-time. Moreover, the simulator has been used in many technical and scientific projects as showed in Chapter 7.

8.1 Contributions

This work had technical and scientific contributions to the fields of Smart Cities, Large-Scale Simulations, and Software Integration. The simulator was also used for educational purposes such as book chapters and supporting tool for Smart City courses. In the following, we summarize the main outcomes of this thesis.

8.1.1 Technical and Scientific Contributions

Smart City Simulator: We developed the simulator, which is available as an open source software to all the Smart City community. The software already implements many

traffic models and is extensible to new city scenarios providing support for urban planning and test for research projects.

Smart City Platform and Simulator Integration: The InterSCity platform and the InterSCSimulator are integrated. Thus, it is possible to develop and test new applications on the platform using the simulated data and also validate new simulations scenarios with the data stored in the platform.

Experiments Using the Simulator: We showed different works that used the simulation to evaluate their proposals such as the Digital Rails autonomous vehicles, the São Paulo bus movement model, and the scalability experiments of the InterSCity platform.

Large-Scale Experiments: We conducted scalability experiments to evaluate the maximum load of the InterSCSimulator using the Google Compute Engine in the cloud. Our tests showed that the simulator is capable of running in large-scale machines with more than 16 CPUs and 100 GBs of memory.

8.1.2 Educational Contributions

Courses: A group of students used the InterSCSimulator in a Smart City course at University of São Paulo to support the tests of a new bus traffic model. The model is based on real data collected from the city of São Paulo buses.

Capstone Project: An undergraduate student used the simulator to develop a simulation of Digital Rails, a new transportation mode that uses autonomous vehicles and semaphore synchronization. The project extended the simulator to allow the simulation of this new transportation approach.

Master Thesis: InterSCSimulator supported the development of two master theses. In one, the InterSCSimulator was used to generate realistic, large-scale workloads to analyze the scalability of InterSCity platform. In another one, the simulator was used as a study-case to the development of an approach to integrate Smart Cities simulators and platforms to allow tests and experiments in Smart Cities platforms, services, and applications.

Educational Material: The material produced from the bibliographic review of Smart Cities and published as book Chapter was used to at least three Smart Cities courses in universities in Brazil and computer science conferences. The review analyzed the main Smart Cities definitions, initiatives, and software platforms.

8.1.3 Publications

Based on the research presented in this thesis, we published seven scientific papers or book chapters. Three regarding the literature review, two about the InterSCSimulator architecture, and two showing the use of the simulator. The complete list of publications is in the following:

Santana, E.F.Z., Chaves, A.P., Gerosa, M.A., Kon, F. and Milojicic, D. Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture. *ACM Computing Surveys (CSUR)*, 2017.

Kon, F.; Santana, E. F. Z. Cidades Inteligentes: Tecnologias, Aplicações, Iniciativas e Desafios. In: José Carlos Maldonado; José Viterbo; Marcio Eduardo Delamaro; Sabrina Marczak. (Org.). *Jornadas de Atualização em Informática.*, Sociedade Brasileira de Computação, 2016.

Kon, F.; Santana, E. F. Z. Computação aplicada a Cidades Inteligentes: Como dados, serviços e aplicações podem melhorar a qualidade de vida nas cidades. In: Flávia Delicato; Paulo Pires. (Org.). *Jornadas de Atualização em Informática.*, Sociedade Brasileira de Computação, 2017.

Santana, E. F. Z., Batista, D. M., Kon, F. and Milojicic, D. S. SCSimulator: An Open Source, Scalable Smart City Simulator. *Tools Session of the Brazilian Symposium on Computer Networks (SBRC)*, Salvador, Brazil, 2016.

Santana, E. F. Z.; Lago, N.; KON, F.; Milojicic, D. InterSCSimulator: Large-Scale Traffic Simulation in Smart Cities using Erlang. *18th Workshop on Multi-agent-based Simulation (MABS)*, São Paulo, Brazil, 2017.

Santana, E. F. Z., Kanashiro, L., Tomasiello, D., Kon, F., Giannotti, M.: Analyzing Urban Mobility Carbon Footprint with Large-scale, Agent-based Simulation. *International Conference on Smart Cities and Green ICT Systems (SMARTGREENS)*, Funchal, Portugal, 2018.

Santana, E. F. Z., Kanashiro, L., Kon, F. Geração de Rastros de Mobilidade para Experimentos em Redes Veiculares. *II Workshop de Computação Urbana (CoUrb)*, Campos do Jordão, Brasil, 2018.

Besides, three other papers were published using the InterSCSimulator as a support tool.

Esposte, A. M., Santana, E. F. Z., Kanashiro, L., Costa, F. M., Braghetto, K. R., Lago, N., Kon, F. Design and evaluation of a scalable smart city software platform with large-scale simulations. *Future Generation Computer Systems (FGCS)*, 2019.

Wen, M., Rosa, T. , Souza, M. C., Aleixo, R. P., Alves, C., Sá, L., Santana, E. F. Z., Kon, F. Criação de Modelo para Simulação de Movimentação de Ônibus a Partir de Dados Reais. *1º Workshop Brasileiro de Cidades Inteligentes (WBCI)*, Natal, Brasil, 2018.

Covas, G., Santana, E. F. Z., Kon, F. Evaluation Exclusive Lanes for Autonomous Vehicle Platoons. *33rd International ECMS Conference on Modelling and Simulation*, Caserta, Italy, 2019.

8.2 Limitations

The main limitations of this work are on the data used for the creation of the São Paulo simulation. The known problems are: 1) the most recent origin-destination survey

available is almost seven years old, and we are using the current city infrastructure and bus line what can cause differences in the population travel time. 2) It is tough to create the exact bus itinerary from the data because it is challenging to have the precise position of the bus stops in the city graph. 3) The origin-destination survey has just the main modal of the trips, but some trips use more than one travel mode such as buses and subway.

Regarding the scalability experiment, the main problem is that we did not reach the limits of the simulator. The tests showed that the simulator scaled almost linearly until the largest experiment executed. About the simulator architecture, one problem is that we can not implement distributed simulations, what is one of the most important features of Erlang, because of the Erlang ETS tables. One crucial future work is to provide a synchronization algorithm to distributed ETS tables.

8.3 Future Work

There are several potential future works regarding the InterSCSimulator such as improvements in the simulator implementation including new features and changes in its architecture, the creation of new study cases, and the inclusion of new Smart Cities scenarios. Regarding the improvements in the simulator implementation:

Distributed Simulations: The current version of InterSCSimulator does not allow distributed simulations, mainly because of the use of ETS tables that are not distributed across an Erlang network. Therefore, to distribute the InterSCSimulator, it will be necessary to implement a synchronization service that updates the ETS table in all Erlang nodes that are executing the simulator.

Real-Time Visualization: The tool that we are using to visualize the simulations only work after the entire log of simulation is already generated. Would be interesting to develop a tool that can get the data directly from the simulator and create an animated visualization of the simulation.

Simulation of Events: We intend to add the occurrence of events in the city that can change traffic behavior such as rain, accidents, and road closures.

Real-Time Integration with a Network Simulator: We already used data generated in the InterSCSimulator with the NS-3 simulator. However, it is possible to integrate the InterSCSimulator with a network simulator such as OMNET++ executing both simulators concurrently. It allows the creation of traffic scenarios influenced by the vehicle network.

Costs Calculation: Add the calculation of costs to the mobility models. It is possible to calculate the costs associated to cars such as gas and parking and also the costs of public transportation. This can be important to optimize costs of transportation for the citizen and compare the different options of transportation in the city.

Regarding new study cases:

Simulate Other Cities: Although the simulator is generic, all the simulations presented in this work are based on the city of São Paulo. It would be important to test the

InterSCSimulator in other city such as New York or Tokyo. Possible, in those new scenarios new scalability problems or issues in the models used can be found.

Use Different data as Input: The OD survey was used to generate the simulated travels in this work. However, there are many other possible data sources such as GPS location of mobility applications such as Google Maps and Waze and data of telecommunications companies.

Regarding the new Smart Cities scenarios:

Waste Management: Would be interesting to simulate the waste management system of a city, including trash trucks, trash disposals, and trash bins. With this scenario, it is possible to measure the efficiency of the system and test algorithms to create the routes to the trucks.

Health-Care: The InterSCSimulator already allows the simulation of buildings that can be the origin or the destination of trips. Using this idea, it could be possible to simulate hospitals that are the destination of people with health problems. Using this scenario is could be possible to analyze the time that the city inhabitants are taking to go to a hospital and measure the necessity of medical units in different regions of the city.

New Transport Modes: Add new transport modes such as bicycles, taxis, and scholar buses. With these new modes, it is possible to simulate all the trips of the origin-destination survey of São Paulo.

References

[Aslam et al. 2018] Aqsa Aslam, Luis Almeida, and Frederico Santos. "A Flexible TDMA Overlay Protocol for Vehicles Platooning". In: *Communication Technologies for Vehicles* (2018), p. 169 (cit. on p. 25).

[Allan and Farid 2015] Deema Fathi Allan and Amro M Farid. "A benchmark analysis of open source transportation-electrification simulation tools". In: *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*. IEEE. 2015, pp. 1202–1208 (cit. on p. 20).

[Agha 1985] Gul A Agha. *Actors: A model of concurrent computation in distributed systems*. Tech. rep. Massachusetts Institute of Technology - Artificial Intelligente LaboratoryB, 1985 (cit. on pp. 2, 21).

[J. C. Anderson et al. 2010] J Chris Anderson, Jan Lehnardt, and Noah Slater. *CouchDB: the definitive guide*. " O'Reilly Media, Inc.", 2010 (cit. on p. 21).

[AlAwadhi and Scholl 2013] Suha AlAwadhi and Hans Jochen Scholl. "Aspirations and realizations: The smart city of Seattle". In: *System Sciences (HICSS), 2013 46th Hawaii International Conference on*. IEEE. 2013, pp. 1695–1703 (cit. on p. 13).

[Aronis and Sagonas 2017] Stavros Aronis and Konstantinos Sagonas. "The shared-memory interferences of Erlang/OTP built-ins". In: *ERLANG 2017*. Association for Computing Machinery (ACM). 2017 (cit. on p. 47).

[Azevedo et al. 2016] Tiago Azevedo, Paulo JM De Araújo, Rosaldo JF Rossetti, and Ana Paula C Rocha. "JADE, TraSMAPI and SUMO: A tool-chain for simulating traffic light control". In: *arXiv preprint arXiv:1601.08154* (2016) (cit. on p. 25).

[Bai et al. 2018] Yun Bai, Jiajie Li, Tang Li, Lingling Yang, and Chenxi Lyu. "Traffic Signal Coordination for Tramlines with Passive Priority Strategy". In: *Mathematical Problems in Engineering* 2018 (2018) (cit. on p. 26).

[Balać et al. 2018] Milos Balać, Henrik Becker, Francesco Ciari, and Kay W Axhausen. "Modeling competing free-floating carsharing operators: A case study for Zurich, Switzerland". In: *Transportation Research Part C: Emerging Technologies* (2018) (cit. on p. 24).

[Barceló et al. 2010] Jaume Barceló et al. *Fundamentals of traffic simulation*. Vol. 145. Springer, 2010 (cit. on p. 19).

[Bakici et al. 2013] Tuba Bakici, Esteve Almirall, and Jonathan Wareham. "A smart city initiative: the case of Barcelona". In: *Journal of the Knowledge Economy* 4.2 (2013), pp. 135–148 (cit. on p. 13).

[Behrisch et al. 2011] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. "SUMO–simulation of urban mobility: an overview". In: *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind. 2011 (cit. on pp. 20, 24).

[Burghout et al. 2006] Wilco Burghout, Haris N Koutsopoulos, and Ingmar Andreasson. "A discrete-event mesoscopic traffic simulation model for hybrid traffic simulation". In: *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*. IEEE. 2006, pp. 1102–1107 (cit. on p. 26).

[Bischoff and Maciejewski 2016] Joschka Bischoff and Michal Maciejewski. "Simulation of city-wide replacement of private cars with autonomous taxis in Berlin". In: *Procedia computer science* 83 (2016), pp. 237–244 (cit. on p. 24).

[Balmer et al. 2008] Michael Balmer, K Meister, and K Nagel. *Agent-based simulation of travel demand: Structure and computational performance of MATSim-T*. ETH, Eidgenössische Technische Hochschule Zürich, IVT Institut für Verkehrsplanung und Transportsysteme, 2008 (cit. on p. 24).

[Buse et al. 2018] Dominik S Buse et al. "Bridging worlds: Integrating hardware-in-the-loop testing with large-scale VANET simulation". In: *Wireless On-demand Network Systems and Services (WONS), 2018 14th Annual Conference on*. IEEE. 2018, pp. 33–36 (cit. on p. 25).

[Camargo Francesquini 2014] Emilio de Camargo Francesquini. "Dealing with actor runtime environments on hierarchical shared memory multi-core platforms". PhD thesis. University of São Paulo, 2014 (cit. on p. 20).

[Caragliu et al. 2011] Andrea Caragliu, Chiara Del Bo, and Peter Nijkamp. "Smart cities in Europe". In: *Journal of urban technology* 18.2 (2011), pp. 65–82 (cit. on pp. 1, 7, 8, 10).

[Chen et al. 2005] Gilbert Chen, Joel Branch, Michael Pflug, Lijuan Zhu, and Boleslaw Szymanski. "SENSE: a wireless sensor network simulator". In: *Advances in pervasive computing and networking*. Springer, 2005, pp. 249–267 (cit. on p. 29).

[Codeca et al. 2017] Lara Codeca, Raphael Frank, Sébastien Faye, and Thomas Engel. "Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation". In: *IEEE Intelligent Transportation Systems Magazine* 9.2 (2017), pp. 52–63 (cit. on p. 25).

[Dameri 2013] Renata Paola Dameri. "Searching for Smart City definition: a comprehensive proposal". In: *International Journal of Computers & Technology* 11.5 (2013), pp. 2544–2551 (cit. on pp. 7, 8).

[Díaz-Díaz et al. 2017] Raimundo Díaz-Díaz, Luis Muñoz, and Daniel Pérez-González. "The Business Model Evaluation Tool for Smart Cities: Application to SmartSantander Use Cases". In: *Energies* 10.3 (2017). issn: 1996-1073. url: http://www.mdpi.com/1996-1073/10/3/262 (cit. on p. 17).

[Du and Sun 2018] Xiaojing Du and Dongshi Sun. "Research on the arterial coordination control of road intersections in port areas based on vissim simulation". In: *2018 Chinese Control And Decision Conference (CCDC)*. IEEE. 2018, pp. 5744–5747 (cit. on p. 26).

[Fernández-Isabel and Fuentes-Fernández 2015] Alberto Fernández-Isabel and Rubén Fuentes-Fernández. "A Model-Driven Engineering Process for Agent-based Traffic Simulations." In: *SIMULTECH*. 2015, pp. 418–427 (cit. on p. 26).

[Fortes et al. 2014] MZ Fortes et al. "Deployment of smart metering in the Búzios City". In: *Transmission & Distribution Conference and Exposition-Latin America (PES T&D-LA), 2014 IEEE PES*. IEEE. 2014, pp. 1–6 (cit. on pp. 10, 12).

[Fourie 2014] Pieter J Fourie. "Reconstructing bus vehicle trajectories from transit smart-card data". In: *Singapore ETH Centre, Future Cities Laboratory Working paper* 986 (2014) (cit. on p. 24).

[Geske et al. 2010] Martin Geske, Przemyslaw Komarnicki, Martin Stötzer, and Zbigniew A Styczynski. "Modeling and simulation of electric car penetration in the distribution power system—Case study". In: *Modern Electric Power Systems (MEPS), 2010 Proceedings of the International Symposium*. IEEE. 2010, pp. 1–6 (cit. on p. 20).

[Giffinger et al. 2007] Rudolf Giffinger et al. *Smart cities-Ranking of European medium-sized cities*. Tech. rep. Vienna University of Technology, 2007 (cit. on pp. 7–9).

[Garzón and Spalanzani 2018] Mario Garzón and Anne Spalanzani. "An hybrid simulation tool for autonomous cars in very high traffic scenarios". In: *15 th International Conference on Control, Automation, Robotics and Vision (ICARCV 2018)*. 2018 (cit. on p. 25).

[Guan 2012] L Guan. "Smart Steps To A Battery City". In: *Government News* 32.2 (2012), pp. 24–27 (cit. on pp. 7, 8).

[Hall et al. 2000] Peter Hall et al. "Creative cities and economic development". In: *Urban studies* 37.4 (2000), pp. 639–649 (cit. on pp. 8, 9).

[Harrison et al. 2010] Colin Harrison et al. "Foundations for smarter cities". In: *IBM Journal of Research and Development* 54.4 (2010), pp. 1–16 (cit. on pp. 7–9).

[Hernández-Muñoz et al. 2011] José M. Hernández-Muñoz et al. "Smart Cities at the Forefront of the Future Internet". In: vol. 6656. 2011, pp. 447–462. doi: 10.1007/978-3-642-20898-0_32. url: http://dx.doi.org/10.1007/978-3-642-20898-0_32 (cit. on p. 9).

[Hülsmann et al. 2014] Friederike Hülsmann, Regine Gerike, and Matthias Ketzel. "Modelling traffic and air pollution in an integrated approach–the case of Munich". In: *Urban Climate* 10 (2014), pp. 732–744 (cit. on p. 20).

[Horni et al. 2016] Andreas Horni, Kai Nagel, and Kay W Axhausen. "The multi-agent transport simulation MATSim". In: *Ubiquity, London* 9 (2016) (cit. on p. 23).

[Jamil et al. 2014] Shuja Jamil, Anas Basalamah, and Ahmed Lbath. "Crowdsensing traces using bluetooth low energy (BLE) proximity tags". In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*. ACM. 2014, pp. 71–74 (cit. on p. 32).

[Janajreh et al. 2013] Isam Janajreh, Liu Su, and Fathi Alan. "Wind energy assessment: Masdar City case study". In: *Renewable energy* 52 (2013), pp. 8–15 (cit. on p. 10).

[Khan and M. Anderson 2016] Tahmina Khan and Michael Anderson. "Accurately Estimating Origin/Destination Matrices in Situations with Limited Traffic Counts: Case Study Huntsville, AL". In: *International Journal of Traffic and Transportation Engineering* 5.3 (2016), pp. 64–72 (cit. on p. 32).

[Kshetri et al. 2014] Nir Kshetri, Lailani L Alcantara, and Yonghoon Park. "Development of a Smart City and its Adoption and Acceptance: the Case of New Songdo". In: *Communications & Strategies* 96 (2014), p. 113 (cit. on p. 10).

[Karnouskos and De Holanda 2009] Stamatis Karnouskos and Thiago Nass De Holanda. "Simulation of a smart grid city with software agents". In: *Computer Modeling and Simulation, 2009. EMS'09. Third UKSim European Symposium on*. IEEE. 2009, pp. 424–429 (cit. on p. 28).

[Kickhofer et al. 2016] Benjamin Kickhofer, Daniel Hosse, Kai Turnera, and Alejandro Tirachinic. *Creating an open MATSim scenario from open data: The case of Santiago de Chile*. Tech. rep. Technical report, VSP Working Paper 16-02, 2016 (cit. on p. 24).

[Kumar et al. 2017] Y Tarun Kumar, V Mahalakshmi Naidu, and CSRK Prasad. "Evaluation of Different Traffic Models for NAD Junction (Visakhapatnam) Using PTV-VISSIM Simulation Software". In: *Journal of Transportation Systems* 2.2, 3 (2017) (cit. on p. 26).

[Krajzewicz et al. 2012] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. "Recent development and applications of SUMO-Simulation of Ur-

ban MObility". In: *International Journal On Advances in Systems and Measurements* 5.3&4 (2012) (cit. on p. 20).

[Liu and Peng 2013] Pu Liu and Zhenghong Peng. "Smart cities in China". In: *IEEE Computer Society* 16 (2013) (cit. on p. 10).

[Lu et al. 2015] Yang Lu et al. "Simmobility mid-term simulator: A state of the art integrated agent based demand and supply model". In: *94th Annual Meeting of the Transportation Research Board, Washington, DC.* 2015 (cit. on p. 27).

[Manville et al. 2014] Catriona Manville et al. "Mapping smart cities in the EU". In: (2014) (cit. on pp. 10, 17).

[Mehdi et al. 2014] Kamal Mehdi, Massinissa Lounis, Ahcène Bounceur, and Tahar Kechadi. "Cupcarbon: A multi-agent and discrete event wireless sensor network design and simulation tool". In: *Proceedings of the 7th International ICST Conference on Simulation Tools and Techniques.* ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). 2014, pp. 126–131 (cit. on p. 29).

[Merkel 2014] Dirk Merkel. "Docker: lightweight linux containers for consistent development and deployment". In: *Linux Journal* 2014.239 (2014), p. 2 (cit. on p. 61).

[Montero Mercadé et al. 2018] Lídia Montero Mercadé et al. "Barcelona Virtual Mobility Lab: the multimodal transport simulation testbed for emerging mobility concepts evaluation". In: *Tenth International Conference on Adaptive and Self-Adaptive Systems and Applications.* International Academy, Research, and Industry Association (IARIA). 2018, pp. 107–110 (cit. on p. 26).

[March and Ribera-Fumaz 2016] Hug March and Ramon Ribera-Fumaz. "Smart contradictions: The politics of making Barcelona a Self-sufficient city". In: *European Urban and Regional Studies* 23.4 (2016), pp. 816–830 (cit. on p. 18).

[Nazário et al. 2014] Débora Cabral Nazário, Igor Vilas Boas Tromel, Mário Antônio Ribeiro Dantas, and José Leomar Todesco. "Toward assessing quality of context parameters in a ubiquitous assisted environment". In: *Computers and Communication (ISCC), 2014 IEEE Symposium on.* IEEE. 2014, pp. 1–6 (cit. on p. 26).

[Nitti et al. 2017] Michele Nitti, Virginia Pilloni, Daniele Giusto, and Vlad Popescu. "IoT Architecture for a Sustainable Tourism Application in a Smart City Environment". In: *Mobile Information Systems* 2017 (2017) (cit. on p. 11).

[Picone et al. 2012] Marco Picone, Michele Amoretti, and Francesco Zanichelli. "Simulating smart cities with DEUS". In: *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques.* ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). 2012, pp. 172–177 (cit. on p. 26).

[Pereira et al. 2015] Ricardo Lopes Pereira, Pedro Cruz Sousa, Ricardo Barata, André Oliveira, and Geert Monsieur. "CitySDK Tourism API-building value around open data". In: *Journal of Internet Services and Applications* 6.1 (2015), pp. 1–13 (cit. on p. 12).

[Papa et al. 2013] Rocco Papa, Carmela Gargiulo, and Adriana Galderisi. "Towards an urban planners' perspective on smart city". In: *TeMA Journal of Land Use, Mobility and Environment* 6.01 (2013), pp. 5–17 (cit. on p. 9).

[Polley et al. 2004] Jonathan Polley, Dionysus Blazakis, Jonathan McGee, Daniel Rusk, and John S Baras. "ATEMU: a fine-grained sensor network simulator". In: *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*. IEEE. 2004, pp. 145–152 (cit. on p. 29).

[Riebl et al. 2015] Raphael Riebl, Hendrik-Jörn Günther, Christian Facchi, and Lars Wolf. "Artery: Extending Veins for VANET applications". In: *Models and Technologies for Intelligent Transportation Systems (MT-ITS), 2015 International Conference on*. IEEE. 2015, pp. 450–456 (cit. on p. 24).

[Stolfi and Alba 2014] Daniel H Stolfi and Enrique Alba. "Red Swarm: Reducing travel times in smart cities by using bio-inspired algorithms". In: *Applied Soft Computing* 24 (2014), pp. 181–195 (cit. on p. 20).

[Sagaama et al. 2018] Insaf Sagaama, Amine Kchiche, Wassim Trojet, and Farouk Kamoun. "Proposal of More Accurate Energy Model of Electric Vehicle For SUMO". In: *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE. 2018, pp. 464–469 (cit. on p. 25).

[Sanchez et al. 2014] Luis Sanchez et al. "SmartSantander: IoT experimentation over a smart city testbed". In: *Computer Networks* 61 (2014), pp. 217–238 (cit. on pp. 1, 11).

[Sargent 2013] Robert G Sargent. "Verification and validation of simulation models". In: *Journal of simulation* 7.1 (2013), pp. 12–24 (cit. on p. 53).

[Stephenson et al. 2012] Martin Stephenson, Giusy Di Lorenzo, and Pol Mac Aonghusa. "Open Innovation Portal: A collaborative platform for open city data sharing". In: *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*. IEEE. 2012, pp. 522–524 (cit. on p. 14).

[Sider et al. 2014] Timothy MN Sider, Ahsan Alam, William Farrell, Marianne Hatzopoulou, and Naveen Eluru. "Evaluating vehicular emissions with an integrated mesoscopic and microscopic traffic simulation". In: *Canadian Journal of Civil Engineering* 41.10 (2014), pp. 856–868 (cit. on p. 26).

[T. Song et al. 2011] Terence Song et al. "Performance evaluation of integrated smart energy solutions through large-scale simulations". In: *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*. IEEE. 2011, pp. 37–42 (cit. on pp. 21, 22).

[X. Song et al. 2017] Xiao Song et al. "Supporting real-world network-oriented mesoscopic traffic simulation on GPU". In: *Simulation Modelling Practice and Theory* 74 (2017), pp. 46–63. ISSN: 1569-190X (cit. on pp. 25, 27, 32, 35).

[Tasharofi et al. 2013] Samira Tasharofi, Peter Dinges, and Ralph E Johnson. "Why do Scala developers mix the actor model with other concurrency models?" In: *European Conference on Object-Oriented Programming*. Springer. 2013, pp. 302–326 (cit. on p. 21).

[Toscano et al. 2012] Luca Toscano, Gabriele D'Angelo, and Moreno Marzolla. "Parallel discrete event simulation with Erlang". In: *Proceedings of the 1st ACM SIGPLAN workshop on Functional high-performance computing*. ACM. 2012, pp. 83–92 (cit. on p. 21).

[Tettamanti et al. 2018] Tamás Tettamanti, Mátyás Szalai, Sándor Vass, and Viktor Tihanyi. "Vehicle-In-the-Loop Test Environment for Autonomous Driving with Microscopic Traffic Simulation". In: *2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. IEEE. 2018, pp. 1–6 (cit. on p. 25).

[Thornton 2013] Sean Thornton. *Chicago's WindyGrid: Taking Situational Awareness to a New Level*. Mar. 2013. URL: http://datasmart.ash.harvard.edu/news/article/chicagos-windygrid-taking-situational-awareness-to-a-new-level-259 (cit. on p. 13).

[Ursachi and Bordeasu 2014] Andrei Ursachi and Dorin Bordeasu. "Smart grid simulator". In: *International Journal of Civil, Architectural, Structural and Construction Engineering* 8 (2014), pp. 519–522 (cit. on p. 28).

[United Nations 2009] United Nations. *Urban and Rural Areas 2009*. 2009. URL: http://www.un.org/en/development/desa/population/publications/urbanization/urban-rural.shtml (visited on 12/13/2015) (cit. on p. 1).

[Uppoor et al. 2014] Sandesh Uppoor, Oscar Trullols-Cruces, Marco Fiore, and Jose M Barcelo-Ordinas. "Generation and analysis of a large-scale urban vehicular mobility dataset". In: *IEEE Transactions on Mobile Computing* 13.5 (2014), pp. 1061–1075 (cit. on p. 25).

[Vlahogianni et al. 2014] Eleni I Vlahogianni, Kostanstinos Kepaptsoglou, Vassileios Tsetsos, and Matthew G Karlaftis. "Exploiting new sensor technologies for real-time parking prediction in urban areas". In: *Transportation Research Board 93rd Annual Meeting Compendium of Papers*. 2014, pp. 14–1673 (cit. on p. 14).

[Waddell et al. 2003] Paul Waddell et al. "Microsimulation of urban development and location choices: Design and implementation of UrbanSim". In: *Networks and Spatial Economics* 3.1 (2003), pp. 43–67 (cit. on p. 28).

[Washburn et al. 2009] Doug Washburn et al. "Helping CIOs understand "smart city" initiatives". In: *Growth* 17 (2009), p. 2 (cit. on pp. 7–9).

[Wolff et al. 2015] A. Wolff, G. Kortuem, and J. Cavero. "Towards smart city education". In: *2015 Sustainable Internet and ICT for Sustainability (SustainIT)*. 2015, pp. 1–3 (cit. on p. 12).

[Xia and Shao 2005] Liping Xia and Yaping Shao. "Modelling of traffic flow and air pollution emission with application to Hong Kong Island". In: *Environmental Modelling & Software* 20.9 (2005), pp. 1175–1188 (cit. on p. 20).

[Yan et al. 2018] Jiaqing Yan et al. "A Study of Bidirectional Green Wave Control Based on Random Optimal Graphical Method". In: *2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS)*. IEEE. 2018, pp. 1180–1184 (cit. on p. 26).

[Zyrianoff et al. 2017] Ivan Zyrianoff, Fabrizio Borelli, and Carlos Kamienski. "SenSE–Sensor Simulation Environment: Uma ferramenta para geração de tráfego IoT em larga escala". In: (2017) (cit. on p. 29).

[Zhou, Tanvir, et al. 2015] Xuesong Zhou, Shams Tanvir, et al. "Integrating a simplified emission estimation model and mesoscopic dynamic traffic simulator to efficiently evaluate emission impacts of traffic management strategies". In: *Transportation Research Part D: Transport and Environment* 37 (2015), pp. 123–136 (cit. on p. 20).

[Zhou and Taylor 2014] Xuesong Zhou and Jeffrey Taylor. "DTALite: A queue-based mesoscopic traffic simulator for fast model evaluation and calibration". In: *Cogent Engineering* 1.1 (2014), p. 961345 (cit. on p. 27).