FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A Modelling Methodology Towards Automated Generation of Road Network Digital Twins

Paulo Jorge Salgado Marinho Ribeiro

Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Rosaldo J. F. Rossetti

September 15, 2023

A Modelling Methodology Towards Automated Generation of Road Network Digital Twins

Paulo Jorge Salgado Marinho Ribeiro

Mestrado em Engenharia Informática e Computação

September 15, 2023

Resumo

O excesso de veículos nas grandes cidades é um problema que se tem agravado nas últimas décadas devido à crescente concentração de pessoas no espaço urbano. Como resultado, o congestionamento nas redes de estradas tornou-se cada vez mais frequente, o que constitui um dos principais desafios que os Sistemas de Transporte Inteligentes (ITS) procuram endereçar nas grandes áreas metropolitanas. Para além de prejudicar a atividade económica, o congestionamento do tráfego afeta principalmente a saúde, em resultado da poluição, provocando, ainda, um elevado número de acidentes. A Via de Cintura Interna (VCI), uma via rápida no Porto, Portugal, apresenta estes problemas.

Embora ainda em regime experimental, a modernização dos sistemas de controlo de tráfego, na área dos ITS, tem trazido soluções inovadoras para a sua resolução. A aplicação de Digital Twins para as redes rodoviárias constitui uma delas. Esta nova tecnologia beneficia a gestão, planeamento e monitorização do trânsito, através de uma versão digital da rede rodoviária atualizada em tempo real, contribuindo para a melhoria da tomada de decisões na sua coordenação.

Na literatura, existem já diversos cenários de simulações que constituem réplicas digitais de várias redes de estradas ao redor do mundo. No entanto, há um fator comum entre estes produtos que atrasa a criação de Digital Twins para novas redes de estradas: todos constituem produtos *ad-hoc*, desenvolvidos especificamente para a rede abordada, com foco nas suas particularidades.

Com este trabalho, pretende-se desenvolver uma framework de geração automática de Digital Twins, recorrendo ao simulador microscópico SUMO, que empregue uma metodologia generalizada de tal forma a ser aplicável a qualquer rede de estradas. Tal ferramenta desbloquearia completamente a monitorização das redes urbanas através do controlo inteligente de tráfego, acelerando significativamente o processo de desenvolvimento desta nova tecnologia. Assim, este estudo visa fomentar a utilização prática de Digital Twins em ambiente de mobilidade automóvel, uma vez que a sua geração automatizada permite aos investigadores direcionar a sua atenção para possíveis extensões sobre a ferramenta base produzida, tais como a exploração de cenários *what-if* ou a previsão de tráfego.

Esta dissertação foca-se, assim, na análise de técnicas de replicação digital de circulação rodoviária, procurando desenvolver uma que considere apenas os aspetos fundamentais das redes. Para isto, será também necessário ultrapassar algumas barreiras relacionadas com a cobertura limitada dos detetores fixos na rede, uma vez que nem todas as regiões estão providas de sensores, desconhecendo-se o seu fluxo.

Para além disto, este trabalho envolve a aplicação da framework a dois cenários: à autoestrada de Genebra, na Suíça, e a parte da rede da VCI, em Portugal. A implementação e validação desta ferramenta será realizada através do uso de dados reais de circulação automóvel, provenientes de Detetores de Loop Indutivo, no primeiro caso, fornecidos pela Open Data Platform Mobility Switzerland (ODPMS), e no segundo, pela empresa ARMIS.

Concluindo, este projeto contribuirá para o domínio da gestão do tráfego, procurando aperfeiçoar as aplicações desta área com a finalidade de promover uma maior fluidez do mesmo.

Abstract

The excess of vehicles in big cities is an issue that has worsened in recent decades due to the increasing concentration of people in urban spaces. As a result, congestion on road networks has become progressively frequent, constituting one of the main challenges addressed by Intelligent Transport Systems (ITS) in large metropolitan areas. In addition to harming economic activity, traffic congestion mainly affects public health due to pollution and causes numerous accidents. The Via de Cintura Interna (VCI), a major highway in Porto, Portugal, presents these problems.

Although still experimental, the modernisation of ITS traffic control systems has brought innovative solutions to address these issues. The application of Digital Twins for road networks is one such solution. This emerging technology benefits traffic management, planning, and monitoring through a digital version of the road network, updated in real-time, improving decision-making in its coordination.

In the literature, there are already several simulation scenarios that represent digital replicas of diverse road networks around the world. However, a common factor among these products hinders the creation of Digital Twins for new road networks: they are all ad-hoc solutions developed specifically for a particular network, focusing on its peculiarities.

This work aims to develop a framework for the automatic building of Digital Twins, relying on the microscopic simulator SUMO, which employs a generalised methodology applicable to any road network. Such a tool would ultimately unlock the monitoring of urban networks through intelligent traffic control, significantly expediting the development process of this new technology. Thus, this study aims to encourage the practical use of Digital Twins within automotive mobility, as its automated generation provides researchers with an extendable baseline toolset and enables ITS engineers to focus on applications such as what-if scenarios or traffic forecasting.

This dissertation thereby focuses on analysing techniques for digital replication of road traffic, seeking to develop one that considers only the fundamental aspects of networks. For this, it will also be necessary to overcome some barriers related to the limited coverage of fixed detectors in the network since not all locations are equipped with sensors, making their flow unknown.

Additionally, this work involves the application of the framework to two scenarios: the Geneva motorway in Switzerland and a segment of the VCI network in Portugal. The implementation and validation of this tool will use real traffic data from Inductive Loop Detectors. In the first use case, the data originates from the Open Data Platform Mobility Switzerland (ODPMS), while the company ARMIS provides it for the second.

In conclusion, this project will contribute to the field of traffic management, seeking to enhance applications in this area, promoting smoother traffic flows.

Agradecimentos

Gostaria de agradecer a todos os que contribuíram de alguma forma para o sucesso desta Dissertação de Mestrado. Deixo os meus sinceros agradecimentos:

À empresa ARMIS, que me acolheu durante este último semestre, tendo-me proporcionado momentos inesquecíveis junto das pessoas que a compõem.

Ao Eng. José Macedo, pela oportunidade de embarcar neste projeto.

Ao meu orientador, Prof. Rosaldo Rossetti, por toda a confiança depositada em mim.

Ao Eng. Krešimir Kušić, autor do trabalho que serviu de inspiração para esta dissertação, cuja ajuda ao longo deste percurso foi inestimável.

Aos amigos que me acompanharam ao longo do curso, pelo companheirismo que foi uma fonte de motivação e apoio que não esquecerei.

Por último, mas definitivamente não menos importante, tenho uma dívida imensurável de gratidão para com a minha família. O seu apoio contínuo, incentivo e crença em mim têm sido os meus pilares de força.

Obrigado a todos por serem parte integrante desta jornada. Sem o vosso apoio, esta dissertação não teria sido concretizada.

Paulo Ribeiro
("Paulinho")

"Porque eu sou do tamanho do que vejo
E não do tamanho da minha altura..."

Fernando Pessoa

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations and Symbols

DT	Digital Twin
ITS	Intelligent Transport Systems
ILD	Inductive Loop Detector
OSM	OpenStreetMap
SUMO	Simulation of Urban MObility
VCI	Via de Cintura Interna
DFC	Dynamic Flow Calibrator
RREF	Reduced Row-Echelon Form
ODPMS	Open Data Platform Mobility Switzerland
WIS	What-If Simulations
JIT	Just-In-Time
POI	Point Of Interest
FD	Fundamental Diagram
CAV	Connected and Autonomous Vehicle
VANET	Vehicular Ad-hoc Network
AS	Actual System
CPS	Cyber-Physical System
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
IoT	Internet of Things

Chapter 1

Introduction

In the last decades, urban mobility has been facing increasing challenges, such as traffic congestion and accidents, which accompany the development of urban scenarios. For example, since the movement of people and goods between cities mainly depends on the road network, problems within the traffic system significantly impact practically all areas of economic activity. Moreover, one of the causes of the current environmental crisis is the use of fossil fuels by conventional automobiles with internal combustion engines, which release gases like carbon dioxide and hydrocarbons. In addition to the damage to the physical health of citizens resulting from pollution, the stress travellers are exposed to also deteriorates their mental health. These are just a few examples of damage caused by traffic congestion, which has been worsening over time.

The Via de Cintura Interna (VCI) highway in Porto, Portugal, is an excellent example of several such traffic problems. As it gives access to most of the urban arteries of Porto and Vila Nova de Gaia, there is a large influx of vehicles on this highway. It thus presents congestion problems, aggravated at peak hours due to people commuting from home to workplaces (primarily in city centres) and vice versa.

All these problems stem from the inefficient management of transport flows. As the city population grows, the vehicle-to-population ratio grows accordingly, exacerbating the situation. As transportation difficulties rise, modernising and digitising transportation through intelligent technologies becomes a priority development area, which led to the emergence of Intelligent Transport Systems (ITS). ITS refers to smart systems using innovative developments in modelling transport systems and regulating traffic flows. It provides end users with more excellent information content and safety, as well as qualitatively increasing the level of interaction between road users compared to conventional transport systems.

A new concept emerges to ensure the effectiveness of the ITS: the digital twin. A Digital Twin (DT) aims to reflect the performance of the real-world scenario by simulating a virtual space. It is a new concept, including creating a new virtual entity in cyberspace, imitating the original physical entities in most aspects, and monitoring, analysing, testing, and optimising it.

This work was carried out in collaboration with ARMIS, a technological solutions company, more precisely with its ITS module. ARMIS developed the platform *Next*, intended for monitor-

ing, managing and planning urban networks through intelligent traffic control, which applies the concept of artificial transport systems.

The initial purpose of this dissertation was to build a DT for the VCI network. However, related literature research led to a new idea capable of solving one of the biggest problems of DTs: the need for a standardised modelling methodology.

Hereupon, this work describes the development of a generalised methodology for building DTs applicable to any road network. The constructed tool was applied and examined in two road networks to demonstrate its robustness. One of those use cases concerns part of the VCI highway, thus contributing to the original purpose of the dissertation.

1.1 Problem Statement

As mentioned before, this work's initial goal was to create a DT of the VCI highway, aimed at reflecting the performance of this road through the simulation of a virtual space, thus allowing its monitoring, analysis, testing and optimization.

ARMIS would supply data from inductive-loop detectors (ILDs) installed beneath the VCI pavement to calibrate the digital model. The aim was to achieve the greatest possible reliability, ensuring that the digital model accurately represents reality.

This task becomes challenging owing to one of the most severe limitations of stationary detectors like ILDs: their limited network coverage. Information is scarce or poor in locations without sensors, making replicating the entire VCI network digitally challenging.

In addition, building a realistic traffic simulation scenario is a time-consuming task. It takes much effort to collect and adapt all the necessary data to replicate the actual behaviour of a given road network. Subsequently, vehicle demand must be converted or generated from the collected information. Then, the data regarding vehicle passages must be imported into the simulation system's architecture to allow model calibration and validation. In addition to these data-related processes, it is also often necessary to make corrections and modifications to the digital representation of the network so that the simulation model can use them. Furthermore, road-related structures such as traffic lights or pedestrian crossings must be appropriately represented and embedded in the scenario.

Each instance of constructing a DT for a new road network requires following this exhaustive process. Building a new DT for the VCI highway would yield a specialized solution suitable to that road network. While this outcome possesses inherent value, it does not facilitate the development of DTs for other networks. Like many others in the literature, it would be just another *ad-hoc* tool developed particularly for a specific road network.

This reasoning led to the emergence of a more relevant research idea, given its innovative and valuable character: the development of a tool adaptable to any road network. More specifically, a tool such that, for a given network and data from road deployed ILDs, automatically generates a DT for that network.

Naturally, this brings new challenges. Such a tool should only be based on the common aspects of all road networks, being as general as possible.

Therefore, this dissertation describes the implementation of a DT construction tool, explaining its logic and the strategies for overcoming several barriers. The ultimate objective of the developed tool is to ensure its applicability to any road network. In order to verify its robustness, it is crucial to subject it to rigorous testing using two scenarios: the VCI highway and the Geneva motorway in Switzerland. This process generated dependable DTs tailored to both networks.

1.2 Motivation

Typically, conventional simulations are employed for medium-term or long-term forecasting purposes, making them well-suited for situations where time sensitivity is not a critical factor in decision-making [35]. However, this type of simulation may not be optimal when an operational decision must be taken quickly, and a fast simulation is necessary to address the real-time dynamics of a physical system.

For this purpose, a symbiotic simulation system can provide an efficient and effective means of achieving short-term forecasting. It consists of a real-time simulation representing a DT running synchronously with a physical system.

Such a real-time simulation, often termed a "base" or "reference" simulation, is commonly employed within symbiotic simulation [4, 32]. It integrates sensor measurements into the base simulation, enhancing precision and reliability in the generated results.

A DT combines the collected data and precisely recorded city signs to get new insights into urban traffic from several perspectives, such as road supply and demand. The knowledge gained allows for optimising the road network structure through traffic simulation and enhancing overall traffic efficiency in the city. Additionally, integrating DT in intelligent transportation helps increase decision-making execution, safety, and vehicle stability beyond expediting smart and safe driving.

On the one hand, this dissertation benefits ARMIS, providing them with a convenient tool for building new DTs, which expedites their development process. Accordingly, the company can complement the *Next* platform with DTs for new road scenarios. Integrating this new technology makes the system more reliable, reducing discrepancies between simulated and real data.

On the other hand, it is highly valuable for the ITS field and for spreading the notion of DTs in this industry, as this term is still in its infancy. Analysis of the available literature indicates that this technology is still insufficiently studied. There is not enough information regarding existing models of ITS services in the literature, making it challenging to develop and implement specific models of transportation infrastructure together with appropriate information systems.

Furthermore, the resulting framework will significantly accelerate the construction of DTs applied to road networks due to its automatic and generalised character.

The spared time by the automatic preparation of a base simulation can be leveraged to develop additional functionalities to augment its capabilities. One such example is the implementation

of what-if simulations (WIS). A base simulation that accurately represents the physical system's current state can help speed up the Just-in-time (JIT) simulation process by generating multiple instances of the base simulation model to conduct various JIT what-if simulations.

Employing WIS enables exploring potential alternative scenarios, facilitating well-informed and timely decision-making. The continuous updating of the base simulation model with real-time data ensures that WIS instances originated from the base simulation just before short-term forecasting yield superior accuracy compared to conventional simulations.

One can achieve greater efficiency by enabling and facilitating such base simulation since the JIT simulation process depends on organising pertinent current data, calibrating the simulation model with those data, and other considerations. When the simulation model becomes ready, it may be too late for JIT decision-making.

Therefore, this dissertation's outcome significantly advances the development of DTs, allowing researchers to shift their focus towards other challenges or ideas ahead of this process.

1.3 Research Questions

Two main research questions guided the research of this dissertation:

- Can a standardised methodology be derived to generate DTs for any road network?
- Can such a tool be applied realistically to a known scenario to help extract relevant information, improving traffic flows?

This work seeks to answer these questions by fulfilling the objectives in the following section.

1.4 Goals

As discussed in the previous sections, ITS are a pivotal technology for improving traffic flows since it effectively addresses road congestion issues. Due to their potential, DTs constitute a significant area of interest in this field.

The primary objective of this dissertation was to develop a framework for building DTs applicable to all road networks. Specifically, the framework should automatically generate a trustworthy DT that reliably simulates the actual traffic behaviour given a representation of the roads and structures of a specific network and the locations and data of the ILDs present in that network.

Moreover, two use cases were analysed to test the robustness of the developed framework. One of them was a motorway in Geneva, Switzerland, the scenario used in the work that describes the model used by this framework [27]. The other was part of the Via de Cintura Interna (VCI) network in Porto, Portugal, a valuable scenario to ARMIS as it may be integrated into its *Next* platform. The descriptive model utilised data from ILDs belonging to the road infrastructure, which this company provided.

In order to achieve the goals indicated above, this dissertation addressed the following objectives:

- Study of the theme's relevant topics and presentation of its state-of-the-art;
- Scrutiny of the type of data generated by ILDs and the related information that can be derived;
- Development of the DT-building framework using the microscopic traffic simulator SUMO;
- Model validation through comparison with sensor data in two different scenarios.

The expected contributions of this dissertation include a thorough analysis of the literature on the subject at hand and a framework for automatedly building DTs applicable to any road network. Furthermore, the DTs of two networks, namely the Geneva motorway and the *Nó de Coimbrões* node of the VCI highway, were generated by applying the framework to these networks and relying on ILDs-generated data.

1.5 Document Structure

In addition to the Introduction, this dissertation contains five more chapters.

The second chapter, Literature Review, exhibits the state of the art of the relevant topics, summarising related articles. It consists of two central sections. The first introduces the fundamental concepts of the investigation. The second section, Related Work, explores previous works relevant to this dissertation. It describes various approaches to building DTs using different techniques and tools while pointing out the advantages and disadvantages of each.

The third chapter, Problem Specification, clarifies the problem at hand, setting the stage for the next chapter, which presents the problem's solution. The section Baseline Methodology details the methodology that served as the basis for developing the DT building framework. Section Problem Formalisation covers the precise definition of the problem, seeking to express it mathematically.

The fourth chapter, Proposed Framework, is the central chapter of this dissertation since it describes the entire implementation of the DT building framework. It describes the modifications implemented in the baseline methodology to enhance its adaptability, ensuring that the framework remains versatile enough to cater to any road network.

Chapter five, titled Results and Analysis, comprehensively examines the results derived from the developed framework. This chapter contains two sections that analyse each specific use case: the Geneva motorway use case and the VCI network use case.

Finally, the last chapter, Conclusions, overviews the document's topics and presents the main conclusions. Then, it sets out its Main Contributions and establishes some Further Improvements that could be made over the current implementation of the framework. The document ends with the idealisation of some Future Work unlocked by this project, aiming to guide forthcoming investigations.

Chapter 2

Literature Review

This chapter aims to provide an advanced and up-to-date picture of the state-of-the-art on the related topics. It is composed of three main sections.

The first section, Intelligent Transport Systems, is organised thematically according to the main concepts related to this field. It summarises the main types of traffic Data Sources, indicating the associated problems and advantages and an in-depth description of the sensors explored in this work, the Inductive Loop Detectors. Next, the second section provides a comprehensive overview of the current knowledge on Digital Twins. It presents their enabling technologies and some examples of their applications. This section ends by presenting a generalised architecture of this new technology and its main challenges. Finally, it ends by introducing the Fundamental Diagrams, which are essential elements for the representation and analysis of traffic.

The third and last section, Related Work, provides a comprehensive overview of prior research pertinent to the dissertation topic. More specifically, several techniques for developing DTs will be presented, supplemented by articles that have implemented them through specific use cases.

Ultimately, a comparative analysis will be conducted among these works, highlighting the need for a universal methodology for developing these tools. This analysis will demonstrate that the implementation of DTs is typically tailored to the specific characteristics of the given road network, resulting in ad-hoc tools impairing broader applicability.

2.1 Intelligent Transport Systems

The efficient functioning of transportation systems is pivotal in the modern world, shaping the accessibility, connectivity, and productivity of regions and nations alike. In an era of mobility and globalisation, the quality of transport infrastructure and services holds substantial implications for economic development, environmental sustainability, and societal well-being. Acknowledging this critical role, delving into the multifaceted challenges that confront transport systems today is imperative.

The work in [42] categorised the main issues concerning transport systems' operation as objective or subjective:

Objective problems:

- increasing vehicle-to-population ratio;
- intensified use of individual transport;
- decreased efficiency of urban passenger transport;
- increased need for city residents to move;
- disproportion between the vehicle-to-population ratio and the pace of road construction;
- problems of urban planning and urban area development.

Subjective problems:

- imperfect system for organising and managing the development of the road transport complex;
- insufficient legislative base at the local and regional level for managing the transport system of the city region;
- insufficient information component when making management decisions;
- insufficient funding for the development of road networks and transport infrastructure;
- unresolved property issues and issues of delineation of property rights and management of transport infrastructure facilities;
- negative impact of the human factor.

Intelligent Transport Systems have made it possible to improve the transportation network and address its primary issues successfully. The main obstacle to introducing ITS is the underdeveloped infrastructure. The presence of sensors, surveillance cameras, IoT and other infrastructure on the road transport network determines the possibility of implementing the system. The many components of smart cities must be intelligently regulated, utilising data provided by physical assets. Thus, one can broadly define ITS as the continuous and quick collection of information regarding roadway traffic conditions using telematic equipment, encompassing the storage, processing, validation and analysis of measured data. It also enables short-term traffic forecasting (in the next 15 minutes or the following day) by utilising real-time sensor data, as intelligence in the transportation industry is quickly expanding. Machine learning (ML) and deep learning (DL) techniques are employed to develop ITS due to the large volume of data produced by transportation systems.

These characteristics make these systems helpful in many applications, from traffic light regulation to delivering immediate replies to police in case of a traffic accident or other unexpected scenario.

Hence, the ITS includes all city services, including traffic police, ambulance, fire departments and other services. In general, ITS helps in solving the following tasks:

- optimisation of the distribution of traffic flows in the network in time and space;
- increasing the capacity of the existing transport network;
- providing travel priorities for a specific type of transport;
- transport management in the event of accidents, catastrophes or measures that affect the movement of vehicles;

- improving road safety, which leads to an increase in traffic capacity;
- reducing the negative environmental impact of transport;
- provision of information on the state of the road to all interested parties.

Having defined what ITS means, it is now necessary to specify its main components and participants:

- transport infrastructure;
- service and software infrastructure;
- vehicles;
- telematic equipment of transport infrastructure elements and automobiles;
- intelligent information boards, road signs and traffic lights with the ability to remotely control them;
- centres for collecting and processing information;
- decision-making and traffic management centres.

One can define three main functions of ITS. The first consists of transport modelling through Digital Twins, a concept explored in the following subsection. The second is concerned with optimising the duration of traffic lights, aiming for the light signal scheme with the shortest transit time. Finally, the third consists of monitoring, planning and managing transport networks. This study focuses on the first function, comparing different ways of modelling road traffic.

An ITS implementation in Chelyabinsk, Russia, mentioned in the article above, illustrates the benefits of this technology at various levels. This ITS can assist authorities in implementing legislation in addition to their practical duties of enhancing road conditions. Some countries require documentation of the projects for developing them on the roads, such as the Russian Federation (REF). It is practical to save all traffic management projects virtually so that the documents are promptly available by just hovering the mouse over the street on the digital map. This way, construction work and the inspection of all technological traffic control devices become easier. For example, since the position of all communications is already in the system, installing a traffic light does not require contacting all local authorities (gas service, administration of electrical networks, and communication providers). Aided by cutting-edge analytical tools, this system can display *heat maps* of a city that depict the concentration of people in various regions or the primary accident locations. Another feature is to show the townspeople's direct correspondence, which provides insight into the townspeople's origins and destinations. A mathematical model of the city's transport and passenger flows creates correspondences based on factual data, making it possible to forecast potential urban transport infrastructure development scenarios.

Transport control centres aim to ensure regular traffic on the road network. These centres used to tackle traffic management issues but now require upgrading through ITS. Among the main functions of these centres are the organisation of an efficient traffic flow and the monitoring of road networks. They are also in charge of notifying participants about alternate routes and coordinating forces during emergencies. These processes provide control over eradicating the consequences

of these scenarios on the road network. Every metropolis has a traffic control centre, where the operators are primarily responsible for making management decisions. The traffic management system comprises many systems for reporting traffic infractions, incident detection, vehicle counters and classifiers. The Transport Control Center receives all of this information in real time, enabling it to get updates on the state and accessibility of urban transportation. Modernisation of the centres is required, first and foremost, due to a rise in the information load on operators, which may result in an emergency scenario. A human working in such a facility is a potential source of errors since their response time is substantially slower than the system's. Furthermore, the volume of equipment monitoring data has risen due to the growth of sensor and computing technology, causing challenges in real-time status evaluation and prediction under traditional modelling conditions. The urge to automate operator tasks rises, leading to ITS implementation. Ideally, the entire system should operate autonomously without human involvement, with employees solely involved in system setup and monitoring. Therefore, with the advent of ITS, the functioning of transport control centres changed qualitatively: operators perform solely a controlling function and influence the adoption of strategic choices on transport network modernisation and development based on analytical projections. Experts in troubleshooting and setting up such a system have become fundamental.

2.1.1 Data Sources

The DT is a data-driven analytic system. It incorporates physical models and historical data to mimic real-world product performance in a virtual space. As such, it must perceive the current state of the physical entity by collecting the surrounding environmental information and communicating it in real time to the virtual model system. The essence of the concept raises the bar for precise detection and efficient transmission of various parameters.

When applied to highways, the central task is to model the real-world road system as accurately as possible throughout its life cycle while organically integrating roads with other components in the transportation system. This activity enables high-quality estimations of the system's performance during its lifespan. These unique properties make the collected data the most valuable asset of this technology, demanding a comprehensive review of the various methods for obtaining it.

The definition of the network state depends on the nature of the data source. The work in [5] grouped available data into three broad types:

- **Static observations:** real-time information obtained in fixed locations in the network, e.g. from induction loops or surveillance systems;
- **Route observations:** usually obtained from GPS-enabled vehicles, smartphones or traffic operators;
- **Global observations:** usually obtained from a secondary source.

The first group frequently comprises metrics such as flow, density, volume, average velocity, occupancy or queue length at specific points in the network. It is one of the primary sources of

real-time information. However, the main limitation of this type of data is the lack of intervisibility between locations with sensors.

The second group often includes metrics such as travel time, queue length and incidence of accidents at a certain point in the network. It entails vehicle-to-cloud communication, allowing the calculation of advisory speed using data gathered from vehicle sensors. Regarding GPS-enabled vehicles, taxis and buses are the most used probe vehicles. Furthermore, with the proliferation of smartphones, it is now simpler to collect information on location, incidents, travel times, and common routes followed by drivers.

Weather reports, special events (such as football matches), and other observations considered beneficial for prediction purposes, like the day of the week, exemplify data from the third group. By using data-driven approaches, it is possible to relate these external information sources, such as incidents and road works, to traffic conditions.

When incorporating this information into the system, its uncertainty may also require assessment, especially when using data from unreliable sources like smartphone usage. Traffic flow modelling uses the gathered data to deduce the features of transportation systems functioning, such as the degree, frequency, and length of congestion, traffic intensity, and loads at specific intervals. Some other metrics are also computed, including average speed and volume transportation of goods, passengers and the respective time cost.

One major obstacle to intelligent transportation is achieving precise information on the current transportation infrastructure. Data modelling and engineering construction are closely tied to the expertise of professionals in many sectors. Furthermore, the accumulated data is of poor quality and low value, making it hard to meet reality's urgent demand and the effect of the rapid application. Data sources are now the primary barrier to the advancement of DT technology, placing restrictions on data analysis and utilisation.

Other challenges include the DT's security, particularly when data comes from IoT devices. There is a pressing need for approaches to increase the credibility of integrated simulation models and their predictions.

This section highlighted the importance of data in modelling traffic systems. Regardless of the traffic forecasting approach, historical data is just as influential as a source of real-time information. This significance is evident in the studies [1] and [2], which showcase ML approaches for traffic prediction using ILD data. While data-driven approaches utilise the network's past to anticipate its evolution, model-driven methods use it to calibrate the parameters used in traffic modelling. Its enormous relevance, associated with the unresolved challenges mentioned above, makes data the main bottleneck for developing DT technology.

It is important to emphasise that the work in this thesis will be based on static observations. More precisely, the data will originate from inductive loop detectors strategically placed across the road networks.

2.1.2 Inductive Loop Detectors

Inductive loop detectors (ILDs) are devices commonly used in traffic control systems to detect the presence of vehicles. Typically, they are placed under the pavement of roads, detecting changes in a magnetic field brought on by a moving vehicle.

The magnetic field, created with a coil of wire, is disturbed when a vehicle drives over the loop, as the metal in the vehicle's undercarriage alters the current flowing through the coil.

By analysing the changes in the current, it is possible to deduce numerous traffic metrics. In addition to generating traffic data, they enable the automation of a few procedures, such as activating traffic lights or controlling gate barriers.

There are diverse varieties of ILDs, including:

- **Single-loop detectors:** the most basic sort of ILD used to detect the presence of a single vehicle;
- **Multi-loop detectors:** often utilised in multi-lane applications to detect the presence of multiple vehicles;
- **Dual-loop detectors:** frequently employed in one-way roadway applications to detect the presence of vehicles in both directions;
- **Smart loop detectors:** the most sophisticated ILDs available that provide extensive information about the vehicle, such as its size, length, and class.

The first type of ILDs, single-loop detectors, is the most commonly used. Its popularity stems from its low costs, resulting from the device's long lifespan and minimal maintenance requirements. They can detect the presence of a vehicle with a high degree of accuracy, even in adverse weather circumstances. Through its utilisation, three essential traffic parameters - speed, volume, and occupancy - can be inferred in real time.

As previously stated, one of the main problems of ILDs is their limited network coverage. Despite this hindrance, these devices are still widely used for traffic data collection. The numerous advantages that come with them account for their widespread use. First, they are cost-effective since they use the existing loop infrastructure. Integrating with already-installed sensors, usually ILDs, would be preferable instead implementing new technologies while disregarding prior expenditures. Furthermore, this technology is nonintrusive and anonymous, unlike other systems that require a specific vehicle identity, like Automated Vehicle Identification (AVI). Additionally, modern sensor technology allows researchers to collect more precise and trustworthy traffic data, leading to more effective traffic monitoring and control. Thanks to their simplicity and robustness, these devices are relatively low cost, easy to install and maintain, and reliable and accurate.

A detector card is a circuit board customarily installed inside a loop detector, responsible for processing the signals generated by the inductive loop in the pavement. This component includes the electronics required to interpret the loop's signals and provide a detection output to the rest of the system.

Detector cards used with traditional ILDs are often bivalent, with the output being either 0 or 1, depending on vehicle presence. However, detector card technology has advanced to the

point where it is now possible to obtain the inductance change across the loop from the vehicle's passage. The rapid scan rate of the detector allows for varied levels of inductance change, which produces a waveform, often known as a vehicle signature.

Vehicle signatures vary depending on vehicle speed and type and display various characteristics. Many features may be derived by exploiting those vehicle signatures directly or indirectly.

The authors in [34] argue that two broad categories of feature vectors exist in this context: vehicle-specific and traffic-specific. The vehicle-feature vectors are mainly dependent on the vehicle type. Vehicle length is an example of this category. These features should remain invariant under the same sensor, installation settings, and driver behaviour. Traffic-specific features like speed and occupancy highly relate to traffic conditions. Another example of these features is the skew rate, representing the slope value at point 0.5 of the normalised signature.

2.1.3 Fundamental Diagrams

Fundamental diagrams (FDs) depict the relationships between several variables in a transportation system, such as traffic flow, density, and average speed. They are used to analyse traffic behaviour and to understand the interactions between various factors in a traffic system.

Typically, they illustrate how a road network's traffic flow (vehicles per unit of time) and traffic density (vehicles per unit length) relate to one another. This relationship is known as the "fundamental relationship" or the "fundamental diagram of traffic flow". The FD describes how traffic flow and density change due to numerous factors such as traffic incidents, road network architecture, and traffic management strategies.

FDs come in various forms, such as capacity diagrams, which show the maximum flow a road network can accommodate under ideal circumstances, and jam diagrams, which show the relation between flow and density under congested conditions. These diagrams are crucial for comprehending traffic behaviour and making sensible transportation management and planning decisions.

Macroscopic Fundamental Diagrams (MFDs) illustrate the relationships between traffic variables at a broader scale, such as the level of an entire road network or a metropolitan region, as opposed to Microscopic Fundamental Diagrams, which do so at a finer scale. They are used to study traffic behaviour at the macroscopic level, particularly to assess the effects of large-scale events, like introducing a new transportation system or building a new road system, on traffic flow and density.

2.2 Digital Twins

As the transportation sector evolves, there is a growing trend towards developing next-generation road infrastructures, integral components of future smart cities. Digital Twins (DTs) arise as a vital instrument for these systems, simplifying their management, analysis, and operation.

This section attempts to answer the following research questions:

- What is a Digital Twin, and what are some of its misconceptions with current and previous definitions?
- What are the enabling technologies associated with this technology?
- What are its main applications?
- What is the status of open research, and what are the challenges of Digital Twins?
- Where and when should a Digital Twin be developed?
- Why should a Digital Twin be used?
- How to design and implement a Digital Twin?

2.2.1 Term Definitions

Formal ideas about Digital Twins have existed since the turn of the century. The term "twins" usage dates back to the National Aeronautical Space Administration (NASA)'s Apollo program when it created two identical space vehicles to allow mirroring of the space vehicle's circumstances during the voyage. In 2012, NASA published a study titled "The Digital Twin Paradigm for Future NASA and US Air Force Vehicles," establishing a crucial milestone in defining Digital Twins. The stated description is as follows:

> "A Digital Twin is an integrated multiphysics, multiscale, probabilistic simulation of an as-built vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its corresponding flying twin." [18, chap. The Digital Twin (Concept)]

This definition is far too specific for NASA's interplanetary vehicle development. The meaning of this term has varied over time to reflect new applications.

Due to the ever-changing definitions, there is a degree of uncertainty around the terminology, making it challenging to describe. On top of that, differentiating DTs from generic computer models and simulations has not been made easier by academia or industry. For instance, some claim that a digital twin is a virtual representation or model that interacts with the physical system throughout its life cycle. Other commonly utilised definitions emphasise the necessity of information exchange between the two spaces, encompassing sensors, data, and models. Some view a digital twin as the cyber component of a cyber-physical system (CPS). The promising future of this technology requires a more definitive definition.

The work in [45] conducts an exhaustive systematic literature review on Digital Twins, seeking to narrow down the meaning of this concept. The authors list the ability of simulation along the

product life cycle, the synchronisation of the cyber system with the physical assets, the integration of real-time data, the behavioural modelling of the physical space and the services provided by the virtual system as the main aspects that characterise the digital twin definition. In light of those above, they define the word as follows:

> "A set of adaptive models that emulate the behaviour of a physical system in a virtual system, getting real-time data to update itself along its life cycle. The digital twin replicates the physical system to predict failures and opportunities for change and prescribe real-time actions for optimising and/or mitigating unexpected events by observing and evaluating the operating profile system."

The uncertainties about this term result in some misconceptions by some authors. Amongst these is the idea that DTs must be an exact 3D model of physical objects. On the other hand, some people mistakenly believe that a Digital Twin is simply a 3D model.

The authors in [14] present three definitions that help identify the general misconceptions in the literature:

- **Digital Model:** a digital version of an existing or planned physical thing, with no automatic data flow between them. Examples of digital models are plans for construction, product designs, and development. The key distinguishing characteristic is no automated data interchange between the physical system and the digital model. In other words, after the production of the digital model, any modification to the actual object has no bearing on the digital model;
- **Digital Shadow:** a digital representation of an object with a one-way flow between the physical and digital object. A change in the physical item's state impacts the digital version, not the other way around;
- **Digital Twin:** a system where data is transferred seamlessly back and forth between an existing physical entity and a digital object. Any modification to the physical thing causes an automatic adjustment of the digital entity and vice versa.

In summary, it has been shown from this review that this concept is a somewhat nebulous term. According to certain writers, like [45] and [11], the fundamental idea behind a Digital Twin is a high-fidelity virtual model of a physical entity with the capacity to replicate and simulate the states and behaviours of the latter throughout its lifespan. These do not require a bidirectional automatic information exchange. Others, such as [14] and [29], go a step further and say that it is best described as the automated data integration between a physical and virtual machine in either direction. The virtual model is updated using information gathered from the physical environment. While operating in real-time, the physical twin enhances its performance by exploiting the knowledge learned from the data.

2.2.2 Enabling Technologies

The task of modelling reality in a digital twin must consider sensors, multifunctional models, data from many sources, services, and other factors. This process is quite complex as it needs a large amount of data. This new tool thus becomes dependent on the evolution of other enabling technologies. The literature focuses mainly on three of them: IoT, Data Analytics and Simulation.

The Internet of Things (IoT) is about providing connected devices with intelligence and the ability to collect information about their surroundings. Interconnected devices enable developers to track and monitor everything we do, thus leading to a more intelligent and efficient world.

A DT must capture the pertinent elements, such as the necessary characteristics and relationships, of the existing system (AS) regarding its operations' contexts and environments. This process includes data collection, storage, calculation, and inference. IoT has increased the volume of data usable in manufacturing, healthcare, and smart city environments, thus enriching the accuracy of the virtual replication of the physical system.

Data Analytics is an umbrella term that groups analytical concepts, as seen throughout the literature. Aside from methods for obtaining statistics from data, Artificial Intelligence (AI) and its subsections, especially Machine Learning (ML) and Deep Learning (DL), stand out. The broad definition of AI dates back to the late 1950s with the concept of creating "intelligent systems". Its subsection, ML, is the creation of algorithms that allow the computer to learn and act for the user without being explicitly programmed. It enables the development of software applications that autonomously gather and analyse data using complex algorithms. DL algorithms learn unstructured and unlabelled data using complex neural networks with autonomous input feature extraction instead of manual extraction. These networks employ machine learning to create deep learning models that can take longer to train because of the considerably more extensive neural networks but result in higher accuracy.

By building a linked physical and virtual twin, DTs can address the issue of seamless IoT and data analytics integration. Combining this technology with artificial intelligence approaches like ML and DL makes it possible to extract valuable information from the collected data about the environment. In addition to real-time data collection, this combination provides effective services to the service provider and the end user.

Furthermore, DTs can utilise simulation to help with decision-making for maintenance or enhancement of the AS based on new needs and/or resource availability. Not only for a general device but also for monitoring whole systems, simulation serves as the foundation for design decisions, validation, and testing. On the one hand, a DT's environment is designed to allow it to imitate the AS's behaviour to simulate how the system reacts. These simulation capabilities may be considered a "static feature" of the DT. Emulation, on the other hand, describes a DT's ability to sync with the AS to perform approximately like it. As a result, this aspect of DT might be referred to as a "dynamic feature".

Sensors and actuators are essential components in CPS, whereas models and data are critical components in DT. Technological advancements in these three domains have become crucial

criteria for promoting DTs.

2.2.3 Implementation Phases

DTs can serve various purposes, including non-digital system design, development, analysis, simulation, and operation to understand, monitor, and/or optimise the AS. This technology is widely used in many areas to understand better, regulate, and optimise the behaviour of complex systems, either at design time (for example, design-space exploration) or runtime (to increase performance/productivity or prevent failures).

The digital twin finds applications throughout the entire product life cycle management (PLM), which splits into three phases: design, production, and service. In the design phase, the DT intends to develop the digital product design before execution. During this phase, the digital twin may be used for conceptual design, detailed design, and virtual verification of a product. In the conceptual design stage, the DT assists designers in formulating functional requirements. Using real-time transmission data can improve the transparency and speed of communication between clients and designers. In the detailed design stage, the DT enables simulation testing throughout the thorough design process to ensure the prototype can deliver the expected performance. Lastly, the digital twin enables simulation and predicts the performance of the physical products based on virtual models during the virtual verification stage.

Regarding the production phase, DTs strive for real-time monitoring and optimisation, as well as anticipating the future condition of the physical twin, preventing downtime and malfunctions. It aids in determining the best combination of parameters and activities to maximise performance and offer projections for long-term planning.

Finally, the service phase relates to the steps after the sale/deployment, including product utilisation and maintenance. In this phase, DTs can provide value-added services support for prognostics and health management (PHM). The PHM is an engineering process for preventing failures and predicting reliability and remaining useful lifetime (RUL). In this case, the DT increases the precision and efficiency of a product's life cycle monitoring.

Currently, relatively few digital twin applications support the entire product life cycle.

2.2.4 General Architecture

As outlined in [45], a DT architecture consists of several components and technologies categorically organised into three main layers: the physical, network, and computing.

The first consists of physical entities identified based on the stage of the product life cycle. The network layer bridges the physical and virtual domains. It exchanges data and information. The computing layer involves the virtual entities emulating the corresponding real entities, including data-driven models and analytics, physic-based models, services, and users.

Each layer comprises some DT components (for example, hardware or software technologies, models, and information structures) that share similarities in their scope of use and interactions, presenting complementary functionalities.

The physical layer components carry real-time data for synchronising the virtual twin with its matching physical twin, promoting anomaly detection, prediction, prescription, and optimisation capabilities. The network layer involves connections and interactions between physical and virtual entities. This layer links all components so that they may share data and information with other components. The Service Oriented Architecture (SOA) method is DT's most commonly utilised middleware architecture. Its principles allow the decomposition of complex and monolithic systems into applications of an ecosystem of elementary and well-defined components. The computing layer is indispensable for the computation and decision-making of digital twins. One can decompose it as an interconnected set of layers, including the following components: data, models, and modelling features.

2.2.5 Main Applications

The numerous applications of this new technology reflect its innovative character and potential. Nowadays, smart cities and manufacturing are the key research areas, with some healthcare-related uses of Digital Twin technology discovered. Other sporadic cases of applications include Maritime Shipping and Aerospace. Due to the fast advancements in connections made possible by the IoT, DTs are becoming increasingly popular. They have the potential to be extremely useful in smart cities. The capacity of smart city services and infrastructures to have sensors and be monitored using IoT devices is precious. As smart cities expand, connectivity and valuable data increase, and DTs become increasingly practical.

The DT is typically applied in contexts of uncertainty and complexity, where working circumstances might change based on external and internal factors. This fact owes to its emulation capability, which allows it to adjust its original configuration and adapt to the present circumstances, known through its synchronous connection with the AS.

2.2.6 Primary Challenges

It is becoming clear that Digital Twin coexists alongside AI and IoT technology, resulting in shared challenges.

For instance, the IT infrastructure must be suitable for the DT to succeed with IoT and data analytics. Without a connected and well-planned IT infrastructure, this technology won't be able to properly accomplish its intended aims.

The next challenge revolves around the data required for a DT. It needs high-quality, noise-free data from a continuous, uninterrupted stream. The DT may underperform if the data is absent, poor, or inconsistent. Planning and analysis of device usage are required to determine the appropriate data to be gathered and used for the effective deployment of a Digital Twin.

The privacy and security issues with DTs present a barrier in an industrial context, constituting the third challenge. There is a considerable risk of disclosing sensitive data due to the enormous volume of data they use. To mitigate this, the primary enabling technologies for DTs must adhere

to the most recent security and privacy conventions. Security and privacy cogitation for DTs data help to address trust difficulties with Digital Twins.

Trust difficulties establish another challenge from both the perspective of the organisation and the user. A DT's technology has to be further addressed and described at a basic level to ensure that end users and businesses comprehend the advantages of a DT. A better understanding helps overcome the barrier of trust. Another method to achieve this is model validation, ensuring that DTs work as intended. In addition, as mentioned in the last paragraph, enabling technologies will shed more light on the procedures used to maintain security and privacy standards throughout the development, promoting user confidence.

The fifth challenge is related to the high expectations of this tool, as potential consumers only see the benefits and feel it will immediately save them time and money. The area is still in its infancy, which must be considered while deciding whether to apply it. Caution is required to underline the obstacles to DT expectations and the need for greater awareness. Strong IoT infrastructure foundations and a better knowledge of the data needed for analytics are essential for organisations to use DTs. Combating the notion that this technology should be employed only because of the prevailing trends is equally tricky. The benefits and drawbacks of its expectations must be explored to take proper action while setting up Digital Twin systems.

Despite DT's difficulties with IoT and data analytics, there are also particular challenges related to its modelling and development.

The absence of a standardised modelling methodology is an example of these challenges. Whether physics-based or designed-based, there has to be a traditional approach from the original design to the DT simulation. Standardised methodologies promote domain and user comprehension while facilitating information flow throughout Digital Twin development and deployment stages.

Another problem stemming from the necessity for standardised use is ensuring that information relating to domain use is conveyed to each development and functional level of Digital Twin modelling. This process guarantees interoperability with fields like IoT and data analytics, enabling Digital Twin's practical use in the future. Domain knowledge is crucial in the future creation of DTs and when leveraging IoT and data analytics.

2.3 Traffic Simulators

Traffic simulators are essential tools in traffic engineering and transportation planning. They provide a means for researchers, planners, and policymakers to understand, assess, and predict traffic dynamics under diverse conditions without real-world experimentation.

There are different kinds of simulators (macroscopic, mesoscopic, and microscopic), each distinguished by the nature of the simulations they can execute. Each simulation type offers varying degrees of detail and computational complexity, rendering them suitable for diverse applications.

Macroscopic simulators, sometimes called flow-based or aggregate models, provide a high-level view of traffic dynamics by focusing on traffic flows without delving into individual vehicle-level considerations. These simulators primarily emphasise traffic flows, density, and speeds at a network-wide level.

Mesoscopic simulators, often called hybrid models, find a middle ground between macroscopic and microscopic models. They offer an intermediate-level view of traffic, delivering more granularity than macroscopic models while maintaining computational efficiency.

Microscopic simulators offer the highest level of detail among the three types. They intricately model individual vehicle movements, behaviours, and interactions. In the vehicular networking community, the behaviour of each vehicle is typically a focal point and requires meticulous modelling, making microscopic simulators the preferred choice.

Comprehensive surveys like [22], [33] and [37] offer valuable insights into the distinct mobility and network simulators accessible to the scientific community. SUMO (Simulation of Urban MObility) emerges as a widely adopted and versatile tool within researchers' diverse spectrum of traffic simulators. As an open-source software developed by the German Aerospace Center (DLR), SUMO employs a microscopic approach to traffic simulation. Its capabilities comprise simulating urban and highway traffic, rendering it highly suitable for transportation research and planning.

SUMO is amenable to macroscopic and microscopic simulations and is compatible with network simulators like Omnet++ and NS3 [31]. Its simulation capabilities encompass a wide range, including multimodal traffic, traffic lights, inductive loops, and other detectors. Moreover, it supports online interaction and closed-loop feedback enabled by the TraCI [50] interface. Additionally, the SUMO community demonstrates a high level of engagement in ITS topics, fostering a dynamic and active environment for research and development in this domain.

In conclusion, SUMO's versatility and capacity to simulate both urban and highway traffic render it an invaluable resource for researchers, urban planners, and transportation engineers. Its open-source nature fosters collaboration and customisation, rendering it a favoured selection for numerous traffic simulation endeavours. SUMO was the simulator chosen to develop the automated DT-building framework for all these reasons. Furthermore, the baseline methodology that inspired this tool finds implementation within SUMO, streamlining the development process.

2.4 Related Work

Before detailing the creation of the framework for the automatic building of Digital Twins, it is relevant to analyse the topic extensively. It is essential to point out that no studies were found with the same intention to automate the process of creating DTs, highlighting this topic's innovative character. However, it is relevant to analyse the various methods of implementing DTs, noting the advantages and disadvantages of each approach and proving the specificity of existing methods explicitly built for the road network involved.

This section provides an enumeration and description of several traffic simulation scenarios, focusing on those works that aim at building realistic DTs and the respective validation of those models. Most of these scenarios rely on SUMO, the selected simulator for the framework devised in this dissertation. Additionally, a summary of the various characteristics of these scenarios, such as the primary data source or the common development steps, will be performed.

The work in [43] presents a large-scale simulation scenario illustrating a complete day of private motorised traffic in Berlin. It represents the most extensive traffic scenario known to this date, encompassing over 2.2 million trips across an 800 km^2 area.

A prior Berlin scenario that inspired this SUMO simulation is worth discussing. The MATSim Open Berlin Scenario [53] runs within a mesoscopic simulator known as MATSim. The traffic demand, encompassing the origins and destinations of people's journeys, was generated using open data such as demographic statistics and calibrated against actual traffic measurements.

Unlike SUMO, MATSim employs an agent-based paradigm for modelling the daily plans of individual agents, representing the traffic demand. These daily plans are iteratively adjusted using co-evolutionary algorithms in conjunction with mobility simulations until all plans achieve stability, reaching a user equilibrium [49]. MATSim generates calibrated traffic demand and supply in this manner. Each person's mode of transportation and route were selected during the calibration with MATSim to attain a user equilibrium eventually. Achieving this state involved iteratively adjusting the transport mode and route until each individual's costs, such as travel time, could not be further improved. Moreover, actual traffic counts were incorporated into the calibration process, bringing the scenario closer to the authentic behaviour of the city. The developed scenario represents 10% of actual mobility in Berlin/Brandenburg.

Returning to the SUMO scenario described in [53], it emerged from the conversion of this previous scenario, which required a few workarounds.

Firstly, the MATSim scenario only replicates 10% of real traffic, accomplished by reducing road capacities. However, this technique is not directly applicable to SUMO. Hence, it becomes necessary to upscale the traffic demand to 100% for a realistic SUMO traffic model, resulting in a considerably larger number of agents in the simulation. To address this substantial volume of traffic participants, the authors restricted the target scenario by eliminating all non-car trips and those originating or ending in Brandenburg. As a result, the simulation solely encompasses the Berlin area traffic.

Secondly, there are significant differences in the traffic dynamics models used by MATSim and SUMO. Consequently, it is not feasible to directly transfer the routes calibrated by MATSim into SUMO. Instead, the authors extracted the calibrated demand from the MATSim scenario and reassigned the traffic using SUMO through iterative route selection until reaching a user equilibrium. Hereupon, the authors provide a three-step description of their methodology.

The initial step was to set up the SUMO's traffic network. Given the road network model differences between MATSim and SUMO, they reconstructed the traffic network from scratch using OpenStreetMap (OSM) data and SUMO's NETCONVERT tool. Furthermore, as OSM only provides limited road information, the resulting network contained several modelling inaccuracies and needed to be modified. More specifically, the authors tweaked the layouts of hundreds of junctions and rectified the inter-lane connectivity using SUMO's NETEDIT platform. Also, they corrected erroneous speed limits and lane numbers caused by numerous issues in OSM data using their local knowledge, satellite images, and Google Street View pictures. Additionally, they utilised signalling information from pre-defined traffic signal programs in SUMO produced during the network's import through NETCONVERT. However, many of these signal programs led to unsatisfactory traffic flows and needed to be manually adjusted.

The second step comprised transferring the traffic demand from the MATSim scenario to SUMO. The former scenario furnishes an individualised day plan for each agent, encompassing all activities (including their duration and type, such as home, work, or leisure) and the trips between these activities (comprising the mode of transport, origin, destination, and departure time). To extract the demand, the authors selected all trips that utilised cars as the chosen mode of transportation. Subsequently, they performed map-matching of the origin and destination locations onto the SUMO road network and retained only those trips for which they could successfully compute a route. This process eliminated all trips beyond the Berlin area. Finally, to expand the traffic demand to 100% of actual traffic, they created nine clones for each trip, randomly shifting the source and target for each clone within 3 km^2 and varying the departure time by 15 minutes.

The third step is traffic assignment, which involves generating a route for each trip to achieve a relatively stable traffic state, reducing overall travel times and minimising teleports in SUMO. In SUMO, teleports occur when vehicles wait too long due to traffic congestion or a deadlock. The vehicle is taken from its current location and reinserted to a different location along the route, with sufficient space to continue the ride. Initially, as the work in [41] recommended, the authors attempted to create these routes using SUMO's MAROUTER tool, which calculates a stochastic user equilibrium without relying on simulations. However, frequent deadlocks and congestion prevented them from adequately applying the tool's calculated results, thus necessitating extensive manual network adjustments in conjunction with running lengthy simulations. Instead, they adopted the conventional approach of iterative dynamic user assignment. For this, they computed a set of initial routes for each trip through the Choice Routing algorithm [12, 23]. Subsequently, they employed SUMO's *duaIterate.py* script, which aims to attain a user equilibrium by iteratively simulating and adjusting route choices. They repeated the process until there was no discernible improvement in overall travel time. Even though this strategy was easily controllable because the

assignment process inherently resolved deadlocks and congestion through iterative route choice, this assignment procedure demanded a significant duration (spanning several weeks) due to the considerable volume of vehicles and routes to process.

To determine how closely the simulated traffic represents reality regarding highway traffic volumes, the authors resorted to publicly accessible traffic count data provided by the City of Berlin[1]. This dataset comprises hourly traffic counts for the past nine years, encompassing over 300 counting locations within the city. In addition, they obtained hourly traffic counts for highways from the official German highway management (BASt)[2]. For the analysis, they chose a random date from the pre-pandemic period: September 27th, 2018, a Thursday corresponding to a typical working day. For comparison purposes, they selected eight random locations within the city and two spots along the main highway A100 traversing Berlin. The corresponding edges of the SUMO network were then selected to gather simulated traffic counts, which were then compared against the actual observed traffic data.

Shifting the focus to the work in [7], it delineates three scenarios centred around two well-recognised roads in Bologna, Italy, Andrea Costa and Pasubio. The study delves into two individual scenarios for each of these roads. Andrea Costa's scenario, meant to simulate the mobility of huge events such as football matches or concerts, covers the vicinity of a football stadium. In a complementary manner, the Pasubio scenario expands upon the Andrea Costa scenario by encompassing the region around a hospital and including frequently used routes to the football stadium.

Despite being implemented in SUMO, the networks of these scenarios are defined as input files to VISSIM, another microscopic simulator. Due to the relatively small geographical size of both VISSIM networks, the authors decided to amalgamate them into a third scenario. Because the areas they cover overlap, this third scenario was obtained by merging them. Apart from encompassing the morning traffic demand, the joint scenario also incorporates the traffic demand associated with a football match. The induction loop data from March 24th, 2010, a football match day, was compared with the traffic flow observed one week earlier. This analysis led to the generation of additional routes from the examined dataset.

These models describe the information about passenger vehicles in an aggregated manner, as it gets the number of vehicles to insert for specific roads at the network's border. As these vehicles traverse the initial route, they encounter designated "routing decision points" where they are randomly assigned new routes per a given distribution. This strategy effectively replicates the observed turn percentages at intersections in reality. Aside from passenger vehicles, both scenarios feature public bus transportation and traffic lights. Pertinent details provided as input include the locations of bus stops, bus routes and schedules, and traffic lights' positions and signal plans.

The authors utilise two datasets comprising detector measurements provided by the municipality of Bologna.

[1] https://api.viz.berlin.de/daten/verkehrsdetektion

[2] https://www.bast.de/DE/Verkehrstechnik/Fachthemen/v2-verkehrszaehlung/zaehl_node.html

The first dataset encompasses measurements taken from November 11th, 2008, to November 13th, 2008, from Tuesday to Thursday. This selection aligns with the notion that Tuesday to Thursday generally represents the typical weekdays regarding traffic patterns. Mondays and Fridays exhibit different traffic patterns. On Mondays, the pattern diverges due to the delayed departure of passenger traffic and, in certain countries, restrictions on heavy delivery traffic on Sundays. On Fridays, the afternoon peak tends to be earlier due to the early business closing times.

The given measurements aggregate into half-hour intervals. There were 636 listed detection sites, with around 90 detectors indicating errors. Each provided temporal value indicates the number of vehicles that passed through the detection location. Speed information is not available since the detection data stems from single ILDs. Also, there is no differentiation between vehicle classes, and each detection site may cover multiple lanes. The quality of the detector data is notable when contrasted with data from other sites, where the count of known errors in the detector data is significantly higher.

The second dataset comprises measurements for the same days but aggregates into 5-minute intervals. The quality remains consistent with the first dataset.

The provided VISSIM scenarios comprehensively describe the infrastructure and demands within the modelled areas, including additional details about public transport. Thus, all available input information from these scenarios was incorporated. Although VISSIM is a microscopic simulation like SUMO, it follows an entirely different concept of road infrastructure modelling. The primary distinction is that, unlike SUMO's graph-based concept involving nodes (intersections) and edges (roads), VISSIM relies solely on roads and their connections. This distinction makes importing the VISSIM network more complex, often necessitating manual adjustments to the results following the initial conversion. Moreover, as SUMO exclusively supports importing VISSIM networks stored in German (VISSIM employs a network description language), the supported networks must also be translated from English into German. The validation process entailed manually comparing the network against images from Google Earth and Google Maps, along with the supported junction telemetries. While the number of lanes was correct for all edges within the VISSIM networks, the connections between lanes across intersections were manually corrected.

For the validation of the simulation, the authors compared the real-world measurements and the simulation outcomes. The authors evaluated the imported networks through the provided induction loop data, where only the flow over the detectors was available and utilised for assessment. From the accumulated findings, the authors conclude that the total number of vehicles simulated in SUMO is reasonably precise. Similarly, the simulation results for a single hour are relatively accurate. However, it is noticeable that the simulation encounters difficulties in effectively replicating the traffic demand in the combined scenario within an hour. The reason is simple: as the scenario's area size increases, the vehicles require a more extensive period to populate it and traverse the available induction loops. In conclusion, a specific duration of simulation time is necessary to populate the scenario with vehicles until achieving the actual network state. This simulation

warm-up period is a recognised requirement in traffic simulations. Usually, it involves employing twice the maximum travel time across the network.

Moving on to the work in [19], the focus lies in examining Connected and Autonomous Vehicles (CAVs) and their efficiency and safety across diverse traffic scenarios involving distinct vehicle and road types. The authors conduct simulations involving distinct vehicle penetration rates, considering multiple degrees of automation.

They examine the impact of CAVs on three types of networks: urban, national, and motorway. The simulation utilises data gathered between 2012 and 2019, from which, for each scenario, they extract three primary traffic states: freeflow, saturated, and congested.

The primary emphasis is microscopic traffic modelling, where vehicle motion arises from longitudinal and lateral models. Longitudinal behaviour (vehicle acceleration) relies on a car-following model that can be customised to mimic diverse driving styles or vehicle capabilities. Lateral behaviour is determined using a lane-changing model, typically concerning decisions grounded in the perception of adjacent lanes. Although these models successfully mimic traffic patterns and flow instabilities, the literature does not agree on using a particular model for each vehicle type.

The impact of CAVs is assessed at a link level, offering a more comprehensive understanding of congestion levels and their propagation throughout the network. This evaluation strategy involves computing three distinct traffic indicators aggregated for each link:

- **Congestion Index (CI):** This metric estimates the congestion level of a road segment and ranges from 0 (free flow) to 1 (all vehicles stopped);
- **Travel Rate (TR):** This indicator monitors vehicle speeds by giving the rate of movement in minutes per kilometre (min/km). It approximates travel time based on the average speed observed within the examined segment;
- **Total Time Spent (TTS):** This metric evaluates traffic efficiency by summing the travel times of all individual vehicles within the analysed segment.

To measure the influence of CAVs on road safety, they use specific safety metrics to estimate the collision probability between cars in the simulation.

Among the scenarios presented by this study, the one of particular interest within the context of this thesis is the simulation involving 100% Human-Driven Vehicles (HDVs), as CAVs deviate from the main topic of this research. This scenario enables observing the current baseline traffic behaviour, aligning with the intended objective. In this regard, the sole level of automation of interest is level 0, corresponding to SUMO's default car-following model.

Demand patterns and volumes were derived by averaging several months of actual data, excluding holidays and weekends, to obtain a typical traffic day (January to April 2012). Distinct types of data were employed based on the three categories of roads:

- For the national and motorway networks, the model uses ILDs data sourced from a dataset provided by Transport Infrastructure Ireland. This dataset included traffic flows aggregated over 5-minute intervals, organised by lane and direction, spanning years until 2019;
- Regarding the urban network, the model uses the Dublin SCATS dataset, which originates from an adaptive traffic light control system in Dublin. This dataset comprises vehicle counts taken every 6 minutes at 480 locations in the city centre.

For each network, the authors chose a specific time of day to exemplify one of three distinct traffic scenarios:

- Free-flow state (typically occurring early in the morning);
- Saturated traffic state (indicative of high traffic levels with initial signs of congestion);
- Congested traffic state (representing the highest traffic demand observed in both simulation and data).

Out of curiosity, the authors' main conclusion is that the advantages brought about by incorporating CAVs concerning efficiency and safety would manifest gradually but not linearly. This finding implies that the transitional phase will hold a pivotal role in the effective implementation of CAVs and warrants the attention of researchers. Additionally, the authors ascertain that congestion wields a more substantial impact on conflicts than penetration rates, highlighting the importance of simultaneously assessing efficiency and safety, contrary to previous studies that treated these factors as separate entities.

The scenarios from the work in [27] are also highly relevant to this context. The methodology employed therein will establish a baseline reference for the framework developed in this thesis. Section Baseline Methodology will expound upon the details of this study's model.

Nonetheless, a brief description can be presented here regarding the two scenarios therein. The first scenario closely aligns with the Digital Twin definition introduced in Term Definitions since it involves a real-time simulation of a road network in Geneva, Switzerland. This scenario's model receives traffic data from ILDs, with minute-level resolution, which is acquired directly in real-time through the Open Data Platform Mobility Switzerland (ODPMS)[3]. The second scenario consists of dynamic traffic generation using offline data from the same source as scenario one.

Another pertinent study in this context is InTAS, a realistic traffic scenario for Ingolstadt, a city in Bavaria, southern Germany [30]. Ingolstadt ranks as the fifth largest city in this state, and its unique characteristics significantly influence its traffic dynamics. For instance, the city hosts a significant industry that accounts for nearly half its workforce, operating round-the-clock in shift patterns. Moreover, the city boasts a high per capita income and low unemployment rate. These factors contribute to a comparatively elevated rate of car usage in Ingolstadt compared to other cities in Germany.

[3] https://opentransportdata.swiss/en/

This work focuses on developing a realistic traffic scenario encircling the city of Ingolstadt. It specifically centres around Vehicular Ad-hoc Networks (VANETs), particularly on vehicle-to-everything (V2X) simulations, considering traffic patterns and driver behaviour. VANETs represent cooperative vehicular networks that rely on wireless communication among vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), also referred to as Road Side Unit (RSU), and vehicle-to-everything (V2X). One of the most prominent advancements VANETs offer is the ability to instantly share pertinent on-road information with other vehicles and RSUs, enhancing overall driving safety and experience. Nevertheless, conducting real-world testing of technologies like VANET can be prohibitively expensive, and replicating test scenarios is often unfeasible. Hence, simulation tools are essential for making testing more affordable and reproducible.

The Ingolstadt Traffic Scenario (InTAS) development encompassed four steps: delineating network topology, traffic demand modelling, scenario simulation, and scenario evaluation. The first step was to create a comprehensive city map incorporating all relevant information. This model considers road topology, building layouts, parking facilities, bus stop locations, and traffic light placements. The second step involved meticulously analysing all available real-world data and identifying a traffic demand model aligned with the dataset that creates a realistic traffic demand representation. The third step centred on adjusting simulation parameters and preparing for the subsequent scenario evaluation.

In the first step, the authors noticed that the converted data contained many streets that did not accurately mirror the real-world scenario. These inaccuracies encompassed discrepancies in the number of lanes, the absence of dedicated turn lanes, and the omission of exclusive bus lanes. Most likely, these issues stem from outdated information retrieved from OSM. Even though OSM undergoes regular updates through a collaborative, wiki-style process, certain places are not precise enough and contain only the street segment, not the number of lanes or exclusive lanes. Therefore, the authors used the NETEDIT tool to make corrections while manually inspecting and validating the map.

In collaboration with the *Ingolstadt Verkehrsmanagement und Geoinformation Office*, a division of the Ingolstadt city administration, an SFTP server housing data from 24 traffic monitoring points was established. The data includes information recorded for each point between September 3rd, 2019, and December 15th of the same year. It details the volume of vehicles passing through these monitored zones continuously over 24 hours, aggregating the total vehicle count in 15-minute intervals. The authors selected values for Tuesdays, Wednesdays, and Thursdays to ensure a representative dataset. The rationale behind this selection, as explained by the *Ingolstadt Verkehrsmanagement und Geoinformation Office*, is that these days typically experience the highest traffic volumes. Additionally, holidays and the days immediately preceding holidays were removed, as traffic patterns may alter on these days. The dataset was divided into two subsets to furnish data for both modelling and validation purposes. One used data from October 2019 to model the traffic demand and evaluate simulation parameters. The other subset was reserved for the validation phase and encompassed traffic data recorded in November 2019. This traffic scenario is also the first SUMO-based scenario to use traffic light programs similar to those used

in real traffic signal systems rather than just standard SUMO programs.

The dataset from the City of Ingolstadt contains detailed information for each sub-district, encompassing data on population, households, employed and unemployed residents, and registered vehicles. These statistics provide a robust foundation for modelling Ingolstadt's traffic. According to the available data, the authors decided to model InTAS using the *activitygen* method[4]. To delineate each driver's routes to reach their respective destinations, the authors employed the *duarouter* tool, designed explicitly for assigning complete routes between origin and destination points, effectively calculating routes for each vehicle. This tool employs the Dijkstra route-planning algorithm to compute the shortest paths across the network. To mitigate potential traffic congestion stemming from this approach, the authors implemented Gawron's method to optimise traffic flow as presented in [17]. This approach computes and optimises routes based on each vehicle's user equilibrium. In SUMO, it consists of the tool *duaIterate*, which iteratively attempts to discover the user equilibrium, i.e. to identify a route for each vehicle without reducing the travel cost [5].

For the validation of the model, the authors compared the total number of vehicles passing through each measurement point in the dataset with the corresponding simulation value obtained from virtual detectors positioned as closely as possible to the location of the actual measurement point. However, the model's performance exhibited limitations, particularly during periods of high traffic volume. This discrepancy might be because the traffic demand modelling method relied on averaging values for population and employment figures across various sub-areas within Ingolstadt. Employing average values can introduce inherent errors. Another potential factor contributing to the observed discrepancies could be the temporal misalignment between demographic and traffic data. The authorities issued demographic data for 2018, while real traffic used to validate InTAS was collected between August and December 2019. Even though the temporal gap between the demographic and traffic data is relatively short, spanning only eight months, it might lead to variations in traffic volumes and introduce errors.

The Luxembourg SUMO Traffic Scenario (LuST) [8] is a realistic simulation scenario with comparable goals to the latter, as it covers all requirements for a common foundation for assessing and testing network protocols and related applications.

This study, focusing on Vehicle-to-X issues and solutions, introduces a scenario that satisfies the following criteria:

- accommodates diverse traffic demands, encompassing congested as well as free-flow patterns;
- allows various scenario dimensions;
- incorporates various types of roads (e.g., residential, arterial, and highway);
- permits multi-modal traffic assessments, encompassing vehicles, public transport, and pedestrians;

[4] https://sumo.dlr.de/userdoc/Demand/Activity-based_Demand_Generation.html
[5] https://sumo.dlr.de/docs/Demand/Dynamic_User_Assignment.html

- delineates a realistic traffic scenario, avoiding gridlocks and unrealistic mobility patterns, spanning one day to capture traffic dynamics during peak hours (high density), throughout the day (moderate density), and during the night (low density).

The authors chose the City of Luxembourg as the scenario for this study based on several considerations. First, its topology resembles many European cities, comprising a central downtown area encircled by various neighbourhoods linked by arterial roads, with highways encircling the city's periphery [10]. Moreover, this city offers diverse traffic statistics, which are invaluable for calibrating traffic demand. Another pivotal factor in this choice is the city's size, which strikes a balance between being sufficiently compact for efficient microscopic simulations yet large enough to emulate the typical congestion patterns encountered in contemporary urban settings. The authors also opted for SUMO since, as a micro-mobility simulator, it enables coupling with vehicular mobility, as vehicular communications studies demonstrate [33].

After choosing the city, the authors utilised OSM to extract its road topology, guided by the discussion of this platform's accuracy in the study [20]. Subsequently, JOSM [21] was employed to select and modify points of interest and road segments manually. During this phase, the authors gathered information on various road types, traffic lights, bus stop locations and names. Furthermore, they recorded supplementary details regarding schools (including their locations and types) for utilisation in the activity generation process, and they captured building geometries to create the necessary polygons for obstacles, which are vital components for wireless network simulations. Notably, the authors adjusted the locations of bus stops in the LuST Scenario from those in the OSM file, trying to align them as closely as possible by reconfiguring the bus routes to match the new bus stop positions.

As this scenario aims for a working SUMO simulation, all the intersections were checked manually for correctness. The authors ensured that no intersection poses an unrealistic bottleneck for traffic flows by employing an iterative procedure utilising JOSM, *netconvert*[6], and SUMO. This iterative procedure was essential for constructing a road network with intersection geometry and segment shapes aligned with SUMO's requirements. These modifications led to minor disparities in the topology, such as variations in the angle between two streets, affecting road shapes. When generating bus stops, bus routes, and the polygons outlining the buildings, the authors had to consider these nuances. Vehicle restrictions were intentionally not imposed on any edge or lane to allow for greater flexibility, meaning that the authors removed lanes reserved for specific vehicle types. Adjustments were made to the number of lanes in specific road segments to replicate real-world traffic patterns closely. Additionally, they tried to standardise the roadways to create a scenario that could be readily altered or expanded.

The authors employed the *activitygen* tool to generate activity demand using publicly available government data accessible through the *Luxembourg National Institute of Statistics and Economic Studies* (STATEC) website[7]. For instance, these data sources included information on population statistics and age distributions. The *activitygen* tool uses a road network definition and population

[6] https://sumo.dlr.de/docs/netconvert.html
[7] https://www.statistiques.public.lu

details to formulate a traffic demand for the scenario. It leverages an activity-based traffic model encompassing various modes of transportation, such as buses, cars, bicycles, and pedestrians, that aids in determining daily activities like work, school, and leisure. The tool's configuration files must contain essential information about the city's topology, demographic statistics, school and workplace locations, and residential areas. The routes provided by this tool were then divided into cars and buses and optimised using the SUMO *duarouter* tool. The traffic demand generated through this process incorporated real-world data from diverse sources, including inductive loops in all intersections featuring traffic lights and highway ramps.

Regarding potential future developments, the scenario benefits from further enhancement by incorporating additional modes of transportation, such as pedestrian and bicycle traffic. Furthermore, concerning the traffic light system, it is essential to highlight that the current implementation employs a static scheduler. In light of this, a prospective avenue for advancement entails implementing a dynamic traffic light system, a feature that could contribute to the scenario's increased realism and adaptability.

To demonstrate that the LuST Scenario behaves realistically, the authors compared it to Google Maps' Typical Traffic option [13]. The traffic patterns are consistent for the 24 hours of simulation compared to the data from Google Maps, validating the scenario's capacity to offer realistic mobility simulations.

Based on the Principality of Monaco, the Monaco SUMO Traffic (MoST) Scenario [9] offers an ideal playground for examining advanced parking management solutions and implementing alternative transport mode applications while considering realistic telecommunication models. It is worth noting that, at the time of this work, a scenario capable of accommodating vulnerable road users like pedestrians, bicycles, and motorbikes was unavailable. Hence, this project seeks to establish a genuine urban mobility scenario that seamlessly incorporates these vulnerable users.

Its development considers future integration of Cooperative Intelligent Transport Systems (C-ITS) technologies. C-ITS relies on mobility data from vehicles and vulnerable road users to enhance road traffic and safety within urban and extra-urban areas.

The authors have chosen SUMO as the mobility simulator due to its highly engaged community on ITS topics and since it provides a socket interface that facilitates seamless communication with other simulators. For instance, network simulators such as OMNet++ and NS3 can interface with SUMO through various tools like Vehicles in Network Simulation (VEINS) [47], iTETRIS [26], and VSimRTI [44].

Finding and modelling a single environment that provides a complete playground required to work on different C-ITS features is complicated. Such systems require a multidimensional environment capable of accommodating geolocation and telecommunication facets. Moreover, they should exhibit high population density and congested traffic patterns to facilitate the exploration of alternative transportation modes and mobility optimisations. Finally, they must be scalable and manageable. Given these criteria, the authors have selected the Principality of Monaco as a prime

case study. This choice owes to Monaco's unique attributes, including its size, distinct characteristics, and the availability of publicly accessible information. Furthermore, the city's compact dimensions provide a manageable and controlled environment well-suited for modelling with a microscopic mobility simulator.

Moreover, the authors chose Monaco City as the scenario to develop, considering that it is:

- **Multidimensional for communications and environmental studies:** Monaco, a coastal city nestled on the slopes of mountains, boasts a layered topology punctuated by tunnels and bridges. This unique topographical layout offers intriguing opportunities for investigating precise positioning and propagation models. Tunnels and bridges contribute to a more realistic multidimensional environment than flat surfaces. Furthermore, when combined with accurate emission models, these multidimensional attributes facilitate exploring eco-routing from energy consumption and environmental impact.
- **Heavily congested for multimodality and parking management solutions:** Most city employees are commuters residing in the surrounding regions. This dynamic results in directional traffic congestion, with significant inbound traffic in the morning and congestion during the evening outbound commute. The city provides a large number of parking spaces and has a well-developed public transportation system in order to accommodate all the commuters. However, despite these measures, traffic congestion persists during peak hours. Given its characteristics, Monaco City is an ideal playground for examining advanced parking management solutions and applications related to multimodal and alternative transportation modes at a city-wide scale.
- **Densely populated for crowd and vulnerable road users applications:** Monaco is a popular tourist destination. In addition to the daily influx of pedestrian commuters, the city welcomes waves of tourists arriving by tour buses, public transport, and, to a lesser extent, private cars. Additionally, a large-scale mobility scenario capable of completely integrating vulnerable users can couple with specific crowds and multi-agent simulators that would not be otherwise scalable across an entire city because of their complexity. These unique characteristics make this scenario an ideal environment for investigating the influence of vulnerable road users on urban traffic dynamics. With it, the authors intend to leverage C-ITS applications to encourage a shift from cars to motorcycles and electric two-wheelers, optimising traffic flow and mitigating congestion.

The initial dataset contains a wealth of pertinent information from OSM. However, it is essential to note that OSM relies on crowd-sourced information, which can sometimes vary in accuracy. This initial dataset was improved using information from open-access websites from Monaco and France. The OSM data includes critical points of interest such as parking lots, car parks, train stations, bus stops, taxi stands, and bicycle-sharing points. These Points of Interest (POIs) are the foundation for constructing the core infrastructure for multimodal transportation and public transit

within the scenario. Placing additional POIs, such as museums, parks, shops, restaurants, workplaces, and educational institutions, is essential for enhancing the realism of the created activity-based mobility. Public transportation data has been sourced from Compagnie des Autobus de Monaco (CAM)[8] and Provence-Alpes-Côte d'Azur (PACA) Mobilité[9]. These sources provide information about the locations and schedules of buses and trains and the availability of electric bicycle stations. The Monaco Parking website[10] was instrumental in parking-related data. This website offers comprehensive information on the location, capacity, schedules, pricing, and availability of many parking facilities within the city. To further enhance the geographical accuracy of the scenario, detailed data on altitude and land registry was extracted from the Institut Géographique National (IGN)[11] database. This additional data was integrated with the existing street topology and Points of Interest (POIs) dataset, resulting in a highly reliable multidimensional city model.

Once more, a prominent challenge faced by the authors pertained to the dimensions and lane count of the network representation. Typically, street geometries tend to undergo alterations when transitioning from a graph to the SUMO representation, often resulting in closer boundaries than reality. In Monaco, the topography of the region itself amplifies this issue. The streets are narrower than the norm, and the buildings are densely situated, leaving minimal room for the adjustments required by the simulator. The perpetual challenge lies in striking a delicate balance between approximation and realism to yield a realistic and functional scenario that the simulator can use.

The generation of activity-based mobility relies on data from the refined dataset, traffic projections, and supplementary demographic insights. This approach captures authentic depictions of the population's daily routines. The enhancements in topology are intimately related to mobility generation. For this reason, the generation process is still ongoing, and, at the time of writing the article, this method could not yet provide realistic mobility during peak hours.

While the MoST Scenario is still in its preliminary stages, it exhibits great potential as a testing ground for exploring C-ITS and alternative transportation modes within a well-regulated environment. This environment facilitates interactions with vulnerable road users within a multidimensional realm considering location and telecommunication challenges.

The proposed future endeavours primarily encompass three avenues:

- enhancing the alignment between the geographic terrain and the street topology;
- refining the activity-based mobility generation, aligning it with actual data, and conducting evaluations;
- concentrating on vulnerable road users and alternative modes of transportation.

Finally, the authors conclude that the primary challenges of modelling a multidimensional traffic scenario lie in the absence of precise altitude and 3D environmental data and the subsequent alignment with the simulation environment. While SUMO can incorporate altitude data, OSM

[8] https://www.cam.mc/
[9] https://www.pacamobilite.fr/
[10] https://monaco-parking.mc/
[11] https://geoservices.ign.fr/

does not furnish such information. Furthermore, transforming topology from a graph to SUMO introduces complexities in mapping the IGN database to the resultant topology. These differences lead to a significant risk of generating unrealistic artefacts within the simulation. Moreover, an accurate depiction of topography is crucial when investigating topics like the influence of electric vehicles, eco-routing, and pollution.

The subsequent study [16] delves into privacy protection within VANETs, a notably challenging domain. Within VANETs, beacon messages, which enclose details such as vehicle position, speed, heading, and other pertinent information, are broadcasted at intervals of up to 10 Hz, rendering them susceptible to reception by any entity within their communication radius. In order to thwart the collection of mobility traces, an observer cannot establish a connection between a driver's messages over extended periods. At the same time, there is a requisite for message authentication to deter the proliferation of counterfeit or tampered messages, which could potentially jeopardise V2X-based assistance functionalities and driver safety. Each participant uses various pseudonyms that do not reveal (or allow for inference of) their true identity. The prevailing consensus for ensuring message authentication in forthcoming VANETs, conducive to privacy preservation, involves periodically altering pseudonym certificates. This approach has been formally embraced in recent standards ([36], [39]). This study evaluates the performance of four distinct pseudonym change strategies and their respective parameters through simulations of two large-scale and realistic traffic scenarios.

Despite the considerable body of research on pseudonym systems, there remains to be clarity regarding the frequency and timing of pseudonym changes required to attain adequate protection against tracking attacks. This study addresses this issue by quantifying the privacy yielded by pseudonym changes through the tracking error of an attacker endeavouring to link the origin and destination of all trips. Specifically, the focus is on an attacker with a limited reception area who can access multiple listening posts [38]. Although pseudonym alterations offer little protection against an attacker with unrestricted coverage ([6], [52]), the authors contend that it is challenging to conduct global surveillance of V2X messages and that a locally constrained attacker is considerably more likely. Additionally, the study assesses the efficiency of pseudonym changes, defined as the ratio between the attacker's tracking error (benefit) and the number of pseudonym changes executed (cost, encompassing factors like storage, bandwidth, computational overhead, and quality of service for V2X applications). According to the authors, this research represents the first examination of pseudonym change strategies for VANETs within large-scale, realistic traffic simulations when confronted with a realistic (local and passive) adversary.

The study's relevance for this context lies mainly in the realistic traffic scenarios utilised to evaluate the various privacy protection strategies. The authors used two distinct synthetic yet realistic large-scale traffic scenarios. Without suitable existing mobility traces for their large-scale analysis, as they often cover only a limited portion of the traffic (e.g., taxis or buses) or restricted scenarios (e.g., on private property), the authors developed a novel scenario. This scenario encompasses a 45 km highway segment located near Stuttgart, Germany, where they simulated 24 hours

of traffic. The second scenario employed in this study is the LuST scenario, introduced earlier in this section.

The authors developed the Stuttgart scenario using OSM data for the road network representation and traffic counts from the German Federal Highway Research Institute[12] and the regional road traffic centre Baden-Württemberg[13] to replicate actual traffic patterns. To enhance the realism of the simulation, the authors categorised vehicles into four types: passenger cars, motorbikes, buses, and trucks. SUMO was used to simulate traffic using the default car-following model, which delivers stochastic driving behaviour for individual vehicles. Real-world acceleration and deceleration parameters were configured for each vehicle class to ensure a more authentic representation of driving behaviour. In order to capture different traffic conditions, the authors generated both a low-traffic variant (between 1 and 4 a.m.) and a high-traffic variant (between 7 and 10 a.m.) for both scenarios.

The assessment of privacy protection strategies revolves around gauging their effectiveness and efficiency in protecting drivers from potential tracking by an adversary with limited coverage. An assessment framework, previously developed by the authors [15], was employed to conduct this evaluation. This framework quantifies the degree of location privacy achieved by considering the tracking error of the attacker. Each evaluation iteration involved a single simulation run for each combination of traffic scenarios, pseudonym change strategy, and associated parameters. The overarching objective was to identify which strategies and parameter settings are most suitable for practical deployment while simultaneously considering the costs associated with pseudonym changes, including considerations such as storage, bandwidth, and computation overhead. These cost-related factors are pivotal in shaping the broader adoption of V2X technology. In the context of an urban scenario, all strategies demonstrated an acceptable level of privacy protection, albeit with a relatively high frequency of pseudonym changes required. Conversely, in a highway scenario, the attacker's algorithm exhibited a notably high degree of tracking success across all strategies, particularly in low-traffic scenarios, even when employing short pseudonym change intervals.

From the observed results, the authors conclude that privacy protection on the highway is far more problematic than in urban contexts, owing to the more uniform traffic caused by the lack of intersections and traffic signals. Nonetheless, across both traffic scenarios under scrutiny, it emerges that the investigated privacy protection strategies exhibit a remarkably akin level of efficiency and efficacy in ensuring privacy. This outcome, while somewhat unexpected, underscores the robustness of these strategies. Additionally, they conclude that the pseudonym change strategy is irrelevant as long as pseudonyms are changed often enough. Periodic pseudonym change is the most straightforward strategy, where vehicles change their pseudonym at regular intervals after starting their journey. Therefore, the authors selected it as the most reasonable choice.

Regarding future work, the authors mention integrating new traffic scenarios in the simulations

[12] http://www.bast.de/DE/Verkehrstechnik/Fachthemen/v2verkehrszaehlung/Stundenwerte.html

[13] http://www.svz-bw.de/info_vm.html

and considering inter-trip tracking. Moreover, the refinement of their attacker algorithm presents an opportunity for enhancement through utilising vehicle-specific details gleaned from beacon messages (such as vehicle length) or driver behaviour. This enhancement could further narrow the list of potential candidates while resolving pseudonym changes.

2.5 Gap Analysis

This chapter consolidated the fundamental concepts associated with Digital Twins, resolving some terminology ambiguities. Furthermore, many studies pertinent to this subject were analysed, particularly emphasising those that developed traffic simulation scenarios based on real-world road networks.

The initial step of acquiring a digital network representation stands out among the typical stages of building these scenarios. As observed, this process has posed considerable challenges for researchers, demanding meticulous manual adjustments to ensure the fidelity of the networks to reality.

Regarding the types of data utilised, although some studies have employed mobile data or traffic demand information gathered through surveys, Inductive Loop Detectors continue to serve as the predominant data source. This preference owes to their data's simplicity, which generally comprises vehicle counts, and to the cost-effectiveness of these devices.

Despite its microscopic behaviour, SUMO has emerged as one of the most widely employed simulators, demonstrating its effectiveness even in large-scale scenarios.

A common aspect among all these works led to the emergence of the idea explored in this dissertation: they were all developed with an intense focus on the road network under study. Consequently, their methodologies were intricately tailored to the peculiarities of the respective scenarios, hindering the potential for the methodology's reuse in different contexts.

Thus, creating a generalised methodology applicable to diverse road networks would bring enormous utility to the ITS field. The following chapters delve into developing an automated tool for building Digital Twins.

Aside from the studies comprehensively discussed here, a few others deserve a brief mention. As such, Table 2.1 consolidates the primary attributes of other works pertinent to the prevailing context.

Table 2.1: Other Related Works

Ref	Year	Network	Data Sources	Virtual Model	Contribution
[3]	2022	The urban area of Barcelona, Spain	Hourly scaled OD flows derived from mobile phone data, gathered through annual mobility surveys	Large-scale 24-hour simulation in SUMO	Effective method for building a large-scale digital replica of traffic
[25]	2022	Every district in the city of Tartu, Estonia	Real data (coarse granularity) from IoT and static geo-data for building OD matrices	Estimation of hourly OD flows for each transportation mode	Simplified field test for estimating modal split within a two-hour interval in a small district
[41], [40]	2022	Metropolitan area of Turin, Italy, and its neighbouring districts	Aggregated traffic and street sensors data, including demand data	Large-scale 24-hour simulation in SUMO	Effective method for building a large-scale digital replica of traffic
[51]	2021	Docklands area in Dublin, Ireland	Dublinked (Open-data store)	3D smart city model with a pedestrian simulation mobility layer, in Unity	Digital feedback from residents on urban planning and policy decisions
[48]	2014	City of Cologne, Germany	Demand data from TAPASCologne initiative of the Institute of Transportation Systems at the German Aerospace Center (ITS-DLR)	Large-scale 24-hour simulation in SUMO	Effective method for building a large-scale digital replica of traffic
[24]	2014	Upper Austria state, Austria	Aggregated historical data from various sources, like traffic counters and floating car data	Traditional offline simulation approach in SUMO, which includes estimated probability distribution of variables	Method for computing short-term traffic data and dynamically adjust demand model
[46]	2014	City of Vila Real, Portugal	Survey and census data collected from PORDATA	Activity-based methodology for generating traffic demand in SUMO	Base scenario for the development of a tool that studies the deployment of Electric Vehicles

Chapter 3

Problem Specification

This chapter clarifies the problem addressed in this dissertation, laying the groundwork for the subsequent chapter, which delineates the proposed solution. First, it describes an approach for producing Digital Twins, which will later serve as the foundation for the proposed framework. Lastly, it presents a mathematical description of the problem, identifying its assorted entities and components.

3.1 Baseline Methodology

Driven by the absence of comprehensive guidelines regarding the systematic utilisation of third-party controllers to dynamically generate and calibrate traffic flow within running simulation scenarios of the microscopic traffic simulator SUMO, the works in [27] and [28] present a methodological approach for building a DT for the Geneva Motorway in Switzerland (DT-GM) in SUMO.

The real-time speeds and vehicle counts, along with their corresponding vehicle classes, are recorded by minute-resolution traffic counters and directly integrated into the active DT-GM, which continuously adjusts itself to its physical correspondence in real time, ensuring calibration on the fly instead of traditional offline calibration.

The traffic counters data, available through the Open Data Platform Mobility Switzerland (ODPMS)[1], is provided by the Swiss Federal Roads Office (FEDRO). ODPMS is Switzerland's customer information platform for public transport and individual mobility.

The source code of the DT-GM model is openly accessible[2], serving as a foundation for further investigation within the research community. However, one must register with ODPMS to get real-time traffic counter data to fully exploit the runtime DT-GM model.

The model described in this study will serve as a baseline for implementing the DT construction framework. Therefore, to successfully generalise this approach, it is essential to understand its methodology clearly.

Thereby, this section clarifies the methodology followed for developing the DT-GM.

[1] https://opentransportdata.swiss/en/cookbook/rt-road-traffic-counters/
[2] https://github.com/SiLab-group/DigitalTwin_GenevaMotorway

3.1.1 The Geneva motorway

DT-GM uses a network corresponding to a section of the A1 motorway in the Geneva region as the environment for microscopic traffic modelling. This motorway network encompasses a sizeable grade-separated interchange that connects to various key locations, including Geneva (east), the border with France (south), and Geneva Airport (north).

The authors initially sketch a schematic diagram to represent the network structure, denoting each road segment with a variable, as depicted in image 3.1. The numbering of the variables is irrelevant, and they will store the corresponding traffic flow $[veh/h]$. The variables will be denoted by x_i if the flow in that segment is unknown or by q_i if the flow is known owing to a nearby sensor.

The network comprises three main entry points and three main exits, all covered by sensors, identified as q_1, q_2, q_3, q_4, q_5, and q_6. In addition, the motorway contains entry ramps x_{11}, x_{12}, x_{15}, and x_{16} and exit ramps x_9 and x_{13}, whose flow is unknown.

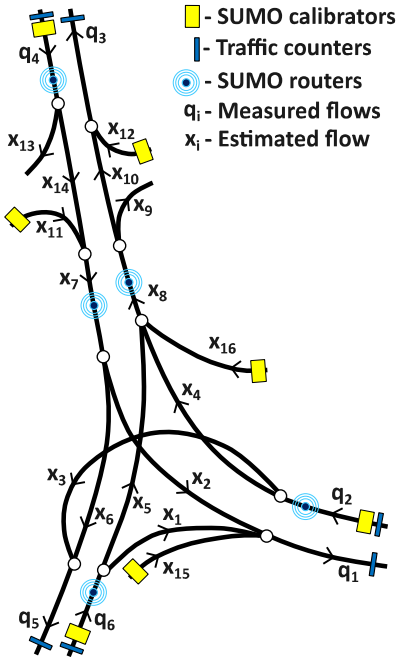


Figure 3.1: Model of the Geneva motorway network - Taken from [27], with permission.

In order to generate a simulation model that accurately reflects the geometry of an actual road network, it is necessary to convert geometric elements with their corresponding attributes into the simulation network model. Therefore, SUMO's NETEDIT module generates a digital motorway network from open-source geographic data from OpenStreetMap (OSM).

The authors began by acquiring a map of Switzerland's territory. Then they used the software `osmconvert` in conjunction with the software `osmosis` to apply filters to isolate the region of interest (Geneva Motorway). Then, using SUMO's `osmNetconvert` program, OSM data was converted into a SUMO network file (with extension ".net.xml") to be loaded into the SUMO engine.

3.1.2 Runtime Flow Estimation

As previously stated, surveying all roads would incur substantial costs from installing and maintaining numerous traffic counters. Consequently, the sensors' limited coverage becomes a severe issue, resulting in data-deficient portions of the network, thus creating uncertainty.

The proposed methodology employs a workaround to this barrier by estimating the traffic flow between locations, leveraging the available data from traffic counters that capture most incoming and outgoing motorway traffic.

Thus, from partial data, a system of linear equations (3.1) can be established based on the principle of traffic flow conservation. According to this law, the network's flow is balanced, meaning that the total flow entering the network equals the total flow exiting the network. This principle applies in the context of a motorway, assuming that the roads operate as one-way streets and any vehicle entering the observed network exits the network at some point (no terminal states in between).

Through this approach, the unknown traffic flows within the road network can be calculated or estimated. Subsequently, the authors proceed to derive the solution (3.2) of the linear system of equations (3.1), which represents the flow balance's general solution in the flow model.

$$
\begin{aligned}
x_1 + x_2 + x_{15} &= q_1 \\
x_3 + x_4 &= q_2 \\
x_4 + x_5 + x_{16} - x_8 &= 0 \\
x_1 + x_5 &= q_6 \\
x_2 + x_6 - x_7 &= 0 \\
x_3 + x_6 &= q_5 \\
x_8 - x_9 - x_{10} &= 0 \\
x_{10} + x_{12} &= q_3 \\
x_{13} + x_{14} &= q_4 \\
-x_7 + x_{11} + x_{14} &= 0
\end{aligned}
\tag{3.1}
$$

$$\vec{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{15} \\ x_{16} \end{bmatrix} = \begin{bmatrix} q_1 \\ 0 \\ q_5 \\ q_2 - q_5 \\ q_6 - q_1 \\ 0 \\ 0 \\ q_2 - q_1 - q_5 + q_6 \\ q_2 - q_1 - q_3 - q_5 + q_6 \\ q_3 \\ 0 \\ 0 \\ q_4 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 0 & -1 & -1 & 0 \\ -1 & 1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_6 \\ x_{11} \\ x_{12} \\ x_{14} \\ x_{15} \\ x_{16} \end{bmatrix} \tag{3.2}$$

In practice, all flows must be non-negative. Therefore restriction (3.3) is added to (3.2).

$$\vec{X} \geq 0 \tag{3.3}$$

This solution of the linear system of equations results in six free variables, given by x_6, x_{11}, x_{12}, x_{14}, x_{15} and x_{16}. The presence of free variables in a linear system of equations indicates that the number of equations is insufficient to uniquely determine the variables' values. The free variables are those that can take any value without affecting the solvability of the system. In general, their number corresponds to the dimension of the solution space of the system, and their presence indicates that the system has multiple solutions or an infinite number of solutions.

Understanding and identifying free variables while solving linear equations and analysing their solution space is crucial. By identifying the free variables, we can parameterise the solution and gain insights into the relationship between the variables within the system.

In this use case, it is necessary to estimate six free variables to obtain positive solutions. Thus, the authors employ a linear programming approach, which enables searching for feasible solutions using the Simplex algorithm.

Thus, to find the solution to the given problem, this methodology defines two steps of the Simplex algorithm. In the first step, the inequalities from (3.3) were used as constraints, and each Simplex run employed randomised cost coefficients of the objective function. This results in the definition of the extreme points of the feasible region for free variables $\vec{X}_{free} = [x_6, x_{11}, x_{12}, x_{14}, x_{15}, x_{16}]^T$.

The positive interval is established based on the calculated bounds for each free variable,

beginning with the smallest positive value and ending with the largest. This interval is then partitioned into ten equal segments, resulting in ten intensity levels for each free variable.

Once the feasible ranges are known, the solution space may be narrowed by specifying the desired intensity vector of the free variables $\vec{X}_{free-des}$. Accordingly, the second step of the Simplex algorithm imposes further constrictions on the range of each free variable based on the desired intensity level. Several Simplex runs again solve such a newly constrained problem.

The best-found solution of $\vec{X}_{free}$ corresponds to the one with the minimum relative error, calculated using the formula $\frac{\|\vec{X}_{feasible}-\vec{X}_{free-des}\|}{\|\vec{X}_{free-des}\|}$, which compares the desired intensity vector with the set of feasible solution vectors obtained.

Following the computation of all flow variables, the Dynamic Flow Calibration (DFC) mechanism determines the distribution of the routes, enabling the distribution of the vehicles (flows) produced by calibrators across the network to fulfil $\vec{X}$. The sections that follow will describe this mechanism.

3.1.3 Runtime Calibration

In SUMO, calibrator objects enable incorporating location-specific variations in traffic flow dynamics and driving behaviour. Once defined in the initial simulation scenario, they allow for dynamic adjustments of traffic flows, vehicle speeds, and other vehicle parameters by assigning predefined vehicle types. This dynamic feature enables the representation of evolving traffic conditions and the simulation of realistic driving scenarios.

Every calibrator is uniquely associated with a specific edge (representing a street within the simulation model) or a particular lane within that edge. Its utilisation requires defining an interval (start, end) whose length establishes the aggregation period when the comparison between observed and desired flows occurs.

The calibration goal is to achieve an appropriate deployment of vehicles at that specific location by the end of the defined time interval. Simultaneously, the spatiotemporal structure of the current traffic should also be maintained as much as feasible [31]. Thus, a calibrator eliminates vehicles that surpass the designated traffic volume while introducing new vehicles (of the specified type) when the regular traffic demand within the simulation fails to reach the desired number of vehicles per hour $[veh/h]$. Additionally, the calibrator assigns desired speeds $[m/s]$ to the vehicles.

An additional file must be imported and loaded during simulation start-up to define each calibrator's attributes. Consequently, in its default configuration, the calibrator functions as a static object, adjusting the traffic flow based on predefined attributes through distinct flows and speeds during different time intervals within the simulation.

When discussing real-time data, this becomes unfeasible, as the flows are unknown at simulation start-up time. Therefore, to solve this problem, SUMO allows access to the calibrator while the simulation is running via TraCI.

TraCI, a "Traffic Control Interface", provides live road traffic simulation access. It enables the retrieval of values from simulated objects and allows for manipulating their behaviour in real-time

during the simulation runtime [50]. TraCI allows invoking a particular calibrator and modifying the flow rate, speed, and vehicle type within the ongoing time interval. This capability facilitates dynamic adjustments within the simulation to reflect desired changes.

This approach establishes short (one-minute-long) intervals corresponding to the frequency of receiving traffic data from actual motorway sensors. As automated counting stations typically involve traffic counters installed per lane, the microscopic simulation model incorporates calibrators positioned at corresponding locations to the actual counters found on the motorway sections within the Geneva region.

Thus, with a one-minute resolution, calibrators are utilised to continually alter the flows to meet the present simulation traffic demand with changes in actual traffic on the actual motorway. Also, during the calibration process, a filter may be applied, which allows for the targeted modification of a specific vehicle type within the traffic flow by the calibrators. This feature enables the simultaneous usage of two calibrators on the same edge or lane, with one dedicated to calibrating cars and the other focused on calibrating heavy vehicles. Using separate calibrators for each vehicle type avoids interference or overlap.

3.1.4 Dynamic Vehicle Rerouting

The correlation between random and real paths determines how realistic the traffic flow is behind or between calibrators. This correlation assumes greater significance as the network expands in size and complexity, particularly between calibrated edges. Rerouting along with calibrators allows the desired traffic flow to match other edges (routes).

Therefore, after the calibrator has inserted the appropriate traffic flow, it may be disseminated across the network by allocating routes to individual vehicles. The distribution of routes can also be probabilistically determined, reflecting the likelihood of a vehicle following a particular route.

In SUMO, calibrator objects can interact with route probe detectors[3]. By specifying the `routeProbe` attribute in the calibrator definition, the probe detector selects a route from the distribution of sampled routes. This approach proves particularly beneficial for extensive city networks with numerous potential routes.

However, the authors predefined routes and route distributions in an additional file loaded at the simulation start-up. These routes are then dynamically assigned to vehicles using TraCI in conjunction with the calibrators. This alternative approach delivered them more control and flexibility within the experiment's context.

These concepts form the basis for introducing the DFC mechanism to reroute the traffic flows computed by (3.2) across the observed motorway. Figure 3.2 illustrates this mechanism, detailed in the following paragraphs.

[3] https://sumo.dlr.de/docs/Simulation/Output/RouteProbe.html

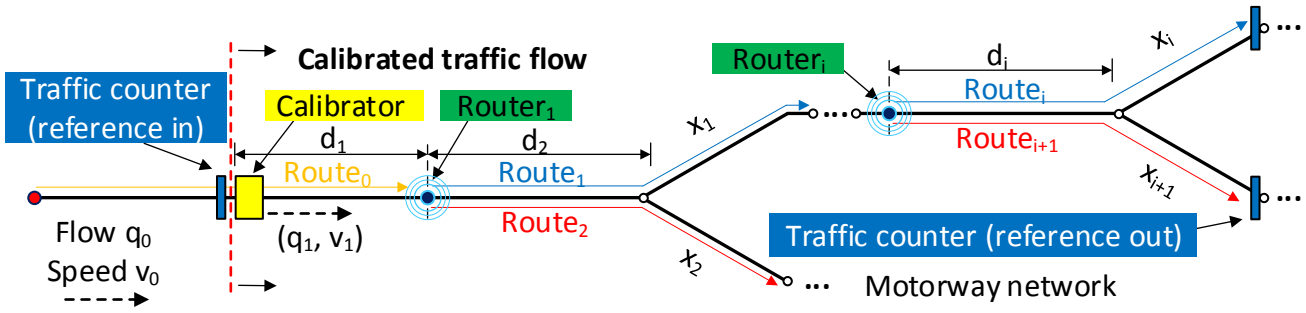


Figure 3.2: Dynamic Flow Calibration (DFC) mechanism - Taken from [27], with permission.

The locations of the traffic counters determine the DT-GM border points. The model is outfitted with calibrators at these places, from which the desired traffic flow (as assessed by traffic counters) is dispersed via the DFC mechanism to match the traffic flow on other motorway segments.

DFC incorporates data about all potential routes originating from a specific point, as manually defined in an additional file. Subsequently, the traffic flow information on the edges, as computed by equation (3.2), is employed to determine the probability distribution of traffic flow across all possible routes. Accordingly, the number of vehicles required to maintain their current direction of travel or divert onto alternative routes is calculated.

Figure 3.2 shows the whole operation of this mechanism. With the initial traffic flow q_0 and speed v_0 entering the model at the starting point, the calibrator dynamically adjusts the traffic flow and speed based on real-time measurements acquired from the traffic counters on the actual motorway. The altered traffic flow (q_1, v_1) continues along the established initial route $Route_0$.

The new route that best fits the required traffic flows on the ensuing areas of the network (calculated by (3.2)) is allocated to the vehicles whenever they arrive at the location where the routing device $Router_1$ is placed (according to the determined probability). For instance, the probability P_{x_1} of a vehicle being assigned to traffic flow x_1 (on $Route_1$) is calculated using the formula $P_{x_1} = 1 - P_{x_2} = 1 - x_2/q_1$. P_{x_1} and P_{x_2} probabilities are computed minute-by-minute and communicated via TraCI to the specific edge where routing occurs.

The route assignment probabilities among two potential routes are represented as ordered pairs, ensuring their sum always equals one. These probabilities take discrete values within the set $\{(P_{x_1}, P_{x_2}) : (1,0), (0.9,0.1), ..., (0.1,0.9), (0,1)\}$. Thus, rerouting presupposes that the required probability distributions have been saved in a separate file and loaded alongside other configuration files before the simulation start-up. This rerouting procedure follows the same principle consistently throughout the motorway model in areas where multiple travel directions are possible.

An extra `marker` is assigned to each vehicle upon insertion into the simulation to mitigate the risk of erroneous rerouting, particularly for vehicles entering the router area late due to potential congestion-induced delays. This marker includes information about the routes calculated by the DFC during the specific time window when the vehicle joined the simulation. Hence, when a vehicle reaches the router area, it is rerouted based on the encoded routes stored in its marker.

Additionally, the authors warn that it is crucial to maintain an adequate distance d_i between the router and the subsequent junction, ensuring sufficient space for seamless lane-change manoeuvres and enabling vehicles to transition smoothly onto their desired travel routes.

Algorithm 1 summarises the execution of the DT-GM microscopic simulation model.

Following the initialisation of all variables, the algorithm loads the initial simulation scenario $\vec{S}$ with all essential SUMO files. The simulation is then initiated and managed through TraCI. SUMO calibrates and updates the simulated traffic scenario at each simulation step.

At regular intervals of 60 seconds in real-time, a fresh request is dispatched to the FEDRO server via ODPMS to acquire up-to-date actual traffic data. The received data, comprising traffic flows ($\vec{q}$), speeds ($\vec{v}$), and vehicle types ($\vec{v}_{type}$), are subsequently relayed to the DFC mechanism. In this manner, calibrators $\vec{C}$ generate the desired traffic flow by considering the speeds and vehicle types traffic counters detect. Furthermore, DFC computes all network traffic flows ($\vec{X}$) and determines the corresponding route distribution probabilities ($\vec{P}$). These probabilities are instrumental in distributing the traffic flows generated by the calibrators throughout the network, ensuring that the overall traffic flow ($\vec{X}$) adheres to the desired conditions for the current one-minute time window.

This process repeats itself until the end of the simulation, regularly saving the DT-GM state. So, the process can be reloaded by loading the saved simulation state and resuming execution.

Algorithm 1 DT-GM algorithm

Init $\vec{S}, \vec{T}, \vec{C}, \vec{P}, \vec{q}, \vec{v}, \vec{v}_{type}, \vec{X}, \vec{X}_{free}, \vec{X}_{free-des}$ ▷ Set parameters and load simulation model
for each simulation step **do**
    if simulation time % 60 [s] == 0 **then**
        Get new actual traffic data via ODPMS: $\vec{q}, \vec{v}, \vec{v}_{type}$
        For given $\vec{X}_{free-des}$ and $\vec{q}$ calculate $\vec{X}$ ▷ DFC computations
        Update $\vec{C}$ and $\vec{P}$ using $\vec{X}$
    Calibrate and update simulation scenario
Save DT-GM simulation state and close simulation

3.1.5 Additional Model Considerations

This last subsection identifies pertinent details of the explained methodology, which come from this network's specificities. Consequently, addressing these matters is imperative when formulating the generalised methodology for building DTs applicable to any road network. The following chapter, Proposed Framework, will elaborate on the solutions to these challenges.

Starting, the model incorporates certain simplifications. To streamline the traffic flow model and minimise unknown variables, minor entrances and exits to the main sections, for which ODPMS does not provide data, have been excluded from the network model. So, manual modifications were made to the network using SUMO's NETEDIT module. The resolution of these motorway topology-related details ensures that the network accurately represents the desired features.

Upon defining the motorway network, the authors generate an initial traffic flow comprising pairs of edges, each consisting of a starting and ending edge. The sequence of edges between the starting and ending edges forms a route, which SUMO automatically assigns in this case. The initial traffic flow unleashes the oscillating traffic demand because the calibrator operates by waiting to observe if any traffic passes through, and if not, it starts yielding traffic to attain the desired traffic volume for the specified calibration period.

Given the use of very short time intervals for calibration, the generated traffic (during peak hours) may appear excessive towards the conclusion of the calibration period, leading to an additional accumulation of vehicles. The initial traffic flow aims to achieve a uniform traffic distribution to address this potential calibrator issue (which looks to be adjusted in the future by SUMO).

As a result, if the traffic counters detect no vehicles, the calibrators remove the incoming vehicles. Conversely, if the traffic volume recorded by the traffic counters exceeds the current generated traffic volume at a specific edge, the calibrator introduces additional vehicles to meet the required demand.

Therefore, these initial flows and the routes utilised in the rerouting process are previously defined in additional files loaded at simulation start-up and meticulously assembled by the authors according to the DT-GM network. Likewise, the placement of routers and calibrators is predetermined before the simulation and explicitly tailored to this network. Thus, all these components should be automatically generated by the DT building framework, as their configuration relies on the topology of the specific network under consideration.

Furthermore, the authors manually deduced the variables and equations according to this network, making them specific to this use case. The same applies to the solution of the linear system of equations used to determine the free variables. An intriguing prospect is automating this entire procedure, relieving the framework user from the burden of performing calculations and rendering this tool completely autonomous.

Moreover, the model's free variables, x_6, x_{11}, x_{12}, x_{14}, x_{15}, and x_{16}, are determined by selecting the desired intensity levels in the vector $\vec{X}_{free-des}$ rather than using absolute values. To estimate these intensity levels, the authors observed GPS traces on the OSM and Google Maps traffic websites concerning the time of day. Subsequently, they conducted runtime test simulations, refining the values accordingly. These intensities should only be considered an approximation rather than a precise depiction of the traffic volume on the respective motorway sections (edges). Despite being approximations, these intensity values are sufficiently accurate for DFC to generate a final feasible solution that satisfies constraint (3.3). The authors established eight intervals for the free variables based on the time of the day, each associated with corresponding desired intensity vectors $\vec{X}_{free-des}$. It is also evident that these intensities are specific to the road segments of this network, so it is relevant to find a way to obtain these values for any road segments whose assigned variable is considered a free variable.

The developed simulation employs mostly default SUMO parameters since the purpose of the study is not to calibrate these simulation parameters. Instead, the focus lies in investigating the resemblance of the DT-GM, configured with basic simulation settings (default parameters),

to actual traffic conditions. Hence, comparing actual and simulated traffic should be primarily considered a measure of how well the model reflects reality rather than an absolute performance metric.

Finally, the data used by the authors of this work follows a format emanating from ODPMS. The desire for a methodology applicable to any network implies establishing a standard format for input data.

3.2 Problem Formalisation

Formalising the problem provides essential groundwork for developing and implementing effective strategies. This formalisation clarifies the relevant components within the system of this problem, along with their interrelations.

The following sections detail the framework's methodology more comprehensibly, encompassing a delineation of the problem's scope, primary objectives, constraints, and decision variables. By specifying the problem space and refining the problem statement, one can set the stage for the subsequent analysis phases, algorithmic design, and performance evaluation.

Hence, in this section, an in-depth examination of problem formalisation is undertaken, delving into the translation of variables, constraints, and objectives into mathematical formulations.

Hereupon, the following definitions are established:

- **Road Network:**

  A road network is a directed graph $G(V,E)$, where V is the set of nodes, and E is the set of directed edges. An edge $\langle v_i, v_j \rangle$ denotes that node v_i is the incoming neighbour of node v_j. In this document, an edge corresponds to a road segment, and a node represents a junction of road segments or an intersection.

- **Network Entry:**

  Given a road network $G(V,E)$, its set of entry nodes V_{en} comprises all the nodes from which vehicles ingress into the network.

- **Network Exit:**

  Given a road network $G(V,E)$, its set of exit nodes V_{ex} comprises all the nodes from which vehicles egress the network.

- **OD Demands:**

  Given a road network $G(V,E)$, the origin-destination (OD) demands during a specific time interval t can be represented as a matrix $M^t = \{m_{i,j}\}$, where each element $m_{i,j}$ indicates the number of vehicles departing from node v_i to node v_j. In this context, $v_i \in V_{en}$ and $v_j \in V_{ex}$.

- **Route/Path:**

  A route/path $p = \langle v_1, v_2, ..., v_{|p|} \rangle$ represents a sequence of nodes encompassing all the nodes traversed by a trip originating from an OD demand. Here, $|p|$ denotes the number of nodes in p. In this context, $v_1 \in V_{en}$ and $v_{|p|} \in V_{ex}$. An OD demand can be associated with multiple routes. Routes will serve to guide the vehicles in the simulation.

- **Traffic Volume/Flows:**

  Given a road network $G(V,E)$, traffic volumes/flows can be represented as a traffic vector $X^t = \{x_i^t\}$, where each element x_i^t indicates the number of vehicles passing by edge e_i during a time interval t. Flows will serve as the primary source for comparing real and simulated data.

- **Split:**

  A split in a road network occurs when a single edge divides into multiple edges, representing a division of traffic flow. Given a splitting edge e_i, with flow x_i^t, the flow division in the split can be represented by $x_i^t = \sum_{n=j}^{k} x_n^t$. This equation means that the sum of flows on the split edges $\{e_j, ..., e_k\}$ equals the flow on the original edge e_i. It ensures the conservation of traffic flow at the point of the split in the road network.

- **Merge:**

  A merge in a road network occurs when multiple edges converge into a single edge, resulting in a combination of traffic flows. Given an edge e_i that results from a merge, with flow x_i^t, the flow combination in the merge can be represented by $\sum_{n=j}^{k} x_n^t = x_i^t$. This equation means that the sum of flows on the merging edges $\{e_j, ..., e_k\}$ equals the flow on the resulting merged edge e_i. This equation ensures the conservation of traffic flow at the point of merge in the road network.

- **Continuation Edge:**

  Given a road network $G(V,E)$, the set of continuation edges C_{e_i} associated with edge e_i is given by $C_{e_i} = \{e_j | e_j \in E, x_j^t = x_i^t$, and there are no splits or merges between e_i and $e_j\}$. The last part of the condition means that the linear path connecting e_i and e_j has no junctions or points where the road splits or merges. This formulation accurately captures the concept of a continuation edge, ensuring that the edges in C_{e_i} have the same flow, as they are connected linearly without any interruptions due to splits or merges.

- **Sensors:**

  Given a road network $G(V,E)$ located in a given geographical area A, S is the set of all ILDs placed within A. The position of these sensors indicates the collection points of real traffic data, which is crucial to bring the model closer to reality.

- **Covered Edges:**

  Given a road network $G(V,E)$, E_c are the edges of that network that contain sensors, so the flows through these edges are known. Thus, $E_c = \{e \in E : \exists s \in S$ placed in $e\}$.

- **Calibrators:**

  The set of calibrators C within a road network refers to points strategically positioned to calibrate the traffic simulation model. These components allow adjustments to model parameters for achieving a closer match between simulation results and observed real-world traffic behaviour.

- **Routers:**

  The set of routers R within a road network represents a collection of locations that serve as decision points for routing vehicles from their origins to their destinations. These components determine vehicle paths within the network, guiding their movement along the interconnected edges. Each router directs vehicles to the desired destinations, defining their paths from the subsequent edges.

Hereupon, the problem can be defined as follows:

Given any road network $G(V,E)$, the central goal of the problem is to devise an automated methodology replicating real-world traffic on this network. This replication should occur reliably leveraging the set of sensors R placed within that network, which provide data for the model calibration accomplished by a set of calibrators C.

The automated framework should comprise the following features to attain this goal:

- **Traffic Flow Replication:**

  The method should generate traffic flows on the network's edges that closely match the actual traffic flows observed in the real-world data collected by the sensors in set R.

- **Calibration:**

  The automated process should leverage the calibrators in set C to refine and adjust the traffic flow estimates, which includes fine-tuning the generated flows to align with the known traffic counts.

- **Sensitivity to Conditions:**

  The methodology should be sensitive to variations in traffic conditions, capturing fluctuations in traffic volume and patterns over different time intervals.

- **Accuracy and Reliability:**

  The generated traffic flows should be accurate and reliable representations of the real-world traffic, minimising discrepancies between the simulated and actual traffic.

- **Efficiency:**

  The automated process should be computationally efficient, allowing for timely traffic replication.

In summary, the problem's overarching goal is to establish a framework that bridges the gap between simulated and real-world traffic by leveraging available data from sensors and calibrators within the road network. This framework should enable accurate traffic replication of any road network, enhancing transportation planning, optimisation, and analysis.

Chapter 4

Proposed Framework

This chapter delineates the complete development process of the proposed framework, aimed at automating the creation of DTs. The framework drastically reduces the effort required to build a DT for any road network, making the process much more accessible. The chapter initiates with an exposition of the tool's architecture, subsequently elucidating its various phases.

4.1 Architecture

The developed framework can be divided into four primary phases, as shown in diagram 4.1. The central components represent these main phases, excluding the last one, depicted with dotted lines. This latter component pertains to the actual execution of the simulation, which this document treats as falling under the fourth phase, Digital Twin Automation. The bulkier black arrows denote manual user inputs, while the remaining arrows indicate the inputs and outputs of the various phases.

The first phase, Data Setup, is dedicated to assembling essential information for the subsequent phases. More precisely, its purpose is to identify the entry and exit points of the networks, as well as the coverage provided by the sensors, i.e., the edges for which flow data is available. Furthermore, it orchestrates the setup of the configuration file for the visualisation of the SUMO simulation. Lastly, this phase handles the organisation of data extracted from the ILDs regarding vehicle passages, structuring it in a manner compatible with the model utilised by the DT.

This phase's inputs include Excel files containing sensor data, one for each ILD. These files must adhere to the ILD standard data format to allow the data's transformation into the format required by the model. Additionally, this phase requires an Excel file consolidating the geographical positions of all the ILDs, precisely their latitude and longitude coordinates. This information is indispensable for determining their coverage and placing these components within the digital network. Another critical input, arising from what could be termed a preliminary stage of the framework, encompasses the SUMO files corresponding to the road networks earmarked for DT creation. In this preliminary phase, the user must simplify this network to enhance the efficiency of the DT while preserving its correspondence with the real-world road network.

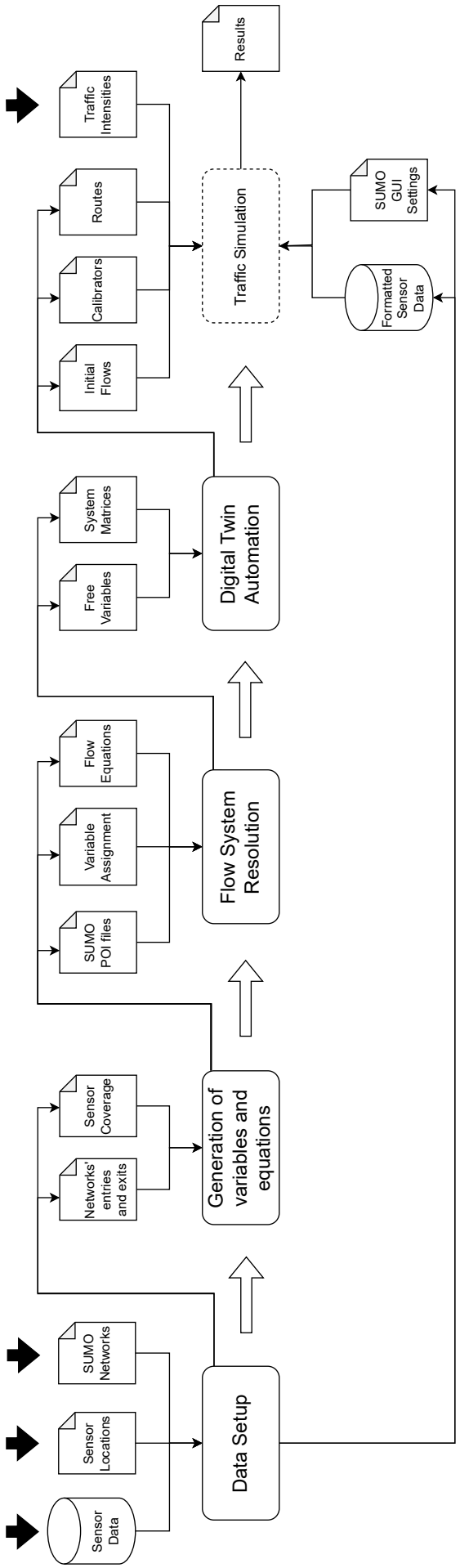


Figure 4.1: Architecture of the proposed framework

The outputs of this phase encompass a singular file that collates information from all sensors, conforming to the model's prescribed format (*sensor_data.xlsx* file within the *data* folder). Furthermore, this phase generates another file containing the entry and exit nodes for all processed networks (*entries_exits.md* in the *nodes* folder) and another file delineating the edges covered by each sensor (*coverage.md* in the *sumo* folder), besides preparing the GUI configuration file (*vci.view.xml* in the *sumo* folder).

The *prepare.py* file in the src folder contains all the code created for this phase.

The second phase (Generation of variables and equations) involves assigning variables to the diverse edges constituting the network, followed by the deduction of flow equations reliant on these variables, thereby relating the flows of the network's various roads. The inputs for this phase encompass the network entries, exits, and sensor coverage, all obtained in the preceding phase, in addition to the network files themselves. Its outputs are the mapping of variables to the assorted roads within the networks (kept as `pickle` files in the *nodes* directory) and the systems of equations for each network (in the *equations.md* file, also in the *nodes* directory). In addition, during this phase, the sensors of each network are recorded, as this information will serve to identify the relevant sensors for the execution of the DT of a specific network. The entirety of the code of this phase is accessible within the *variables.py* file located within the *src* folder.

The third phase, Flow System Resolution, focuses on resolving these systems of equations, aiming to detect the systems' free variables. Therefore, its inputs are the various equations deduced in the previous phase, and its output is the *free_variables.md"* file in the *nodes* folder, which contains the free variables of the system as well as the matrices and vectors required to calculate the remaining flow values from these variables. The *solver.py* file in the *src* folder contains the code for this phase.

Finally, the last phase, Digital Twin Automation, refers to the execution of a given network's DT. Its inputs comprise all of the information derived from the preceding phases, along with the *intensities.md* file in the *nodes* folder, which, as explained in subsection Traffic intensity on the free variables' edges, must be manually assembled by the user based on the free variables obtained from the third phase and the traffic intensities inferred from Google Maps. This phase results in the execution of the DT simulation in SUMO, outputting result files for each running hour, which contain the calculated and measured flows from the simulation. These files, located in the *results* folder within the *sumo* directory, will subsequently be harnessed in Chapter Results and Analysis for an in-depth assessment of the DT's performance. The code comprising all DT operations is available within the *digital_twin.py* file in the *src* directory.

4.2 Configurations

The configuration of all framework settings takes place within the config.ini file located in the project's root folder. Within this file, users can specify their input files' directories and define the simulation's desired parameters.

This file comprises five sectors, each dedicated to a specific type of input, more specifically:

- ***dir*:**

  Here, the user can customise primary directories, including the *data* directory that stores all sensor data and their corresponding locations, and the *nodes* directory, which contains the SUMO networks, as well as some network-related files generated by the framework, like the variable assignments induced for each network. Another example of a directory that can be changed is the *results* folder, which will store the simulation results files, later used to analyse the framework's performance in Chapter Results and Analysis.

- ***sensors*:**

  This sector identifies relevant files relating to sensors, including their locations, coverage and respective data on vehicle passages.

- ***nodes*:**

  In this sector, the user specifies all SUMO networks, along with the paths to files generated by the framework. These files encompass the entries and exits of each network, the respective sensors, equations, and free variables. Furthermore, the user must indicate the traffic intensities input file here, as elucidated in section Traffic intensity on the free variables' edges.

- ***sumo*:**

  This sector identifies the relevant files for the simulation. These encompass the SUMO binary file required for execution, the simulation configuration files (*.sumocfg* files), and the XML file configuring the simulation visualisation.

- ***params*:**

  This last sector enables the user to adjust simulation parameters. More precisely, it is possible to change the delay, which allows pausing the simulation for a specified number of milliseconds between each simulation step to improve visualisation, the number of simulated hours of traffic, and the step length, which defines the duration of each simulation step in seconds. Furthermore, this sector also allows modifying a value that slows down the simulation's execution speed, another that sets the number of seconds to wait before removing old vehicles from the permanent distribution lists (routing control), and the number of iterations for the Simplex algorithm.

Let us now go over how to use the framework. To commence, the user should create a Python virtual environment. These self-contained environments isolate Python packages and dependencies, facilitating the management of distinct projects. Virtual environments help avoid version conflicts, ensuring reproducibility and upholding a clean and organised development environment. Then, after activating the newly constructed virtual environment, the project's libraries must be installed.

To accomplish this, the user must execute the following commands:

```
python -m venv env
.\env\Scripts\activate
pip install -r requirements.txt
```

Listing 4.1: Activate Python Virtual Environment

Following these initial steps, the user is ready to execute the framework. For this purpose, one must use the commands defined in the Makefile within the project's root folder. The Makefile is a versatile and extensively employed build automation tool, empowering developers and data scientists to specify, arrange, and automate various tasks associated with a project. The framework's pipeline, elaborated in the preceding section, can be delineated through this tool, thereby simplifying the execution of its various stages.

More precisely, the user can run the following commands:

- ***make*:**

  This command executes the framework in its entirety. More specifically, it sequentially executes the four framework phases given by the subsequent commands.

- ***make prepare*:**

  This command initiates the first phase of the framework, which is responsible for preparing sensor data and helpful information for the subsequent phases, as elaborated in section Data Setup.

- ***make variables*:**

  Running this command commences the second phase of the framework, which is responsible for assigning variables to the road network edges and deducing flow equations, as detailed in section Generation of variables and equations.

- ***make solve*:**

  This command executes the third phase of the framework, which solves the systems of linear equations derived in the previous phase, retrieving the free variables of the system and the pertinent matrices for its resolution. Section Flow System Resolution explains this procedure.

- ***make run*:**

  Executing this command initiates the fourth and ultimate phase of the framework, which involves the actual simulation of traffic on the road network selected by the user. The reasoning of this entire phase is made clear in Section Digital Twin Automation.

- ***make results*:**

  This command triggers the analysis of the outcomes produced by the simulation during the final phase of the framework. Its execution yields the graphs expounded in Chapter Results and Analysis to assess the framework's performance.

- ***make clean*:**

  Lastly, this command automatically cleans files generated during the project's execution, returning it to its initial state after deleting the produced outputs.

4.3 Data Setup

The first phase of framework execution involves data preparation for simulation usage. In addition, this phase also includes deducing the coverage of existing sensors in each network, which is crucial for variable assignment and flow equation derivation in phase Generation of variables and equations. The same is true for the network's entries and exits, likewise discovered in this stage. Finally, the simulation's visualisation is prepared, with the definition of the values of some parameters, specifically its delay.

4.3.1 Sensor Data

A standard format for the sensors' input data is necessary to achieve a methodology applicable to any network. The framework adopts the same format used by DT-GM, as it aligns with its baseline methodology and feeds the model effectively.

This format comprises an Excel file containing multiple spreadsheets. The first sheet includes a timestamp column, arranged in ascending chronological order, representing the hours of a given day. This timestamp information enables inferring the corresponding periods for the sensor data, which is also chronologically organised. Subsequently, individual spreadsheets are provided for each sensor, displaying the respective vehicle count data. The counts group into two types of vehicles, cars and trucks, with the latter representing larger vehicles. Thus, each sensor has four columns denoting car flow, car speed, truck flow, and truck speed. Each row in these spreadsheets corresponds to the vehicle counts recorded during a specific minute of a given day.

Naturally, while building new DTs, processing the raw data from ILDs will be necessary to transform it into the desired format. The data is already in this format in the Geneva Motorway use case, so additional processing was not required. However, in the VCI use case, the data provided by ARMIS was unclustered, thus necessitating this processing.

Therefore, this processing seeks to resort solely to standard aspects of ILDs data. Each row of this data typically corresponds to a vehicle passage, coupled with the passage's ID, respective timestamp, inferred vehicle class and detected speed. There may also be additional properties, such as the vehicle's number of axles, length, or indication if it is going in the opposite direction or speeding. However, these attributes are not pertinent to the required format.

As in the case of the DT-GM input file, the processing seeks to compile all of the sensor data from a particular network into a single Excel file. It begins by recording the set of timestamps present in the data from each sensor and verifying if there are any inconsistencies in these times (e.g., mismatched timestamps among sensors on the same network). If they are valid, the program creates a spreadsheet with these timestamps. Afterwards, it maps the existing classes to the two types of vehicles used by the model. Classes A and B are categorised as cars (smaller vehicles), while C and D are considered trucks (larger vehicles).

Then, the program groups the data of these two vehicle types into one-minute blocks, creating the four columns mentioned above for each sensor. The flows (for cars and trucks) correspond to the count of IDs of the passages of vehicles of that type. As for the speeds, it calculates the average speeds of these passages for the current minute.

With this processing, the framework gains the ability to cope with data from ILDs in any road network, contributing to its generic nature. It is essential to acknowledge that the current approach employs data from the past due to the unavailability of real-time access to sensor data for the VCI use case. The framework may, however, be readily modified to consume real-time data by obtaining the accumulating vehicle counts every minute. Alternatively, the model can receive each vehicle passage individually for even greater precision, updating the simulation at each step accordingly.

4.3.2 Sensor Coverage

Identifying the roads with known flow (from sensors), denoted as covered edges, is paramount. Flows on roads lacking sensors will be estimated based on the flows of these covered edges, representing actual traffic on those roads. Therefore, it is convenient to identify the covered edges in this initial phase before deriving the flow equations.

The locations of the sensors must be known for this purpose, being available in the file *sensor_locations.xlsx* inside the *data* folder. Due to its significance, this file will serve as one of the input files for the framework. Each row in the file corresponds to a sensor associated with the network to which it belongs (VCI/Geneva Motorway), its equipment ID and respective coordinates in the format $(longitude, latitude)$.

For each sensor, the program carries out a coverage detection process. The process initiates by converting the geographical coordinates $(longitude, latitude)$ to Cartesian (x, y) coordinates in the simulation's network. This conversion is necessary as SUMO's simulation space operates on a two-dimensional Cartesian coordinate system.

Afterwards, it seeks the lanes nearest to the sensor's coordinates using a SUMO function that retrieves a list of neighbouring lanes within a specified distance (radius) from a given point in the simulation's road network. In this case, the program employs a radius of 50 meters, leading to the retrieval of all lanes within that distance from the sensor. Then, the program sorts the returned list in ascending order based on their distances, retrieving the first element of the list, which corresponds to the lane nearest to the sensor.

This procedure determines the network edge where the sensor is situated. However, the coverage of the sensor may extend beyond that edge. Some edges may function as a continuation of that edge, forming a continuous path without any entries or exits. This document will designate these linear edges as continuation edges for clarity and consistency, as defined in Problem Formalisation. So, the sensor also covers the continuation edges of the sensor edge. As a reminder, this occurs due to the principle of traffic flow conservation, stated in subsection Runtime Flow Estimation. This principle ensures the absence of terminal states within the network, meaning all vehicles entering it must ultimately exit. As a result, both the sensor edge and the corresponding continuation edges will have the same flow.

Hence, implementing a mechanism to determine these continuation edges becomes necessary. The mechanism starts by gathering the incoming and outgoing edges of the sensor edge, as continuation edges work both ways. In other words, the continuation edges preceding and following the sensor edge carry the same flow.

Then, starting the analysis in the opposite direction to the direction of the sensor edge (preceding edges), the program checks the number of incoming edges retrieved. If the sensor edge has multiple incoming edges, it results from a merge, so there are no continuation edges in the preceding direction. If it only has one, the program still needs to check the number of outgoing edges of that incoming edge. This verification must happen because that edge may have more than one outgoing edge, indicating a split before the sensor edge. Consequently, that incoming edge is not a continuation edge. Otherwise, if that edge only has one outgoing edge, it is a continuation edge, and the sensor's coverage incorporates it.

The processing does not end there, as this continuation edge may have further continuation edges in the preceding direction. Thus, it will be necessary to repeat this procedure, but now for the incoming edges of the continuation edge. The process ends when it finds an edge that is not a continuation edge.

It is also necessary to carry out this process in the opposite direction, exploring the subsequent edges to the sensor edge. So, starting from the outgoing edge of the sensor edge (if it only has one), the program checks its incoming edges, verifying that it does not result from any merge, and stores the continuation edges along the way.

After checking both directions, the mechanism acquires the entire coverage of the sensor, which is a list of its covered edges. Lastly, it registers this information in the file *coverage.md* in the *sumo* folder, displaying the IDs of the covered edges for each sensor.

Algorithm 2 comprises this whole mechanism. It uses the function 3 to find the sensor edge.

4.3.3 Network Entries and Exits

The framework's next phase, Generation of variables and equations, initiates the assignment of network variables through its entries and exits. Hence, it is essential to determine them at this stage. So, a search for the entries and exits of each network defined in the *config.ini* file begins, registering them in the *entries_exits.md* file within the *nodes* folder.

Algorithm 2 Sensor coverage detection mechanism

```
Init network                                                    ▷ Load the road network
procedure GEN_COVERAGE
    radius ← 50
    for each row of the sensor locations file do
        coverage ← []
        sensor_id ← equipment ID
        long, lat ← sensor location coordinates (longitude, latitude)
        x, y ← network.convertLonLat2XY(long, lat)              ▷ Convert coordinates to 2D space
        edge ← get_closest_edge(network, x, y, radius)          ▷ Find the sensor edge (3)
        previous_edges ← list(edge.getIncoming().keys())
        following_edges ← list(edge.getOutgoing().keys())
        while len(previous_edges) == 1 do                       ▷ Continuation edges before the sensor
            outgoing_edges ← list(previous_edges[0].getOutgoing().keys())
            if len(outgoing_edges) > 1 then
                break
            coverage.append(previous_edges[0].getID())
            previous_edges ← list(previous_edges[0].getIncoming().keys())
        while len(following_edges) == 1 do                      ▷ Continuation edges after the sensor
            incoming_edges ← list(following_edges[0].getIncoming().keys())
            if len(incoming_edges) > 1 then
                break
            coverage.append(following_edges[0].getID())
            following_edges ← list(following_edges[0].getOutgoing().keys())
    Write the sensor coverages to the coverage.md file in the sumo folder
```

Algorithm 3 Function to find the lane closest to the sensor coordinates

```
Init network                                                    ▷ Load the road network
function GET_CLOSEST_LANE(x, y, radius)
    lanes ← network.getNeighboringLanes(x, y, radius)
    if len(lanes) > 0 then
        distancesAndLanes ← [(dist, lane) for lane, dist in lanes]
        Sort the list distancesAndLanes by the first element (distance)
        _, closestLane ← distancesAndLanes[0]                   ▷ Get the lane closest to the sensor
        return closestLane
    else
        raise "No edges found within radius for these coordinates"
```

This process involves examining all network nodes individually. The analysis concerns the amount of incoming and outgoing edges of the node. If the node has no incoming edges, it is an entry node. If it has no outgoing edges, it is an exit node. Recall that a network cleanup is performed immediately before this phase, removing all isolated nodes, i.e. those not connected to any edge.

There is also an additional case in which a node can simultaneously be an entry and exit node. This situation occurs when a node has an incoming edge and an outgoing edge that are not interconnected, so the node has no connections. Thus, the entry road into the network is entirely independent and unrelated to the exit road. The highlighted node in the VCI's Nó do Amial network (Figure 4.2) is an example of the latter scenario since it contains both an entry (the road on the left) and an exit (the road on the right).

Algorithm 4 demonstrates this process of finding the entries and exits of a given network.

Algorithm 4 Mechanism to find the entry and exit nodes of a given network

```
1:  Init nodes                                                    ▷ Load all network nodes
2:  procedure GEN_ENTRY_EXIT_NODES
3:      entry_nodes ← []
4:      exit_nodes ← []
5:      for n in nodes do
6:          in_edges ← n.getIncoming()
7:          out_edges ← n.getOutgoing()
8:          if len(in_edges) == 0 then
9:              entry_nodes.append(n)
10:         else if len(out_edges) == 0 then
11:             exit_nodes.append(n)
12:         else if len(in_edges) == 1 and len(out_edges) == 1 and not n.getConnections() then
13:             entry_nodes.append(n)
14:             exit_nodes.append(n)
15:     Write the entry and exit nodes to the entries_exits.md file in the nodes folder
```

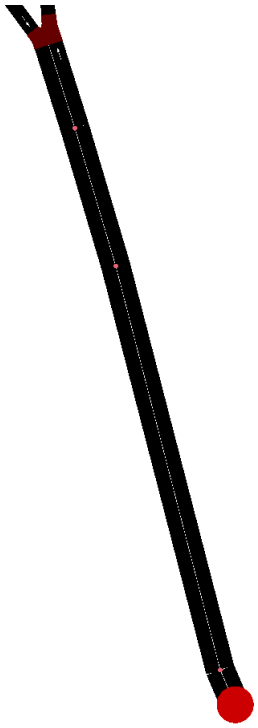


Figure 4.2: Example of a node that is both a network's entry and exit. Taken from the VCI's Nó do Amial network.

4.3.4 GUI Settings

Additionally, the program sets the simulation delay to 20 milliseconds. This value slows the simulation by pausing for the specified number of milliseconds between simulation steps. By default, the delay is set to 0, which may result in an overly fast simulation that is difficult to observe. So, the framework raises the delay value to enhance the visualisation of the process, making it more discernible for viewers.

The simulation visualisation is set up in file *vci.view.xml*, located within the *sumo* folder, often known as a GUI-settings file within SUMO environments. This file defines a scheme named `speedLarge`, specifying the visualisation parameters for each simulation entity, including vehicles, pedestrians, and network edges. This scheme will be vital for viewing the POIs corresponding to the network variables and routers generated in the second phase of the framework execution, as it enables viewing them on a large scale, regardless of the network size. The file also defines colour schemes for each entity, assigning them colours based on specific attributes. For instance, the colours of the vehicles can be displayed based on their waiting time or CO_2 emissions, where light colours may represent low values of these attributes and dark colours indicate higher values. Similarly, a scaling scheme can be employed to represent these attributes using the size of the vehicles.

Users can experiment with all these configurations while the simulation runs by interacting with the *View Settings* dialogue in the SUMO GUI.

4.4 Generation of variables and equations

As previously mentioned, the DT-GM authors manually deduced the variables and equations, considering the Geneva Motorway network. For the framework to work for any road network, it is necessary to automate this entire process, relieving the framework user from the burden of performing calculations and rendering this tool fully autonomous. Therefore, this second phase of the framework applies the automatic assignment of variables to the roads in the network and the subsequent derivation of flow equations.

4.4.1 Assignment of variables to network roads

Variables and equations are generated individually for all networks defined in the *config.ini* file. The variable assignment is stored as a Python dictionary in a `pickle` file, with each network having its own. This file type serves to serialise a Python object structure, allowing its effortless deserialisation.

The variable assignment process involves examining the entire network by individually analysing its nodes through a processing queue, which stores the nodes to be analysed next. It is essential to note that this attribution occurs at the level of the lanes, not the edges, due to instances where the lanes of a particular edge have different flows, as they are interconnected to different edges. Hereupon, the format of this dictionary that stores the outcome of this entire process is defined here:

```
{
    'edge_id_1': {
        'root_var': 'variable_1',
        'lane_id_1': 'variable_2',
        'lane_id_2': {
            'connection_1': 'variable_3',
            'connection_2': 'variable_4'
        },
        'lane_id_3': 'variable_5',
        ...
    },
    'edge_id_2': {
        'root_var': 'variable_6',
        'lane_id_4': 'variable_7',
        ...
    },
    ...
}
```

The dictionary comprises network edge IDs as keys, each containing an inner dictionary with additional information for the respective edge. Within the inner dictionary, there is always a `root_var` key, establishing the total flow variable for that road. Then, there is a key for the ID of each lane belonging to that edge. The values associated with the lane IDs can be either variable strings or nested dictionaries representing the lane's connections. This last nesting accommodates situations where a single lane has multiple associated variables due to connections leading to more than one edge. This structure allows for a flexible representation of edge-related information.

Before delving into the algorithm designed for this procedure, it is vital first to describe the variable generation process, including its nomenclature and representation.

For its nomenclature, the required parameters include the edge to which the variable is to be associated, the edges covered by all the sensors (discovered through the mechanism of subsection Sensor Coverage), and the current count of the variables as their numbering increases during the assignment process. The name of the flow variable starts with "q" if its value is known, meaning if sensors cover the corresponding road. Otherwise, it begins with "x", following the baseline methodology. Therefore, when generating a new variable and associating it with an edge, the framework checks if it is one of the edges covered by the sensors. Moreover, if a sensor covers the edge, the framework stores it in a dictionary containing the sensors of the current network and their respective covered edges, later saving them in the file *network_sensors.md* in the *nodes* folder. This information will be helpful in the last phase of the framework, Digital Twin Automation.

Regarding the representation of the variables, the program creates a POI file for each network, also in the *nodes* folder like the road network files. Thus, the user may easily visualise the generated variables by opening the desired network on the NETEDIT platform and loading the corresponding POIs in File → Additionals and Shapes → Load Additionals. To visualise the points on a large scale, the configuration file for the GUI view, described in GUI Settings, must be loaded. This action is done by navigating to Edit Coloring Schemes → Import View Settings.

Each variable will be assigned a pinpoint with unique features based on its defined type. Three different types are specified, described in Table 4.1. Additionally, as described in Table 4.2, seven different colours are assigned to the pinpoints, each with a specific meaning. These colours mainly enable distinguishing which stage of the algorithm generated the variable.

Table 4.1: Types of pinpoints and corresponding positions and colours

Type of pinpoint	**Position**	**Colours**
router	At 75% of the edge length	green
flow variable	At the center of the edge (50% of its length)	cyan, dark magenta (128,0,128), blue, yellow, yellowish brown (128,128,0)
flow variable (lane)	At 85% of the edge length	navy blue (0,0,128)

Note: For positioning the pinpoint on the chosen edge, the program obtains the edge shape, which mimics its geometry, from a polyline representing the lane's centre line. Then, an interpolation allows the POI to be positioned based on the specified distance.

Table 4.2: Meaning of the pinpoint colours

Colour	**Meaning**
Green	Represents a router
Yellow	Represents a network entry
Yellowish Brown (128,128,0)	Represents a network exit
Cyan	Represents a road resulting from a simple split or merge (not occurring at a complex node)
Dark Magenta (128,0,128)	Represents a road resulting from a merge that occurs at a complex node
Blue	Represents a variable forcibly assigned in the second iteration of the algorithm due to pending merges
Navy Blue (0,0,128)	Represents a division of the flow of a given edge by its lanes

Returning to the variable assignment algorithm, it starts by analysing the network's entry and exit nodes, determined in the previous phase of the framework, in Network Entries and Exits.

Regarding the entry nodes, the program verifies that each has only one outgoing edge and raises an exception otherwise. The next step is to create a variable for each network entry edge whose lanes carry this same variable. Then, per the tables above, the yellow pinpoint of type `flow variable` is formed. Finally, the program adds each of these edges to the process list.

Similarly, a brief check is performed for the network's exit edges to ensure each has just one incoming edge. The program also generates a variable for each exit and the corresponding yellowish-brown pinpoint of type `flow variable`. All lanes belonging to these edges also share the same variable. Afterwards, extra processing is necessary to associate these same generated variables with the continuation edges of the exit edges. The procedure is similar to detecting the sensors' coverage in Sensor Coverage. The algorithm must know when to stop processing a given sequence of nodes, which is when it discovers an exit edge or a continuation edge of that exit edge, whose variables are already assigned, hence the need for this procedure.

Finally, after these initial assignments, the main algorithm is called for assigning the intermediate variables of the network. Lastly, after generating all variables, the framework creates the network's POIs file.

The main structure of the variable assignment algorithm, as shown in 5, comprises this reasoning.

Algorithm 5 General structure of the variable assignment algorithm

```
1:  Init network, entry_nodes, exit_nodes   ▷ Load the road network and its entry and exit nodes
2:  function GEN_COVERAGE
3:      variable_count ← 1
4:      variables ← { }
5:      process_list ← []
6:      for each entry node do
7:          if len(entry.getOutgoing()) > 1 then
8:              raise "Entry node has more than one outgoing edge. Please adapt the network."
9:          entry_edge ← entry.getOutgoing()[0]
10:         variable ← the next variable name
11:         Update variables, assigning variable to the entry edge and its lanes
12:         gen_pinpoint(entry_edge, variable, "flow variable", "yellow")
13:         Add entry_edge to process_list and increment variable_count
14:     for each exit node do
15:         if len(exit.getIncoming()) > 1 then
16:             raise "Exit node has more than one incoming edge. Please adapt the network."
17:         exit_edge ← exit.getIncoming()[0]
18:         variable ← the next variable name
19:         Update variables, assigning variable to the exit edge and its lanes
20:         gen_pinpoint(exit_edge, variable, "flow variable", "128,128,0")
21:         Increment variable_count
22:         // Assign variable to the continuation edges of the exit edge
23:         previous_edges ← list(exit_edge.getIncoming().keys())
24:         while len(previous_edges) == 1 do
25:             outgoing_edges ← list(previous_edges[0].getOutgoing().keys())
26:             if len(outgoing_edges) > 1 then
27:                 break
28:             Update variables, assigning variable to the continuation edge of the exit edge
29:             previous_edges ← list(previous_edges[0].getIncoming().keys())
30:     equations ← calculate_intermediate_variables(process_list, variables)
31:     Create the POIs file for the current network in the nodes folder
32:     return equations
```

The algorithm's main logic, which entails assigning the network's intermediate variables, may be broken into two iterations.

The first iteration involves executing the central processing for the process list derived from the initial step, which comprises the network's entry nodes. Following this iteration, the program traverses all network edges, and those that have no variable assigned are stored (`pending_edges`). Some of these edges may depend on other pending edges due to their interconnections. Deleting edges whose preceding edges are already on the list reduces the list of pending edges, preventing unnecessary processing.

The execution of the first iteration retrieves some edges that result from merges that could not be resolved in this iteration because the merging edges have no associated variable (`pending_merges`). An example of this situation occurs in roundabouts. As they lack a starting entry due to their circular format, the first iteration cannot resolve them because, for a given entry in the roundabout, the algorithm must know the flow of the roundabout section that connects to that entry to process the subsequent edge of the roundabout, which results from merging the entry with the previous edge of the roundabout.

As a result, the algorithm's second iteration involves traversing these edges, retrieving the corresponding merging edges, and forcibly assigning a new variable to those without one. This procedure applies solely to edges in `pending_merges` that are part of the previously discovered `pending_edges` to ensure that these edges were not processed later in the iteration. Then, the corresponding pinpoint of type `flow variable` is generated, now marked in blue to highlight that the algorithm forcibly assigned the variable during its second iteration. Afterwards, the program adds the merging edge to the process list and invokes once again the central processing.

The described process is repeated in this second iteration until no pending merges remain. After both iterations conclude, the program removes outdated POIs, corresponding to those forcibly assigned (blue colour) but later processed regularly. Finally, the program creates the `pickle` file that stores the network variables.

Function 6 displays the iterations of the variable assignment algorithm.

Algorithm 6 Iterations of the Intermediate Variables Assignment Algorithm

```
1:  Init network                                                  ▷ Load the road network
2:  function CALCULATE_INTERMEDIATE_VARIABLES(process_list, variables)
3:      equations ← {}                                            ▷ First Iteration
4:      pending_merges ← process(process_list, variables, equations, pending_merges)
5:      pending_edges ← []
6:      for each edge in network.getEdges() do
7:          if edge.getID() not in variables then
8:              pending_edges.append(edge)
9:      for each pend_edge in pending_edges do
10:         previous_edges ← list(pend_edge.getIncoming().keys())
11:         for each pe in previous_edges do
12:             if pe in pending_edges then
13:                 pending_edges.remove(pend_edge)
14:                 break
15:     while pending_merges do                                   ▷ Second Iteration
16:         following_edge ← pending_merges.pop()
17:         merging_edges ← following_edge.getFromNode().getIncoming()
18:         for each edge in merging_edges do
19:             if edge in pending_edges and edge.getID() not in variables then
20:                 Generate a new variable and assign it to edge and its lanes
21:                 gen_pinpoint(edge, variable, "flow variable", "blue")
22:                 Increment variable_count and add edge to the process_list
23:                 break
24:         pending_merges ← process(process_list, variables, equations, pending_merges)
25:     Remove outdated POIs
26:     Register the variables assignment in a pickle file in the nodes folder
27:     return equations
```

It now remains to explain the central processing of the algorithm, which is responsible for analysing all the edges in the `process_list`.

Processing one edge at a time, it starts by obtaining the destination node of the edge and the respective connections. Subsequently, it builds a dictionary based on these connections (`connection_pairs`), whose keys are the source edges (`from_edges`), and values contain the respective destination edges (`to_edges`) and the connections that link them. Then, it reverses the dictionary, yielding a new dictionary with the `to_edges` as keys and the `from_edges` and corresponding connections as values (`reversed_pairs`).

These two dictionaries will be crucial for quickly analysing the current node as they facilitate counting incoming and outgoing connections of each edge. For instance, one can easily

find the `from_edges` of the current node, represented by the keys of the first dictionary, and its `to_edges`, represented by the keys of the reversed dictionary.

The number of `from_edges` and `to_edges` of the current node determines the scenario to be analysed, resulting in four possible scenarios.

The first scenario, also the simplest, occurs when the node contains only one `from_edge` and one `to_edge`. In such a case, the destination edge is a mere continuation of the source edge. If the source edge has no variable assigned yet, the node cannot be processed, and the algorithm proceeds to the next edge in the `process_list`. Otherwise, the destination edge adopts the same variable as the source edge for all its lanes since the flow remains the same along these edges. Subsequently, the program adds the destination edge to the end of the `process_list` and moves on to the next edge in the list.

The second scenario pertains to a split, wherein an edge splits into two or more edges, resulting in a flow division. Hence, it occurs when the current node encompasses a single `from_edge` and several `to_edges`. In this situation, generating and assigning new variables to the edges resulting from the split becomes essential unless they already have associated variables. Ultimately, the program adds the edges with newly generated variables to the `process_list`, continuing processing afterwards.

The third scenario relates to a merging junction, where numerous edges combine into one, resulting in a flow confluence. So, it occurs when the node has multiple `from_edges` but only one `to_edge`. In this case, the first step is determining whether the merge is processable. For this, the program verifies if any lane of the merging edges has no assigned variable. If so, the node is not processable, and the edge resulting from the merge is added to the `pending_merges` to be addressed in the second iteration of the algorithm. Otherwise, a new variable is generated for that edge if it does not already have a variable assigned. Then, all merging edges are removed from the `process_list`, as they no longer require processing. Conversely, the outgoing edge is added to this list to continue the processing. Lastly, if present, the outgoing edge can be removed from the pending merges list, as the merge has already been processed, eliminating the need for further handling in the second iteration of the algorithm.

The generated pinpoints for these first three scenarios are of type `flow variable` and colour cyan.

Finally, the fourth scenario involves more complex nodes with several source and destination edges. These nodes can contain any scenarios mentioned above: continuation edges, splits and merges.

The program begins by separately analysing each of the `from_edges`. If it has only one `to_edge`, it corresponds to the first scenario, and this continuation edge and its lanes are assigned the variable `root_var` of the `from_edge`. The process list is then updated to include the `to_edge`.

Another case analysed here occurs when the `from_edge` leads to several `to_edges`, representing the second scenario, a split. In this situation, the `from_edge` flow must be divided into numerous variables, one for each lane that leads to a different `to_edge`. The `root_var` of the

`from_edge`, which sets the overall flow of that edge, is kept. It then generates a variable for each lane of the `from_edge` that leads to a different destination edge. This time, the generated pinpoint is placed at lane level, labelled as `flow variable (lane)`, and coloured navy blue to indicate that these variables resulted from dividing the flow of an edge by its lanes. Finally, the program adds the `from_edge` to a list called `divided_edges`, which allows it to track the edges whose flow has already been divided by their lanes. This list will help determine when merges of lanes from different edges can be processed.

The program then proceeds to explore the `to_edges` of the current node and analyse each individually. The first possible scenario to encounter here is also the continuation edges. Most of these were already processed in the previous step when analysing the `from_edges`. However, some continuation edges could not be processed earlier because the connection linking them to the `from_edge` involves only some `from_edge` lanes. It is only possible to process these continuation edges after dividing the `from_edge` flow by its lanes. Thus, if the `from_edge` flow has already undergone division and the `to_edge` does not yet have a variable, the program assigns the variables from the preceding lanes to their corresponding lanes. After this, it adds the `to_edge` to the process list. Another possible scenario is the merging of lanes from different edges. Processing this case is only possible when all `from_edges` have an assigned variable. Otherwise, the program adds the `to_edge` to the `pending_merges` list for processing in the second iteration of the algorithm and proceeds to process the next `to_edge`. If processable, the program generates a new variable for the merged edge. It also generates a pinpoint of type "flow variable" and coloured dark magenta, indicating that the generated variable resulted from a merge in a complex node. Lastly, it adds the `to_edge` to the process list.

Finally, router generation happens during this process. Whenever the program identifies a splitting edge, a new router is created and positioned on the edge just before that splitting edge to ensure adequate space for vehicle rerouting. The program then creates the pinpoint of each router with a green colour.

Function 7 shows this central processing of the variable generation approach.

In short, this subsection has described the assignment of variables to the network's edges. As seen, the topology of road networks involves several scenarios to consider in this assignment, including flow divisions and confluences. It is important to note that the equations are generated simultaneously with this process, as detailed in the following subsection.

Algorithm 7 Central processing of the Variable and Equation Generation Algorithm

Init *network* ▷ Load the road network
function PROCESS(*process_list*, *variables*, *equations*, *pending_merges*, *divided_edges*)
    while *process_list* **do**
        Get the next *edge* in the *process_list* and the *connections* of its destination node
        Build the *connection_pairs* and *reversed_pairs* dictionaries
        conn_incoming ← list(*connection_pairs*.keys())
        conn_outgoing ← list(*reversed_pairs*.keys())
        if len(*conn_incoming*) == 1 and len(*conn_outgoing*) == 1 **then**
            if *conn_incoming*[0] not in *variables* **then**
                continue
            previous_vars ← the previous variables assigned to the outgoing edge
            Update the outgoing edge's variables with the `root_var` of the preceding edge
            if *previous_vars* != *variables*[*conn_outgoing*[0]] **then**
                Update the outdated equations by replacing the old variables with the new ones
            Add the outgoing edge to the *process_list*
        else if len(*conn_incoming*) == 1 and len(*conn_outgoing*) > 1 **then**
            for each *edge* in *conn_outgoing* **do**
                if *edge* not in *variables* **then**
                    Generate a new variable for *edge* and the corresponding pinpoint (cyan)
                    Add *edge* to the *process_list*
            Create a new equation $X_s = X_{o_1} + X_{o_2} + ... + X_{o_n}$, adding it to *equations*
            Place a new router on the edge preceding the splitting edge if not already there
            Generate the corresponding router pinpoint (green)
        else if len(*conn_incoming*) > 1 and len(*conn_outgoing*) == 1 **then**
            Check if the merge is *processable* (if all merging lanes have a variable assigned)
            if *processable* **then**
                if *conn_outgoing*[0] not in *variables* **then**
                    Generate a new variable for *conn_outgoing*[0]
                    Generate the corresponding pinpoint (cyan)
                    Remove all merging edges from the *process_list*
                    Add the outgoing edge to the *process_list*
                Create a new equation $X_m = X_{i_1} + X_{i_2} + ... + X_{i_n}$, adding it to *equations*
                Remove the outgoing edge from the *pending_merges*, if present
            else
                Add the outgoing edge to the *pending_merges*

```
36:         else
37:             for each from_edge in connection_pairs.keys() do
38:                 if only one to_edge linked with from_edge and from_edge in variables then
39:                     if to_edge not in variables then
40:                         Update the to_edge's variables with root_var of the preceding edge
41:                         Add to_edge to the process_list
42:                 else if many to_edges, from_edge in variables and not in divided_edges then
43:                     for each to_edge do
44:                         Assign new variable to the from_edge lanes that lead to to_edge
45:                         Generate the corresponding pinpoint (navy blue)
46:                     Create a new equation X_s = X_l1 + X_l2 + ... + X_ln, adding it to equations
47:                     Add from_edge to the divided_edges
48:             for each to_edge in reversed_pairs.keys() do
49:                 if only one from_edge which is already in divided_edges then
50:                     if from_edge not in variables then
51:                         continue
52:                     if to_edge not in variables then
53:                         Update the to_edge's variables with the variables of the preceding lanes
54:                         Add to_edge to the process_list
55:                 else if to_edge not in variables then
56:                     Check if all from_edges have been processed
57:                     if any from_edge not processed then
58:                         Add to_edge to the pending_merges
59:                         continue
60:                     Generate a new variable for to_edge
61:                     Generate the corresponding pinpoint (dark magenta)
62:                     Create a new equation X_m = X_l1 + X_l2 + ... + X_ln, adding it to equations
63:                     Add to_edge to the process_list
64:                 else
65:                     Check if the merge equation is ready to be added (with updated variables)
66:                     If so, add the equation X_m = X_l1 + X_l2 + ... + X_ln to equations
67:     return pending_merges
```

4.4.2 Derivation of flow equations

There are two stages to the derivation of the flow equations. The first stage occurs concurrently with the variable assignment process, yielding many simple equations that relate these variables. The second stage revolves around simplifying and formatting these equations, creating a more manageable system of linear equations, and preparing it for its resolution, described in Flow System Resolution.

The initial stage requires acting in all four scenarios outlined in the previous subsection, including certain sub-cases within the fourth scenario.

In the first scenario, referring to continuation edges, the program obtains the variables assigned to the destination edge, if any. After assigning the variables of the preceding edge to the destination edge, it assesses whether these new variables differ from the prior ones. In cases of disparity, the equations require updating, as there might be outdated equations using the old variables. This rectification entails going through each equation, substituting old variables with those arising from this new assignment.

In the second scenario, which entails splits, the program creates a new equation that translates the consequent division of flows. As a result, this equation will follow the format $X_s = X_{o_1} + X_{o_2} + ... + X_{o_n}$, where X_s is the `root_var` of the splitting edge, whose flow equals the sum of the flows from all edges emerging from this split (outgoing edges). The network equations list is then updated to include that new equation.

The third situation, which refers to merges, likewise involves the creation of a new equation. However, only when the merge is processable, i.e., when all lanes of the merging edges have an assigned variable. In this case, the equation will take the format $X_m = X_{i_1} + X_{i_2} + ... + X_{i_n}$, where X_m is the `root_var` of the edge resulting from the merge, whose flow equals the sum of the flows of the merging edges (incoming edges). The equation is then also added to the list of equations.

Finally, acting on the fourth scenario's sub-cases is also necessary. First, in the case of dividing the flow of an edge by its lanes, the resulting equation will follow the format $X_s = X_{l_1} + X_{l_2} + ... + X_{l_n}$, where X_s is the total flow of the edge that underwent the division, corresponding to the sum of the variables generated for its lanes that lead to different destination edges. Furthermore, in cases involving merges within complex nodes where the `to_edge` has not yet received a variable assignment, the formulated equation takes the format $X_m = X_{l_1} + X_{l_2} + ... + X_{l_n}$. In this equation, X_m corresponds to the `root_var` of the edge resulting from the merge, encapsulating the summation of flows of the merging edges/lanes. Otherwise, if the `to_edge` already holds an assigned variable (such as when merges lead to network exits), one must check if the equation is ready to be added by verifying if all merging edge lanes have associated variables. Upon meeting this criterion, the equation takes the format $X_m = X_{l_1} + X_{l_2} + ... + X_{l_n}$, where X_m denotes the edge stemming from the merge, whose flow corresponds to the sum of the flows of the merging lanes.

The second stage of the equations' derivation process unfolds after the algorithm's central processing, which yields many flow equations, is complete. This stage begins by simplifying

the linear system by combining some equations and thus reducing their number within the system. Following this, the resultant equations undergo formatting to prepare them for the system resolution in the forthcoming phase of the framework, Flow System Resolution. This formatting consists of moving the constants to the right side of the equation and the variables to the left, besides ordering the variables and reducing the number of negative signs for improved readability.

The simplification process involves iterating through all equations. Note that, in the initial equations, the left side of the equation contains only a single variable. Thus, for each equation E_{q_1}, we search the remaining equations to find those whose right side comprises the variable on the left side of E_{q_1}. In those cases, the program replaces the variable with the respective expression (right side of E_{q_1}). It is worth highlighting that this procedure prioritizes replacing variables with higher numbering, aiming to retain those with lower numbers for simplicity. The new equation thus preserves the second equation's left side, and its right side is the result of substituting the variable within the second equation's right side for the expression from the right side of the first equation. After adding this new equation, the program adds the initial equation (E_{q_1}) to a list of outdated equations. Also, if no simplification has occurred because the variable is not present on the right side of any equation, the initial equation (E_{q_1}) remains within the system. After this simplification process, the program removes the outdated equations, culminating in a more manageable system.

Afterwards, these equations are formatted to make them conducive to system resolution. For every equation, the program moves the constants (numbers and q_i variables with known flow) from the left side to the right and the variables from the right side to the left. Subsequently, it counts the number of negative terms in the equation. If it exceeds the number of positive terms, it reverses its signs by multiplying both sides of the equation by -1. Finally, it concludes by ordering the terms per the variable numbering, ending with writing the equations to the *equations.md* file in the *nodes* folder.

4.5 Flow System Resolution

The main objective of this third phase of the framework is to find the free variables of the equation system derived in the previous phase. Accomplishing this requires solving the linear system of equations. Multiple approaches can address this problem, including Gaussian elimination or matrix inversion. These techniques involve performing row operations on the system matrix to simplify it and eventually express the variables through a series of independent variables (free variables).

The program starts by obtaining all the x_i variables present in the equations, ordered by their numbering. Afterwards, it constructs the augmented matrix for the system of linear equations, with dimensions MxN, where M is the number of equations, and N represents the count of existing variables plus one to account for the equations' constant expressions in the matrix's final column. This construction entails traversing the equations and generating a row of size N for each equation. Each position within the row corresponds to a specific variable, with the index of that position

denoting the variable's position within the sorted variable list. Within this row, the position for a given variable adopts the coefficient value of that variable in the equation. Each cell will usually take one of three values: 0 if the variable is not present in the equation, 1 if it is present as a positive term, and -1 if it is present as a negative term. Then, in the row's last position, the equation's constant expression is added, corresponding to its right side, owing to the formatting conducted in the previous phase of the framework. The augmented matrix of the system of linear equations is the result of this processing.

The next step is obtaining the augmented matrix's reduced row-echelon form (RREF). The RREF of a matrix is a specific form achieved through a series of elementary row operations. Its simple structure facilitates the analysis and resolution of linear algebraic problems. A matrix is often converted into its RREF using the Gaussian elimination approach, which involves several steps, including pivoting, scaling, and zeroing out entries. There are three types of elementary row operations on a matrix's rows:

- Swapping two rows;
- Multiplying a row by a non-zero scalar;
- Adding a multiple of one row to another row.

In row-echelon form, each row has a leading coefficient (or pivot) that is the first non-zero element of the row. The variables corresponding to the columns containing leading coefficients are called pivot variables. Conversely, columns with no leading coefficients refer to free variables. So, after converting the matrix to the RREF, determining the system's free variables is straightforward.

Subsequently, the program determines the inequality constraint matrix A. Each row of this matrix specifies the coefficients of a linear inequality constraint, specifically (3.3). In linear programming problems, constraints constitute linear inequalities of the form $Ax \leq b$, where:

- A is the inequality constraint matrix, where each row represents a constraint, and each column corresponds to a decision variable;
- x is the vector of decision variables to be determined;
- b is the inequality constraint vector.

The inequality constraint matrix is essential to the typical linear programming problem form. It helps describe the relationships between the decision variables and the constraints systematically and mathematically. Solving a linear programming problem entails determining the values of the decision variables that satisfy the constraints while optimizing the objective function, and the inequality constraint matrix plays a crucial role in this process. It is built by traversing all of the RREF matrix rows and appending a row with the values of the coefficients of the free variables in the current row of the matrix.

Determining the inequality constraint vector is another imperative task. The inequality constraint vector (b) corresponds to the constants on the right-hand side of the inequality constraints in a linear programming problem. It establishes the upper limits or bounds for each inequality constraint. Each vector element corresponds to a distinct constraint and sets forth the maximum

allowable value for the left-hand side of the inequality. Constructing this vector is straightforward, as it only involves extracting the final element from each row of the RREF matrix.

The next step is to build the $X_{particular}$ vector and the X_{null} matrix. The former originates from the inequality constraint vector, exhibiting a dimension equivalent to the total number of variables. This vector has the value of the inequality constraint vector at the positions corresponding to the free variables and the value 0 in the remaining ones. Likewise, the X_{null} matrix is crafted based on the inequality constraint matrix, and its dimension also equals the total variable count. The program adds a row for each variable. If it is a free variable, it inverts the signs of the respective row of the inequality constraint matrix and appends it to the matrix. Otherwise, it creates a new row with a dimension equal to the number of free variables, with all elements set to 0 except for the position of the free variable at hand, which holds a value of 1.

Assuming that the system of equations is consistent (has at least one solution) and independent (no equation is a linear combination of any other), then $FV = V - E$, where FV is the number of free variables, V is the total number of variables, and E is the number of equations in the system. Therefore, the program performs this verification to validate the calculations.

The framework performs this process for all existing networks, storing all built components (free variables, A, b, $X_{particular}$, X_{null}) in the `free_variables.md` file in the `nodes` folder.

4.6 Digital Twin Automation

This final section elucidates the Digital Twin implementation process and the various solutions developed to make the framework suitable for any road network. Recall that the implementation model draws from the model outlined in work [27], described in Section Baseline Methodology.

4.6.1 Initial collection of information

The program begins by gathering data from the files created in the previous framework phases, namely the network entries and exits, the sensors distributed throughout the network along with their edge coverage, and the edges where the routers stand. Other information collected includes the network variables and their corresponding edges, the free variables, and the actual data from the ILDs directly extracted from the unique file generated in Sensor Data.

Following this, the program prepares data structures for storing all the information necessary for the DT implementation. One is the permanent distributions, stored in a dictionary called `perm_dists`. This dictionary contains a key for each router. It holds information regarding the number of new vehicles entering the network and the route distribution applied by the router throughout each simulation interval. Another dictionary relates to sensors, storing the values of the flows and speeds of cars and trucks for each sensor during the current minute. In addition, another one controls the vehicles in the network, keeping a list of IDs of the vehicles that cross each entry and exit.

4.6.2 Calibrator Generation

Before commencing the simulation, it is essential to prepare the calibrators. These components will be pivotal in approximating the simulation's behaviour to the actual traffic, as elucidated in Runtime Calibration.

The first step involves determining the number and placement of calibrators. The framework installs a calibrator at each network entry, thus enabling reasonable network control because adjustments to entry traffic will ripple through the entire road system. Specifically, each entry features two calibrators corresponding to distinct vehicle types (cars and trucks), enabling separate management of these categories. As for their placement, the framework places them right after the entry edge, at the beginning of the subsequent edge, to allow the calibrator to read the incoming traffic generated by the initial flows and adjust it as needed.

When a vehicle crosses or is inserted by a calibrator, it necessitates a new route to dictate its onward journey through the network. Establishing these calibrator routes is a prerequisite, accomplished prior to simulation by creating an additional calibrator file. Since the system already uses routers to assign new vehicle routes according to the deduced probability distributions, the calibrators' routes can be defined simply as possible. Thus, each calibrator will have only one route associated with it, whose search begins at the edge where the calibrator is located and ends in two possible scenarios:

- It reached a router edge;
- It reached a network exit edge.

In the first case, upon departing from the calibrator's location, the vehicle will proceed to the nearest router, where it will be assigned a new route according to the computed probability distributions. The second case happens when the path ahead of the calibrator does not include routers (because there are no splits in that path), so the defined route will end with the vehicle leaving the network. Therefore, these two scenarios ensure the provision of a continuous route for all vehicles, culminating in their eventual departure from the network.

This route search process employs a depth-first search (DFS) methodology, which terminates promptly upon reaching any of the scenarios above. It is important to note that in the first scenario, the framework concludes the route at the edge immediately following the router's edge to ensure that vehicles do not terminate their route prematurely before rerouting. Algorithm 8 describes this search mechanism based on prior knowledge of the edges containing routers.

Algorithm 8 Depth-First Search (DFS) of calibrator routes

```
1: Init network, router_edges                    ▷ Load the road network and the router edges
2: paths ← []
3: dfs(start_edge, [])
4: function DFS(current_edge, path)
5:     Add current_edge to the path
6:     if len(current_edge.getOutgoing()) == 0 or (current_edge in router_edges and
   current_edge != start_edge) then ▷ Check if reached the end of a route (another router or an
   exit)
7:         if len(current_edge.getOutgoing()) == 1 and len(current_edge.getOutgoing()[0].getIncoming())
   == 1 then
8:             Add current_edge.getOutgoing()[0] to the path    ▷ end the route at the edge after
   the router if possible
9:         Add path to the paths list
10:        path.pop()
11:        return
12:    for each successor_edge in current_edge.getOugoing() do
13:        dfs(successor_edge, path)
14: return paths
```

After obtaining all this information, the program saves the calibrator definitions to an additional file the simulation will load. It generates a `calibrator` tag for each component (including the distinction between vehicles and trucks, with a unique tag for each). Then, it establishes a `flow` tag for each calibrator in each simulation minute, wherein parameters such as the route, `jamThreshold` (a predefined threshold for identifying and mitigating unexpected congestion if the mean edge speed drops below the set threshold), and departure speed are defined. The calibrators will be accessed at simulation runtime via TraCI, and the DT will adjust the flow of the ongoing minute based on the computed values. The framework generates the additional calibrators file within the *calibrators* directory in the *sumo* folder.

4.6.3 Generation of initial flows

The initial flows described in Additional Model Considerations are essential to unlock the calibrators' performance. Its automated generation constitutes another challenge of this framework. The definition of these flows also occurs in an additional file loaded by the simulation, generated by the framework within the *flows* directory in the *sumo* folder.

The framework generates two flow tags for each network entry, one for each type of vehicle (cars and trucks). For both types, the specification of start and end times is imperative, delineating the active period of these flows. In this case, the flows should persist throughout the entire simulation. So, `beginTime` takes the value 0, and `endTime` takes the total value of seconds.

Specifying these flows' departure and arrival edges is also necessary, so developing a mechanism to define them becomes crucial. The departure edge is naturally the network entry edge. Similar to the process of obtaining the routes of the calibrators, detailed in the previous subsection, the arrival edge is discovered by traversing the network from the entry edge until finding an edge that contains a router or an exit edge. The arrival edge will usually correspond to the last edge of this search. In other words, it might either be a network exit or a router edge. However, in the latter case, the program tries to end the initial flow at the edge following the router's edge. This precautionary measure ensures that the initial route assigned to the inserted vehicles persists until the intended rerouting, thus preventing an abrupt termination of their trajectories.

4.6.4 Route Generation

Another crucial information pre-required for the DT's operation is the network's internal route configuration for vehicles. More precisely, the framework must establish the routes starting from each router edge in the network, as these router components will assign them to vehicles within the simulation, considering the calculated probability distributions. To ensure seamless vehicle circulation across the network, the procedure for route discovery is similar to the one used for calibrator route definition. The only difference lies in the initiation point, as the search starts from each router's edge. Multiple potential routes stem from the splitting edge that follows the router. However, the same Depth-First Search (DFS) algorithm employed earlier can be applied here, yielding all the pathways originating from this branching point. Likewise, the search for each path concludes upon reaching a new router or a network exit.

Hence, for each router, the framework starts searching for possible paths from its edge, creating a distinct route tag for each path. Each route is then assigned a unique colour from a predefined list. This way, when engaging with the visualization menu mentioned in GUI Settings, users can observe simulation vehicles coloured according to their active route.

Subsequently, the program establishes a set of route probability distributions for each router, adhering to the methodology elucidated in Dynamic Vehicle Rerouting. Each distribution includes all possible combinations of probabilities that guide vehicle choices concerning routes linked to the specific router. In most cases, splits will result in two possible paths, so these distributions will consist of pairs of probabilities that add up to 1. Thus, the program generates conceivable combinations of these probability pairs.

Afterwards, the program creates a `routeDistribution` tag for each distribution and assigns it a unique ID. This tag is then supplemented with child `route` tags, each representing a route and its associated probability within the given distribution.

Finally, the framework creates the routes file in the *routes* directory inside the *sumo* folder. After computing the best suitable probability distribution for the current minute and a specific router during the simulation, the DT consults this file to apply the proper distribution.

4.6.5 Traffic intensity on the free variables' edges

As explained in Runtime Flow Estimation, the baseline methodology partitions the value range of each free variable into ten distinct intensity levels. As a result, the Simplex algorithm can reduce the solution space, contingent on the chosen intensity level.

As mentioned in Additional Model Considerations, the intensities employed represent a specificity of the road network used in the model outlined by the methodology authors. This peculiarity is due to the predefined intensities in the DT-GM for each of the six roads associated with the model's free variables. The authors derived these intensities by manually observing GPS traces on the OSM and Google Maps traffic websites on various daytimes.

As such, the intensities allocated to the roads associated with the free variables emerge as a vital framework input. Their deduction must occur between the third and fourth phases of the framework, following the determination of free variables and preceding the DT execution.

Unfortunately, no feasible method has been identified to automate this process fully, rendering it challenging to eliminate this input from the framework and relieving users of the obligation to prepare it. This absence of solutions predominantly stems from restricted access to historical traffic data and the relatively underdeveloped state of certain APIs that hinder automated data retrieval based on the coordinates of the target road.

However, an alternative manual approach was discovered, less exhaustive than the approach employed in the work above. This approach involves a single preparation step facilitated by utilizing historical traffic data offered by Google Maps[1]. This information is accessible via an interactive window that adjusts the depiction of road traffic on the current map. Two display modes are available: real-time traffic (Live Traffic) and the average traffic level on roads (Typical Traffic). The latter mode is the most interesting in this context, as the aim is an average traffic intensity on the roads corresponding to the free variables. The interactive window allows selecting the desired weekday and the corresponding time for viewing the average traffic levels. In ascending order of congestion level, the four colours used to depict the traffic conditions are green, orange, red, and dark red. Green denotes a free-flow state, while dark red represents the highest congestion level, where vehicles are nearly stationary.

The authors of the baseline methodology established time-of-day intervals with which they link a certain intensity, regardless of the weekday to which the data corresponded. In contrast, this framework allocates an intensity to each hour of the day, considering the specific day of the week. It is obtained by directly observing average traffic levels on Google Maps and deducing the appropriate intensity value (one of ten levels) based on the colours of the free variable's road and its adjacent roads. Before the simulation starts, the program determines the weekdays corresponding to the data's timestamps. It is important to note that for the period between 10 p.m. and 6 a.m., no data is available on Google Maps. Given that traffic during this period can be regarded as negligible, the assigned desired intensity is 0.

[1] https://www.google.com/maps/@46.1613894,6.0999767,15z/data=!5m1!1e1?hl=en&entry=ttu

In short, the framework estimates the flow in the free variables' edges using the traffic intensities defined in the *intensities.json* file in the *nodes* folder. These vary according to the time of day and the respective day of the week. Nonetheless, users are required to manually collect these intensities following the third phase of the framework through observation of historical data provided by Google Maps.

4.6.6 Traffic Model

After gathering all the required information and extracting the simulation parameters from the *config.ini* configuration file, the framework is ready for initiating the DT execution. Among these parameters lies the total duration of hours for the simulation, the specified interval of seconds for removing old vehicles from permanent distributions (routing control), and the step length, which defines the duration of each simulation step in seconds. In this case, the step length is set at 0.25, meaning that four simulation steps correspond to one second of real-time. Additionally, the user can alter a parameter that governs the simulation's speed, delaying the visualisation of the simulation for a deeper comprehension of its behaviour.

For every hour, the framework initiates a distinct simulation session aiming for an iterative execution, producing well-defined states that can be resumed and analysed individually. As equation (4.1) indicates, the process commences by calculating the total number of simulation steps.

$$total\_steps = total\_hours \times 3600 \times \frac{1}{step\_length} \tag{4.1}$$

Then, at each step, it verifies whether one second has elapsed based on the condition on line 15. If this is the case, it updates the current minute's flow and speed by adding the number and speed of new vehicles that entered the network during that second. The sum of the speeds is then divided by the number of new vehicles when one minute has passed to obtain the average speed for the current minute. This procedure, outlined in algorithm 9, uses a list that stores the vehicles that have recently accessed the network but have yet to reach the internal roads, thereby facilitating the identification of new vehicles. It is conducted for every network entry, as the flow and speed records are handled separately for each, simplifying the final validation of results. Network exits are subject to the same flow and speed controls.

The flow update process involves observing vehicles on two edges, namely, *start_edge* and *next_edge*. First, the program identifies new, unaccounted-for vehicles in the *start_edge*. Subsequently, the flow data is updated by including passages made by these newly registered vehicles, and the program adds these vehicles to the list of already accounted-for vehicles still on the initial edge. Additionally, the *next_edge*'s vehicles are verified, eliminating the registration from those already counted, as they have already entered or left the network.

The framework must conduct the vehicle counting procedure for all network entry and exit points. In the case of network entries, *start_edge* and *next_edge* should represent two sequential edges along the entry pathway, with *start_edge* being closer to the entry node, as the counting procedure described above must happen in the direction of entrance into the network. Conversely,

in the case of network exits, *next_edge* assumes the edge closest to the exit node since vehicle counting must happen toward egress from the network. The method for choosing these two edges prioritises edges equipped with sensors, choosing them as *next_edge* to ensure vehicle counts closely align with the real data. This strategic selection allows the counting process to occur near the physical sensor locations. The selection is done differently for entries and exits when no sensors are in their paths. Regarding network entries, the framework selects the edge nearest to the first encountered branching point as *next_edge* and the preceding edge as *start_edge*. Concerning network exits, it selects the last two edges of the departure path. Additionally, this selection strategy avoids opting for excessively short edges (with less than 150 meters), ensuring enough space for the counting process. This approach for selecting edges produced the best results, discussed in section Geneva motorway use case, achieving the desired levels of simulation reliability.

Algorithm 9 Update of the current minute's flow and speed according to the new vehicles entering the network every second

```
1:  function EDGEVEHPARAMETERS(start_edge, next_edge, oldVehIDs)
2:      leaving_count ← traci.edge.getLastStepVehicleIDs(next_edge)
3:      intersection ← set(oldVehIDs).intersection(leaving_count)
4:      for each vehicle in intersection do ▷ Remove vehicles that already crossed the count edge
5:          Remove vehicle from oldVehIDs
6:      // Get the vehicles on the vehicle count edge
7:      current_vehicles ← traci.edge.getLastStepVehicleIDs(start_edge)
8:      newVehIDs ← []
9:      for each vehicle in current_vehicles do
10:         if vehicle not in oldVehIDs then
11:             Add vehicle to the newVehIDs list
12:     flow, speed ← 0
13:     for each vehicle in newVehIDs do
14:         flow+ = 1
15:         speed+ = traci.vehicle.getSpeed(vehicle)
16:         Add vehicle to oldVehIDs
17:     return flow, speed, oldVehIDs, newVehIDs
```

After that, the program checks if one minute of real time has passed. If so, it updates the constant values (q_X) by adding the flows detected by the sensors on the lanes of the respective edge, computing also the average speed of the vehicle passages. Moreover, it records the data for that minute, encompassing actual flows (detected by the ILDs) and simulation flows at the network entries and exits covered by sensors. The same occurs for the entries and exits without sensors, whereby the estimated flow and the flow of the simulation's current minute for that particular edge are stored. The Total Time Spent (TTS), the cumulative time vehicles expend within the simulation, is also saved, measured in hours. A stack designated `controlFile` stores all this data, amassing the real/calculated flows and the actual simulation flows for each elapsed minute. Later,

this data will be saved in a results file for the ongoing hour for further analysis in Chapter Results and Analysis. Lastly, the program resets the stored flow and speed values so that the analysis can move on to the following minute.

To continue the analysis, it first needs to fetch the flow values for the upcoming minute from the sensors, thereby updating the values of the q_X constants. Afterwards, as described in the previous subsection, the framework defines the desired intensity levels for the free variables' roads based on the current simulation time and corresponding weekday by reading these values from the *intensities.md* file. Subsequently, the DT computes the free variables' values by applying the Simplex algorithm, yielding the resultant X_{free} vector. From the estimated values for these variables, the program calculates the values of the remaining system variables using the matrices $X_{particular}$ and X_{null} derived in the third phase of the framework (Flow System Resolution). Then, it updates the TTS value by adding the product of the vehicle count in the ongoing simulation minute and the conversion factor $\frac{60}{3600}$, which converts the unit to hours. Next comes the calibrator configuration via TraCI, which must be modified to reflect computed flows.

The program then determines the probability distributions for the splits associated with each router. The ratio between the variable of an edge originated by a given split and the variable of the splitting edge reflects the likelihood that a vehicle will take that particular road upon this split. Thus, according to the probabilities set computed for each router, the program selects the most suitable probability distribution from those defined in the route file generated in subsection Route Generation.

Afterwards, the program links each router's probability distribution to the newly added vehicles to the network. This attribution serves as a marker, indicating the current probability distribution of the routes for that vehicle, meaning that it is ready to be routed by the DFC mechanism. Every minute, the DT verifies if the defined interval for vehicle cleanup has passed, and if so, it stores the IDs of all vehicles in the network. This list will help update the permanent distributions list by removing vehicles that have already left the network and no longer require routing.

Then, each router dynamically assigns vehicle routes according to the embedded probability distribution model. This procedure is done individually for each router by identifying vehicles traversing the router's edge and applying the corresponding route distribution. After this step, the vehicle possesses an assigned route, so the program removes it from the list of vehicles in the permanent distribution, which contains vehicles with no specified route.

Upon completing each hour, the program saves the `controlFile` data structure into an Excel file, thereby preserving the recorded real/calculated flows and simulation flows for later comparison in Results and Analysis. It is also worth noting that the simulation's state can be saved in an XML file every hour, enabling the recovery of this state and resuming the simulation.

Algorithm 10 combines the main steps of the DT created by the framework.

Algorithm 10 Framework's Digital Twin

Init *network* ▷ Load all information generated in the previous phases of the framework
procedure DIGITALTWIN
    Initialize the necessary data structure (*routers*, *sensors*, *oldVehIDs*)
    Generate the *calibrators* file
    Generate the initial flows file
    Generate the routes file
    Read the simulation parameters defined in the configuration file
    $total\_steps \leftarrow total\_hours * 3600 * (1/step\_length)$
    $current\_hour \leftarrow 0$
    while $current\_hour < total\_hours$ **do**
        Start the SUMO simulation via TraCI
        $step \leftarrow 0$
        while $step < total\_steps$ **do**
            Mark the start of a new simulation step via TraCI
            if $step\%(1/step\_length) == 0$ **then** ▷ A second has passed
                Update of the current minute's flow and speed
            if $step\%(60 * (1/step\_length)) == 0$ **then** ▷ A minute has passed
                if $step > 0$ **then**
                    Save the current minute's flows and speeds (real/calculated and simulated)
                    Save the current TTS in the $controlFile$ stack as well
                    Reset flow and speed values for the current minute
                Read from real sensor data the flow and speed for the next minute
                Update the values of the q_X variables according to the new values
                Define free variables' intensity levels based on current weekday and hour
                Run the Simplex algorithm, obtaining the $X_{complete}$ vector
                Update the TTS value
                Set the flows of the calibrators according to the calculated flows
                For each router, calculate the route distribution probabilities
            if $step\%(1/step\_length) == 0$ **then** ▷ A second has passed
                if $step\%(60 * (1/step\_length)) == 0$ **then** ▷ A minute has passed
                    Assign the computed probability distribution to new vehicles in the network
                if $step > 0$ **then**
                    if $sim\_time\%time\_clean == 0$ **then**
                        Remove vehicles that have already left the network from *perm_dists*
                    for *router* in *routers* **do**
                        Assign routes to vehicles on the *router* based on probability distribution
            Slow down or speed up the simulation based on $time\_sleep$
            if $step\%(3600 * (1/step\_length)) == 0 and step > 0$ **then** ▷ An hour has passed
                Save the $controlFile$ content in an Excel file
            $step += 1$
        End the SUMO simulation via TraCI

Chapter 5

Results and Analysis

The primary aim of this chapter is to unravel the performance and effectiveness of the proposed framework for building Digital Twins when subjected to various real-world scenarios. It scrutinizes the outcomes of simulations conducted over different periods and under diverse traffic conditions. The objective is to appraise the accuracy of the generated Digital Twins by contrasting simulated data with real-world data. The optimal scenario entails nearly flawless digital replication of the network traffic, with values closely resembling those recorded by the ILDs.

The leading traffic flow metric employed for the analysis is the count of vehicles on various roads within the network during the assessed timeframe. The result files produced by the framework will serve as input for this stage, as they contain both the actual vehicle counts and the simulated counts for both network entry and exit edges, regardless of sensor coverage. Accordingly, the generated graphs illustrate the fluctuations in vehicle volume on various roads throughout the evaluated periods of the day. Moreover, this chapter states potential causes for the observed traffic patterns, considering the specific road context.

5.1 Geneva motorway use case

The first evaluated scenario corresponds to the one examined in work [27], which pertains to the Geneva motorway network discussed in The Geneva motorway. Analysing this scenario enables a direct comparison between the outcomes generated by the framework and the method proposed in that study, namely, DT-GM. The objective is to ascertain whether the framework adapts to the network's topology, yielding outcomes that align with the ad-hoc approach outlined in the first work.

After applying the framework to this use case, utilising the intensities gathered through the process outlined in Traffic intensity on the free variables' edges, it became evident that the results were not optimal. Consequently, experiments were conducted using the intensity intervals defined by the DT-GM methodology, slightly improving the outcomes.

Therefore, this analysis will compare three approaches: the DT-GM methodology and two executions of the proposed framework, one utilising intensities derived from Google Maps and the other employing the intensity intervals defined by the DT-GM authors.

The evaluation will encompass both vehicle entrances and exits. These graphs display the real vehicle volumes, minute by minute, in a salmon colour, representing the data from the ILDs. Towards a more straightforward observability, two smoothed curves are introduced, one for real data and another for simulated data, employing a similar approach to the graphs demonstrated in the first study. These smoothed curves are generated through a moving average, considering 20 samples of one-minute traffic volumes.

First, Figure 5.1 analyses the framework's performance on roads covered by sensors to assess its ability to accurately replicate the traffic volumes detected by these sensors.

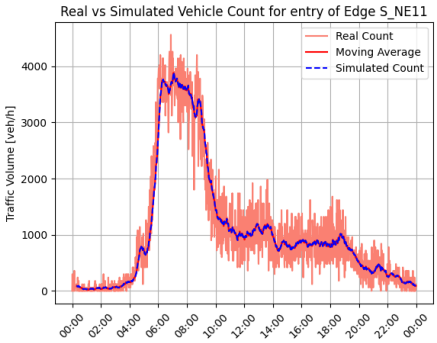
Starting the analysis with the southbound main segment of the motorway, Figure 5.1a illustrates the progression of traffic flow on a chosen workday (specifically, the 24th of March 2022) from the French border to Switzerland. In contrast, Figure 5.1d displays the traffic flow from Switzerland towards the border. The overall traffic demand within the studied motorway region exhibits a symmetrical pattern. Commuters from France play a substantial role in generating the morning peak hour (occurring from 7 to 10 a.m.) within the observed segment. Conversely, travellers returning to Geneva at the end of the day contribute to the afternoon peak hour (6 to 8 p.m.).

Starting the analysis with the southbound main segment of the motorway, Figures 5.1a, 5.1b and 5.1c illustrate the progression of traffic flow on a chosen workday (specifically, the 24th of March 2022) from the French border to Switzerland. In contrast, Figures 5.1d, 5.1e, and 5.1f display the traffic flow from Switzerland towards the border. The overall traffic demand within the studied motorway region exhibits a symmetrical pattern. Commuters from France play a substantial role in generating the morning peak hour (occurring from 7 to 10 a.m.) within the observed segment. Conversely, travellers returning to Geneva at the end of the day contribute to the afternoon peak hour (6 to 8 p.m.).

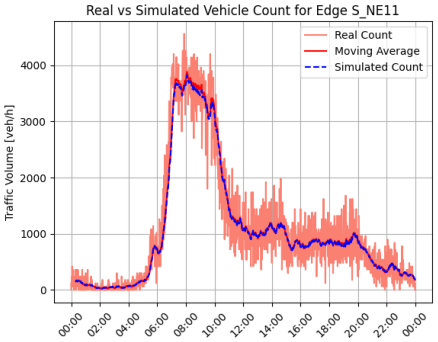
The remaining figures are related to the traffic inflow and outflow of the north and east zones, respectively.

Figure 5.1: Comparison of the results of the DT-GM methodology and the proposed framework on all edges covered by sensors on the Geneva motorway network

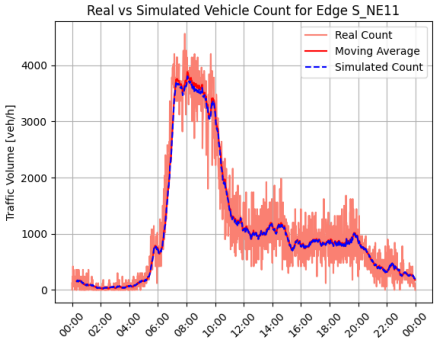
Inflow from the southern part of the network (q_6)



(a) DT-GM Methodology

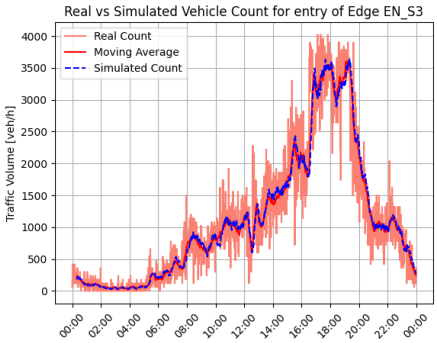


(b) Framework w/ Google Maps intensities

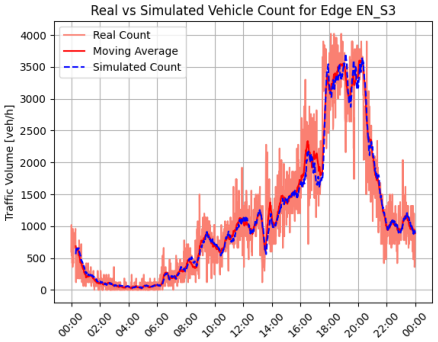


(c) Framework w/ DT-GM intensities

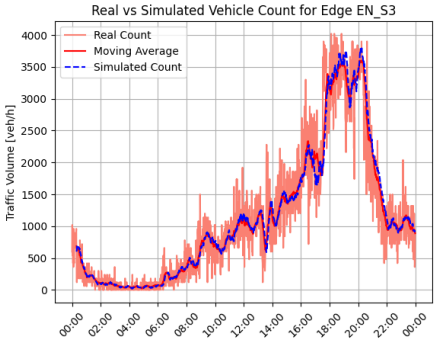
Outflow from the southern part of the network (q_5)



(d) DT-GM Methodology

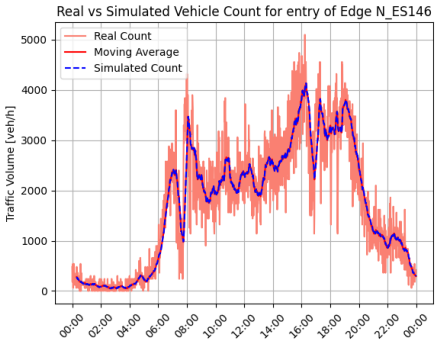


(e) Framework w/ Google Maps intensities

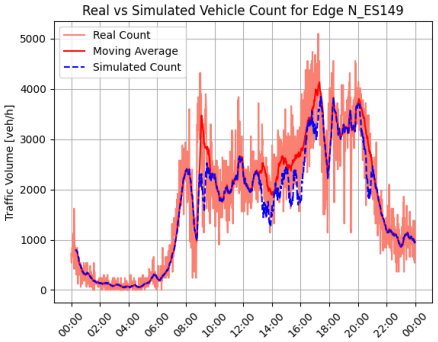


(f) Framework w/ DT-GM intensities

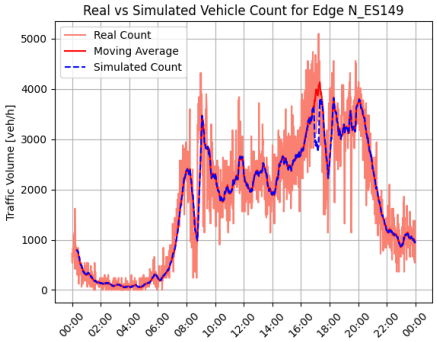
Inflow from the northern part of the network (q_4)



(g) DT-GM Methodology



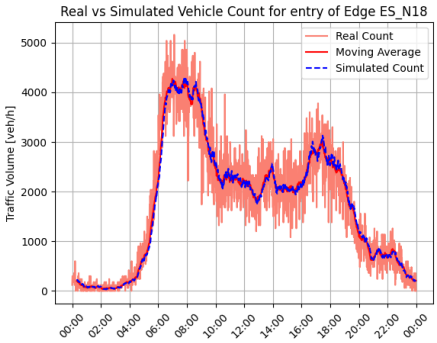
(h) Framework w/ Google Maps intensities



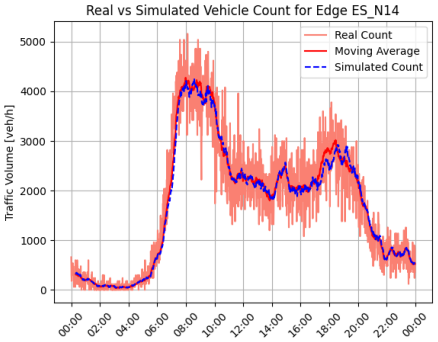
(i) Framework w/ DT-GM intensities

Figure 5.1: Comparison of the results of the DT-GM methodology and the proposed framework on all edges covered by sensors on the Geneva motorway network

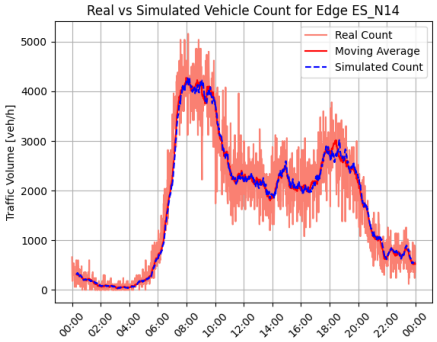
Outflow from the northern part of the network (q_3)



(j) DT-GM Methodology

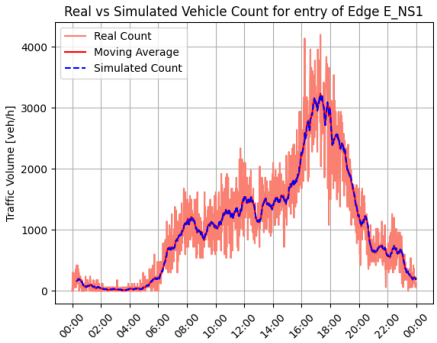


(k) Framework w/ Google Maps intensities

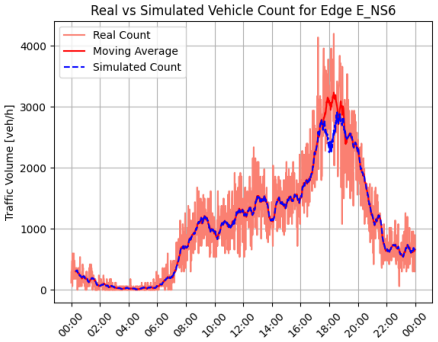


(l) Framework w/ DT-GM intensities

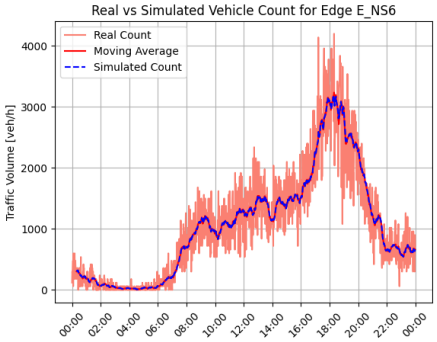
Inflow from the eastern part of the network (q_2)



(m) DT-GM Methodology

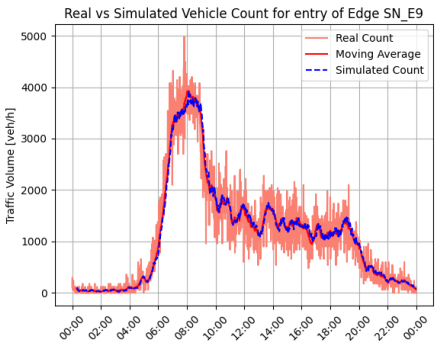


(n) Framework w/ Google Maps intensities

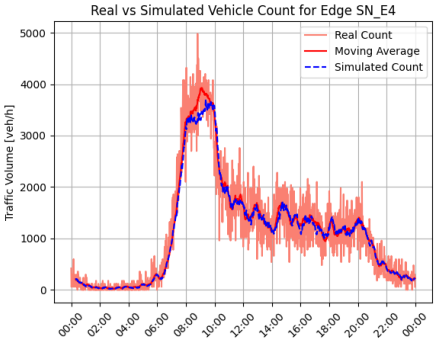


(o) Framework w/ DT-GM intensities

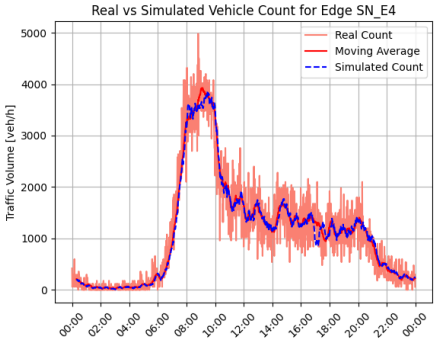
Outflow from the eastern part of the network (q_1)



(p) DT-GM Methodology



(q) Framework w/ Google Maps intensities



(r) Framework w/ DT-GM intensities

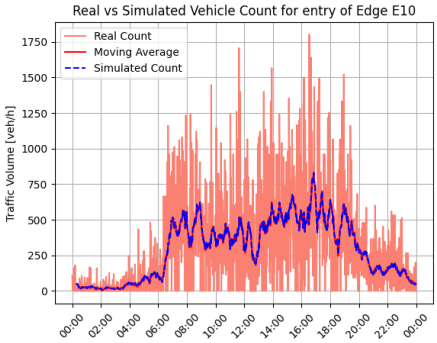
Commuter traffic from France substantially contributes to the morning peak hours in the eastbound direction toward Geneva. Furthermore, eastbound traffic is partially composed of commuters from the north who use the motorway as an alternative route to avoid driving through the city centre when heading to the southern part of Geneva. Similarly, for northbound traffic, there is a significant contribution from commuters travelling southbound from France and eastbound from Geneva, particularly during the morning peak hours. However, traffic entering the motorway at entry with flow x_{12} also plays a role in contributing to traffic during these peak hours. Conversely, the afternoon peak hours exhibit an opposite pattern as commuters return, resulting in a symmetrical reversal of traffic flow.

The authors justify the minor temporal shifts between measured and simulated outflows by citing the simplifications in their model. Specifically, they excluded certain local motorway entry ramps for which ODPMS lacked traffic data. Consequently, the program introduces additional traffic on the included entry ramps (with flows x_{11}, x_{12}, x_{15}, and x_{16}) to achieve the necessary overall traffic flow equilibrium on the observed motorway segment. In addition, compared to the symmetric measurement points for outflows, the spatial relocation of calibrators on the east, north, and south primary sections may lead to a slight delay in the flow model's response time. This delay results in a rightward shift in the time required to attain the desired flow on the motorway section's exits (as shown in Figures 5.1d, 5.1j, 5.1p), as vehicles must travel a longer distance to reach their intended destinations. Nonetheless, it is essential to note that this temporal shift is on a small-scale time resolution, considering that the DT-GM runtime adjustment is minute-based. Therefore, despite this slight time adjustment, the accuracy remains high. The cumulative error remains negligible due to the fine-grained temporal resolution in place.

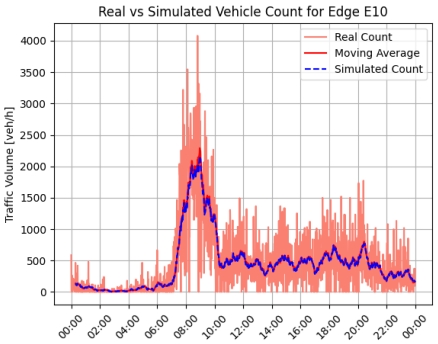
What follows are the results for the remaining network entries and exits, devoid of sensors, therefore having unknown flow. Figure 5.2 displays the comparison between the three approaches for these roads. In these instances, the simulated traffic volume is compared with estimates generated during the framework's execution, as the actual values remain unknown due to the absence of sensors. It is worth noting that the traffic volume on these secondary entries and exits is lower than on the primary roads analysed above.

Figure 5.2: Comparison of the results of the DT-GM methodology and the proposed framework on all edges without sensors on the Geneva motorway network

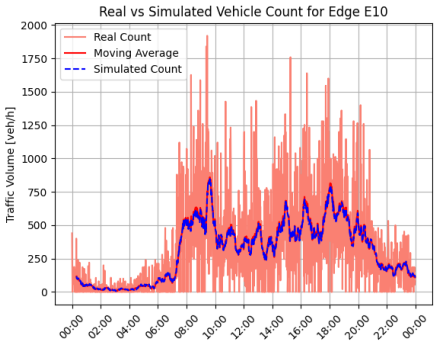
Inflow from secondary network entry (edge E10)



(a) DT-GM Methodology

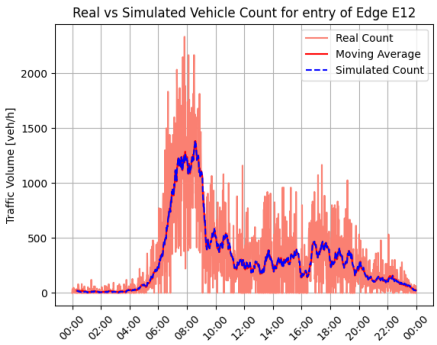


(b) Framework w/ Google Maps intensities

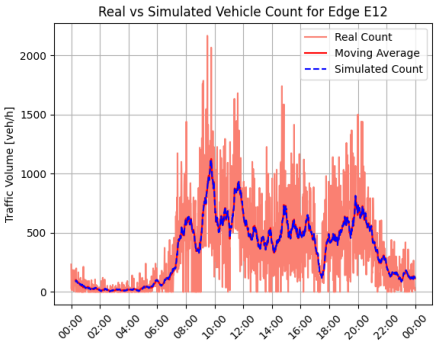


(c) Framework w/ DT-GM intensities

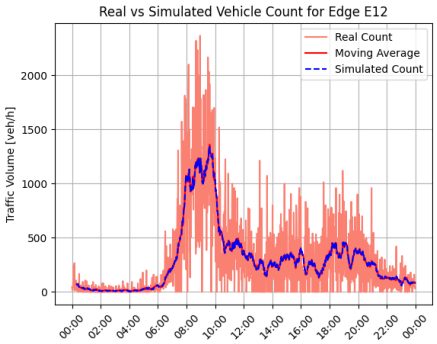
Inflow from secondary network entry (edge E12)



(d) DT-GM Methodology

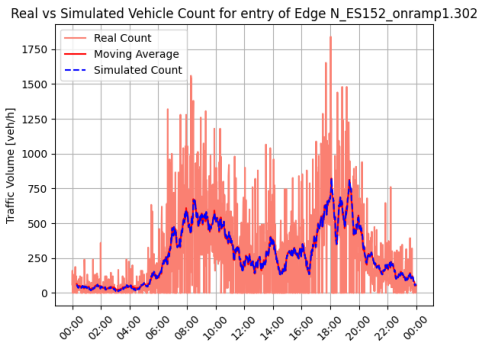


(e) Framework w/ Google Maps intensities

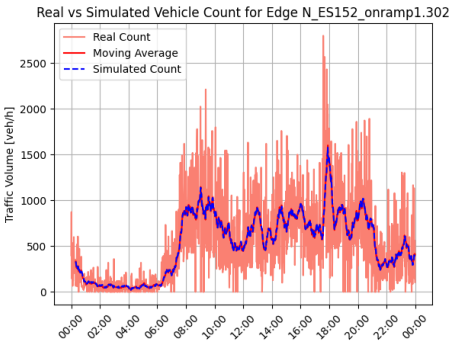


(f) Framework w/ DT-GM intensities

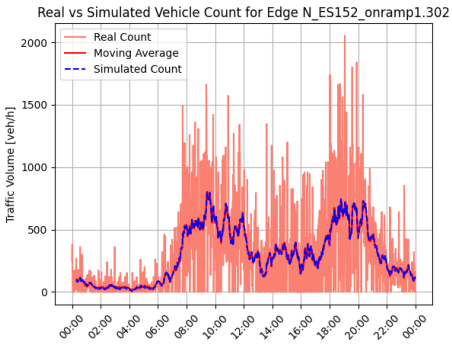
Inflow from secondary network entry (edge N_ES152_onramp1.302)



(g) DT-GM Methodology



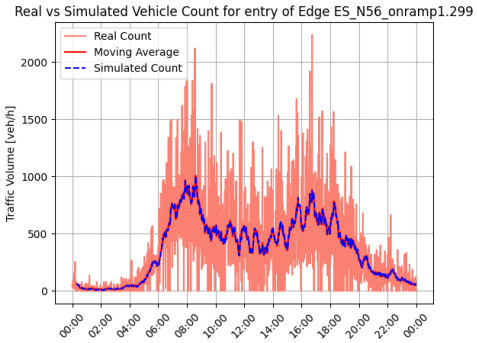
(h) Framework w/ Google Maps intensities



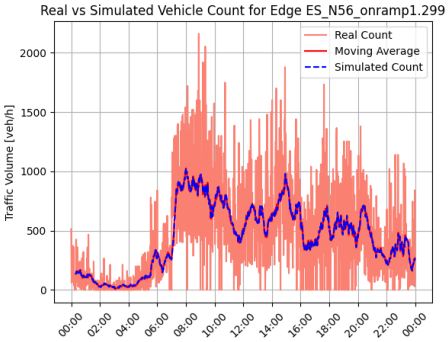
(i) Framework w/ DT-GM intensities

Figure 5.2: Comparison of the results of the DT-GM methodology and the proposed framework on all edges without sensors on the Geneva motorway network

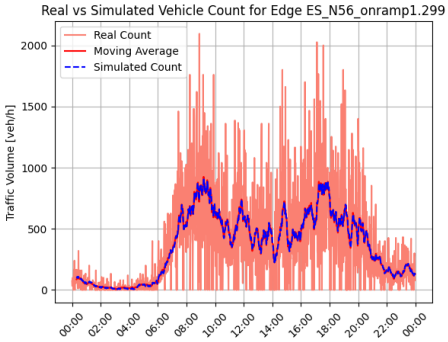
Inflow from secondary network entry (edge ES_N56_onramp1.299)



(j) DT-GM Methodology

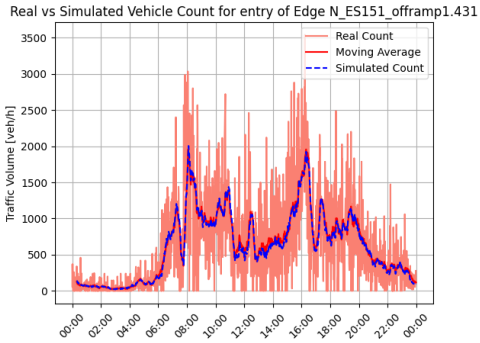


(k) Framework w/ Google Maps intensities

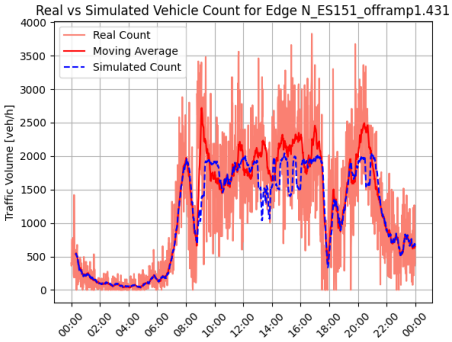


(l) Framework w/ DT-GM intensities

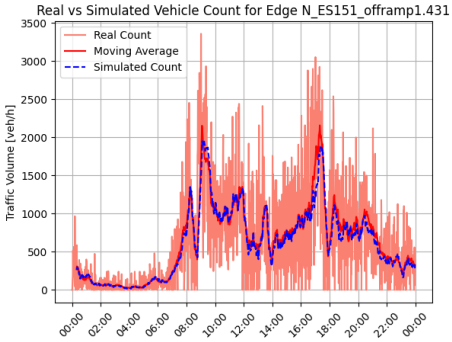
Outflow from secondary network exit (edge N_ES151_offramp1.431)



(m) DT-GM Methodology

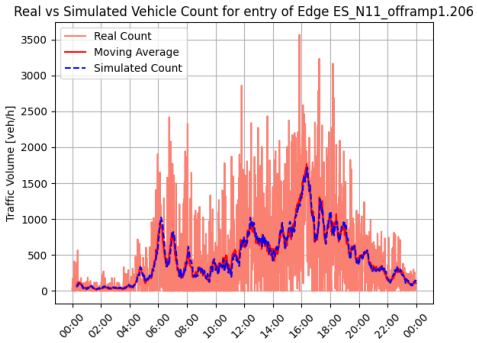


(n) Framework w/ Google Maps intensities

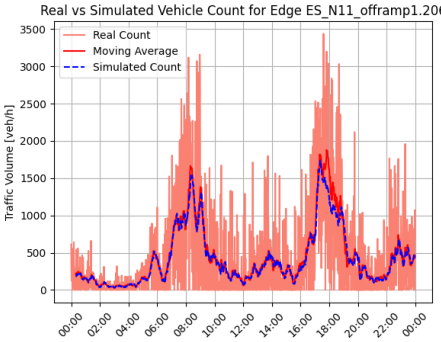


(o) Framework w/ DT-GM intensities

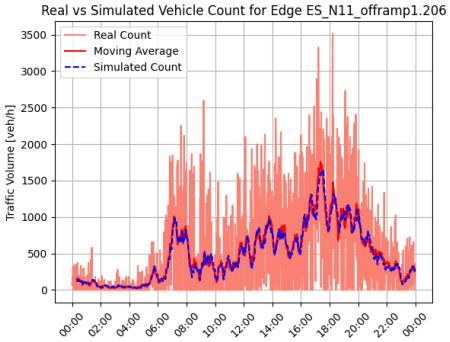
Outflow from secondary network exit (edge ES_N11_offramp1.206)



(p) DT-GM Methodology



(q) Framework w/ Google Maps intensities



(r) Framework w/ DT-GM intensities

The results indicate no significant difference between the traffic generated by the runtime DT-GM (blue curves) and the real motorway traffic (red curves). At the one-minute traffic granularity, morning and evening peak hour traffic and stationary flow between peak hours correlate well with actual traffic.

As can be observed, the framework's performance declined when employing intensities sourced from Google Maps. Simulated traffic encountered challenges in mirroring actual traffic patterns, with this divergence becoming particularly pronounced during periods of elevated traffic volumes, such as peak hours. However, there is a slight improvement when adopting the intensity intervals specified by DT-GM. Nonetheless, some disparities persisted, particularly during periods of heightened traffic congestion.

The observed data on the framework performance reveals that the simulated traffic struggles to match the actual traffic patterns at specific entrances and exits where traffic is estimated. Consequently, this performance discrepancy indirectly influences the primary entrances and exits equipped with sensors, thus contributing to the marginal variations seen in the prior graphs.

Further experimentation with the framework led to the conclusion that these disparities stem from the cumulative error of small changes in the selection of vehicle counting edges and the placement of calibrators and routers. In the DT-GM methodology, the authors meticulously adjusted the positioning of these components, resulting in slightly better results. The framework is limited to the automated generation of these components, and by applying a standardised logic to all entrances and exits, the final positions may not be optimal for all cases.

More precisely, these situations arise because specific calibrators may be located on edges with inadequate characteristics, such as insufficient length or unsuitable shape, hindering traffic generation and impeding lane-changing manoeuvres for the inserted vehicles. Similarly, vehicles frequently require lane changes according to their revised destinations after rerouting, which may be impeded by the lanes' attributes at the routing edge.

Notwithstanding the minor disparities observed in traffic patterns during peak hours, the framework demonstrated a remarkable alignment with the network in this specific use case, effectively replicating the traffic behaviour on this road network.

5.2 VCI network use case

The second use case is one of the VCI nodes comprising several entry and exit points, namely, the Coimbrões node. This decision arose from an in-depth analysis of each node along this highway, considering the data acquired from the sensors deployed on it. The data used for this use case gathers vehicle passages on the 24th of May, 2023. Image 5.3 illustrates the sensors' locations within the VCI ring, with data sourced from the ARMIS company. Each green pinpoint on this map identifies a sensor with data available for the evaluated day. In contrast, a red pinpoint indicates data unavailability due to a device breakdown or inactivity. The black circle highlights the VCI node covered in this section (Coimbrões Node).

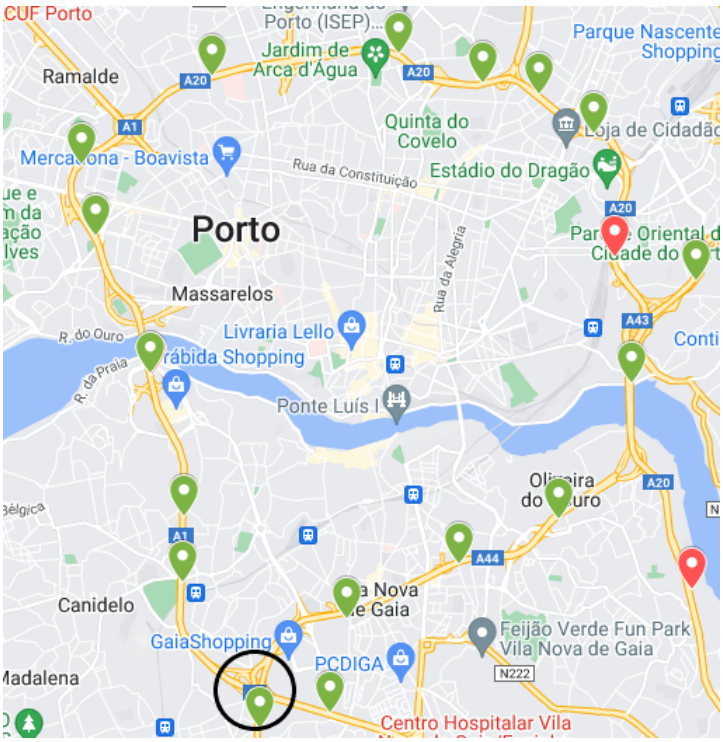


Figure 5.3: Location of the sensors from the VCI network

The entire network of the VCI ring was extracted from OSM and subsequently converted into a SUMO network. Following this step, the network was divided into multiple nodes, each comprising a collection of entry and exit points for this highway. Due to flaws arising from the initial network conversion, common to several related works, as seen in Related Work, it was necessary to manually correct specific details of the networks of these nodes, such as the number of road lanes and their respective directions.

A total of 20 nodes underwent analysis, and their respective SUMO networks are available in VCI nodes' networks and equations. Within the provided dataset, a single sensor supplies data for all lanes within a specific segment, spanning both directions. The selection of the Coimbrões node as the focal point of analysis stems from its reputation on this highway, apart from its level of complexity and sufficient sensor coverage. Specifically, the existing sensors encompass three entry

points and three exit points within its network. Image A.16 illustrates this node's network after the framework's application, as indicated by the generated pinpoints during the variable assignment process.

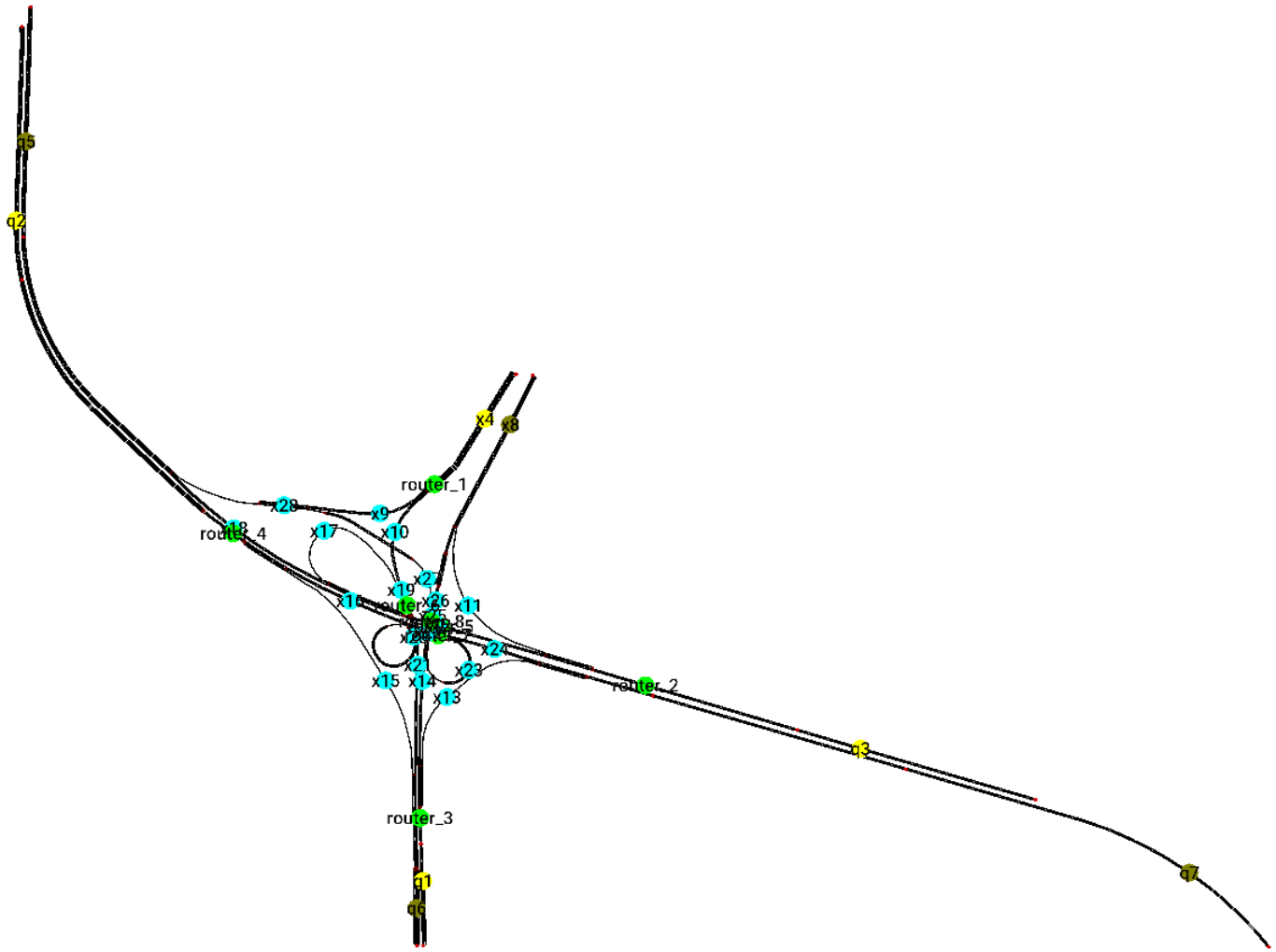
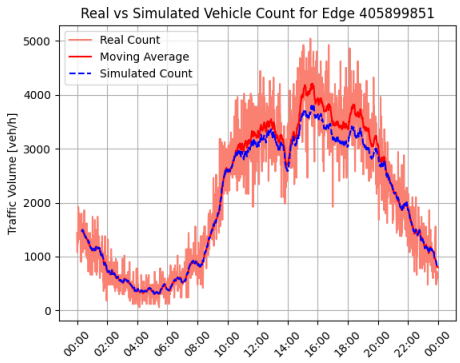


Figure 5.4: SUMO Network of the VCI node "Nó de Coimbrões"

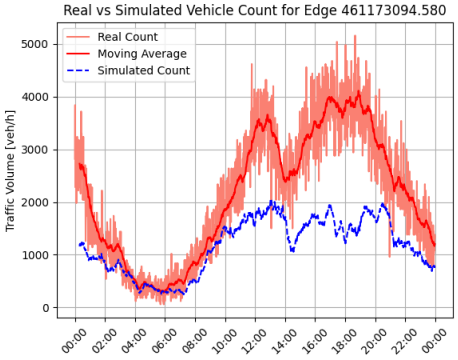
The networks of the remaining nodes have likewise gone through the second phase of the framework, as displayed in VCI nodes' networks and equations, with the corresponding pinpoints representing the assigned variables. It is crucial to emphasise that the equations for these networks were meticulously formulated through a manual, labour-intensive process, and all match with the framework's outputs. Therefore, the automated variable and equation generation procedure achieved a remarkable 100% efficacy rate, proving effective across all 21 networks examined (including the Geneva motorway). The systems of equations generated for all networks are also accessible in VCI nodes' networks and equations.

Returning to the Coimbrões Node, before commencing the framework's final phase, it became imperative to validate the data originating from the sensors within its network in conjunction with ARMIS. Specifically, the objective was to align the lanes referenced in the sensor data with the

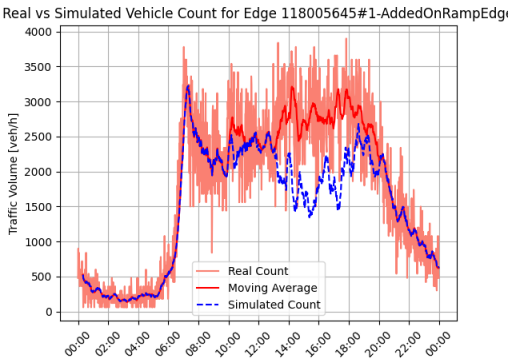
SUMO network while discerning their respective orientations. Following the data separation, the final framework phase was initiated, with the outcomes of the edges covered by sensors presented in Figure 5.5.



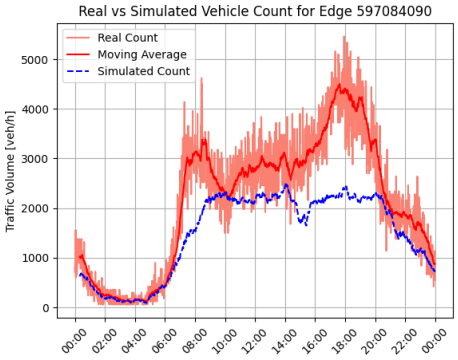
(a) Inflow from the western part of the network



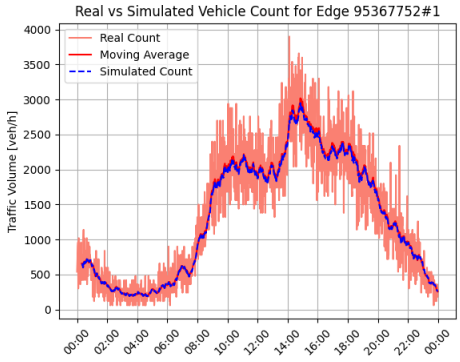
(b) Outflow from the western part of the network



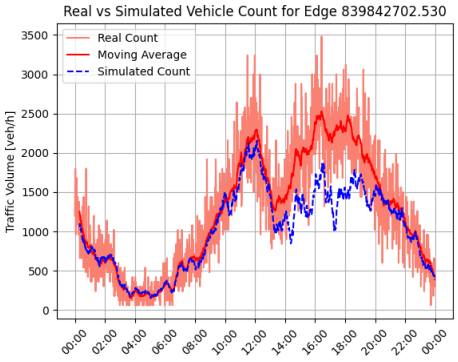
(c) Inflow from the southern part of the network



(d) Outflow from the southern part of the network



(e) Inflow from the eastern part of the network



(f) Outflow from the eastern part of the network

Figure 5.5: Framework results on all edges covered by sensors on the VCI's "Nó de Coimbrões" network, using intensities derived from Google Maps

As observed, the results for this use case are comparatively less satisfactory than those of the first use case.

Unlike this network, the Geneva network underwent a meticulous manual adjustment process through topological modifications, rendering it highly suitable to the employed methodology. For example, the DT-GM network comprises extensive roads, which provide sufficient space within its representation for flow counting, calibration, and rerouting processes. Conversely, the network of this VCI node consists of short road segments where some of these processes are applied. For instance, there is a small road segment with a router at the network's centre, at the intersection of roads originating from the north and south leading to the VCI ring. The limited space available for these processes can lead to incorrect route assignments, resulting in vehicles that prematurely exit the simulation before completing their intended routes. Consequently, the final traffic counts may not align precisely with the sensor-derived values.

Furthermore, the traffic intensities associated with the roads corresponding to the free variables were derived from Google Maps using the manual procedure outlined in Traffic intensity on the free variables' edges. As observed in the previous use case, the intensities acquired through this method may not accurately portray the network's traffic dynamics, potentially contributing to disparities in the results.

Lastly, as mentioned above, the differentiation between vehicle counts in both directions (entrances and exits from the network) was conducted manually with ARMIS. Due to difficulties in identifying the lanes recorded by the sensors in the physical network, this process involved a degree of uncertainty. Consequently, the vehicle count data may have been assigned to the incorrect edges (wrongly defined directions), potentially leading to anomalous simulation behaviour, as inconsistency in data impedes a continuous traffic flow within the broader network context.

Despite all this, the simulated volume curve attempts to mirror the actual volumes, albeit with difficulties caused by the potential factors outlined herein.

Chapter 6

Conclusions

This document detailed the implementation of a framework designed to automate the building of Digital Twins. This work seeks to revolutionise the process of conceiving such tools, abruptly reducing the effort required for their development.

Initially, the document introduced the context and motivation of this dissertation, emphasising the theme's relevance. Subsequently, it explored various topics related to the theme under investigation, showcasing pertinent information gathered during an extensive literature review.

More specifically, this review sought to assess the current state of the art of Digital Twins in the context of road traffic. This technology has emerged as a relatively new and promising domain, albeit largely unexplored. With ambiguous definitions, the term Digital Twin became highly contentious in the literature. Thus, this study strived to dispel these misconceptions and clearly define the benefits, challenges, and enabling technologies associated with Digital Twins.

An analysis of similar works on creating Digital Twins for diverse road networks ensued. Here, the goal was to fathom the main techniques employed for the operation of this tool, spanning from model calibration methodologies to procedures for analysing and evaluating the obtained outcomes. Within this analysis, a salient observation emerged: the methodologies adopted by these correlated studies proved overly tailored to the specific road networks under consideration. As a result, this observation sparked the idea to develop an automated framework for building Digital Twins, with the overarching goal of alleviating researchers from the laborious process of producing these tools anew each time they sought to apply them to a novel network. Hence, the framework enables the immediate and automated generation of a Digital Twin, empowering researchers to focus on other enhancements for the baseline Digital Twin, such as integrating traffic forecasting or delving into the analysis of what-if scenarios.

Then, the document described the framework's baseline methodology, which draws inspiration from the model outlined in work [27]. However, this methodology necessitates generalisation to be adaptable to any road network. So, the document also presents the solutions formulated to address the diverse challenges stemming from this generalisation.

Finally, the framework's outcomes were analysed and evaluated through a direct comparison with the results of the DT-GM model. Furthermore, the framework underwent execution on a

subset of VCI nodes, encompassing numerous ingress and egress points of this motorway, and the resulting outcomes in this specific scenario were also evaluated. Fortunately, the framework's outcomes demonstrated the feasibility and correctness of this approach, achieving dependable traffic replication in both scenarios.

All code developed for the proposed framework will be publicly available on GitHub[1] to foster further research.

Lastly, the upcoming section will outline this dissertation's Main Contributions, followed by another section comprising a discussion of Further Improvements to the framework's methodology. The document concludes with several suggestions for Future Work that could be undertaken on the DTs generated by the framework.

6.1 Main Contributions

Among the prominent contributions of this dissertation, the most notable is the devised framework designed for the automated building of Digital Twins, with applicability to any road network. This tool holds the potential to propel the DT concept within the domain of traffic networks, offering a substantial acceleration in its developmental process. This advancement will enable researchers to redirect their attention towards other intricate challenges.

Moreover, this work presents two use cases of the developed tool. These applications yielded a DT of the Geneva motorway in Switzerland and DTs for several nodes of the VCI highway in Porto, Portugal. As demonstrated in Chapter Results and Analysis, the performance of all DTs generated by the framework demonstrates remarkable outcomes, substantiating their capacity to replicate real-world traffic dynamics faithfully.

In addition, its contributions include a thorough analysis of the literature on Digital Twins and other related concepts.

6.2 Further Improvements

The current methodology demands that the network's entry and exit points comprise two linear edges. This requirement stems from the present mechanism of counting vehicle flows and speeds by continually contrasting the vehicle presence on these edges every second, as elaborated in Traffic Model. However, due to the relatively small scale of the networks used by the framework, it is plausible that entry and exit points might only consist of a singular edge followed by an immediate split. As a result, users might need to manually extend the network through a new OSM conversion or even introduce an artificial edge, non-existent in reality.

Therefore, it would be interesting to explore alternative approaches for tallying flows and speeds to address the constraint of entry and exit points requiring two linear edges. A solution to consider involves leveraging SUMO's Induction Loops Detectors[2] components. These virtual

[1] https://github.com/paulinho-16/Master-Thesis

[2] https://sumo.dlr.de/docs/Simulation/Output/Induction_Loops_Detectors_%28E1%29.html

sensors are positioned within the road network to emulate the operational characteristics of real-world traffic sensors. They collect simulated traffic data, facilitating a subsequent comparison with real-world data.

More importantly, testing the framework for additional networks other than the two use cases discussed in this document would be highly beneficial. Validating the framework's efficacy across diverse networks would bolster its broad applicability. Moreover, the framework could also be evaluated against other traffic schedules, considering periods with greater or lesser mobility, for example. Such testing would ensure that the framework adapts to diverse traffic conditions.

Lastly, another substantial enhancement would be automating the acquisition of average traffic intensities on the free variables' edges, which currently relies on manual user input. However, achieving this automation would require the free and complete availability of historical traffic data through an API, a resource currently unavailable, as stated in Traffic intensity on the free variables' edges.

6.3 Future Work

This framework yields base simulations that accurately capture the physical system's state. For a given scenario, it becomes compelling to derive multiple instances of the base simulation model to explore diverse what-if scenarios. These simulations can test various hypothetical situations, such as different traffic demands, changes in infrastructure like expansions or closures of roads, or technological innovations like the deployment of autonomous vehicles. In addition, incorporating a traffic forecasting component atop the base model would be significant. The forthcoming insights into traffic conditions carry numerous benefits in transportation, enabling the anticipation of decisions.

Furthermore, adapting the framework to accommodate real-time data from a continuous data streaming source would be pertinent. This adjustment would be relatively easy, given that the current methodology's operational structure aligns well with this scenario. The study in [27] proves this assertion by creating a real-time version of the model using real-time data from the ODPMS.

Lastly, using this framework to generate a Digital Twin for a large-scale network would present both a significant challenge and potential value. A compelling strategy could involve partitioning the network into subsections, encompassing multiple entries and exits, and initiating numerous framework instances for these subnetworks. The utmost challenge would entail effectively interconnecting these executions and their outcomes to formulate the complete network model. The complete VCI network comprises a great use case for this scenario since it could be simulated through numerous framework instances centred on the subnetworks established in section VCI network use case.

References

[1] Ishteaque Alam, Mohammad Fuad Ahmed, Mohaiminul Alam, João Ulisses, Dewan Md. Farid, Swakkhar Shatabda, and Rosaldo J. F. Rossetti. Pattern mining from historical traffic big data. In *2017 IEEE Region 10 Symposium (TENSYMP)*, pages 1–5, 2017.

[2] Ishteaque Alam, Dewan Md. Farid, and Rosaldo J. F. Rossetti. The prediction of traffic flow with regression analysis. In Ajith Abraham, Paramartha Dutta, Jyotsna Kumar Mandal, Abhishek Bhattacharya, and Soumi Dutta, editors, *Emerging Technologies in Data Mining and Information Security*, pages 661–671, Singapore, 2019. Springer Singapore.

[3] Javier Argota Sánchez-Vaquerizo. Getting Real: The Challenge of Building and Validating a Large-Scale Digital Twin of Barcelona's Traffic with Empirical Data. *ISPRS International Journal of Geo-Information*, 11(1), 2022.

[4] Heiko Aydt, Stephen John Turner, Wentong Cai, and Malcolm Yoke Hean Low. Research issues in symbiotic simulation. In *Proceedings of the 2009 Winter Simulation Conference (WSC)*, pages 1213–1222, 2009. ISSN: 1558-4305.

[5] Joaquim Barros, Miguel Araujo, and Rosaldo J. F. Rossetti. Short-term real-time traffic prediction methods: A survey. In *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 132–139. IEEE, 2015.

[6] A.R. Beresford and F. Stajano. Location privacy in pervasive computing. 2(1):46–55, 2003. Conference Name: IEEE Pervasive Computing.

[7] Laura Bieker, Daniel Krajzewicz, Antonio Pio Morra, Carlo Michelacci, and Fabio Cartolano. Traffic simulation for all: a real world traffic scenario from the city of Bologna. 2014.

[8] Lara Codeca, Raphael Frank, and Thomas Engel. Luxembourg SUMO Traffic (LuST) Scenario: 24 hours of mobility for vehicular networking research. In *2015 IEEE Vehicular Networking Conference (VNC)*, pages 1–8, 2015. ISSN: 2157-9865.

[9] Lara Codecá and Jérôme Härri. Towards multimodal mobility simulation of C-ITS: The Monaco SUMO traffic scenario. In *2017 IEEE Vehicular Networking Conference (VNC)*, pages 97–100, 2017. ISSN: 2157-9865.

[10] Paolo Crucitti, Vito Latora, and Sergio Porta. Centrality measures in spatial networks of urban streets. 73(3):036125, 2006. Publisher: American Physical Society.

[11] Romina Eramo, Francis Bordeleau, Benoit Combemale, Mark van den Brand, Manuel Wimmer, and Andreas Wortmann. Conceptualizing Digital Twins. 39(2):39–46, 2022. Conference Name: IEEE Software.

[12] Ricardo Ewert, Kai Martins-Turner, Carina Thaller, and Kai Nagel. Using a Route-based and Vehicle Type specific Range Constraint for Improving Vehicle Routing Problems with Electric Vehicles. 52:517–524, 2021.

[13] Rodric C. Fan, Xinnong Yang, and James D. Fay. Using location data to determine traffic information, 2003.

[14] Aidan Fuller, Zhong Fan, Charles Day, and Chris Barlow. Digital Twin: Enabling Technologies, Challenges and Open Research. 8:108952–108971, 2020.

[15] David Förster, Frank Kargl, and Hans Löhr. A framework for evaluating pseudonym strategies in vehicular ad-hoc networks. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, WiSec '15, pages 1–6. Association for Computing Machinery, 2015.

[16] David Förster, Hans Löhr, Anne Grätz, Jonathan Petit, and Frank Kargl. An Evaluation of Pseudonym Changes for Vehicular Networks in Large-Scale, Realistic Traffic Scenarios. 19(10):3400–3405, 2018. Conference Name: IEEE Transactions on Intelligent Transportation Systems.

[17] Christian Gawron. An iterative algorithm to determine the dynamic user equilibrium in a traffic simulation model. *International Journal of Modern Physics C*, 9(03):393–407, 1998.

[18] Edward Glaessgen and David Stargel. The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles. In *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*. American Institute of Aeronautics and Astronautics, 2012. _eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2012-1818.

[19] Maxime Guériau and Ivana Dusparic. Quantifying the impact of connected and autonomous vehicles on traffic efficiency and safety in mixed traffic. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, 2020.

[20] Mordechai Haklay. How Good is Volunteered Geographical Information? A Comparative Study of OpenStreetMap and Ordnance Survey Datasets. 37(4):682–703, 2010. Publisher: SAGE Publications Ltd STM.

[21] Mordechai Haklay and Patrick Weber. OpenStreetMap: User-Generated Street Maps. 7(4):12–18, 2008. Conference Name: IEEE Pervasive Computing.

[22] Jerome Harri, Fethi Filali, and Christian Bonnet. Mobility models for vehicular ad hoc networks: a survey and taxonomy. 11(4):19–41, 2009. Conference Name: IEEE Communications Surveys & Tutorials.

[23] Karl Huebner, Bjoern Schuenemann, and Ilja Radusch. Sophisticated Route Calculation Approaches for Microscopic Traffic Simulations. "1"(2), 2015.

[24] Karl-Heinz Kastner, Robert Keber, Petru Pau, and Martin Samal. Real-Time Traffic Conditions with SUMO for ITS Austria West. In Michael Behrisch, Daniel Krajzewicz, and Melanie Weber, editors, *Simulation of Urban Mobility*, Lecture Notes in Computer Science, pages 146–159. Springer, 2014.

[25] Kaveh Khoshkhah, Mozhgan Pourmoradnasseri, Amnir Hadachi, Helen Tera, Jakob Mass, Erald Keshi, and Shan Wu. Real-Time System for Daily Modal Split Estimation and OD Matrices Generation Using IoT Data: A Case Study of Tartu City. *Sensors*, 22(8), 2022.

[26] Daniel Krajzewicz, Robbin J. Blokpoel, Fabio Cartolano, Pasquale Cataldi, Ainara Gonzalez, Oscar Lazaro, Jérémie Leguay, Lan Lin, Julen Maneros, and Michele Rondinone. iTETRIS - A System for the Evaluation of Cooperative Traffic Management Solutions. In Gereon Meyer and Jürgen Valldorf, editors, *Advanced Microsystems for Automotive Applications 2010*, VDI-Buch, pages 399–410. Springer, 2010.

[27] Krešimir Kušić, Rene Schumann, and Edouard Ivanjko. Building a Motorway Digital Twin in SUMO: Real-Time Simulation of Continuous Data Stream from Traffic Counters. In *2022 International Symposium ELMAR*, pages 71–76, 2022. ISSN: 1334-2630.

[28] Krešimir Kušić, René Schumann, and Edouard Ivanjko. A digital twin in transportation: Real-time synergy of traffic data streams and simulation for virtualizing motorway dynamics. 55:101858, 2023.

[29] Jiewu Leng, Hao Zhang, Douxi Yan, Qiang Liu, Xin Chen, and Ding Zhang. Digital twin-driven manufacturing cyber-physical system for parallel controlling of smart workshop. 10(3):1155–1166, 2019.

[30] Silas C. Lobo, Stefan Neumeier, Evelio M. G. Fernandez, and Christian Facchi. InTAS – The Ingolstadt Traffic Scenario for SUMO, 2020.

[31] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wiessner. Microscopic Traffic Simulation using SUMO. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582, 2018. ISSN: 2153-0017.

[32] Malcolm Yoke Hean Low, Kong Wei Lye, Peter Lendermann, Stephen John Turner, Reman Tat Wee Chim, and Surya Hadisaputra Leo. An agent-based approach for managing symbiotic simulation of semiconductor assembly and test operation. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, AAMAS '05, pages 85–92. Association for Computing Machinery, 2005.

[33] Francisco J. Martinez, Chai Keong Toh, Juan-Carlos Cano, Carlos T. Calafate, and Pietro Manzoni. A survey and comparative study of simulators for vehicular ad hoc networks (VANETs). 11(7):813–828, 2011. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wcm.859.

[34] Seri Oh, Stephen G. Ritchie, and Cheol Oh. Real-Time Traffic Measurement from Single Loop Inductive Signatures. 1804(1):98–106, 2002. Publisher: SAGE Publications Inc.

[35] Bhakti Stephan Onggo, Navonil Mustafee, Andi Smart, Angel A. Juan, and Owen Molloy. Symbiotic Simulation System: Hybrid Systems Model meets Big Data Analytics. In *2018 Winter Simulation Conference (WSC)*, pages 1358–1369, 2018. ISSN: 1558-4305.

[36] Panagiotis Papadimitratos, Levente Buttyan, Tamas Holczer, Elmar Schoch, Julien Freudiger, Maxim Raya, Zhendong Ma, Frank Kargl, Antonio Kung, and Jean-Pierre Hubaux. Secure vehicular communication systems: design and architecture. 46(11):100–109, 2008. Conference Name: IEEE Communications Magazine.

[37] Lúcio Sanchez Passos, Rosaldo J. F. Rossetti, and Zafeiris Kokkinogenis. Towards the next-generation traffic simulation tools: a first appraisal. In *6th Iberian Conference on Information Systems and Technologies (CISTI 2011)*, pages 1–6, 2011.

[38] Jonathan Petit, Djurrre Broekhuis, Michael Feiri, and Frank Kargl. Connected vehicles: Surveillance threat and mitigation. *Black Hat Europe*, 11(2015):1, 2015.

[39] Jonathan Petit, Florian Schaub, Michael Feiri, and Frank Kargl. Pseudonym Schemes in Vehicular Networks: A Survey. 17(1):228–255, 2015. Conference Name: IEEE Communications Surveys & Tutorials.

[40] Marco Rapelli, Claudio Casetti, and Giandomenico Gagliardi. TuST: from Raw Data to Vehicular Traffic Simulation in Turin. In *2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 1–8, 2019. ISSN: 1550-6525.

[41] Marco Rapelli, Claudio Casetti, and Giandomenico Gagliardi. Vehicular Traffic Simulation in the City of Turin From Raw Data. 21(12):4656–4666, 2022. Conference Name: IEEE Transactions on Mobile Computing.

[42] Andrey Rudskoy, Igor Ilin, and Andrey Prokhorov. Digital Twins in the Intelligent Transport Systems. 54:927–935, 2021.

[43] Karl Schrab, Robert Protzmann, and Ilja Radusch. A Large-Scale Traffic Scenario of Berlin for Evaluating Smart Mobility Applications. In Eftihia G. Nathanail, Nikolaos Gavanas, and Giannis Adamos, editors, *Smart Energy for Smart Transport*, Lecture Notes in Intelligent Transportation and Infrastructure, pages 276–287. Springer Nature Switzerland, 2023.

[44] Björn Schünemann. V2X simulation runtime infrastructure VSimRTI: An assessment tool to design smart traffic management systems. 55(14):3189–3198, 2011.

[45] Concetta Semeraro, Mario Lezoche, Hervé Panetto, and Michele Dassisti. Digital twin paradigm: A systematic literature review. 130:103469, 2021.

[46] João Soares, Cristina Lobo, Zita Vale, and P. B. de Moura Oliveira. Realistic traffic scenarios using a census methodology: Vila real case study. In *2014 IEEE PES General Meeting | Conference & Exposition*, pages 1–5, 2014. ISSN: 1932-5517.

[47] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. 10(1):3–15, 2011. Conference Name: IEEE Transactions on Mobile Computing.

[48] Sandesh Uppoor, Oscar Trullols-Cruces, Marco Fiore, and Jose M. Barcelo-Ordinas. Generation and Analysis of a Large-Scale Urban Vehicular Mobility Dataset. 13(5):1061–1075, 2014. Conference Name: IEEE Transactions on Mobile Computing.

[49] Kay W. Axhausen, Andreas Horni, and Kai Nagel, editors. *The Multi-Agent Transport Simulation MATSim*. Ubiquity Press, 2016. Accepted: 2016-12-31 23:55:55.

[50] Axel Wegener, Michał Piórkowski, Maxim Raya, Horst Hellbrück, Stefan Fischer, and Jean-Pierre Hubaux. TraCI: an interface for coupling road traffic and network simulators. In *Proceedings of the 11th communications and networking simulation symposium*, CNS '08, pages 155–163. Association for Computing Machinery, 2008.

[51] Gary White, Anna Zink, Lara Codecá, and Siobhán Clarke. A digital twin smart city for citizen feedback. 110:103064, 2021.

[52] Björn Wiedersheim, Zhendong Ma, Frank Kargl, and Panos Papadimitratos. Privacy in inter-vehicular networks: Why simple pseudonym change is not enough. In *2010 Seventh International Conference on Wireless On-demand Network Systems and Services (WONS)*, pages 176–183, 2010.

[53] Dominik Ziemke, Ihab Kaddoura, and Kai Nagel. The MATSim Open Berlin Scenario: A multimodal agent-based transport simulation scenario based on synthetic demand modeling and open data. 151:870–877, 2019.

Appendix A

VCI nodes' networks and equations

A.1 *Nó do Areinho* (6 equations)

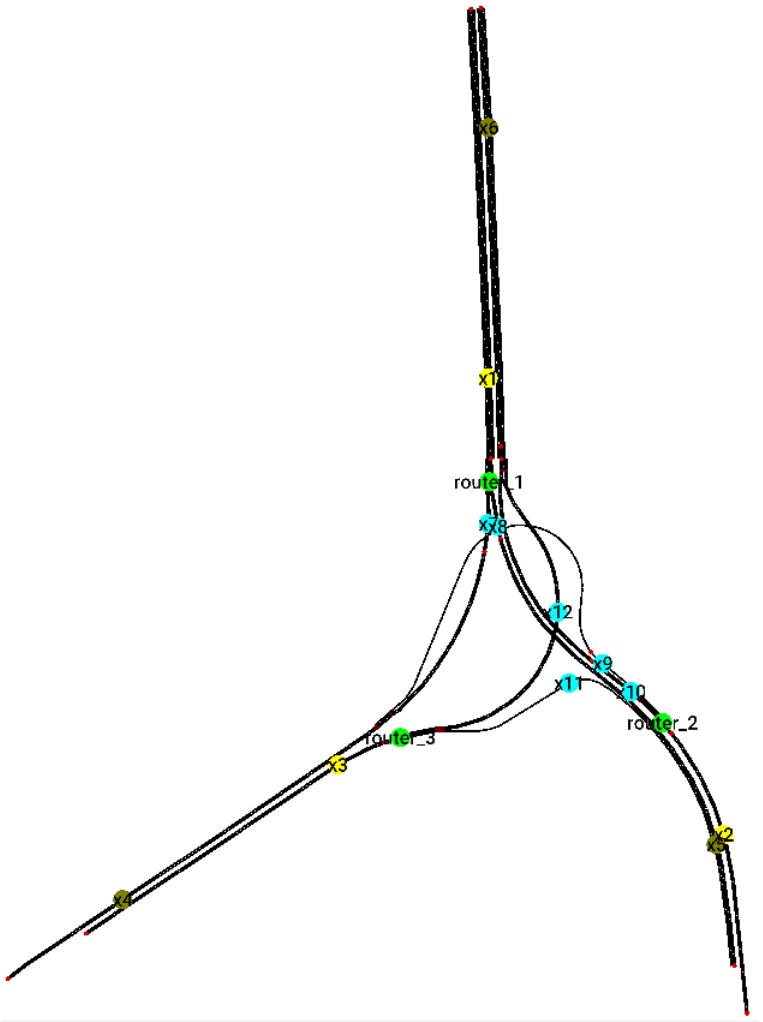


Figure A.1: VCI's *Nó do Areinho* node network and assigned variables

$$
\begin{aligned}
x_{10} + x_{12} - x_6 &= 0 \\
x_{11} + x_{12} - x_3 &= 0 \\
x_7 + x_8 - x_1 &= 0 \\
x_7 + x_9 - x_4 &= 0 \\
x_8 + x_{11} - x_5 &= 0 \\
x_9 + x_{10} - x_2 &= 0
\end{aligned}
\tag{A.1}
$$

A.2 *Nó do Freixo* (13 equations)

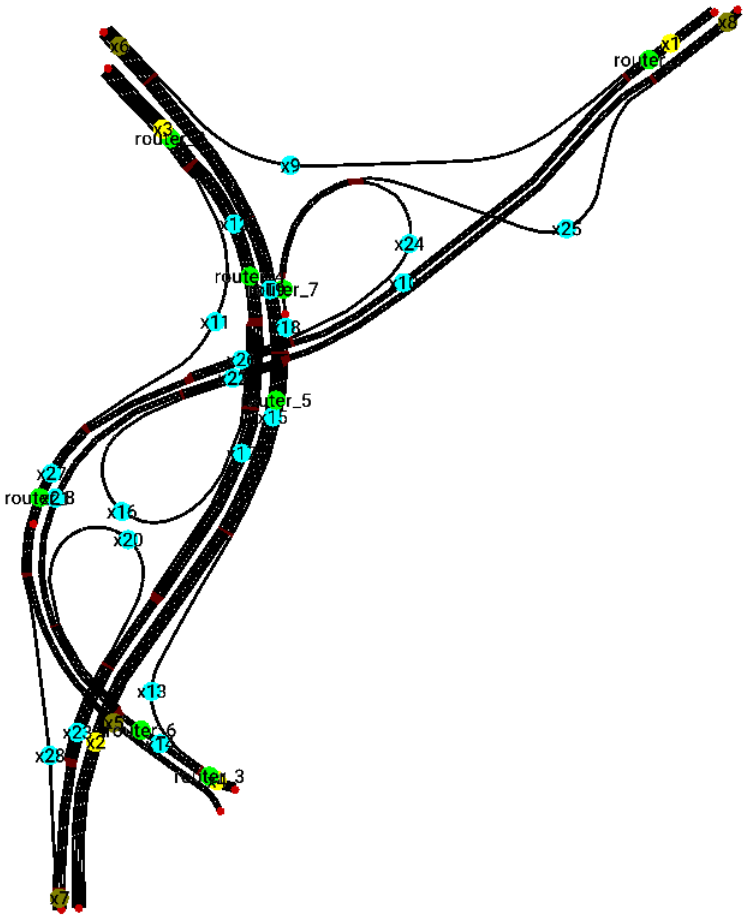


Figure A.2: VCI's *Nó do Freixo* node network and assigned variables

$$
\begin{aligned}
x_{10} + x_{11} + x_{24} - x_{27} &= 0 \\
x_{11} + x_{12} - x_3 &= 0 \\
x_{13} + x_{14} - x_4 &= 0 \\
x_{16} + x_{17} - x_{12} &= 0 \\
x_{16} + x_{21} + x_{25} - x_8 &= 0 \\
x_{17} + x_{20} + x_{28} - x_7 &= 0 \\
x_{18} + x_{19} - x_{15} &= 0 \\
x_2 + x_{13} - x_{15} &= 0 \\
x_{20} + x_{21} - x_{14} &= 0 \\
x_{24} + x_{25} - x_{18} &= 0 \\
x_5 + x_{28} - x_{27} &= 0 \\
x_9 + x_{10} - x_1 &= 0 \\
x_9 + x_{19} - x_6 &= 0
\end{aligned}
\tag{A.2}
$$

A.3 *Nó da Avenida 25 de Abril* (14 equations)

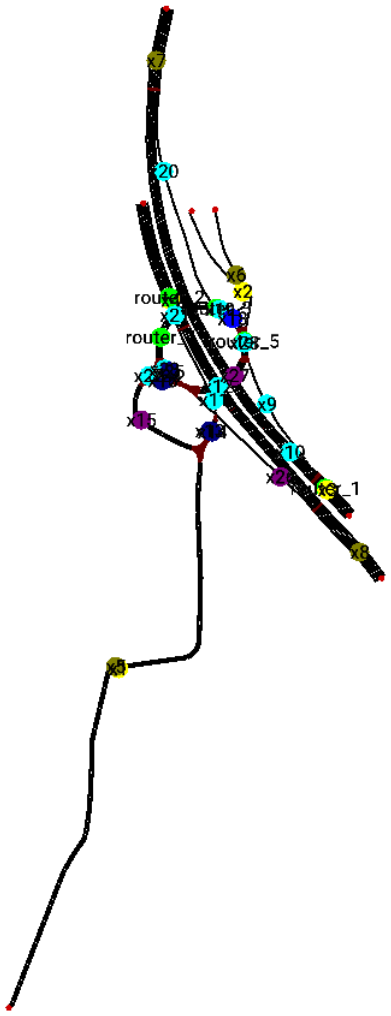


Figure A.3: VCI's *Nó da Avenida 25 de Abril* node network and assigned variables

$$
\begin{aligned}
x_1 + x_{13} - x_{15} &= 0 \\
x_{10} + x_{20} - x_7 &= 0 \\
x_{11} + x_{12} - x_4 &= 0 \\
x_{12} + x_{16} + x_{24} - x_8 &= 0 \\
x_{13} + x_{14} - x_{11} &= 0 \\
x_{14} + x_{22} - x_5 &= 0 \\
x_{16} + x_{17} - x_{15} &= 0 \\
x_2 + x_{18} - x_{19} &= 0 \\
x_{20} + x_{21} - x_{19} &= 0 \\
x_{22} + x_{23} - x_{21} &= 0 \\
x_{24} + x_{25} - x_{23} &= 0 \\
x_6 + x_{18} - x_{28} &= 0 \\
x_9 + x_{10} - x_3 &= 0 \\
x_9 + x_{17} + x_{25} - x_{28} &= 0
\end{aligned}
\tag{A.3}
$$

A.4 *Nó do Mercado Abastecedor* (19 equations)

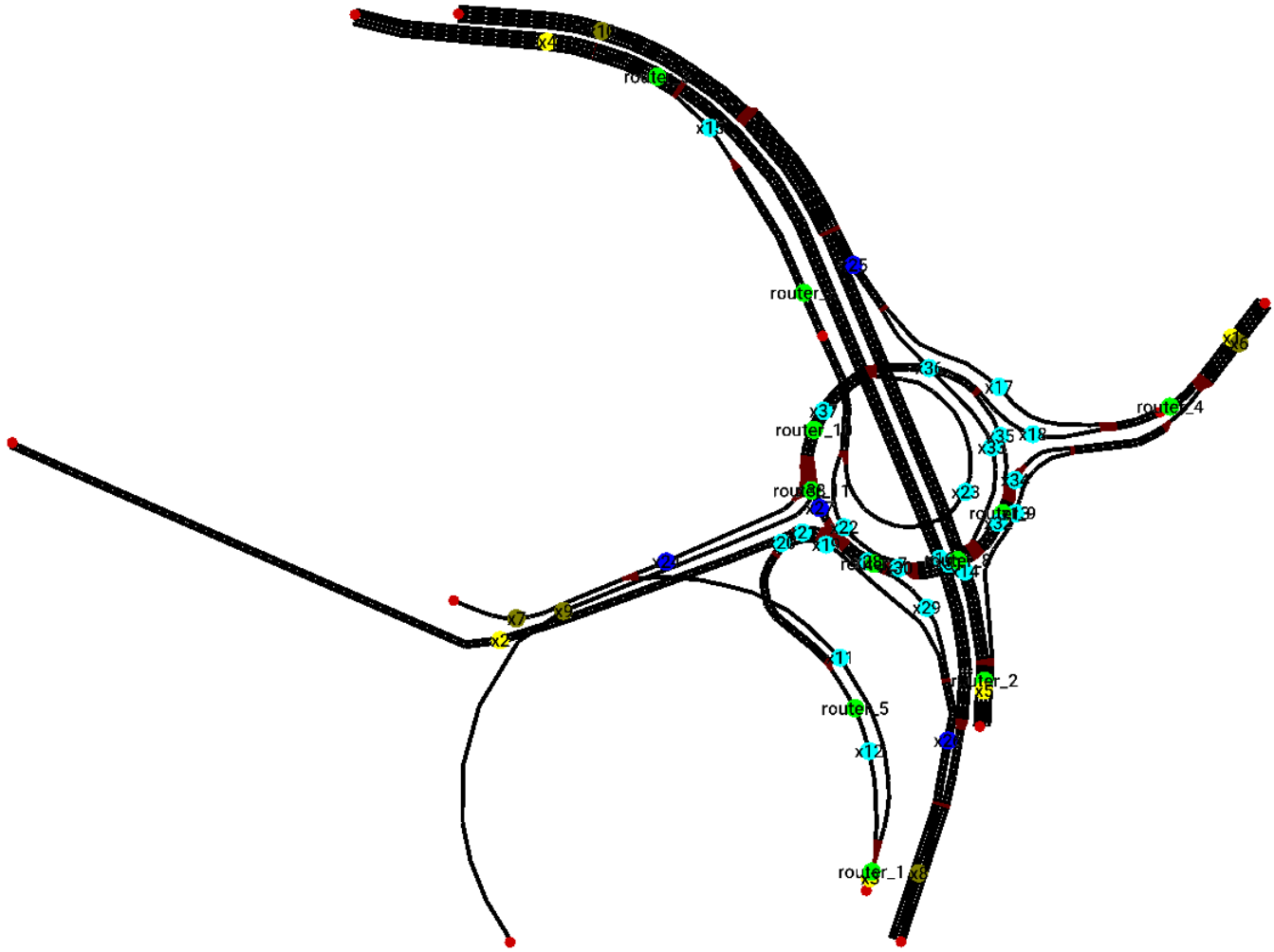


Figure A.4: VCI's *Nó do Mercado Abastecedor* node network and assigned variables

$$
\begin{aligned}
x_{11} + x_{12} - x_3 &= 0 \\
x_{11} + x_{24} - x_7 &= 0 \\
x_{13} + x_{14} - x_5 &= 0 \\
x_{13} + x_{34} - x_6 &= 0 \\
x_{14} + x_{25} - x_{10} &= 0 \\
x_{15} + x_{16} - x_4 &= 0 \\
x_{16} + x_{26} - x_8 &= 0 \\
x_{17} + x_{18} - x_1 &= 0 \\
x_{17} + x_{33} - x_{25} &= 0 \\
x_{18} + x_{23} + x_{35} - x_{37} &= 0 \\
x_{19} + x_{20} - x_{12} &= 0 \\
x_{19} + x_{29} - x_{26} &= 0 \\
x_2 + x_{20} + x_{27} - x_{28} &= 0 \\
x_{22} + x_{23} - x_{15} &= 0 \\
x_{22} + x_{30} - x_{31} &= 0 \\
x_{29} + x_{30} - x_{28} &= 0 \\
x_{32} + x_{33} - x_{31} &= 0 \\
x_{34} + x_{35} - x_{32} &= 0 \\
x_9 + x_{24} + x_{27} - x_{37} &= 0
\end{aligned}
\tag{A.4}
$$

A.5 *Nó das Antas* (11 equations)

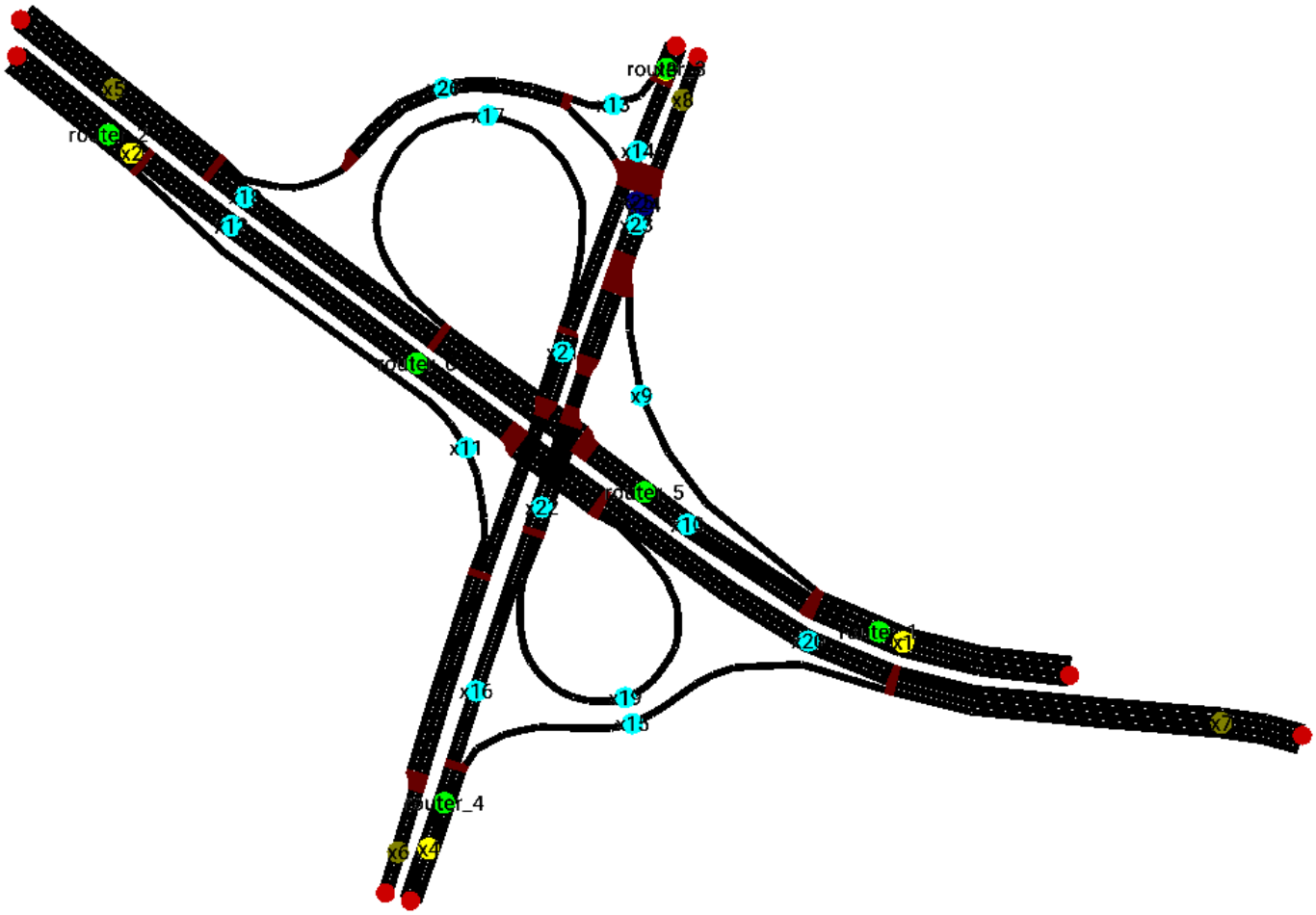


Figure A.5: VCI's *Nó das Antas* node network and assigned variables

$$
\begin{aligned}
x_{11} + x_{12} - x_2 &= 0 \\
x_{11} + x_{14} + x_{17} - x_6 &= 0 \\
x_{13} + x_{14} - x_3 &= 0 \\
x_{13} + x_{18} + x_{25} - x_5 &= 0 \\
x_{15} + x_{16} - x_4 &= 0 \\
x_{15} + x_{20} - x_7 &= 0 \\
x_{17} + x_{18} - x_{10} &= 0 \\
x_{19} + x_{20} - x_{12} &= 0 \\
x_{24} + x_{25} - x_{23} &= 0 \\
x_9 + x_{10} - x_1 &= 0 \\
x_9 + x_{16} + x_{19} - x_{23} &= 0
\end{aligned}
\tag{A.5}
$$

A.6 *Nó de Entre-Douro-e-Minho* (6 equations)

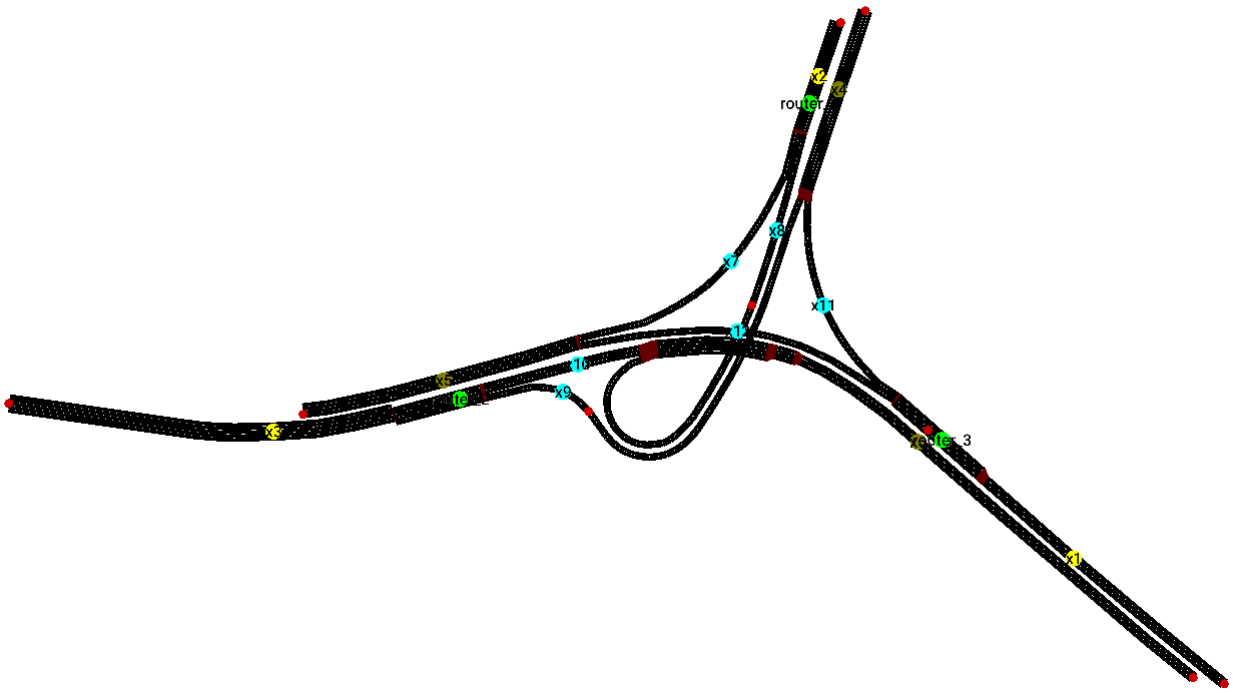


Figure A.6: VCI's *Nó de Entre-Douro-e-Minho* node network and assigned variables

$$
\begin{aligned}
x_{11} + x_{12} - x_1 &= 0 \\
x_7 + x_{12} - x_5 &= 0 \\
x_7 + x_8 - x_2 &= 0 \\
x_8 + x_{10} - x_6 &= 0 \\
x_9 + x_{10} - x_3 &= 0 \\
x_9 + x_{11} - x_4 &= 0
\end{aligned}
\tag{A.6}
$$

A.7 *Nó de Paranhos* (16 equations)

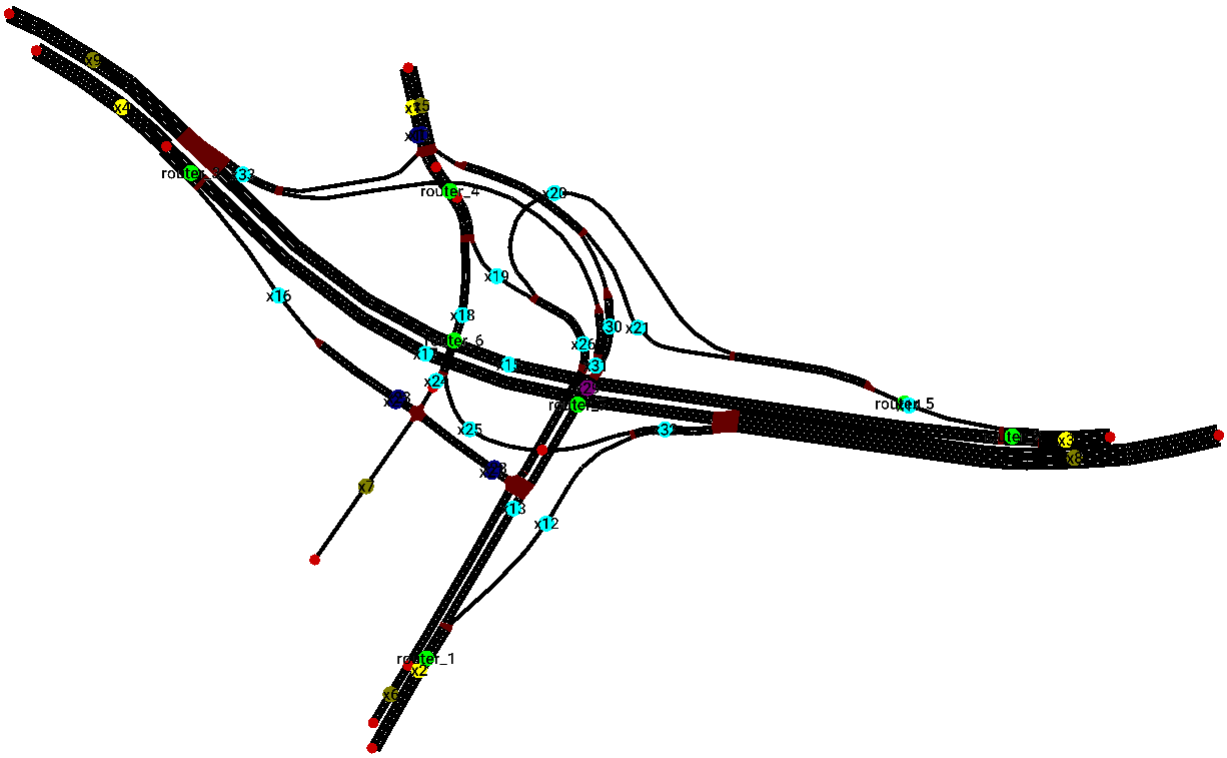


Figure A.7: VCI's *Nó de Paranhos* node network and assigned variables

$$
\begin{aligned}
x_{10} + x_{11} - x_1 &= 0 \\
x_{10} + x_{15} + x_{31} - x_9 &= 0 \\
x_{12} + x_{13} - x_2 &= 0 \\
x_{12} + x_{17} + x_{25} - x_8 &= 0 \\
x_{13} + x_{28} - x_{29} &= 0 \\
x_{14} + x_{15} - x_3 &= 0 \\
x_{16} + x_{17} - x_4 &= 0 \\
x_{18} + x_{19} - x_{11} &= 0 \\
x_{19} + x_{20} + x_{27} - x_6 &= 0 \\
x_{20} + x_{21} - x_{14} &= 0 \\
x_{21} + x_{30} - x_5 &= 0 \\
x_{22} + x_{23} - x_{16} &= 0 \\
x_{22} + x_{24} - x_7 &= 0 \\
x_{24} + x_{25} - x_{18} &= 0 \\
x_{27} + x_{28} - x_{23} &= 0 \\
x_{30} + x_{31} - x_{29} &= 0
\end{aligned}
\tag{A.7}
$$

A.8 *Nó de São João Bosco* (2 equations)

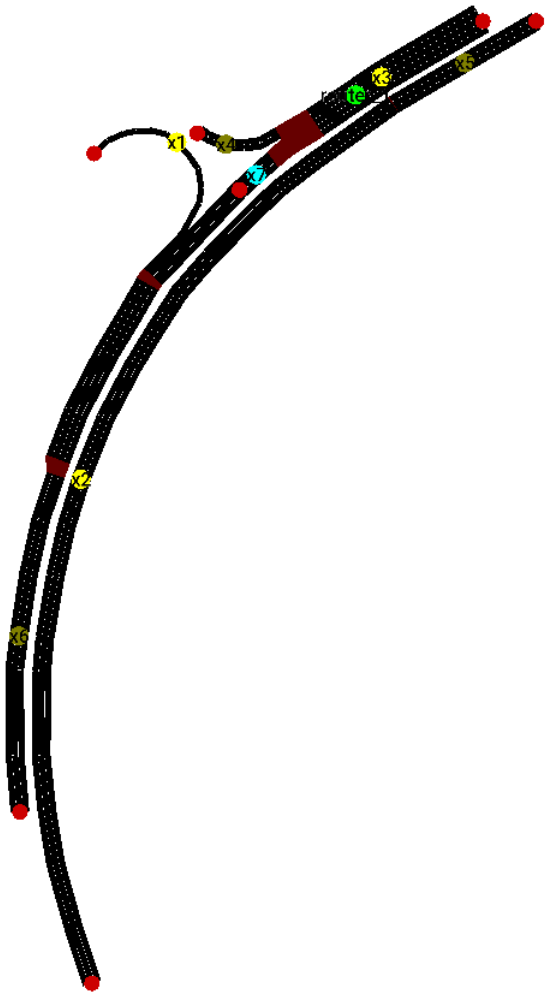


Figure A.8: VCI's *Nó de São João Bosco* node network and assigned variables

$$\begin{aligned} x_1 + x_7 - x_6 &= 0 \\ x_4 + x_7 - x_3 &= 0 \end{aligned} \tag{A.8}$$

A.9 *Nó da Avenida da Boavista* (12 equations)

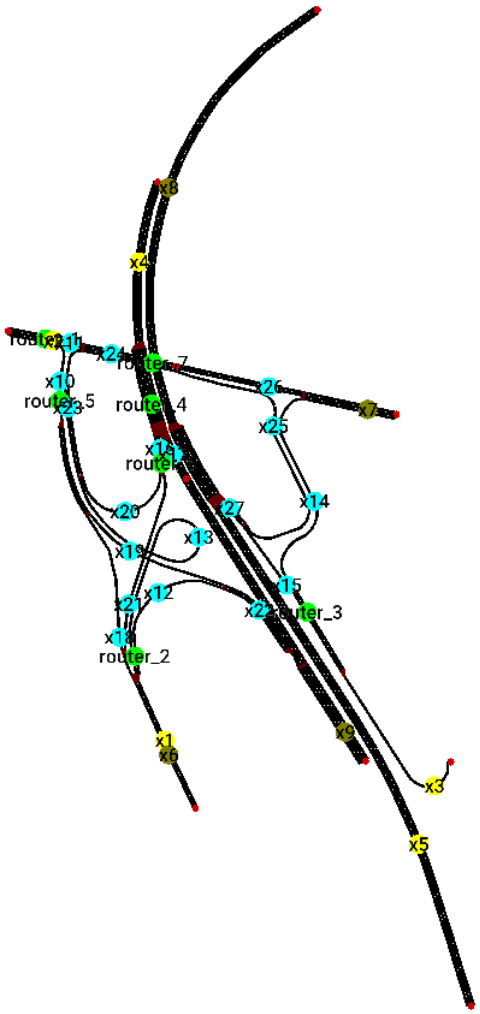


Figure A.9: VCI's *Nó da Avenida da Boavista* node network and assigned variables

$$
\begin{aligned}
x_{10} + x_{11} - x_2 &= 0 \\
x_{11} + x_{13} + x_{20} - x_{24} &= 0 \\
x_{12} + x_{13} - x_1 &= 0 \\
x_{12} + x_{17} + x_{19} - x_9 &= 0 \\
x_{14} + x_{15} - x_3 &= 0 \\
x_{14} + x_{26} - x_7 &= 0 \\
x_{16} + x_{17} - x_4 &= 0 \\
x_{18} + x_{19} - x_{10} &= 0 \\
x_{18} + x_{21} - x_6 &= 0 \\
x_{20} + x_{21} - x_{16} &= 0 \\
x_{25} + x_{26} - x_{24} &= 0 \\
x_5 + x_{15} + x_{25} - x_8 &= 0
\end{aligned}
\tag{A.9}
$$

A.10 *Nó do Amial* (22 equations)

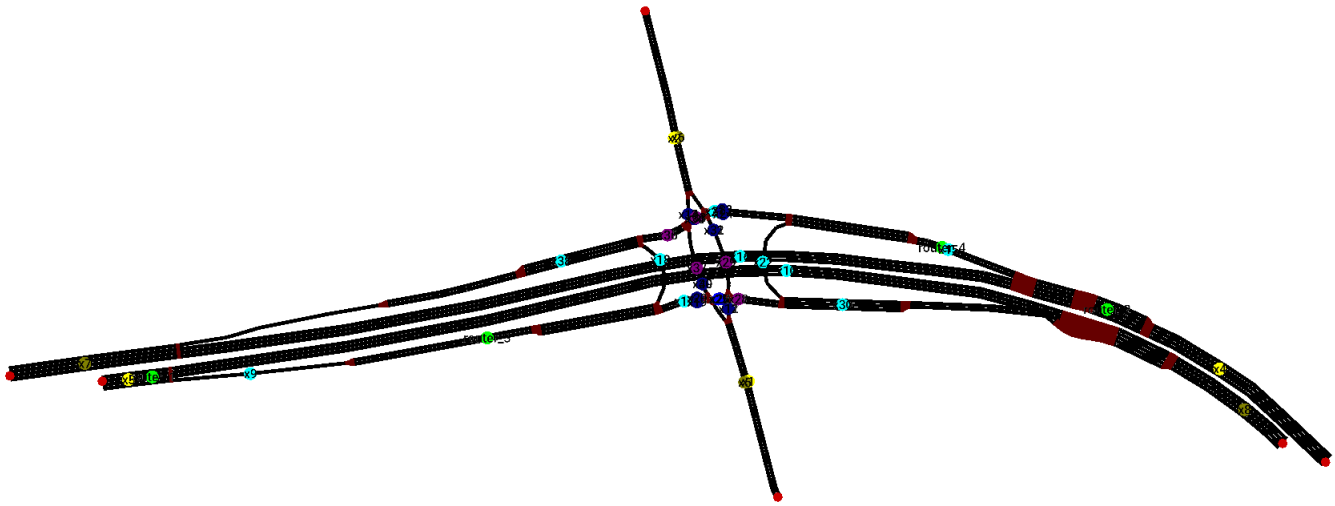


Figure A.10: VCI's *Nó do Amial* node network and assigned variables

$$
\begin{aligned}
x_{10}+x_{22}+x_{28}-x_8&=0\\
x_{11}+x_{12}-x_1&=0\\
x_{11}+x_{22}+x_{26}-x_{30}&=0\\
x_{12}+x_{27}-x_{29}&=0\\
x_{13}+x_{14}-x_2&=0\\
x_{13}+x_{18}+x_{34}-x_{38}&=0\\
x_{14}+x_{35}-x_{37}&=0\\
x_{15}+x_{16}-x_4&=0\\
x_{16}+x_{18}+x_{36}-x_7&=0\\
x_{17}+x_{18}-x_9&=0\\
x_{19}+x_{20}-x_{17}&=0\\
x_{19}+x_{39}-x_5&=0\\
x_{20}+x_{40}-x_{25}&=0\\
x_{21}+x_{22}-x_{15}&=0\\
x_{23}+x_{24}-x_{21}&=0\\
x_{23}+x_{31}-x_6&=0\\
x_{24}+x_{32}-x_{33}&=0\\
x_{26}+x_{27}-x_{25}&=0\\
x_{31}+x_{32}-x_{29}&=0\\
x_{34}+x_{35}-x_{33}&=0\\
x_{39}+x_{40}-x_{37}&=0\\
x_9+x_{10}-x_3&=0
\end{aligned}
\tag{A.10}
$$

A.11 *Nó da Via Norte* (12 equations)

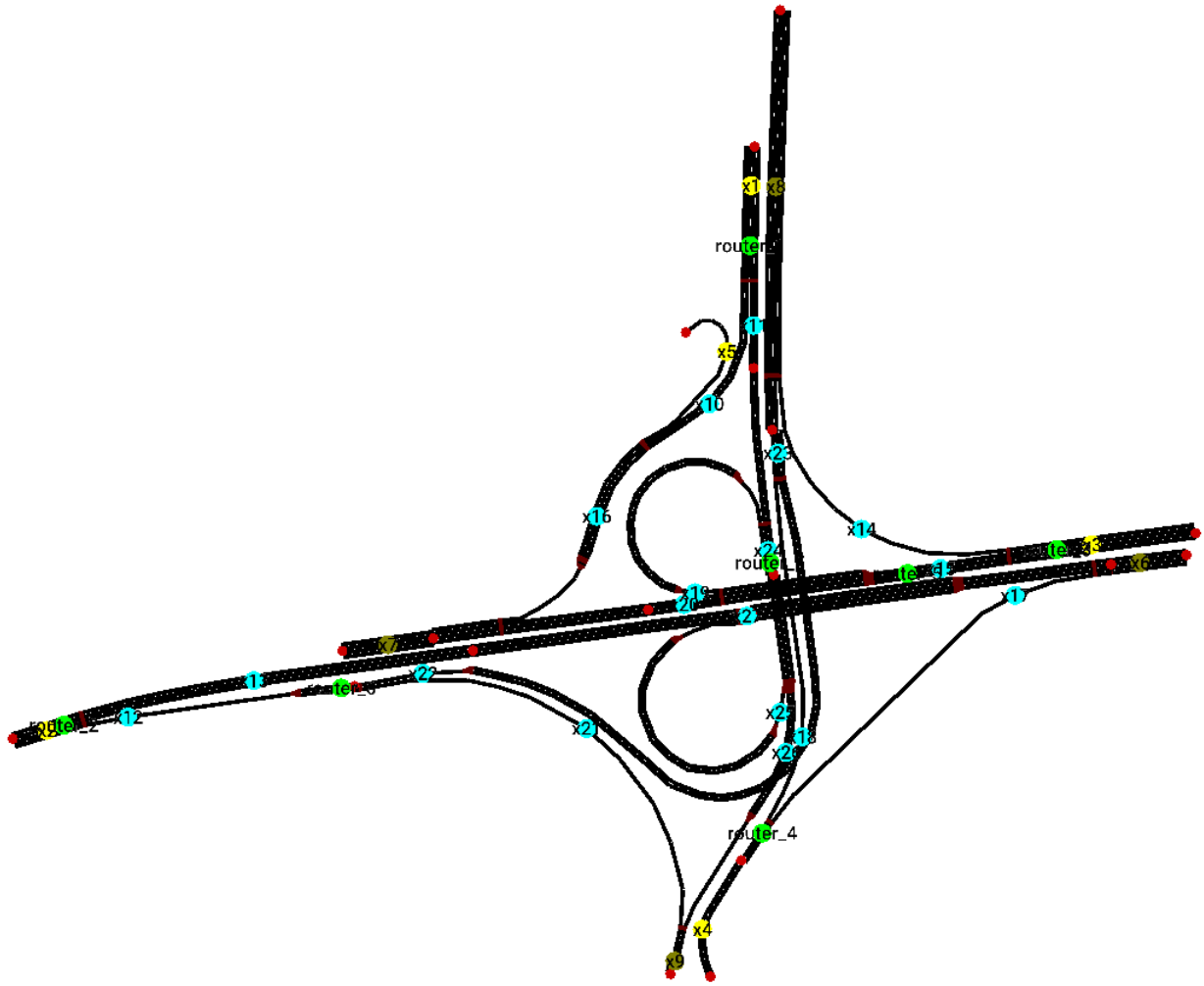


Figure A.11: VCI's *Nó da Via Norte* node network and assigned variables

$$
\begin{aligned}
x_{10} + x_{11} - x_1 &= 0 \\
x_{11} + x_{19} - x_{24} &= 0 \\
x_{12} + x_{13} - x_2 &= 0 \\
x_{13} + x_{17} + x_{25} - x_6 &= 0 \\
x_{14} + x_{15} - x_3 &= 0 \\
x_{14} + x_{18} + x_{22} - x_8 &= 0 \\
x_{17} + x_{18} - x_4 &= 0 \\
x_{19} + x_{20} - x_{15} &= 0 \\
x_{21} + x_{22} - x_{12} &= 0 \\
x_{21} + x_{26} - x_9 &= 0 \\
x_{25} + x_{26} - x_{24} &= 0 \\
x_5 + x_{10} + x_{20} - x_7 &= 0
\end{aligned}
\tag{A.11}
$$

A.12 *Nó da Associação Empresarial* (20 equations)

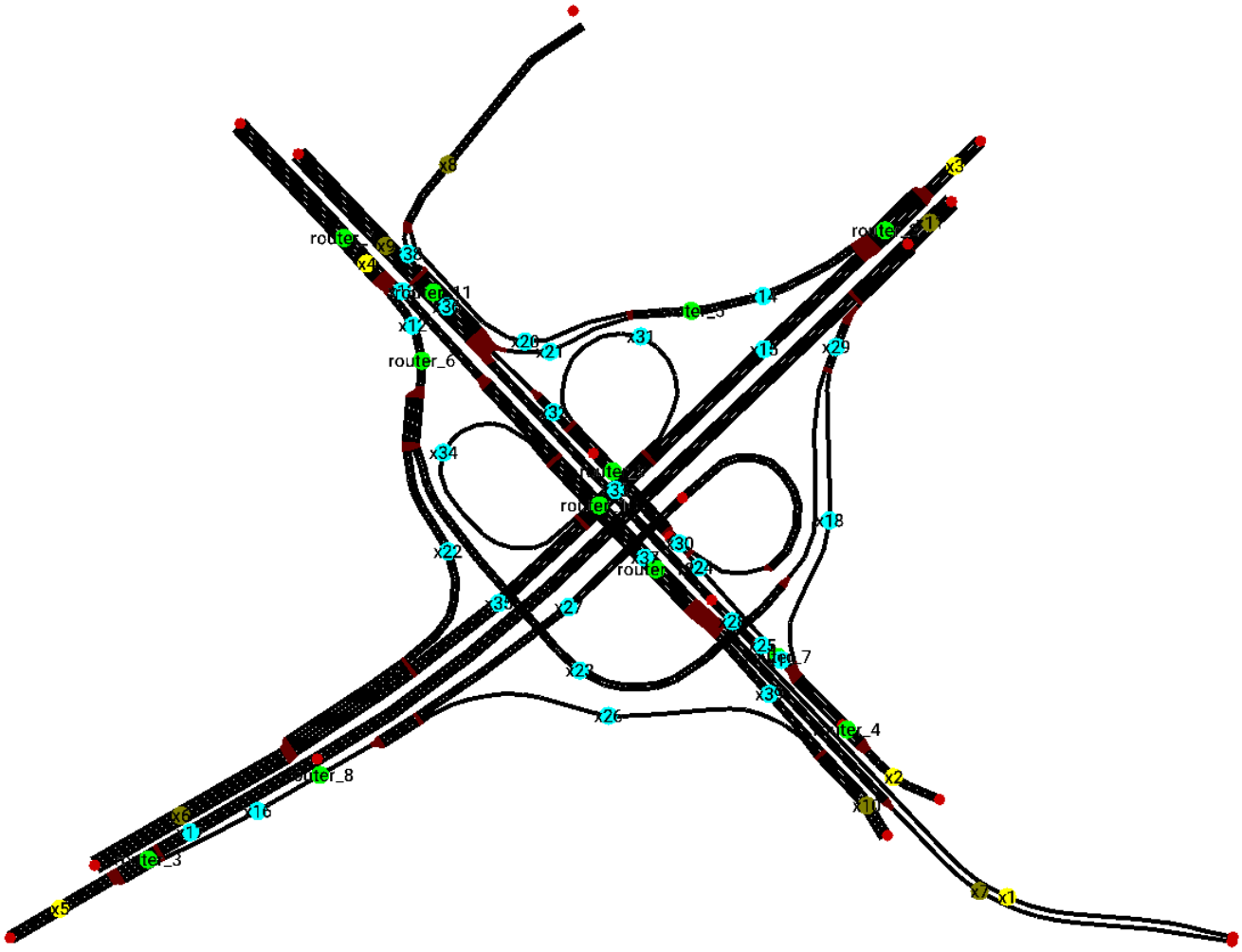


Figure A.12: VCI's *Nó da Associação Empresarial* node network and assigned variables

$$
\begin{aligned}
x_1 + x_{21} + x_{25} + x_{32} - x_{36} &= 0 \\
x_{12} + x_{13} - x_4 &= 0 \\
x_{13} + x_{34} - x_{37} &= 0 \\
x_{14} + x_{15} - x_3 &= 0 \\
x_{15} + x_{31} - x_{33} &= 0 \\
x_{16} + x_{17} - x_5 &= 0 \\
x_{17} + x_{18} + x_{23} - x_{11} &= 0 \\
x_{18} + x_{19} - x_2 &= 0 \\
x_{20} + x_{21} - x_{14} &= 0 \\
x_{20} + x_{38} - x_8 &= 0 \\
x_{22} + x_{23} - x_{12} &= 0 \\
x_{22} + x_{35} - x_6 &= 0 \\
x_{24} + x_{25} - x_{19} &= 0 \\
x_{24} + x_{27} - x_{30} &= 0 \\
x_{26} + x_{27} - x_{16} &= 0 \\
x_{26} + x_{39} - x_{10} &= 0 \\
x_{31} + x_{32} - x_{30} &= 0 \\
x_{34} + x_{35} - x_{33} &= 0 \\
x_7 + x_{39} - x_{37} &= 0 \\
x_9 + x_{38} - x_{36} &= 0
\end{aligned}
\tag{A.12}
$$

A.13 *Nó da Via Panorâmica* (5 equations)

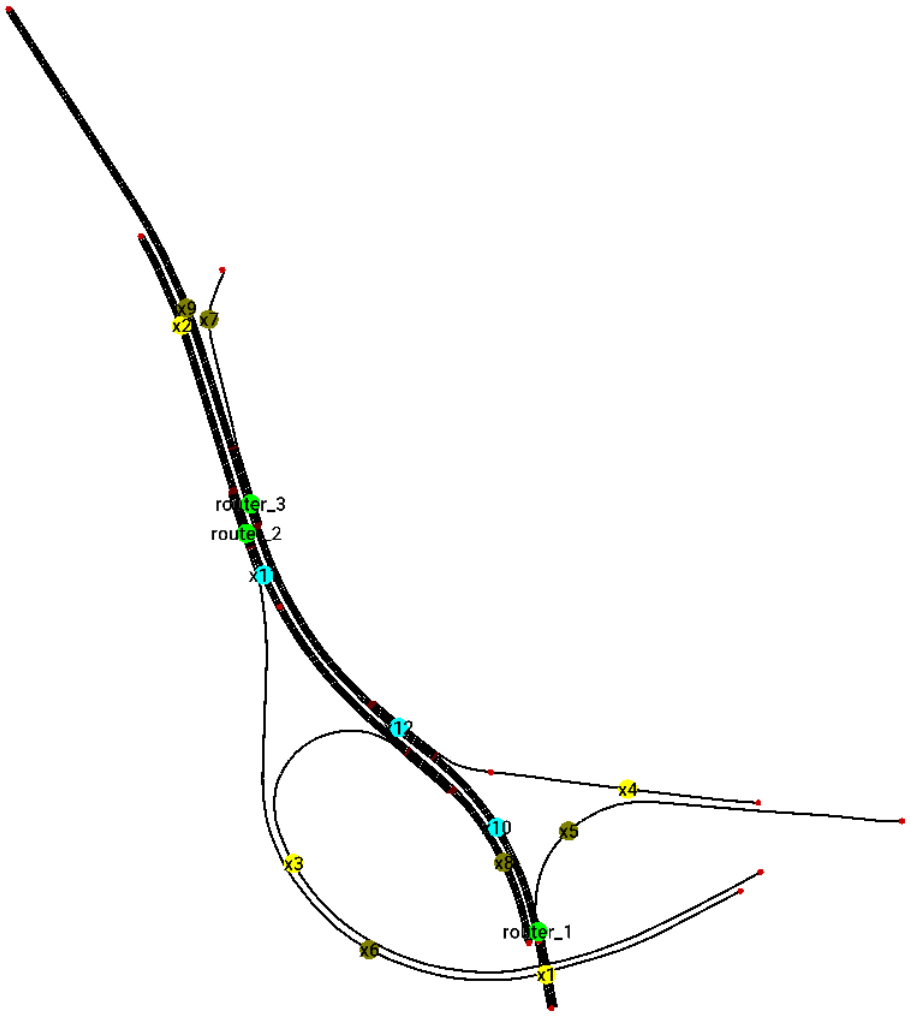


Figure A.13: VCI's *Nó da Via Panorâmica* node network and assigned variables

$$
\begin{aligned}
x_3 + x_{11} - x_8 &= 0 \\
x_4 + x_{10} - x_{12} &= 0 \\
x_5 + x_{10} - x_1 &= 0 \\
x_6 + x_{11} - x_2 &= 0 \\
x_7 + x_9 - x_{12} &= 0
\end{aligned}
\tag{A.13}
$$

A.14 *Nó da Afurada* (17 equations)

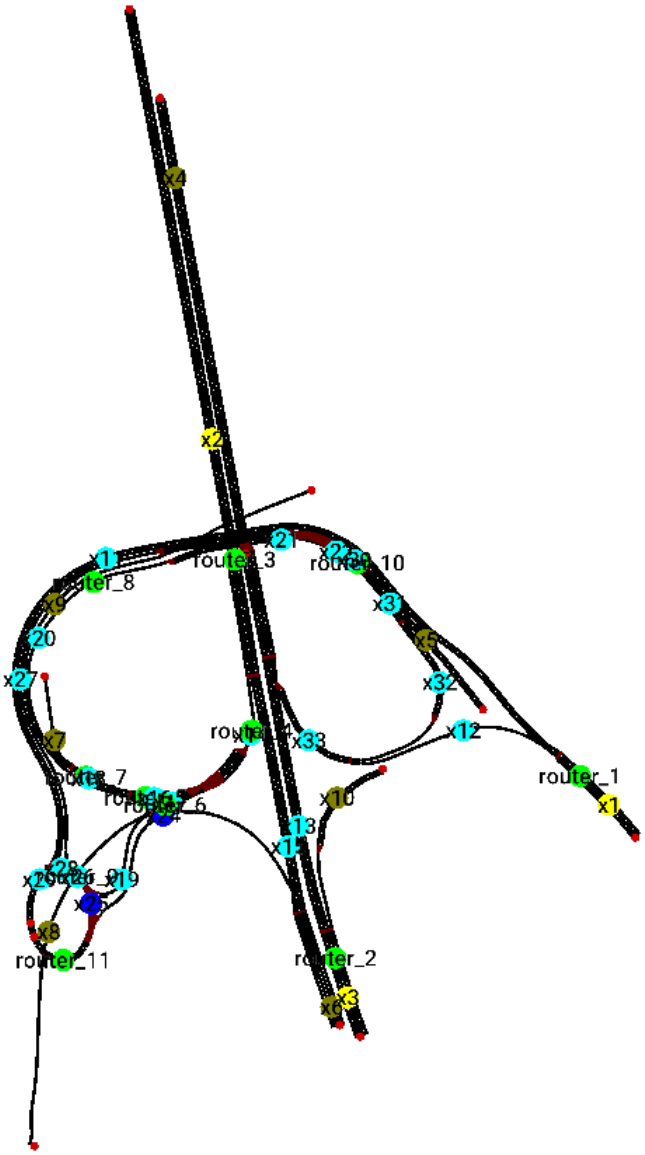


Figure A.14: VCI's *Nó da Afurada* node network and assigned variables

$$
\begin{aligned}
x_{10} + x_{13} - x_3 &= 0 \\
x_{11} + x_{12} - x_1 &= 0 \\
x_{11} + x_{28} - x_{29} &= 0 \\
x_{12} + x_{13} + x_{32} - x_4 &= 0 \\
x_{12} + x_{22} + x_{31} - x_{33} &= 0 \\
x_{14} + x_{15} - x_2 &= 0 \\
x_{15} + x_{24} - x_6 &= 0 \\
x_{16} + x_{17} - x_{14} &= 0 \\
x_{19} + x_{25} - x_{26} &= 0 \\
x_{21} + x_{22} - x_{20} &= 0 \\
x_{21} + x_{27} - x_{30} &= 0 \\
x_{24} + x_{25} - x_{29} &= 0 \\
x_{27} + x_{28} - x_{26} &= 0 \\
x_5 + x_{31} - x_{30} &= 0 \\
x_7 + x_{20} - x_{18} &= 0 \\
x_8 + x_{19} - x_{17} &= 0 \\
x_9 + x_{18} - x_{16} &= 0
\end{aligned}
\tag{A.14}
$$

A.15 *Nó das Devesas* (4 equations)

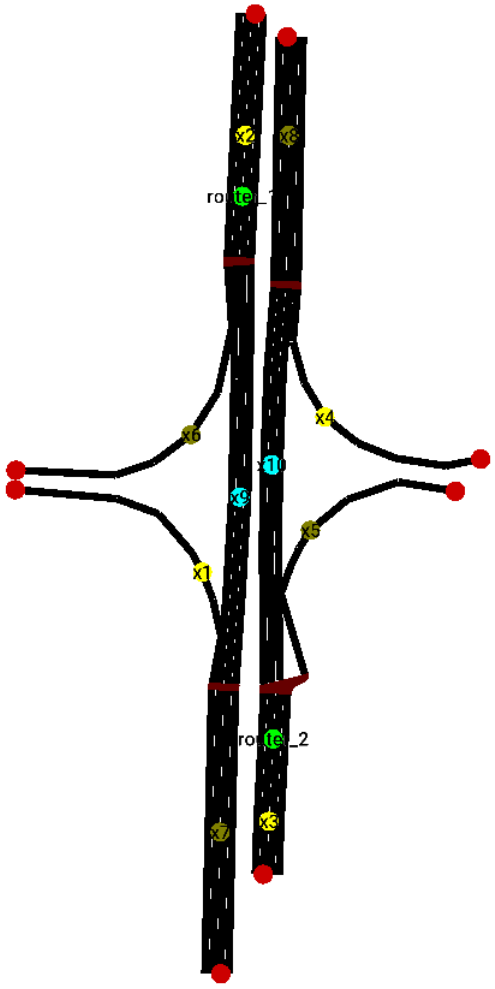


Figure A.15: VCI's *Nó das Devesas* node network and assigned variables

$$\begin{aligned} x_1 + x_9 - x_7 &= 0 \\ x_4 + x_{10} - x_8 &= 0 \\ x_5 + x_{10} - x_3 &= 0 \\ x_6 + x_9 - x_2 &= 0 \end{aligned} \tag{A.15}$$

A.16 *Nó de Coimbrões* (15 equations)

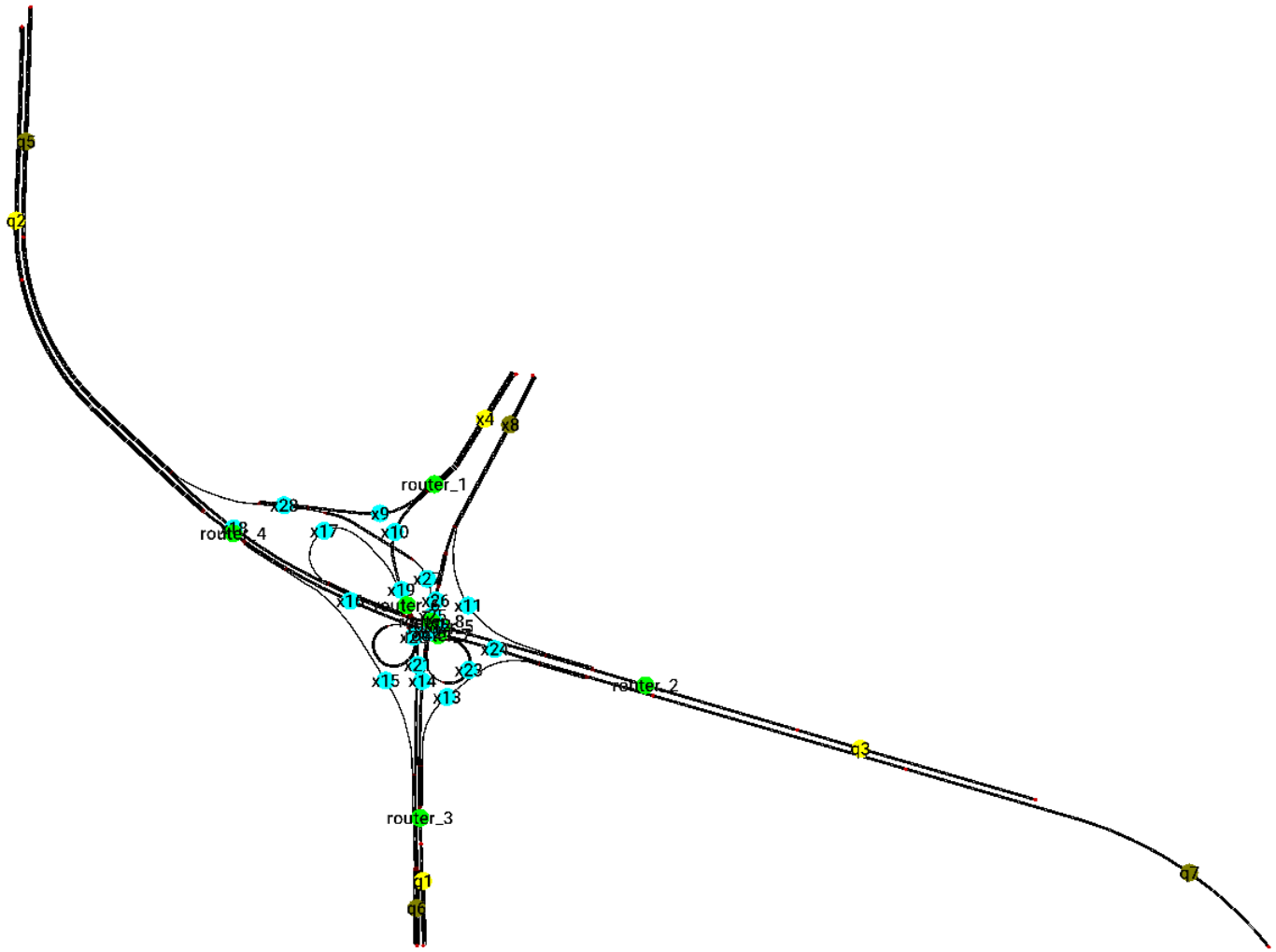


Figure A.16: VCI's *Nó de Coimbrões* node network and assigned variables

$$
\begin{aligned}
x_{10} + x_{17} - x_{19} &= 0 \\
x_{11} + x_{12} &= q_3 \\
x_{11} + x_{26} - x_8 &= 0 \\
x_{13} + x_{14} &= q_1 \\
x_{13} + x_{24} &= q_7 \\
x_{14} + x_{23} - x_{25} &= 0 \\
x_{15} + x_{16} &= q_2 \\
x_{15} + x_{21} &= q_6 \\
x_{16} + x_{20} - x_{22} &= 0 \\
x_{17} + x_{18} - x_{12} &= 0 \\
x_{20} + x_{21} - x_{19} &= 0 \\
x_{23} + x_{24} - x_{22} &= 0 \\
x_{26} + x_{27} - x_{25} &= 0 \\
x_9 + x_{10} - x_4 &= 0 \\
x_9 + x_{18} + x_{27} &= q_5
\end{aligned}
\tag{A.16}
$$

A.17 *Nó do Continente* (16 equations)

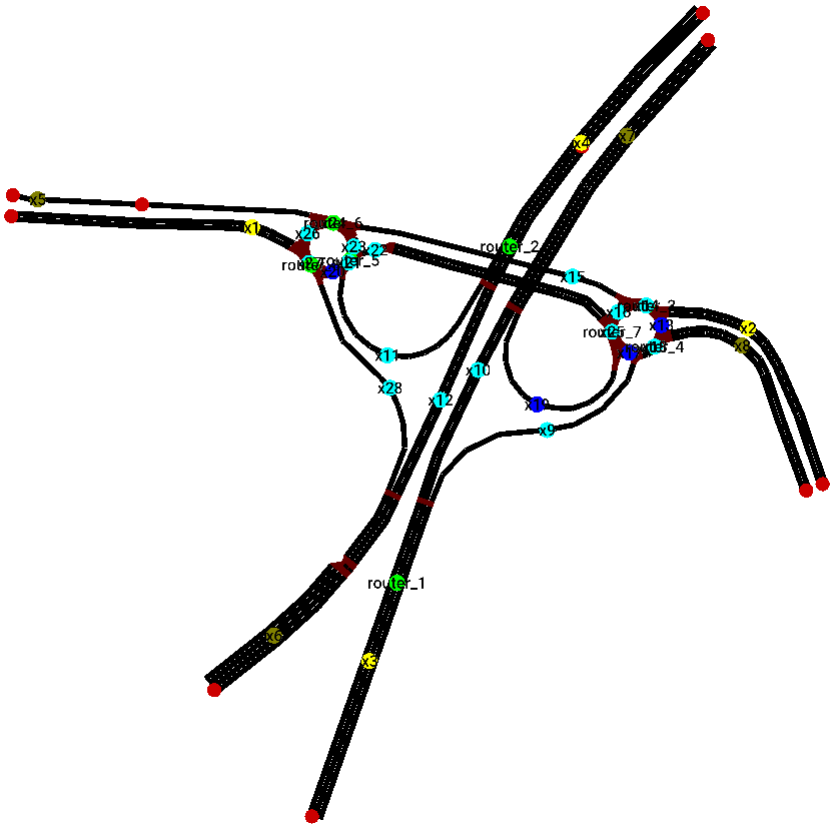


Figure A.17: VCI's *Nó do Continente* node network and assigned variables

$$
\begin{aligned}
x_1 + x_{26} - x_{27} &= 0 \\
x_{10} + x_{19} - x_7 &= 0 \\
x_{11} + x_{12} - x_4 &= 0 \\
x_{11} + x_{20} - x_{21} &= 0 \\
x_{12} + x_{28} - x_6 &= 0 \\
x_{15} + x_{16} - x_{14} &= 0 \\
x_{15} + x_{23} - x_{24} &= 0 \\
x_{16} + x_{22} - x_{25} &= 0 \\
x_{17} + x_{19} - x_{25} &= 0 \\
x_2 + x_{13} - x_{14} &= 0 \\
x_{20} + x_{28} - x_{27} &= 0 \\
x_{22} + x_{23} - x_{21} &= 0 \\
x_5 + x_{26} - x_{24} &= 0 \\
x_8 + x_{13} - x_{18} &= 0 \\
x_9 + x_{10} - x_3 &= 0 \\
x_9 + x_{17} - x_{18} &= 0
\end{aligned}
\tag{A.17}
$$

A.18 *Nó da Barrosa* (18 equations)

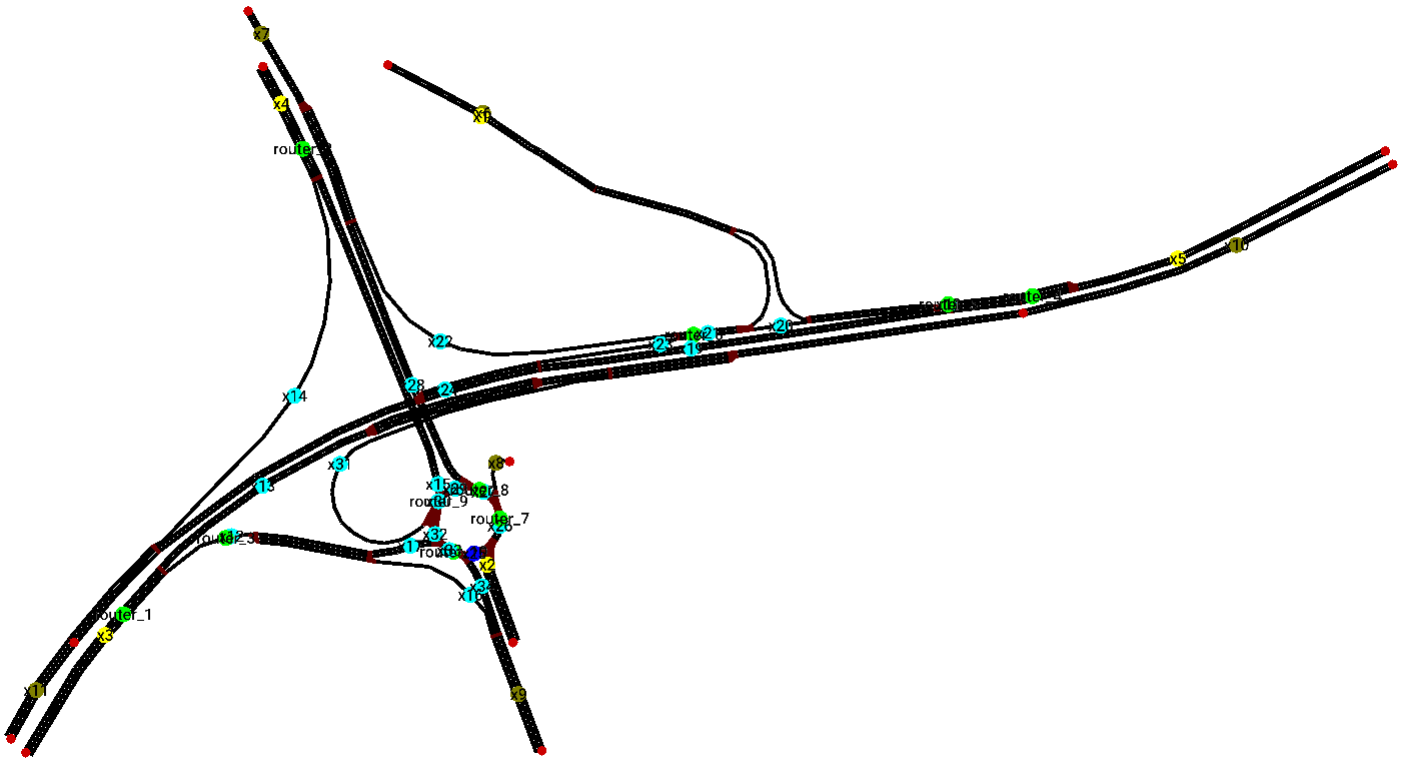


Figure A.18: VCI's *Nó da Barrosa* node network and assigned variables

$$
\begin{aligned}
x_1 + x_{20} - x_{21} &= 0 \\
x_{12} + x_{13} - x_3 &= 0 \\
x_{13} + x_{31} - x_{10} &= 0 \\
x_{14} + x_{15} - x_4 &= 0 \\
x_{14} + x_{19} + x_{23} - x_{11} &= 0 \\
x_{15} + x_{29} - x_{30} &= 0 \\
x_{16} + x_{17} - x_{12} &= 0 \\
x_{16} + x_{34} - x_9 &= 0 \\
x_{17} + x_{32} - x_{33} &= 0 \\
x_{18} + x_{19} - x_5 &= 0 \\
x_2 + x_{25} - x_{26} &= 0 \\
x_{22} + x_{23} - x_{21} &= 0 \\
x_{22} + x_{28} - x_7 &= 0 \\
x_{25} + x_{34} - x_{33} &= 0 \\
x_{28} + x_{29} - x_{27} &= 0 \\
x_{31} + x_{32} - x_{30} &= 0 \\
x_6 + x_{20} - x_{18} &= 0 \\
x_8 + x_{27} - x_{26} &= 0
\end{aligned}
\tag{A.18}
$$

A.19 *Nó da Rotunda do Atlântico* (21 equations)

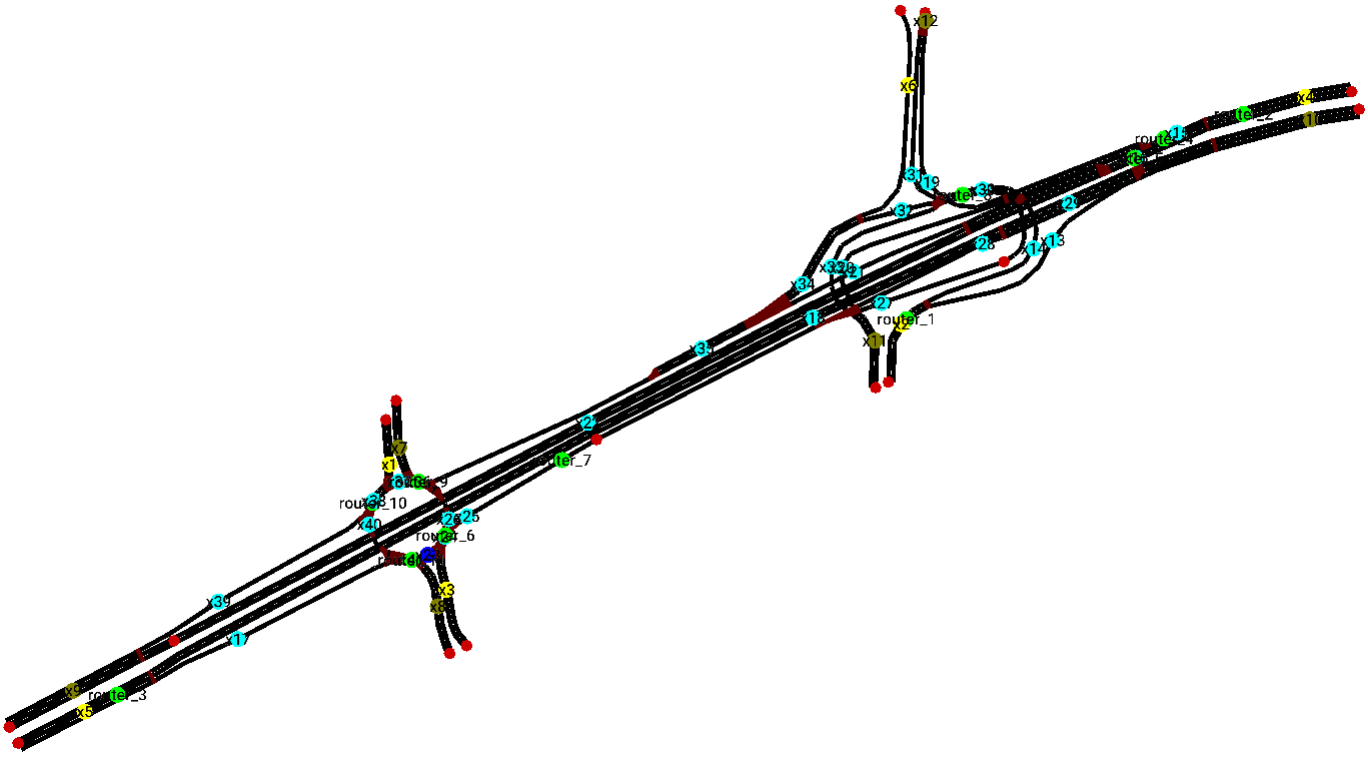


Figure A.19: VCI's *Nó da Rotunda do Atlântico* node network and assigned variables

$$
\begin{aligned}
x_1 + x_{37} - x_{38} &= 0 \\
x_{13} + x_{14} - x_2 &= 0 \\
x_{13} + x_{18} + x_{28} - x_{10} &= 0 \\
x_{14} + x_{27} - x_{30} &= 0 \\
x_{15} + x_{16} - x_4 &= 0 \\
x_{17} + x_{18} - x_5 &= 0 \\
x_{17} + x_{40} - x_{41} &= 0 \\
x_{19} + x_{20} - x_{15} &= 0 \\
x_{19} + x_{31} - x_{12} &= 0 \\
x_{20} + x_{33} - x_{11} &= 0 \\
x_{21} + x_{22} - x_{16} &= 0 \\
x_{21} + x_{26} + x_{34} - x_{36} &= 0 \\
x_{22} + x_{39} - x_9 &= 0 \\
x_{25} + x_{26} - x_{24} &= 0 \\
x_{27} + x_{28} - x_{25} &= 0 \\
x_3 + x_{23} - x_{24} &= 0 \\
x_{31} + x_{32} + x_{33} - x_{30} &= 0 \\
x_{39} + x_{40} - x_{38} &= 0 \\
x_6 + x_{21} + x_{32} - x_{35} &= 0 \\
x_7 + x_{37} - x_{36} &= 0 \\
x_8 + x_{23} - x_{41} &= 0
\end{aligned}
\tag{A.19}
$$

A.20 *Nó de Gervide* (16 equations)

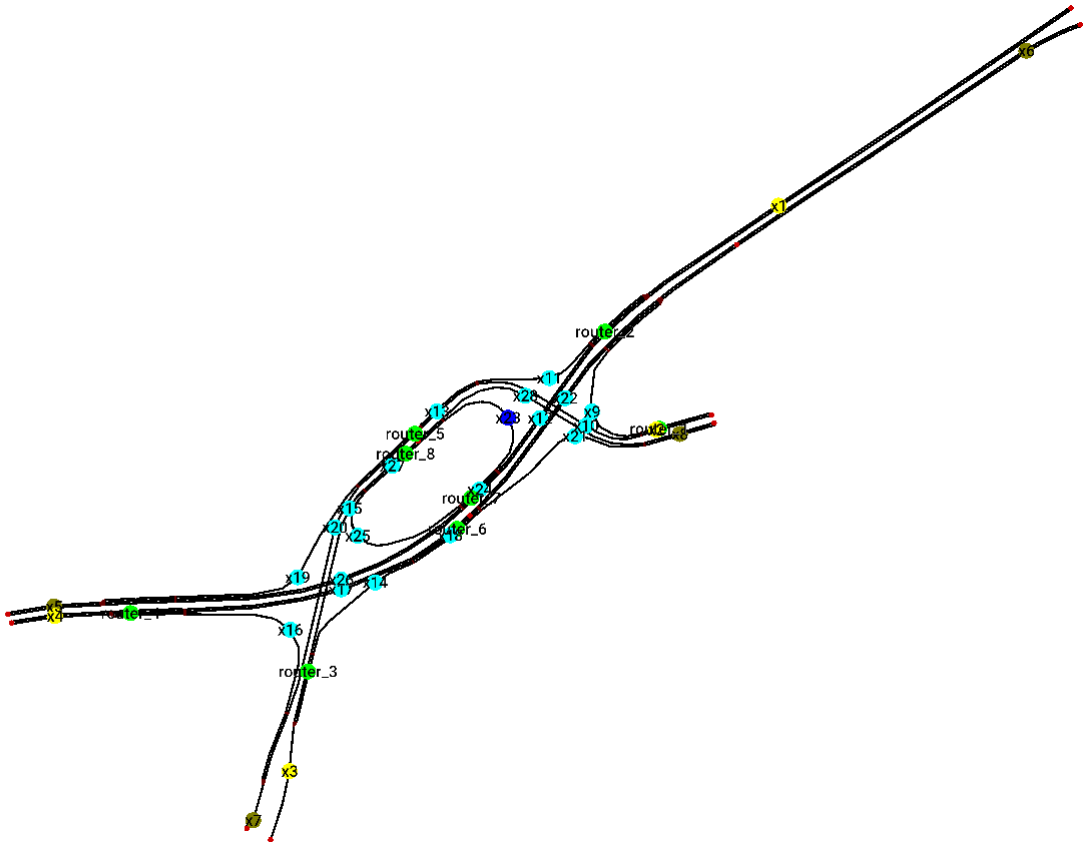


Figure A.20: VCI's *Nó de Gervide* node network and assigned variables

$$
\begin{aligned}
x_{10} + x_{11} - x_{13} &= 0 \\
x_{11} + x_{12} - x_{1} &= 0 \\
x_{12} + x_{23} - x_{24} &= 0 \\
x_{14} + x_{15} - x_{3} &= 0 \\
x_{14} + x_{17} - x_{18} &= 0 \\
x_{15} + x_{25} - x_{27} &= 0 \\
x_{16} + x_{17} - x_{4} &= 0 \\
x_{16} + x_{20} - x_{7} &= 0 \\
x_{19} + x_{20} - x_{13} &= 0 \\
x_{19} + x_{26} - x_{5} &= 0 \\
x_{21} + x_{22} - x_{18} &= 0 \\
x_{21} + x_{28} - x_{8} &= 0 \\
x_{23} + x_{28} - x_{27} &= 0 \\
x_{25} + x_{26} - x_{24} &= 0 \\
x_{9} + x_{10} - x_{2} &= 0 \\
x_{9} + x_{22} - x_{6} &= 0
\end{aligned}
\tag{A.20}
$$