Using the InterSCSimulator to Evaluate Systems and Urban Scenarios

Eduardo Felipe Zambom Santana, Fabio Kon

[1]Departamento de Ciência da Computação – Universidade de São Paulo (USP)
São Paulo – SP – Brasil

{efzambom,kon}@ime.usp.br

Abstract. *The InterSCSimulator is a scalable, open-source Smart City simulator. The primary use of this tool is to evaluate complex Smart City scenarios and to test Smart City systems such as software platforms and applications. This simulator was already used in different contexts, such as to generate workload to Smart City platforms experiments, to evaluate the impacts of autonomous vehicles systems, and to assess the impact of a new subway line in São Paulo. This paper presents the main features and use cases of the InterSCSimulator, providing an information source to possible new users and contributors to this project.*

Resumo. *O InterSCSimulator é um simulador de Cidades Inteligentes escalável e de código aberto. O principal uso dessa ferramenta é para avaliar cenários complexos de Cidades Inteligentes e para facilitar os testes de sistemas de Cidades Inteligentes como plataformas e aplicações. Esse simulador já foi utilizado em diferentes contextos como na geração de dados para testes de plataformas de Cidade inteligentes, na avaliação de sistemas de carros autônomos e na avaliação de impact de uma nova linha de metrô na cidade de São Paulo. Esse artigo apresenta os prinicipais requisitos funcionais e casos de uso do InterSCSimulator, sendo uma importante fonte de informação para possíveis novos usuários e contribuidores desse projeto.*

1. Introduction

Performing tests and experiments of Smart Cities systems can be a challenging activity for different reasons. For example, it is hard for a research group to have all the required equipment for large scale experiments such as sensors and actuators. Moreover, some scenarios need more complicated and expensive equipment, such as vehicles and smart meters. The use of simulators that can simulate smart city scenarios and devices can help in the development and experiments of commercial and research applications.

InterSCSimulator is an open-source, large-scale, smart city simulator [Santana et al. 2017][1]. It has been used to simulate different scenarios in the last years, such as smart parking applications, subway lines, and bus models. InterSCSimulator can simulate different entities, such as vehicles, sensors, and subway stations. Also, experiments showed that the simulator could execute very large-scale scenarios with

[1]InterSCSimulator repository - `https://github.com/ezambomsantana/smart_city_model`

more than 18 million entities in one-day simulation [Santana 2019]. All videos, papers, and documentation are available in the InterSCity project site[2].

The simulator is useful in several distributed systems and computer networks contexts, such as vehicular networks, Internet of Things, and large scale distributed applications. This paper presents the implementation of InterSCSimulator. In Section 2, we show the main architectural components of the simulator and describe the features of the InterSCSimulator, such as the map representation and the traffic models. Also, in Section 3, we describe some of the projects that already used the InterSCSimulator. Finally, in Section 4, we present our conclusions and point out future work.

2. InterSCSimulator

The main features of the InterSCSimulator are the representation of the city map, the city train and subway systems, the simulation of sensors, and the mobility models. Also, the architecture of the simulator allows the addition of new smart city entities and models.

2.1. Features

There are two aspects of the InterSCSimulator features, the representation of the city entities, and the models that control the behaviors of the entities. For example, we implemented the city road and subway network using a graph model. The list of the most important features of InterSCSimulator are:

- The **city road network** is a digraph with the city roads modeled as edges and the intersections as nodes. There are many attributes in the streets, such as maximum speed and elevation. Also, it is possible to model special lanes in the graph, such as cycle paths and exclusive bus lanes. The **train and subway network** is also modeled as a graph, the stations are the nodes, and the edges connect the neighboring stations. The main attribute of the edges is the average travel time between the connected stations.
- There are different **vehicles** in the simulator, such as cars, buses, and bikes. All of them have mobility models that calculate the speed of the vehicles in the edges of the city road network.
- The **bus system** includes bus stops and timetables. People can wait for the bus in the stops and enter the buses when one vehicle comes in a link with a stop.
- **People** can travel between different points in the city road network using different transportation modes. Also, people can use the train/subway system when they are in nodes that contain a station.
- The **sensors** can generate data using simple or complex models. Also, it can create data based on events of other entities. For example, a temperature sensor can generate values based on a statistical distribution in time intervals. Also, a parking spot sensor can change its state when a car arrives or leave the spot.
- **Locations** can generate people that travel to other locations. For example, we can model a hospital which generates and attracts travel using a distribution.
- It is possible to define **Traffic Lights** in the road network intersections. For example, those actors can be used to improve traffic models or to test traffic synchronization algorithms.

[2]InterSCSimulator site - `https://interscity.org/software/interscsimulator/`

2.2. Architecture

InterSCSimulator has a three-layer architecture. The first one is the Sim-Diasca, a framework that provides general-purpose simulation features such as a simulation time manager, a load balancer, and a base actor model [Song et al. 2011]. The second layer is composed of the smart city agents developed in this research and also the city infrastructure. The third layer is the scenarios that researchers can create using smart city agents. Figure 1 presents the complete architecture of the InterSCSimulator.

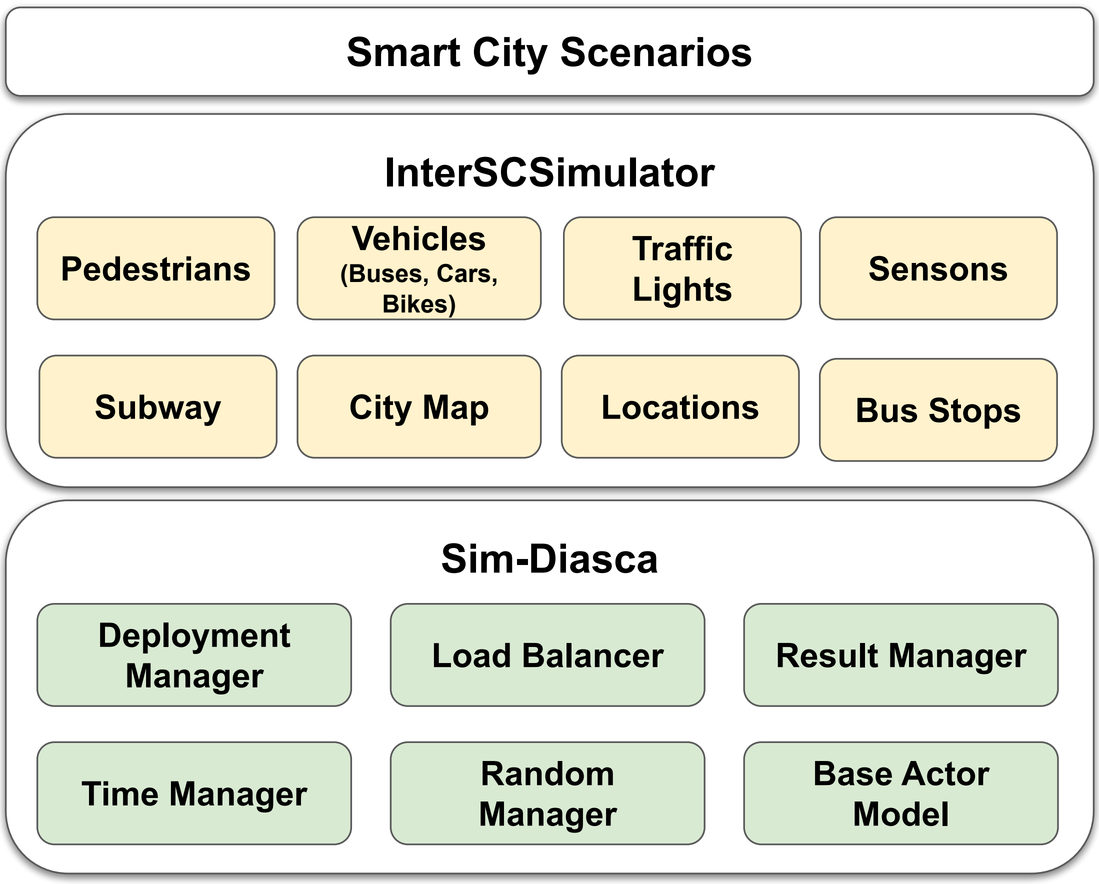

Figure 1. InterSCSimulator Architecture

Figure 2 presents how the InterSCSimulator works. The simulator has a set of input files, some of them required, and other optional. With those inputs, the **Scenario Definition** component reads the files and creates the actors. After, the **Scenario Execution** simulates the events and save a log with all the events in the output file.

The InterSCSimulator input files describe the entities and infrastructure that will be simulated. There are three required files (the first three in Figure 1) and other optional files. The three required files are **config.xml** that describe general configuration such as the simulation time, **map.xml** with the city graph, and **trips.xml** with all the people and travels that must be simulated. Moreover, there are optional files such as the one to describe the subway system, the bus system, traffic signals, and sensors.

In the execution end, the simulator saves an XML or CSV file with all the simulated events. Examples of events are car movement from a link to another one in the city

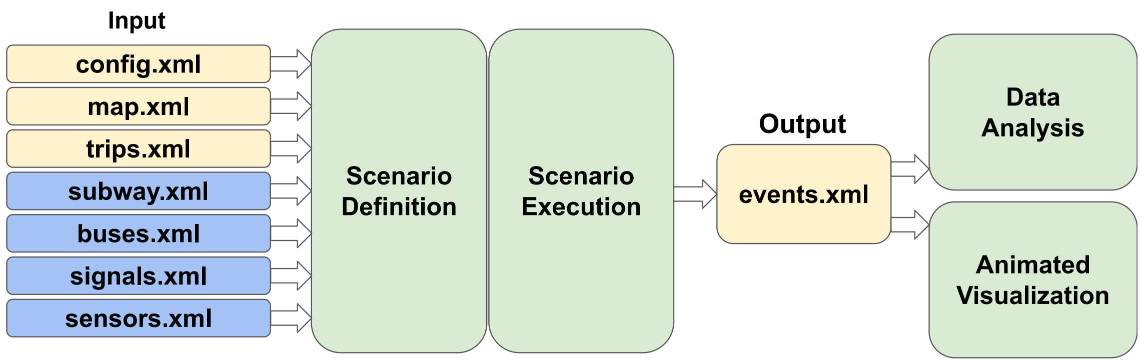

Figure 2. InterSCSimulator Components

graph, data generated by a sensor, or the beginning of one subway travel. All the events have the simulation time, the actor that generated the event, and the location of the event. Some events have other attributes, such as the total distance and time in a final travel event. Then, we can use this file to make an analysis of the data or make an animated visualization of the simulation, such as Figure 3.

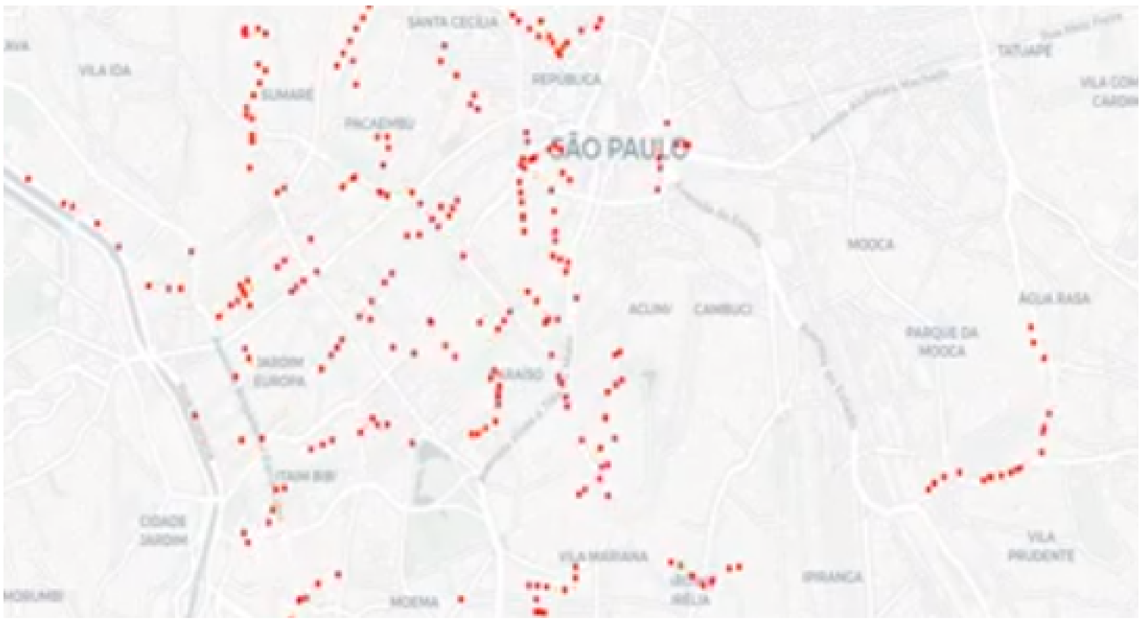

Figure 3. Simulation Visualization

3. Use Cases

The InterSCSimulator was already used in different contexts. For example, we used the simulator to evaluate the scalability of the InterSCity Platform [Del Esposte et al. 2017] and also to simulate a new subway line in the city of São Paulo. All the use cases are based on the São Paulo simulation scenario. To create this scenario, we used the following datasets:

- Origin-Destination (OD) Matrix derived from a survey conducted by the city subway company for the year 2012[3]. This survey contains more than 42 million travels for a day in the city.

- The map of the city based on OpenStreetMap[4]. We created a digraph with more than 50 thousand nodes and 180 thousand links.
- The bus lines and stops of the city provided by the Municipal Transportation Secretary. The city has more than 15 thousand buses, and almost 20 thousand bus stops.
- The subway network of the city provided by the Metrô Company. We created a graph with 89 nodes, all the stations of the city.

3.1. InterSCity platform experiments

The InterSCity platform is an open-source microservices-based platform to enable collaborative research, development, and experiments in smart cities [Del Esposte et al. 2019]. The platform handles requirements to support the development and deployment of integrated smart city services and applications in different domains such as transportation, health-care, and environmental monitoring. We integrated the InterSCSimulator and the InterSCity to perform the platform scalability experiments and also to test smart city applications. The integration is performed in two ways:

- **Application Request:** An agent access an application deployed in the platform, making an HTTP request to a platform service that sends a response with the data requested. The simulator agent must handle the response and change its behavior.
- **City Infrastructure:** The simulator also simulates city sensors that generate data and send it to the platform. This data is sent to the platform using a queue system deployed on the platform.

Figure 4 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 receives 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.

The InterSCity research group tested a Smart Parking application using 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 on the platform.

As it is not easy to perform scalability experiments in real environments, the platform developers 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, based on the São Paulo scenario. Each car made one or more requests to the platform, searching for a parking spot. 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.

[3]Origin-Destination Survey - `http://goo.gl/Te2SX7`

[4]OpenStreetMap - https://www.openstreetmap.org

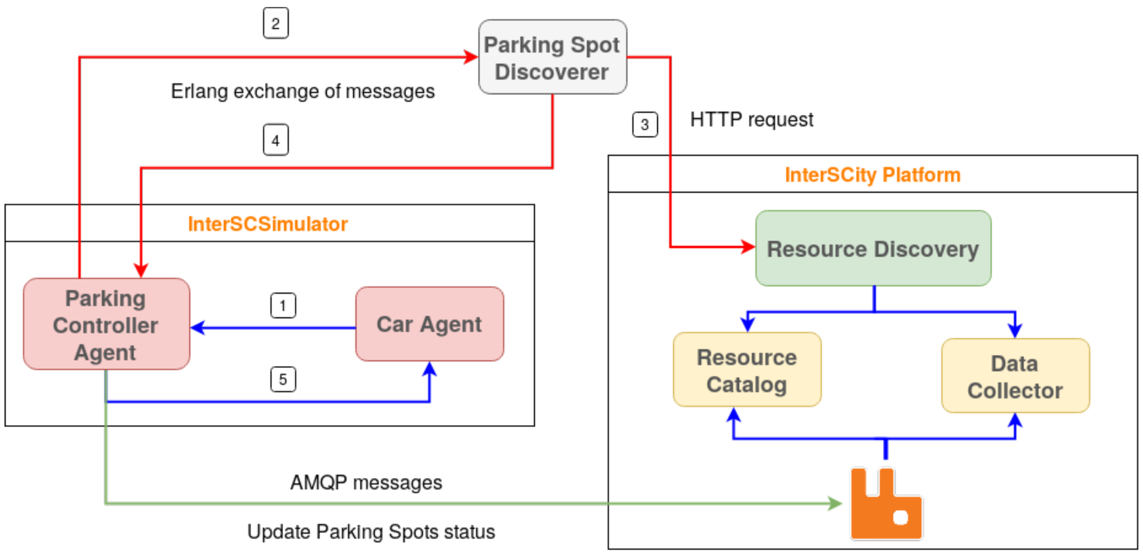

Figure 4. Platform Response Time

3.2. Digital Rails

Autonomous vehicles (AVs) technology brings new solutions and challenges for urban mobility. The development and adoption of AVs have the potential to reduce traffic jams and increase traffic safety. However, despite the advancement of automation technology in both research and commercial environments, fully 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 conventional vehicles, with minimal changes to the current cities' infrastructure. DR consists of dedicated lanes for AVs that allow AV platoons to traverse arterial roads at high speeds, and synchronized traffic signals coordinate 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. Designers first elaborated on the proposal for Digital Rails at a design consultancy firm called Questtonó[5].

The analyses evaluated the impact that such a system could have on traffic using simulations based on the city of São Paulo. The DR simulations expanded the InterSC-Simulator implementing a couple of features on the simulator, such as the synchronized traffic signals, the exclusive AV lanes, and the AV movement model. Figure 5 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 the São Paulo scenario. We executed four simulations, 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 travel time was 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

[5]`https://www.questtono.com/en/`

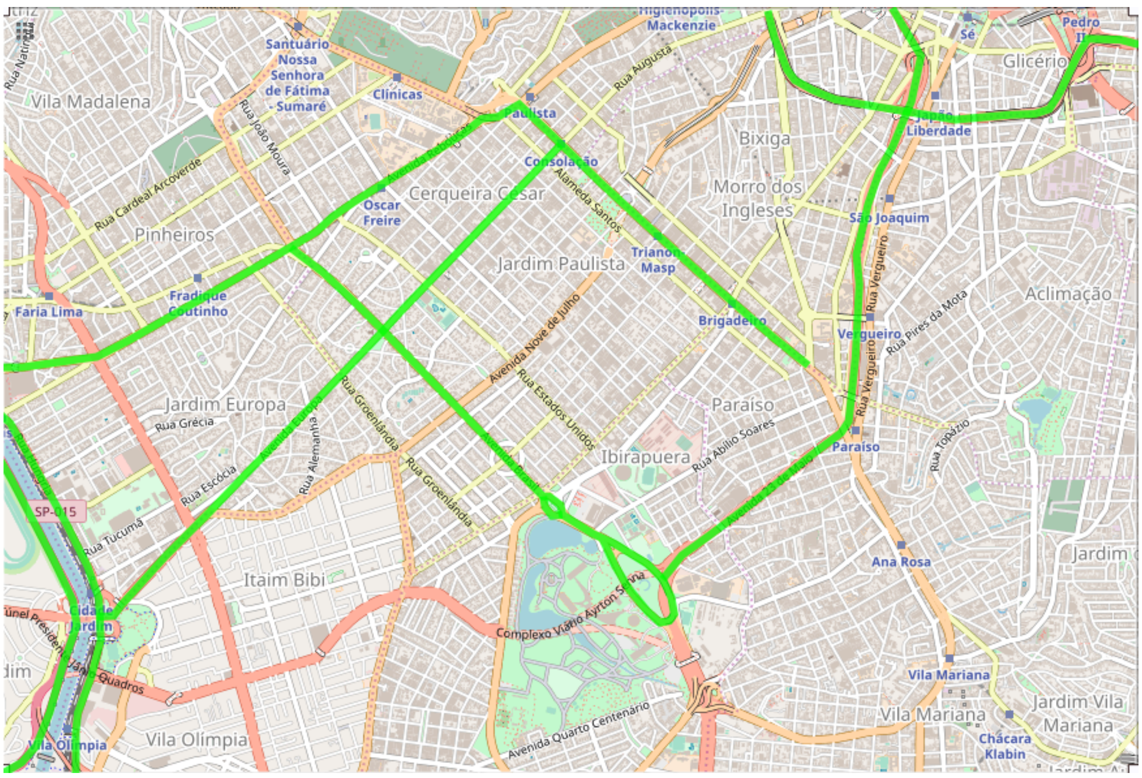
Figure 5. DR network simulated in the city of São Paulo

the benchmark.

We also analyze travel times considering only vehicles that are not able to use DR. 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.

3.3. São Paulo Subway

InterSCSimulator allows the comparison of large-scale mobility scenarios. The Traffic Engineering Group from the Polytechnic School from the University of São Paulo used the simulator to evaluate the impact of a new subway line under construction in the city of São Paulo, especially in the Paraisópolis community, one of the largest underprivileged neighborhood of the city. This subway line will have two stations in the community, and it will have a significant impact on the population's access to quality transportation.

In their work, they examined four simulated scenarios based on the São Paulo scenario 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. They simulated the entire community population (approximately 44 thousand people) and other cars from the city to generate car traffic. In each scenario, people that used car or bus in the original simulation changed the travel mode to the subway when the required walking distance in the destination to a subway station was smaller than 1.5 kilometers.

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.

4. Conclusions and Future Work

InterSCSimulator was already useful for the support of different research, such as the experiments of distributed systems, the simulation of several city scenarios, and to test new traffic models. The simulator is scalable, allowing the test of small or extensive scenarios and is extensible, supporting the development of new scenarios for different research groups. The code of the simulator, all the generated datasets, and supporting tools are open-source.

As future work, we intend to improve the architecture of the simulator to allow distributed execution, it is already possible, but it is not easy to configure. We also aim to create a real-time visualization tool, as the current tool only works with the complete events file. Regarding new scenarios, we plan to develop new actors such as trash bin and trash trucks, taxis, and smart meters.

References

Del Esposte, A. d. M., Santana, E. F., Kanashiro, L., Costa, F. M., Braghetto, K. R., Lago, N., and Kon, F. (2019). Design and evaluation of a scalable smart city software platform with large-scale simulations. *Future Generation Computer Systems*, 93:427–441.

Del Esposte, A. M., Kon, F., Costa, F. M., and Lago, N. (2017). Interscity: A scalable microservice-based open source platform for smart cities. In *SMARTGREENS*, pages 35–46.

Santana, E. F. Z. (2019). *InterSCSimulator: a scalable, open source, smart city simulator*. PhD thesis, Universidade de São Paulo.

Santana, E. F. Z., Lago, N., Kon, F., and Milojicic, D. S. (2017). Interscsimulator: Large-scale traffic simulation in smart cities using erlang. In *International Workshop on Multi-Agent Systems and Agent-Based Simulation*, pages 211–227. Springer.

Song, T., Kaleshi, D., Zhou, R., Boudeville, O., Ma, J.-X., Pelletier, A., and Haddadi, I. (2011). Performance evaluation of integrated smart energy solutions through large-scale simulations. In *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*, pages 37–42. IEEE.