



SUMO2014 - Modeling Mobility with Open Data

Proceedings

Berichte aus dem DLR-Institut
für Verkehrssystemtechnik

Band 24



DLR

Deutsches Zentrum
für Luft- und Raumfahrt

Reports of the DLR-Institute of Transportation Systems

Volume 24

Proceedings of the SUMO2014 Modeling Mobility with Open Data

May 15 + 16, 2014
DLR, Berlin - Adlershof

Publisher:

Deutsches Zentrum für Luft- und Raumfahrt e.V.
Institut für Verkehrssystemtechnik
Rutherfordstraße 2, 12489 Berlin-Adlershof

ISSN 1866-721X

DLR-TS 1.24

Berlin, May 2014

Institute Director:
Prof. Dr.-Ing. Karsten Lemmer

Preface

Dear reader,

You are holding in your hands a volume of the series „Reports of the DLR-Institute of Transportation Systems“. We are publishing in this series fascinating, scientific topics from the Institute of Transportation Systems of the German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt e.V. - DLR) and from his environment. We are providing libraries with a part of the circulation. Outstanding scientific contributions and dissertations are here published as well as projects reports and proceedings of conferences in our house with different contributors from science, economy and politics.

With this series we are pursuing the objective to enable a broad access to scientific works and results. We are using the series as well as to promote practically young researchers by the publication of the dissertation of our staff and external doctoral candidates, too. Publications are important milestones on the academic career path. With the series „Reports of the DLR-Institute of Transportation Systems / Berichte aus dem DLR-Institut für Verkehrssystemtechnik“ we are widening the spectrum of possible publications with a bulding block. Beyond that we understand the communication of our scientific fields of research as a contribution to the national and international research landscape in the fiels of automotive, railway systems and traffic management.

This volume contains the proceedings of the SUMO2014 – Modeling Mobility with Open Data, which was held from 15th to 16th May 2014 in Berlin-Adlershof, Germany. SUMO is a well established microscopic traffic simulation suite which has been available since 2002 and provides a wide range of traffic planning and simulation tools. The conference proceedings give a good overview of the applicability and usefulness of simulation tools like SUMO ranging from new methods in traffic control and vehicular communication to the simulation of complete cities. Another aspect of the tool suite, its universal extensibility due to the availability of the source code, is reflected in contributions covering parallelization and interfacing improvements to govern microscopic traffic simulation results.

The major topic of this second edition of the SUMO conference is open data. Several articles cover the acquisition and refinement of traffic networks as one of the fundamental data sources. Subsequent specialized issues such as data models for emissions and Bluetooth simulation are targeted as well. The conference’s aim was bringing together the large international user community and exchanging experience in using SUMO, while presenting results or solutions obtained using the software or modeling mobility with open data. Let you inspire to try your next project with the SUMO suite. There are many new applications in your environment.

Prof. Dr.-Ing. Karsten Lemmer

**All contributions have been double refereed as abstract and full paper
by the International Scientific Committee**

International Scientific Committee

Michael Behrisch (German Aerospace Center, Germany)

Laura Bieker (German Aerospace Center, Germany)

Robbin Blokpoel (Imtech Traffic & Infra, Netherlands)

David Eckhoff (Universität Erlangen, Germany)

Jakob Erdmann (German Aerospace Center, Germany)

Jérôme Härrı (EUROCOM, France)

Daniel Krajzewicz (German Aerospace Center, Germany)

Mario Krumnow (Technische Universität Dresden, Germany)

Christoph Sommer (Universität Paderborn, Germany)

Andreas Schadschneider (Universität zu Köln, Germany)

Peter Wagner (German Aerospace Center, Germany)

Organisation Committee

Michael Behrisch (German Aerospace Center, Germany)

Melanie Knocke (German Aerospace Center, Germany)

Table of Contents

1 A situational awareness approach to intelligent vehicle agents.....	1
<i>Vincent Baines, Julian Padget, Bath University, UK</i>	
2 Traffic simulation for all: a real world traffic scenario from the city of Bologna	19
<i>Laura Bieker, Daniel Krajzewicz, German Aerospace Center , Germany; AntonioPio Morra, Carlo Michelacc and Fabio Cartolano, Comune di Bologna, Italy</i>	
3 Interface between proprietary Controllers and SUMO	27
<i>Robbin Blokpoel, Imtech Traffic & Infra, The Netherlands</i>	
4 Network Conversion for SUMO Integration	35
<i>Robbin Blokpoel, Imtech Traffic & Infra, The Netherlands</i>	
5 Can Road Traffic Volume Information Improve Partitioning for Distributed SUMO?	45
<i>Ulrich Dangel, Quentin Bragard, Anthony Ventresque, Liam Murphy, Lero@UCD, School of Computer Science and Informatics, University College Dublin, Ireland; Patrick McDonagh, Lero@DCU, School of Electronic Engineering, Dublin City University, Ireland</i>	
6 Models enabling Simulations of V2X Applications regarding Emergency Vehicles in Urban Environment	55
<i>Michael Düring, Mark Gonter, Falco Thiel, and Florian Weinert, Volkswagen AG, Germany</i>	
7 Lane-Changing Model in SUMO	77
<i>Jakob Erdmann, German Aerospace Center , Germany</i>	
8 Second Generation of Pollutant Emission Models for SUMO	89
<i>Daniel Krajzewicz, Michael Behrisch, Peter Wagner, German Aerospace Center, Germany; Stefan Hausberger, 3130 Institut für Verbrennungskraftmaschinen und Thermodynamik, Austria; Mario Krumnow, Technische Universität Dresden, Institut für Verkehrstelematik, Germany</i>	
9 Modelling Bluetooth Inquiry for SUMO	103
<i>Michael Behrisch, Gaby Gurczik, German Aerospace Center, Germany</i>	
10 Stochastic Optimization of Signal Plan and Coordination using Parallelized Traffic Simulation.....	115
<i>Junchen Jin, Xiaoliang Ma, ITS Lab, Traffic and Logistics, KTH Royal Institute of Technology, Sweden</i>	
11 DFROUTER – Route estimate method based on detector data	127
<i>TeRon V. Nguyen, Matthew Fullerton, Institute of Transportation, Technische Universität München, Germany; Daniel Krajzewicz, German Aerospace Center , Germany; Son T. Mai, University of Munich, Germany</i>	

12 TraCI4MATlab: Re-engineering the Python implementation of the TraCI interface	145
<i>Andrés F. Acosta Gil, Jairo Espinosa, Facultad de Minas, Universidad Nacional de Colombia, Colombia; Jorge E. Espinosa, Politécnico Colombiano Jaime Isaza Cadavid, Colombia</i>	
13 TOMS – Traffic Online Monitoring System for ITS Austria Wests .	157
<i>Karl-Heinz Kastner, Petru Pau, RISC Software GmbH, Austria</i>	
14 A Railway Simulation Landscape Creation Tool Chain Considering OpenStreetMap Geo Data	167
<i>Christian Rahmig, Andreas Richter; German Aerospace Center , Germany</i>	
15 Advanced Traffic Light Information in OpenStreetMap for Traffic Simulations.....	177
<i>David Rieck, Fraunhofer FOKUS, Berlin, Automotive Services and Communication Technologies, Germany; Björn Schünemann, Ilja Radosch, Technische Universität Berlin, OKS/ Daimler Center for Automotive IT Innovations, Germany</i>	
16 Towards a Mobile-based ADAS Simulation Framework.....	185
<i>João S. V. Gonçalves, Rosaldo J. F. Rossetti, Departamento de Engenharia Informática, Faculdade de Engenharia da Universidade do Porto, Laboratório de Inteligência Artificial e Ciência de Computadores; João Jacob, António Coelho, Rui Rodrigues, Departamento de Engenharia Informática, Faculdade de Engenharia da Universidade do Porto, INESC TEC, Portugal</i>	
17 Map matching and cycling infrastructure analyses with SUMO and python	195
<i>Joerg Schweizer, Federico Rupi, University of Bologna, DICAM, Italy</i>	
18 Authors Index	207

1 A situational awareness approach to intelligent vehicle agents

Vincent Baines, Julian Padget

Bath University, UK

{V.F.Baines, J.A.Padget}@bath.ac.uk

1.1 Abstract

As an increasing number of technological developments are made in the field of autonomous vehicles, the question of what intelligent system(s) will be placed around these vehicles both for the pursuit of individual goal and conformance to regulations as part of a wider collective of vehicles becomes pertinent, especially in the context of a mixed environment of autonomous and human controlled vehicles. The requirement to conform both to the law and with social conventions, in unpredictable circumstances, poses the problem of how to encode such knowledge.

This paper adopts a Situational Awareness approach to agent knowledge, from low level perceptions, through to high level projection of future events, and explores a number of traffic scenarios where agents adopt different plans based on expected future states. A variant on such reactions is also presented, where the use of institutional governance frameworks is adopted to enforce certain behaviour, offering a 'late binding' mechanism for socially complex situations.

Keywords: multiagent systems, autonomous vehicles.

1.2 Introduction

This instruction file for Word users may be used as a template. Kindly send the final and checked Word and PDF files of your paper to us. You should make sure that the Word and the PDF files are identical and correct and that only one version of your paper is sent. It is not possible to update files at a later stage. Please note that we do not need the printed paper.

Developments in the field of autonomous vehicles are already visible on the roads of the world and likely to increase in both quantity and importance with time. Having been demonstrated operating individually – vehicles such as Google's [21] and more recently Nissan's [13] – as well as collectively in convoys [5], the question is raised of how can groups of intelligent vehicles act together in order to achieve: (i) their own goals, (ii) those of the larger collective, and (iii) those of society as a whole?

We start from the assumption that some communication between vehicles is a necessity (and an inevitability) to facilitate coordination, an assumption supported by a recent announcement from the US National Highway Traffic Safety Administration (NHTSA) [22] that Vehicle to Vehicle (V2V) communication devices may become mandatory in a year. With such technology set to enable V2V communication, there follows the consideration of how much information needs to be exchanged in order to manage cooperation and coordination

between vehicles. We consider this issue in the context of Endsley's [11] Situational Awareness work, that is, "the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future". This provides a framework in which to consider knowledge exchange between system components, that is, 'low level' perception data (e.g. a vehicle's x,y position) through to 'high level' projection considerations (e.g. given speed and orientation, there will be a collision with that detected vehicle).

Given such an environment, it becomes possible to explore what levels of data (quantitative, qualitative) and communication (high frequency, low frequency) are effective in resolving complex interactions between vehicles. We can also take account of social conventions (e.g. in a given context, what does a flash of headlights indicate) as well as regulation (e.g. at red traffic lights with an emergency vehicle approaching, what action to select).

To investigate these questions, we have built a distributed framework, connecting intelligent agents (implemented using the Jason [6] platform), a rich simulation environment (SUMO [17]), data analysis tools, plus system- and domain-specific visualization tools, that allows components to publish and subscribe to information as required. Through the selection of appropriately abstract message types, components are able to process and react to information regardless of whether the data originates from the real world, or a simulation. SUMO is used to provide a realistic traffic and vehicle simulation component, with an intelligent agent layer controlling representations of autonomous vehicles, in order to explore what interactions between 'vehicles of the future' and 'vehicles of the past' may look like. This is coupled with an institutional framework [10], capable of issuing obligations to these vehicles in an attempt to maximise the broader collective needs and resolve complex social situations. Finally, the simulation is based as far as possible on real world information, using Open Source Map (OSM) data to build 3D models and SUMO maps, combined with realistic traffic flows. For this aspect, data was used from the UK Highways Agency Traffic Flow Database System (TRADS [1]), where vehicle trips for a section of the M25 motorway over a 15 minute period have been extracted, and are used to build flows in SUMO.

In summary, the contributions of this work are: (i) extending the scope of vehicle control to incorporate the use of intelligent agents (ii) integrating open map data to allow geographically situated simulations and visualizations (iii) utilizing real-world vehicle data to reproduce actual initial conditions, and (iv) capturing conventions and regulations in institutional models to provide guidance to vehicle agents.

1.3 Research Background

As discussed in Section 1.2 this work attempts to adopt themes from Endsley's [11] Situational Awareness (SA) work in knowledge representation and exchange. This work considers Endsley's concepts of perception, comprehension and projection as transitions between 'low level' data at the perception phase (e.g. a vehicle's xyz location) through to 'high level' data at the projection phase (e.g. an upcoming traffic light will be red when arrived at, given current speed and current state of light).

The Belief-Desire-Intention (BDI) [7] model is used in the intelligent agents implemented in this work, allowing some analogies to be drawn between SA levels and BDI (e.g. low level beliefs

to agent perceptions, high level projections to agent plans). The Jason [6] multiagent platform is used for the agent component of the work, integrated into the 'Bath Sensor Framework' (BSF) [18] which forms the simulation backbone.

Earlier work [3] considered Hourizi's [14] findings (identifying relationships between aircraft accidents and lack of SA) as a motivation to improve shared knowledge between vehicle convoys; in essence to communicate less but understand more. This theme is developed further as any communication network will have some performance limits, and as the results presented later in Section 1.4 show, limitations have been identified with the simulation framework deployed here, that affect the ability of agents to perform their tasks. Thus, there is a need to communicate both at an appropriate level (e.g. within the SA context) and at an appropriate rate.

Effort to explore cooperative vehicle communication seems timely, with increasing progress in vehicle convoys such as the 'SAfe Road TRains for the Environment' (SARTRE) project [5] demonstrating the ability of vehicles to function as vehicle platoons. Continued announcements of increasingly sophisticated autonomous vehicles such as Google's car [21], the Volkswagen based 'MadeInGermany' [12] vehicle, Nissan's self-drive [13], etc, demonstrate the increasing maturity of real world vehicles suitable for this work. Whilst SUMO [17] currently provides the simulation of the vehicles (and allows safe experimentation), the Bath Sensor Framework enables low overhead substitution of one component for another, thus improving the relevancy of findings presented in this paper for potential real world applications.

Motivation to explore scenarios based on projection of future states is provided by news announcements [25] claiming that controlling vehicle speed based on upcoming traffic lights could reduce CO2 emissions by 15 percent. Similarly, experimentation in vehicle to traffic light communication [15] is also seeking to improve fuel consumption and reduce emission levels.

An institutional framework is adopted in order to provide a 'late binding' mechanism for agent behaviour, supporting the resolution of complex social conventions (e.g. is a flash of headlights to indicate move out of the way, or an offer to provide space to pull out) as well as an enforcement of dynamic global requirements (e.g. obey this variable speed limit). The enforcement of some requirement for the larger collective benefit contrary to an individual's gain has been demonstrated in other domains [4], while the need for multiple institutions [9] interacting (e.g. no lane change in a variable speed limit zone, but permissible to make way for an emergency vehicle) will be considered in future work.

Having established the research background around this work, the simulation framework is now presented in detail.

1.4 Simulation Framework

One of the central objectives of our work is to use so-called 'intelligent agents' in the context of large-scale agent-based simulation. Such agents have been perceived as mismatched with ABM because of the clearly higher computational requirements. Our aim here is to use a few such agents situated in an environment populated by many more conventional agents, in order to develop and evaluate behaviours that can operate effectively in typical scenarios. Consequently, we are using data obtained from the UK Highways Agency to construct such typical scenarios by generating SUMO vehicle populations that reflect real-world data collected from the M25 (a motorway that goes around London UK). This section provides a technical overview of the framework, as well as performance findings.

1.4.1 Technical overview

These informal requirements have led to the creation of a distributed environment called the 'Bath Sensor Framework' (BSF) [18], whose primary features are: (i) a bus-like communications system based on the eXtended Messaging and Presence Protocol (XMPP) [27], and (ii) a publish/subscribe interface that can be implemented for a variety of programming languages (we currently use Java, C# and Python). Similar XMPP-based approaches have been demonstrated in other distributed applications [24, 26]. A notable additional aspect of our framework is the adoption of two de-facto standard messaging formats: (i) Resource Description Framework (RDF), allowing the association of semantics with messages by reference to common ontologies, and (ii) JSON, allowing the low-overhead communication of structured data. Simulation components interact via publish-subscribe, where each component provides a 'Sensor' (output) and 'Sensor Client' (input), that connect to topic nodes in an XMPP server.

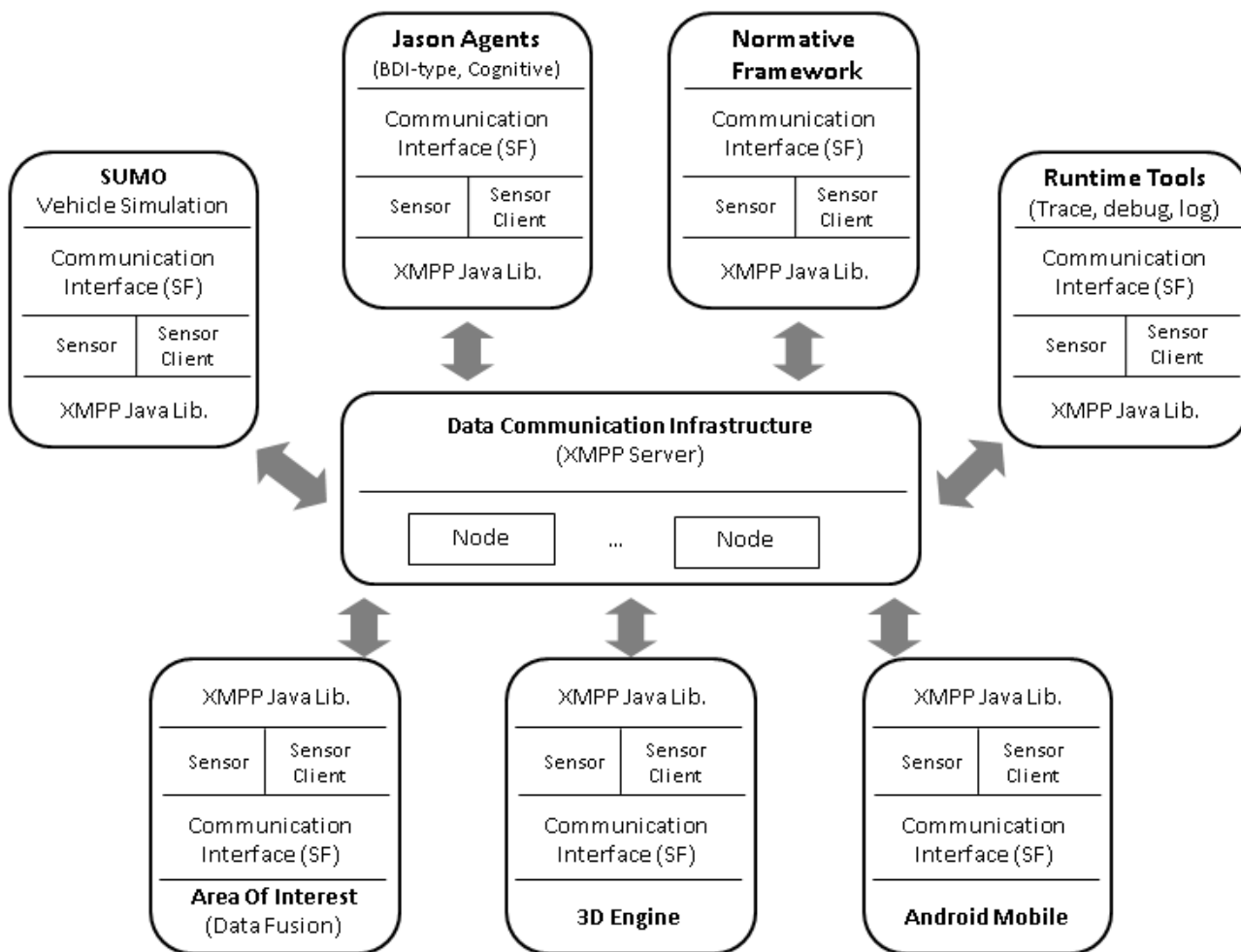


Figure 1-1: Illustration of available BSF system components

A sketch of the framework appears in Figure 1-1 populated with some of the components making up a typical instantiation of the framework as used for the work reported here:

- The SUMO interface is based on the traci4j library, allowing commands to be sent to SUMO (based on received input via BSF subscriptions) and information extracted from SUMO and published out to the BSF. This component also controls the update rate of SUMO, allowing the processing and creation of BSF messages between each simulation step.

- The Jason component provides an **intelligent agent** capability, and in the case of vehicle scenarios allows Jason agents to request the creation of a SUMO vehicle, which they then control. A similar approach has been employed to the control of non-player-characters in Second Life [19].
- The **normative framework** component introduces the element of institutions [10] into the simulation, allowing obligations to be issued to simulation participants (e.g. for a vehicle to slow down, or move out of the way), following the principles set out in [2] and developed for Jason in [20].
- The **Area of Interest** (AOI) component acts as a data fusion module relative to individual vehicles for a given 'interest volume'. This is based on the assumption that the agents controlling a vehicle have a greater interest in certain events and states near to their current location, and reduces the level of noise arising otherwise from being informed about the entire simulation state. This reports information such as upcoming traffic light states given the vehicle's current route, vehicles in the same lane which may become collision hazards, and so on.
- A **3D engine** component is used to provide a human observer with a variety of views to the simulation. As this subscribes to multiple feeds, it is able to display: (i) basic spatial information (e.g. a 3D view of the SUMO simulation, traffic light states) (ii) vehicle state information (e.g. lights, smoke if crashed), (iii) augmented with other system component information (e.g. calculated collision volumes, Jason agent belief state data), as well as system information (e.g. messages per second graph). The visualizer has proved an essential tool in debugging, as the task of understanding unintended behaviours with distributed intelligent system can be very challenging otherwise.
- Finally, there are some **runtime tools**, one of which is the RDF Monitor suite that provides analysis tools for the messages being exchanged over the BSF. This covers measures from lower level performance metrics (e.g. message delivery time, volume) to higher level simulation specific metrics (SUMO fuel consumption, mean speed). New metrics can readily be added, collected and displayed. There is also a database logger and replayer tool, allowing simulations to be recorded, analysed via SparQL queries, and replayed or stepped through as required.

All simulation components are built around the Open Source Map (OSM) data format. This has been imported into SUMO, and a corresponding 3D model built using the *osm2world* tool [16]. Some modification to *osm2world* were necessary to ensure accurate correlation between the 3D model and SUMO vehicle positions, but the two now match closely. Therefore, all tools, models, data, and code can be provided open source to the community and are available for download. Similarly, whilst the RDF message vocabulary in use has not been formally specified, there is a reasonable coherence of terms and structures used. This helps to integrate both new simulation components, as well as analysis tools which might query the RDF (Allegrograph) database directly.

This framework also allows consideration of what we consider 'impedance matching' between simulation components. Publishers of BSF data include both their own 'sensor name' as well as some data definition (e.g. metres per second, miles per hour) in the 'sensor reading', which offers subscribers some control over what data they process. For example, considering three simulation components: the Jason agent layer, the SUMO simulation, and the 3D viewer, there is a substantial difference between suitable data rates. Whilst a 3D engine could be required to run at 30 frames per second for a human observer [8], this would require SUMO to have a simulation step of 0.03 seconds to match as a 1:1 data source rate. Whilst rendering engines are well suited to such frequencies, simulation (including data

extraction and publishing) at such rates becomes challenging. Furthermore, to continue the 1:1 ratio the Jason agents would also have to operate at such frequency. It was found in earlier experimentation with vehicle convoys [3], that agents can quickly suffer from data deluge: too many queued position updates to be processed in a given time step, and so the agents start reacting to 'stale' data causing incorrect action selection. Whilst tools have been developed to identify overall BSF performance (presented in the following section), we have realized the need for a (non-functional) design requirement to take account of appropriate data rates for each system component.

1.4.2 Framework performance

Whilst the core code and design behind the BSF has been stable for a few years now, improvements on how it is deployed have lead to a steady improvement in measured performance. As an overview we can consider the key components being the publisher, the XMPP messaging server, the subscriber, and the supporting infrastructure.

Most improvements in BSF reliability have been found in improving the XMPP server, both in terms of software and hardware configuration. Issues such as poorly configured WiFi cards and lossy networks, combined with poor out of the box configuration in some XMPP server software, led to the development of some basic 'RDF Utilities' being included in the BSF. The criticality of having such reliability has led to a number of network tests being included as part of the build process (and reported back to a Jenkins¹ build server) as otherwise any simulation may appear to work but have totally unreliable results.

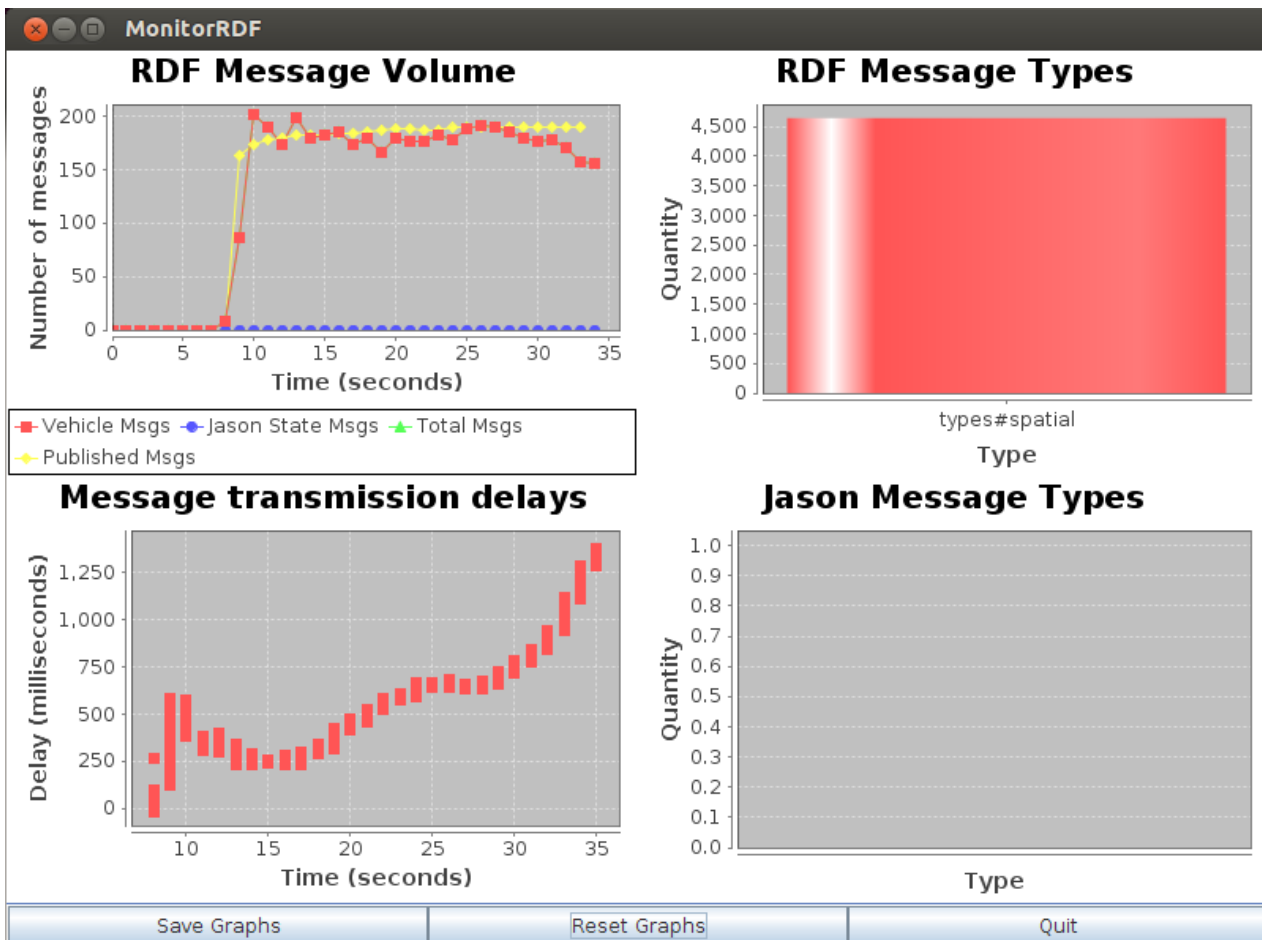


Figure 1-2: rdfMonitor Tool

¹ Jenkins Continuous Integration server, <http://jenkins-ci.org>

Two key tools are used, both are included in the 'RDF Utilities' suite. Firstly 'rdfMonitor' which subscribes to SUMO and Jason data, and displays a set of realtime graphs of performance. As BSF data readings include a timestamp encoded during message creation in the publish process, this monitor is able to plot the message delivery time, functioning essentially as a 'ping' tool for BSF messages. Supplementing this, graphs are provided for overall message volumes, quantity of messages by type, and Jason message types.

Two key tools are used, both are included in the 'RDF Utilities' suite. Firstly 'rdfMonitor' which subscribes to SUMO and Jason data, and displays a set of realtime graphs of performance. As BSF data readings include a timestamp encoded during message creation in the publish process, this monitor is able to plot the message delivery time, functioning essentially as a 'ping' tool for BSF messages. Supplementing this, graphs are provided for overall message volumes, quantity of messages by type, and Jason message types.

Figure 1-2 shows the 'rdfMonitor' GUI, and here a number of characteristics can be seen. The 'RDF Message Volume' and 'Message transmission delays' form the most interesting features in this example, highlighting that the communication network is currently saturated and messages are suffering from increased delays over time. The test tool creating these RDF messages (described shortly) also creates an RDF message with details of how many messages were created in the last time step (shown as 'Published Msgs' series in RDF Message Volume graph), in this case a reasonably steady number of just under 200 every update. By contrast, the received number of messages can be seen to be frequently below this value (shown as 'Vehicle Msgs' series in RDF Message Volume graph). If the received message count is lower than the published message count, then there are unprocessed messages awaiting, i.e. a queue is growing. This correlates with the 'Message transmission delays' graph, where as the simulation time increases, processed messages have an increasing delay, i.e. the queue of delayed messages is growing.

The component creating this test data is known as the 'rdfTest' tool, and is developed to run either in publish or subscribe mode. In publish mode, data is generated either a specified steady state rate, or published as fast as possible. In subscribe mode, output is simply the number of messages received per second. This has allowed quantified testing of improvements to BSF configuration, and to define the current 'safe' operating characteristics e.g. maximum messages per second before a backlog will be formed. It is also expected that increasing the number of subscribers will affect the performance of the system, but as the BSF configuration used in these experiments so far have involved a low number of subscriptions, this has not been specifically investigated.

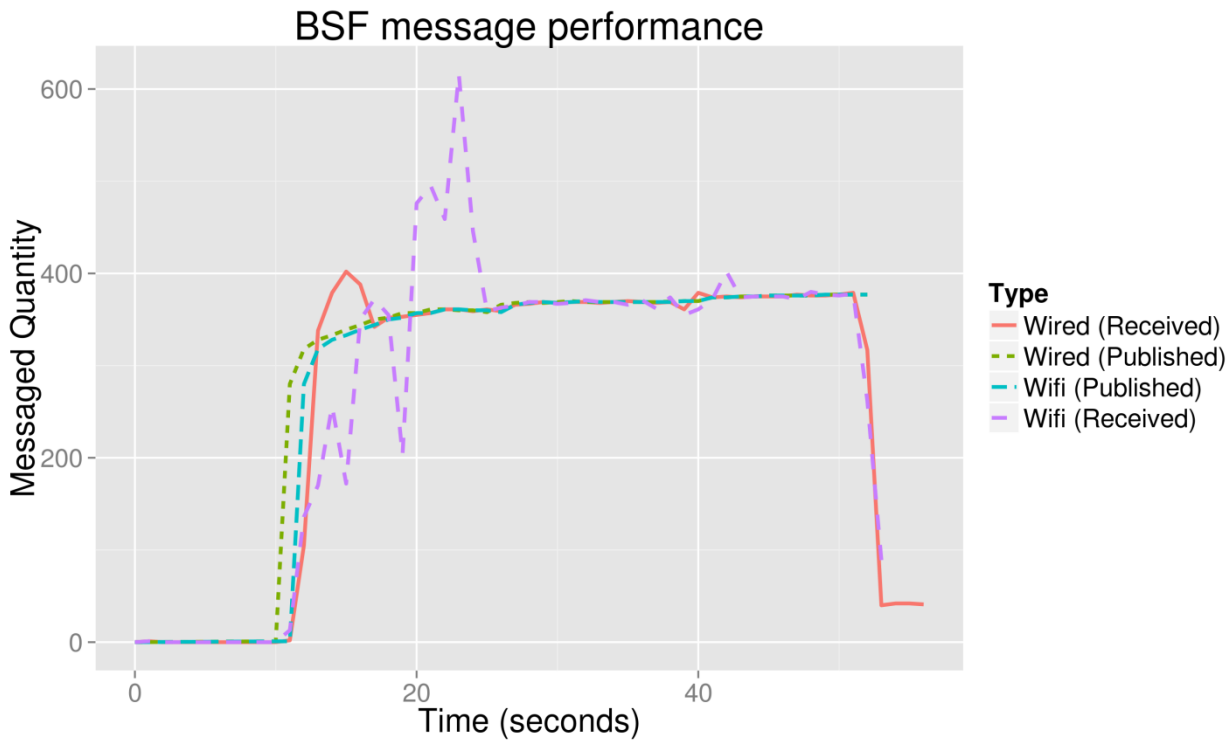


Figure 1-3: Message delivery performance

The use of these tools helps establish an operating envelope for the intended configuration. In Figure 1-3 the current optimised configuration of the BSF infrastructure in use for these experiments can be seen. This highlights that the communication volume could be approaching values where not having the luxury of wired networks (i.e. in V2V communications) is starting to have an impact. Whilst the received messages per second for the wired BSF subscriber closely match the published rate, much more variation can be seen in the BSF subscriber using a WiFi network.

Currently the largest delay in the publish and subscribe components is the time taken in serialization of the RDF messages. It is for this reason that JSON has also been explored as an alternative, and both message types are implemented and easily interchangeable. JSON offers a significantly improved serialization performance, but with a reduction in the additional vocabulary provided with the RDF messages. Examples of the variation in serialization performance are available [23] which relates to the earlier discussion of identifying the required data rate between specific system components.

Significant effort has been spent in improving the overall system performance and developing tools to assess whether the system is performing reliably in realtime, as without timely and reliable message exchange, unexpected behaviour occurs. Previous work [3] identified where running CPU intensive components (Jason and 3D Viewer) could impact the performance of both (e.g. insufficient reasoning cycles for Jason, frame rate drop off for 3D Viewer) resulting in unexpected agent behaviour. Whilst some design decisions have been taken in an effort to improve the stability of the system (e.g. BDI agents with plan failure mechanisms, SA approach to communication of higher level information rather than low level data at high frequency) there is still a time critical nature to message exchange that is necessary to maintain expected agent behaviour. However, if an assessment of network performance is made using the included BSF tools, and message volume is kept below the identified maximum value, then we find that repeated reliable simulations runs are achievable.

Having discussed the simulation framework in detail, the vehicle scenarios built upon the BSF implementation are now presented.

1.5 Experimental Scenarios

This paper presents two scenarios, using the platforms and tools described above, through which the ‘comprehension’ and ‘projection’ elements of situational awareness are explored. The institutional framework plays an essential role in each of these because it provides a form of behavioural specification of what a vehicle agent *ought* to do in a given situation. Thus, rather than loading each agent with every conceivable behaviour for every conceivable situation, it is instead able to acquire that behaviour via an instance of an institution that is created when a situation arises, while still retaining the autonomy to decide whether to follow the direction given by the institution. In this way, it becomes possible to encode different regulations and different conventions, delivering them through (multiple) institutional models, enabling both experimentation with regulations and with their combinations² 1 as well as re-use. The details of these two scenarios are now presented in more depth.

1.5.1 Scenario 1: Motorway change lane request

In this scenario, we are interested in examining the benefit institutions can have in resolving inter-vehicle requests. In the UK there are a variety of visual and audible cues used to transmit some intention or request to another vehicle. These can range from clear legal obligations (e.g. blue flashing lights of emergency vehicles create an obligation to allow that vehicle past) to the more ambiguous (e.g. a flash of headlights can indicate some hazard, or a desire to overtake). Given the improved capacity to communicate via V2V technology, this “headlight flash” request is explored in conjunction with an institution, allowing one vehicle to inform the institution of its desire to overtake, and for the institution to resolve this (by issuing an obligation to the other vehicle to change lanes).

Figure 1-4 shows this scenario in the 3D viewer, where a semi-transparent rectangular volume is projected ahead to indicate the ‘collision volume’, as computed by the Jason agent based on its current speed. Based on the distance ahead to a vehicle detected in this collision volume, Jason agents can take various actions. In this scenario, two variations are presented between a Jason agent as the leading car (V1) and a Jason agent as the following car (V2). Both variations share a similar starting set of events:

1. Vehicle V1 injected at 8 seconds into SUMO from Jason with speed of 29m/s
2. Vehicle V2 injected at 11 seconds into SUMO from Jason with speed of 30m/s
3. After 7 seconds, V2 increases speed to 32m/s
4. V2 agent belief added of `aoiVehicleDetection` with position of V1, which triggers call to `checkCollisionVolume`
5. If V1 within collision volume, then agent belief added `detectionInCollisionZone(Name, Distance)`
6. If Distance is less than 45m then agent plan `brakeHard` is triggered, if Distance greater than 45m and less than 65m then plan `flashlights` is triggered.

² Conflict between regulations is inevitable and while there are mechanisms to resolve these (not discussed here), in the first instance, the decision about which regulation to follow can be left to the vehicle agent.

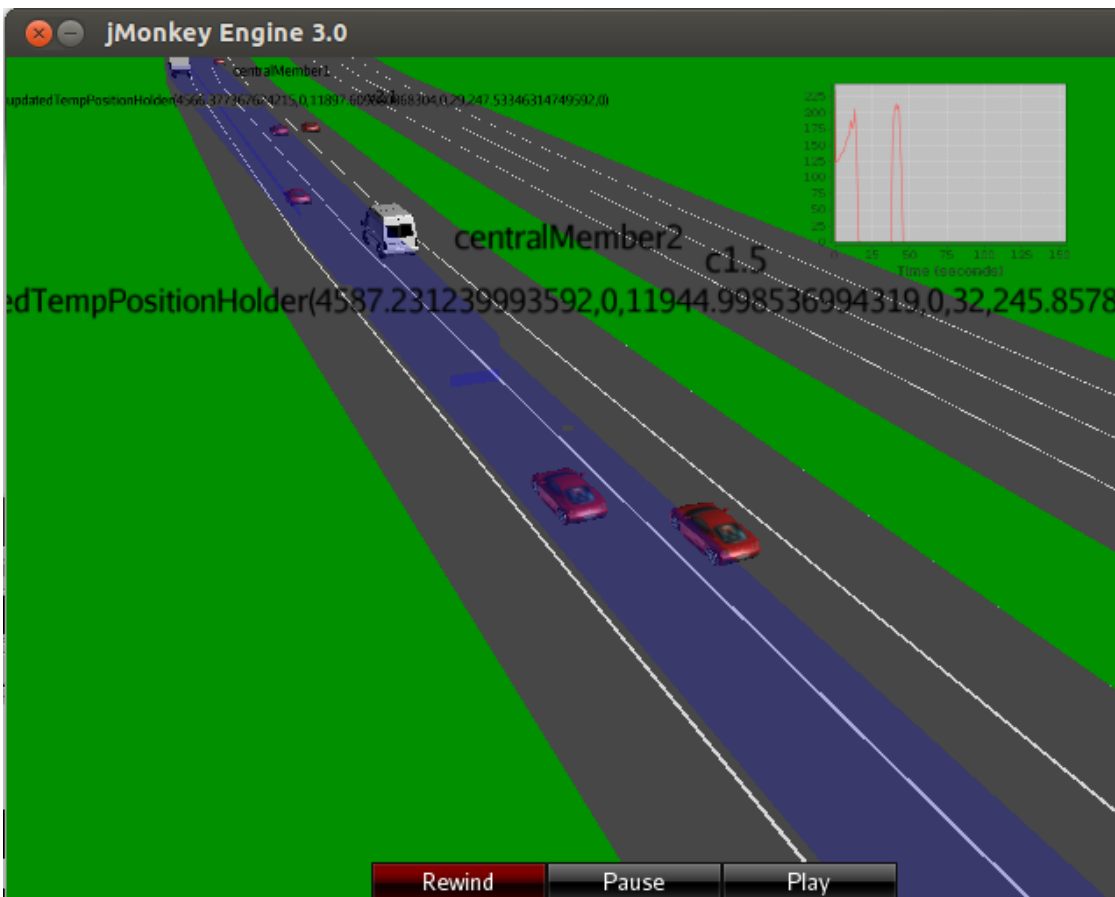


Figure 1-4: M25 Scenario in 3D viewer

Lights flash with no institution

In this case, when approaching the vehicle, V2 flashes lights at V1 but there is no response. V2 continues to gain on V1 until the distance is below 45m. The `brakeHard` plan causes the vehicle to slow to 10m/s until V1 is outside of the collision zone, after then it resumes the previous speed. This is behaviour cycles as V2 gains on V1 again, and is considered a possible extension to investigate the Variable Speed Limit further work described later.

Lights flash with institution

This repeats the same background as the previous baseline except that now the institution is active:

1. V2 publishes the event `flashLights(V1)` to the institution.
2. Institution issues permission for V1 to change lane `perm(changeLane(Agent))` and also the obligation `obl(changeLane(Agent), deadline, violation)`
3. V1 agent receives `changeLane` which triggers a `quickLaneChange` request to be sent to SUMO
4. The TraCI4j interface to SUMO implements `quickLaneChange` by changing one lane across.
5. V1 moves to inside lane, and V2 is able to overtake.

1.5.2 Scenario 2: City traffic lights

In this scenario, the capability for reasoning about future states is explored, combining the Situational Awareness concept of 'projection' with the Area Of Interest component. A city context is used, based on Bath in the UK, which generates more complex routes as well as interactions with traffic lights. It is the effect of such traffic lights interactions which is

explored in the current scenario, investigating the role institutions could play in managing vehicles' speed in order to coordinate with traffic light states.

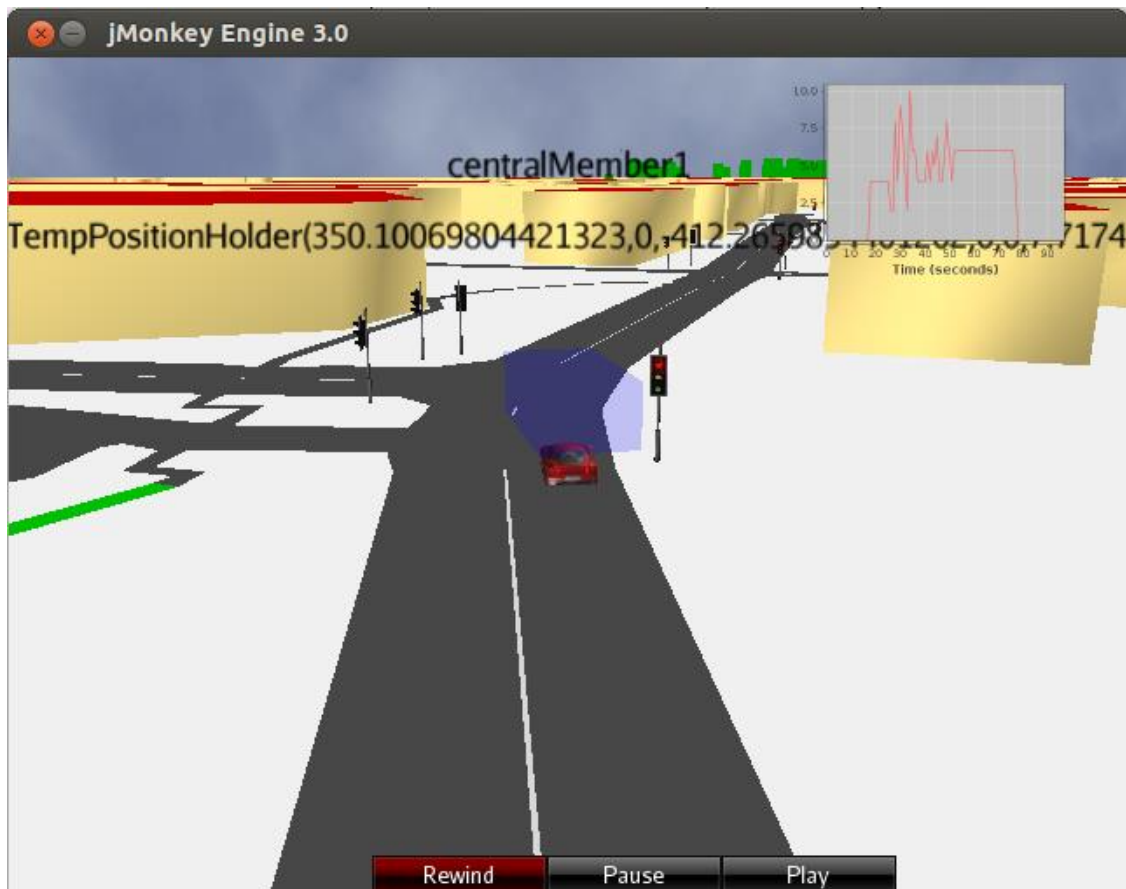


Figure 1-5: City Scenario in 3D viewer

The context of this scenario is shown in Figure 1-5 where the vehicle can be seen stationary at the first traffic light it encounters. As with the previous scenario, two variations are presented: firstly a baseline with no institution active and secondly with an institution issuing obligations to slow down depending on the distance to, and state of, upcoming traffic lights.

City journey with no institution

This case is quite simple, as no activity takes place from the institution, and the vehicle simply drives along the predetermined route.

1. Jason vehicle is inserted in SUMO simulation at 30 seconds.
2. Vehicle progresses along route, stopping at red traffic lights.

City journey with institution

In this case, the vehicle receives obligations from the institution, and so the procedure followed is slightly more complex:

1. Jason vehicle is inserted in SUMO simulation at 30 seconds.
2. Area of interest module retrieves vehicle route info, and identifies upcoming traffic lights controlling sections of that route.
3. Area of interest module publishes `upcomingLight`, `Distance`, `Colour` of detected first upcoming traffic light on vehicle's approach.
4. Institution framework reacts to `upcomingRedLight` and issues permission `reduceSpeed(Agent)` and obligation `obl(reduceSpeed(Agent), deadline, vioQueue(Agent))`

5. Agent receives `reduceSpeed` , triggering `shortCruise` plan.
6. This plan reduces vehicle speed to 7m/s for 35 seconds, after which speed control returns to SUMO.
7. Vehicle arrives at traffic lights after they have turned back to green and proceeds along route.

Having provided the detail of the two scenarios being considered in this paper, the experimental results are now presented.

1.6 Results

We now present some simulation results for the motorway and traffic lights scenarios described in the previous section.

1.6.1 Scenario 1: Motorway change lane request

In this scenario, the desire is to measure what impact the institution has in maintaining average traffic speed and preventing congestion. To do so, a metric has been developed to measure the speed of each vehicle, and the distance to the vehicle ahead. Initially the use of detector locations was considered, but this mechanism is only able to report gaps between adjacent vehicles at one location, whereas we need to establish inter-vehicle gaps in a region of the simulation. Another possibility would be to use multiple detectors at regular intervals over a multi-kilometer section of road, but given the focus on intelligent vehicles in our simulation, we chose to use a vehicle-centric rather than an infrastructure derived metric, leaving the latter for future investigation.

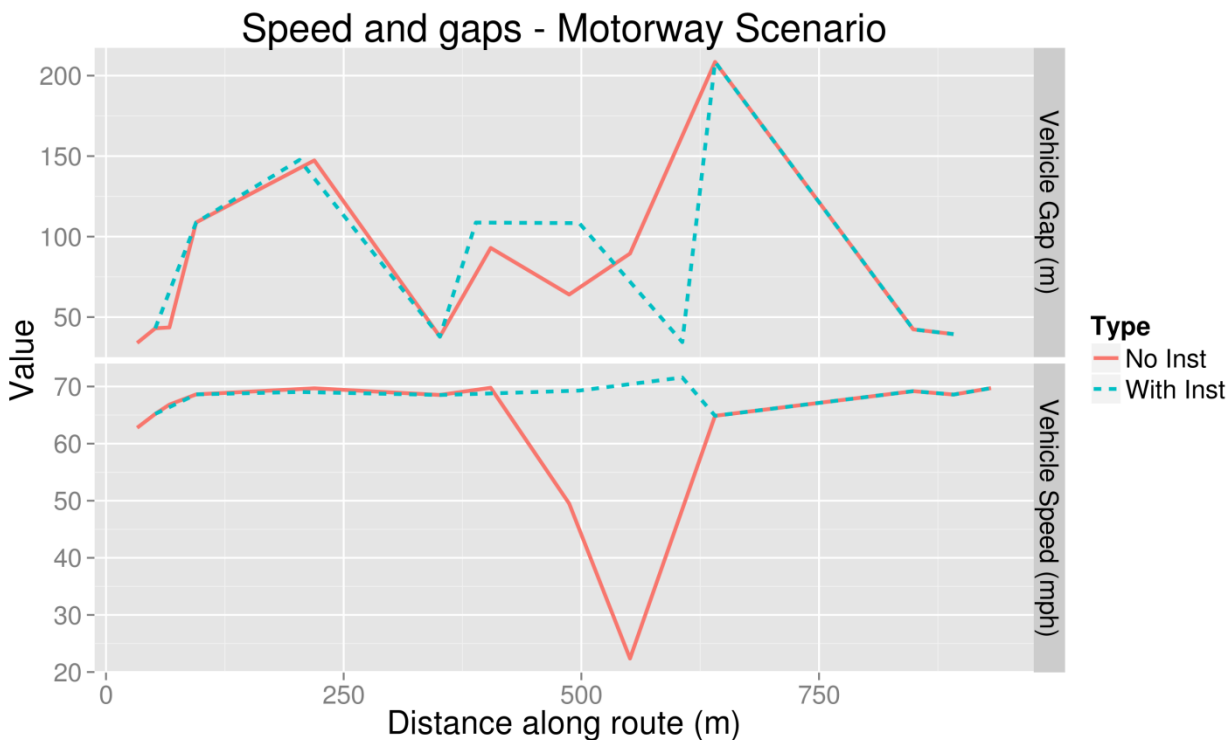


Figure 1-6: Vehicle speed and gaps

The results of this scenario can be seen in Figure 1-6, which presents both the vehicle speeds and distance to the vehicle in front, of each vehicle along the route. The most significant observation is in vehicle speeds approximately 500m along the route, where with the

institution active this remain at a reasonable constant 70mph. With the institution not active, as outlined in Section 1.5 V2 will continue to gain on V1 until it is forced to perform a hard brake to avoid a collision. The impact of this can be seen in the reduced speed both of V2, and furthermore the vehicle behind V2 has also been forced to brake to avoid colliding into V2.

The results shown in the vehicle gap section are more difficult to draw any strong conclusions from. It can be seen that either side of this disruption (i.e. ahead by 625m or behind by 375m) is largely identical for both with and without institution variations, confirming that this is a localised disturbance. In this particular scenario, the gap becomes difficult to interpret, as this concerns vehicles in a single lane, and as soon as V1 complies with the obligation to change lane, there is an immediate disruption to the reported gaps. However, with no institution active, there does seem to be some decrease in gaps. As V2 had to perform a hard brake (as shown by speed decrease in upper graph), it must have become close to V1, which would show as a decreased gap. But as V2 brakes hard, the vehicles behind will start to become closer (until they also brake) and so there is likely to be a decrease in gaps for a number of vehicles behind V2. This management of this of this kind of 'ripple effect' is considered in the future work section for possible localised variable speed limit institutions.

1.6.2 Scenario 2: City traffic lights

In this scenario, the desire is to measure effect of the institution in managing vehicle arrival times at traffic lights. The key metric being used is the measurement of fuel consumption, as the premise is that by modifying the vehicle speed such that it arrives at the traffic lights when they are green, it will result in less wasted fuel sat idling, and less fuel consumed in accelerating from stationary.

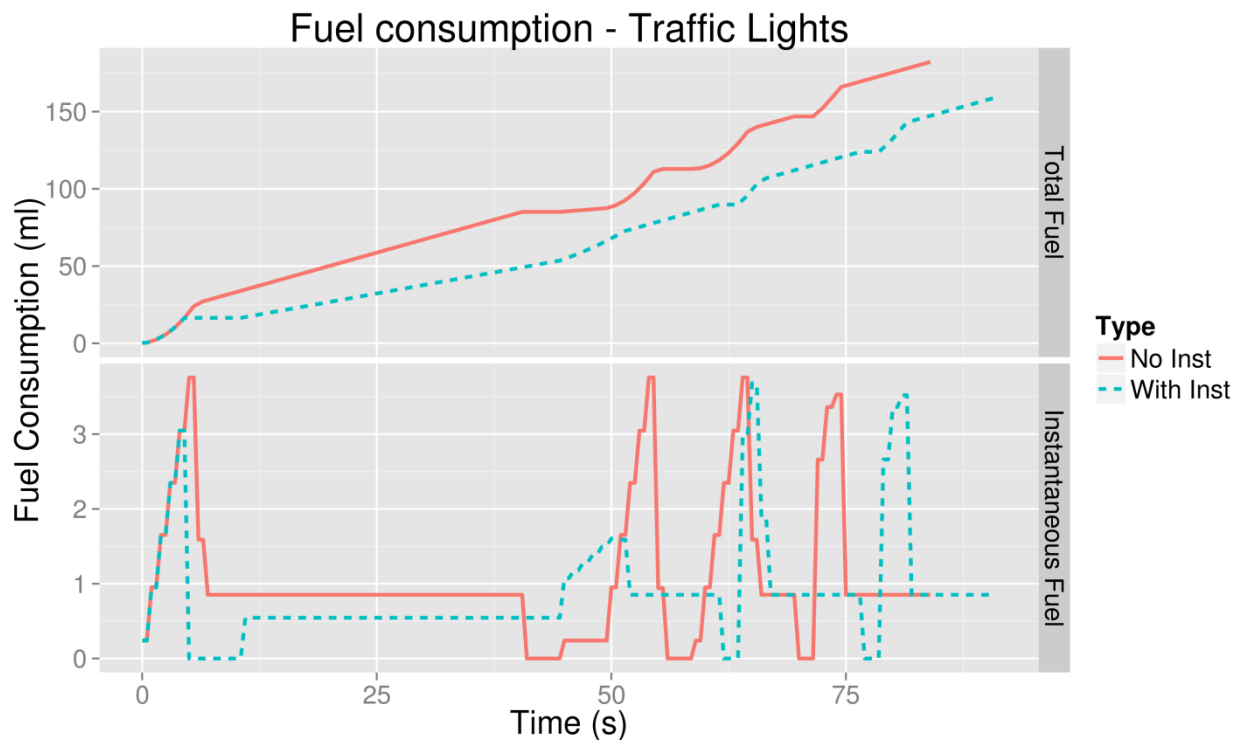


Figure 1-7: Impact on fuel consumption

In Figure 1-7 the results of two experiments are shown, showing the contrast between running the scenarios with and without institution involvement. In the case of no institution, the vehicle progresses along its route, until it is held up by a red light at a junction. This results in fuel expended while sat idling, and also in fuel required to accelerate from rest after

the light changes to green. In contrast, with the institution active, it issues an obligation for the vehicle to slow down due to the state of an upcoming traffic light. By doing so, the vehicle arrives at the light when it is green, and the graph shows this results in a fuel saving. The expectation is that there would be an increase in journey time, but in fact the vehicle only loses approximately ten seconds which can be traded off against the saved fuel. However, it can also be seen that considerable fuel saving is made simply due to the slower speed adopted by the vehicle due to the institution obligation to slow down. Further analysis is required with more variance in the scenario, as well as alternative fuel consumption models.

Whilst elements of the scenario presented may not be totally realistic at this stage (e.g. reduced speed value is very low, signal sequence may not stay red for such a length of time), the ability of SUMO to represent fuel and emission consumptions in different use cases, coupled with improved development of the institutions in use, suggests a promising avenue for exploration.

1.7 Discussion and future work

The work presented here demonstrates the use of an opensource solution combined with realistic traffic data, real world metrics and a sophisticated architecture, in modelling a number of vehicle scenarios. Two scenarios were considered: a city based investigation into the effect on an institution model in regulating arrival times at traffic lights in order to improve fuel consumption, and a motorway based investigation into the use of an institution issuing obligations to move out of the way in order to prevent excessive braking and acceleration of following vehicles.

Whilst the results presented in this work suggest promise, further work is planned to validate the metrics used and develop improvements. The gap-speed measure is still relatively new and requires development to run for multiple lanes, which will be a useful measurement when trying to identify the impact of vehicles switching lanes (e.g. in the flash lights scenario presented in this paper).

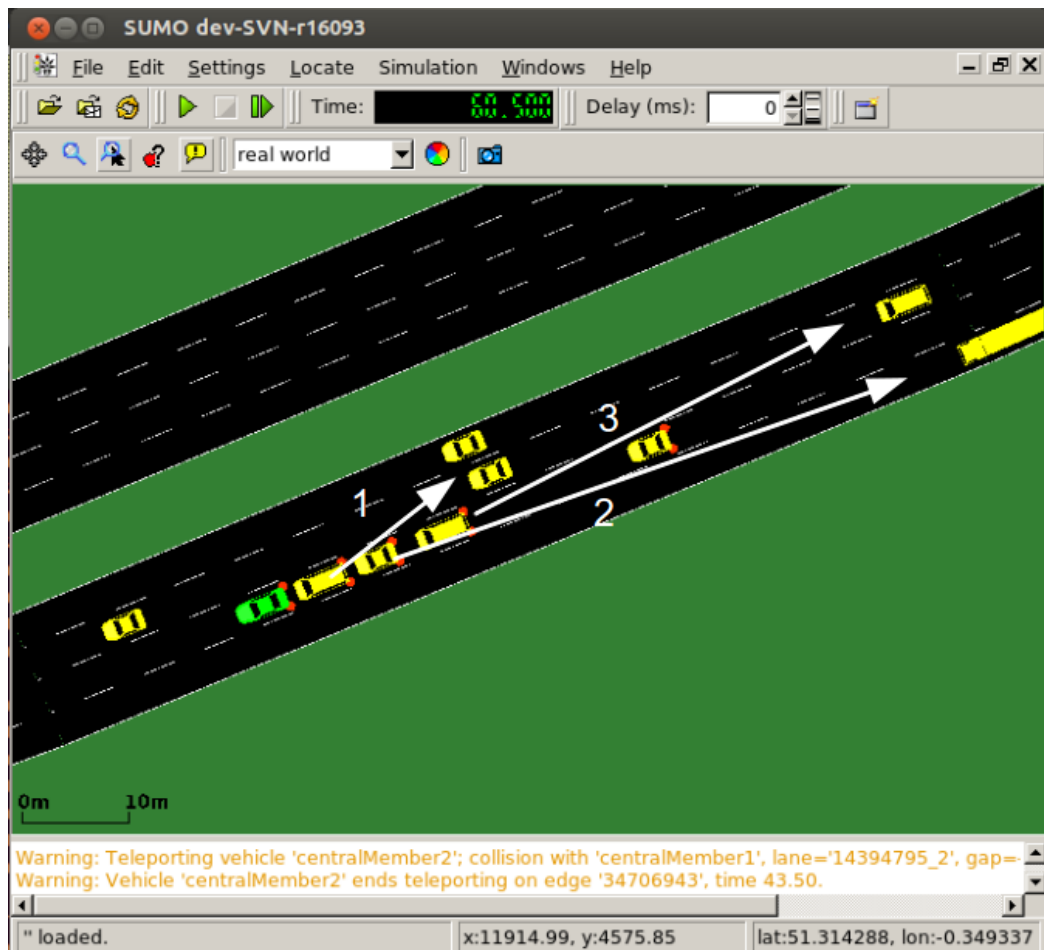


Figure 1-8: Incident management example

Having demonstrated potential benefits in the combination of the SA approach and institutional governance in the two scenarios presented, future work is planned to extend these into a larger, multiple institution scenario. One such example under consideration is shown in Figure 1-8, where a stationary vehicle is causing disruption. This particular example is an extension to the existing flash-lights scenario, where the following vehicle did not reduce its speed for some reason and collided with the vehicle ahead. There is now some post-accident traffic flow management required, with a potential dynamically instantiated institutional agreements indicated numerically. Institution number 1 allows the first vehicle to move one lane right but requires the approaching vehicle to slow down. Similarly institution number 2 allows the second queued vehicle to move one lane left, but requires the oncoming lorry to slow down, and finally institution number 3 allows the last stationary vehicle to move one lane right.

Another planned scenario involves the use of variable speed limits and how institutional governance of such an obligation could be of benefit. Currently such speed limits are implemented at a coarse granularity, affecting substantial road sections and subject to non-compliance by drivers (or at least, some percentage 'error' in speed judgement over the specified limit). The first variation of this scenario would enforce a uniform speed, and then a second variation would investigate the use of localised variable speed limits i.e. around an accident, or to reduce a congestion wave.

1.8 References

- [1] UK Highways Agency. Traffic flow database system. Accessible via <https://trads.hatris.co.uk>. accessed 26th Jan 2014.
- [2] N. Alechina, M. Dastani, and B. Logan. Programming norm-aware agents. In Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2, AAMAS '12, pages 1057–1064, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.
- [3] V. Baines and J. Padget. Communication and metrics in agent convoy organization. In 7th International Workshop on Agents in Traffic and Transportation (ATT 2012 at AAMAS 2012), pages 69–77, June 2012.
- [4] T. Balke. Towards the Governance of Open Distributed Systems: A Case Study in Wireless Mobile Grids. PhD thesis, University of Bayreuth, September 2011.
- [5] C. Bergenheim, Q. Huang, A. Benmimoun, and T. Robinson. Challenges of platooning on public motorways. In 17th World Congress on Intelligent Transport Systems, 2010.
- [6] R. H. Bordini, J. F. Hbner, and M Wooldridge. Programming multi-agent systems in AgentSpeak using Jason. Wiley, 2007.
- [7] M. E. Bratman, D. J. Israel, and M. E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355, 1988.
- [8] Kajal T. Claypool and Mark Claypool. On frame rate and player performance in first person shooter games. *Multimedia Syst.*, 13(1):3–17, 2007.
- [9] O. Cliffe, M. De Vos, and J. A. Padget. Specifying and reasoning about multiple institutions. In *Coordination, Organizations, Institutions, and Norms in Agent Systems Lecture Notes in Artificial Intelligence*. Springer, 2007.
- [10] Owen Cliffe, Marina De Vos, and Julian Padget. Answer set programming for representing and reasoning about virtual institutions. In Katsumi Inoue, Ken Satoh, and Francesca Toni, editors, CLIMA VII, volume 4371 of *Lecture Notes in Computer Science*, pages 60–79. Springer, 2006.
- [11] Mica R. Endsley. Toward a theory of situation awareness in dynamic systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1):32–64, 1995.
- [12] Freie Universitat Berlin. Autonomous car navigates the streets of Berlin. <http://autonomos.inf.fu-berlin.de/news/press-release-92011>, September 2011. accessed October 8th, 2011.
- [13] Nikki Gordon-Bloomfield. Nissan takes Japanese PM on autonomous LEAF test drive. <http://transportevolved.com/2013/11/11/nissan-takes-japanese-pm-on-autonomous-leaf-test-drive/>, November 2013. accessed January 19th, 2014.
- [14] R. Hourizi. Awareness beyond mode error. PhD thesis, University of Bath, 1999.
- [15] Kyeong Tae Kim. STVC: Secure Traffic-light to Vehicle Communication. In ICUMT, pages 96–104. IEEE, 2012.
- [16] Tobias Knerr. Merging elevation raster data and openstreetmap vectors for 3d rendering. Master's thesis, University of Passau, May 2013.
- [17] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of SUMO - Simulation of Urban MObility. *International*

- Journal On Advances in Systems and Measurements, 5(3&4):128–138, December 2012.
- [18] JeeHang Lee, Vincent Baines, and Julian Padget. Decoupling cognitive agents and virtual environments. In Frank Dignum, Cyril Brom, Koen V. Hindriks, Martin D. Beer, and Deborah Richards, editors, CAVE, volume 7764 of Lecture Notes in Computer Science, pages 17–36. Springer, 2012.
- [19] JeeHang Lee, Tingting Li, and Julian Padget. Towards polite virtual agents using social reasoning techniques. *Computer Animation and Virtual Worlds*, 24(3-4):335–343, 2013.
- [20] JeeHang Lee, Julian Padget, Brian Logan, Natasha Alechina, and Daniela Dybalova. Runtime norm compliance in BDI agents. In International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 2014. IFAAMAS, 2014. To appear.
- [21] J. Markoff. Google cars drive themselves, in traffic. <http://www.nytimes.com/2010/10/10/science/10google.html>, October 2010. accessed October 8th, 2011.
- [22] Nathan Naylor. U.s. Department of Transportation Announces Decision to Move Forward with Vehicle-to-Vehicle Communication Technology for Light Vehicles. <http://www.nhtsa.gov/About+NHTSA/Press+Releases/2014/USDOT+to+Move+Forward+with+Vehicle-to-Vehicle+Communication+Technology+for+Light+Vehicles>, February 2014.
- [23] Eishay Smith. JVM-Serializers. <https://github.com/eishay/jvm-serializers/wiki>, 2013. accessed February 2nd, 2014.
- [24] L. Stout, Michael A. Murphy, and S. Goasguen. Kestrel: an XMPP-based framework for many task computing applications. In Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers, MTAGS '09, pages 11:1–11:6, New York, NY, USA, 2009. ACM.
- [25] Volkswagen. Audi in the simTD large-scale test study: the “traffic light info online” project. http://www.volkswagenag.com/content/vwcorp/info_center/en/news/2013/06/audi_simTD.html, June 2013. accessed January 19th, 2014.
- [26] J. Wagener, O. Spjuth, E. Willighagen, and J. Wikberg. XMPP for cloud computing in bioinformatics supporting discovery and invocation of asynchronous web services. *BMC Bioinformatics*, 10(1):279, 2009.
- [27] XMPP Standards Foundation. The XMPP standards foundation homepage. Retrieved from <http://www.xmpp.org>, 20130129, no date.

2 Traffic simulation for all: a real world traffic scenario from the city of Bologna

Laura Bieker¹, Daniel Krajzewicz¹, Antonio Pio Morra², Carlo Michelacci² and Fabio Cartolano²;

*¹German Aerospace Center, Rutherfordstraße 2, 12489 Berlin, Germany;
{Laura.Bieker, Daniel.Krajzewicz}@DLR.de*

*²Comune di Bologna, Bologna, Italy
{AntonioPio.Morra, Carlo.Michelacci}@comune.bologna.it, cartolano@ibinnovation.eu*

2.1 Abstract

It is time consuming to set up a realistic traffic simulation scenario. Even though data is available, a large effort is needed to gather, convert and adapt all the data needed to replicate a part of a real road network. Available road networks usually have to be corrected and adapted to the used simulation model. The demand has to be converted or even generated using given measurements. The measurements must be imported into the simulation system's architecture to allow the models' calibration and validation. Additional road side structures must be converted into a proper representation and embedded into the scenario. Therefore, three real world traffic simulation scenarios of the city of Bologna are made available to the public. With these scenarios researchers are able to start their investigations with little preparation effort and can concentrate on their research questions.

Keywords: Real world traffic scenario, open data, validation

2.2 Traffic simulation and Open Data

For modelling real world scenarios the traffic simulation needs input traffic and infrastructure data about the real-world traffic conditions. The quality of the input data is crucial for realistic simulation results. Therefore the following input data for the simulation are needed:

1. Representation of the road network
2. Representation of the demand
3. Representation of real traffic lights
4. Representation of infrastructure

Without these representations a realistic traffic simulation is hardly possible. But collecting, processing and validating the input data is time consuming. Sometimes it is difficult to receive real world data especially the real traffic light signal plans are rarely open to the public. Consequently, it is difficult for traffic researchers to analyse and evaluate their traffic light algorithm under real world conditions. Thus, real world scenarios from Bologna are prepared and described in this work and will be made publicly available within the SUMO package [1] [2]. By making the scenarios available to the public the scenarios can be used for further research with little effort.

2.3 Bologna Scenarios

The simulation scenarios presented in this paper have been built in the project iTETRIS (“An Integrated Wireless and Traffic Platform for Real-Time Road Traffic Management Solutions”). iTETRIS was co-funded by the European Commission between 2008 and 2011 and was concerned in developing a simulation system for evaluations of large-scale traffic management solutions that work via vehicular communications [3]. A large part of the project was dedicated to determining and modelling of real-world traffic. Major contribution on this task was performed by the municipality of Bologna who was a project partner in iTETRIS. Besides describing the situation and the problems in Bologna, this municipality also delivered initial ideas for traffic management applications and additionally a large set of data and simulation scenarios.

The given data included representations of the areas around the “Andrea Costa” and the “Pasubio” roads, as input files for the commercial microscopic traffic simulation Vissim, a product of PTV AG. Each of the scenarios included the demand for Bologna’s peak hour (8:00am – 9:00am). Additional data sets supported by the municipality of Bologna included positions of traffic lights, traffic light plans, inductive loop positions and measures and many others. A further scenario, “joined”, was implemented within iTETRIS by merging both Vissim scenarios.

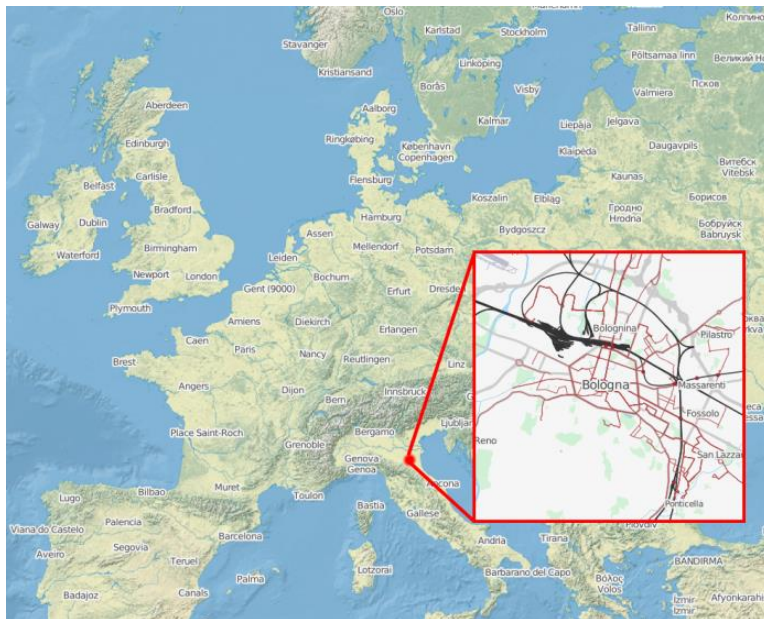


Figure 2-1: Location of Bologna

In the following the three regions of Bologna and the traffic simulation scenario are described.

2.3.1 Andrea Costa

The Andrea Costa scenario includes the area around the football stadium and was set up to simulate the mobility of big events such as football matches or concerts.

Table 2-1: Selected views on the A. Costa scenario from iTETRIS

Location in Bologna	SUMO Road Network

2.3.2 Pasubio

The Pasubio scenario extends the Andrea Costa scenario about the area around the hospital and includes also common routes to the football stadium.

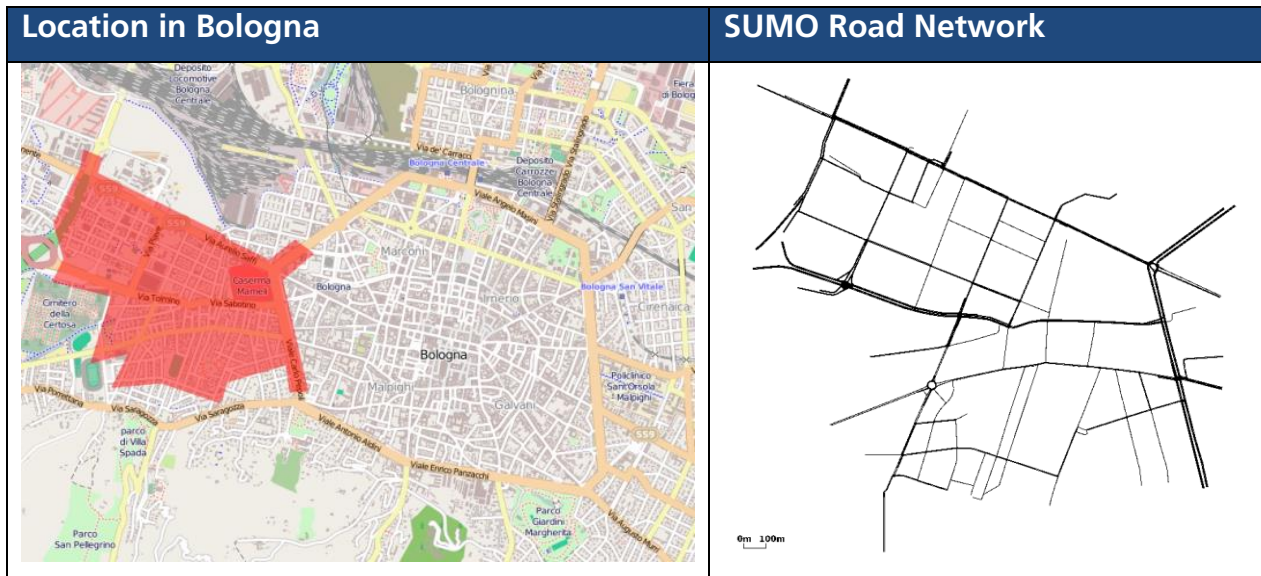
Table 2-2: Selected views on the Pasubio scenario from iTETRIS

Location in Bologna	SUMO Road Network

2.3.3 Andrea Costa and Pasubio joined scenario

Because of the relatively small areal size of the VISSIM networks “Andrea Costa” and “Pasubio”, it was decided to generate one scenario based on both. This scenario should be obtained by joining both given, as the areas they cover are overlapping. In addition to the traffic demand for the morning traffic demand, the Andrea Costa-Pasubio joined scenario also includes the traffic demand for a football match. For generating a higher traffic demand the induction loop data from 24 March 2010 when a football match took place was compared to the traffic flow on the day one week in prior. Additional routes were generated from the analysed data set.

Table 2-3: Selected views on the "joined" scenario from iTETRIS



2.4 Development of the Scenario

2.4.1 Traffic Network

The supported VISSIM scenarios model two areas located east to the inner city ring. They describe a slightly pruned road network not containing few smaller streets. Traffic lights are defined, including their positions and signal plans. Passenger vehicles in each network are described in an aggregated manner: the numbers of vehicles to insert are given for certain roads located at the network's border. Following their initial route, the vehicles pass certain "routing decision points" at which they get a new route assigned randomly, according to a given distribution. This method is used for reproducing the turn percentages at intersections measured in reality. Both scenarios describe the traffic at the morning peak hour, between 8:00 a.m. and 9:00 a.m.

In addition to the passenger vehicles, both scenarios include a description of the public bus transport taking place in the regarded area. Both, positions of the bus stops and bus routes and schedules are given.

2.4.2 Traffic lights

Definitions of traffic lights were given as telemetry files in dwg format and according signal time plans given in Excel format. One may note that the Excel sheets contain "variable phases" used by the UTOPIA system to adapt the traffic lights to the current demand on the controlled roads.

2.4.3 Traffic Demand

Two datasets containing detector measures were supplied by the municipality of Bologna. The first one contains measures from the days 11.11.2008-13.11.2008, Tuesday to Thursday. Choosing these days is conforming to the fact that Tuesday to Thursday are usually the only "common" weekdays in means of traffic. Mondays and Fridays have different traffic shapes; on Mondays, the shape is differing due to the slightly later departure of passenger traffic and

in some countries due to prohibitions of heavy delivery traffic on Sundays. On Fridays, the afternoon peak is often earlier, due to earlier end-of-business times.

The given measures were aggregated into intervals of half an hour. 636 detection sites were listed, where about 90 detectors (11.11.2008: 95, 12.11. 2008: 92, 13.11. 2008: 91) reported an error marked as the value "-1". Each of the given time line values represents the number of vehicles that passed the detection site. Because the detection is done using single induction loops, no speed information is available. Also, no distinction between different vehicle classes is given. Each detection site may cover more than one lane. The detector data is of a very good quality compare to other sites where the known number of errors in the detector data is much higher.

The second data set contains the measures for the same days, aggregated into intervals of 5min. The quality corresponds to the data set aggregated into 1800s.

The given VISSIM scenarios describe both the infrastructure and the demands within the modelled areas with a great detail and including additional information about public transport. Due to this, all information available within these inputs was imported.

Although VISSIM is a microscopic simulation just as SUMO, it follows a completely different concept of modelling the road infrastructure. The main difference is that VISSIM is not using a graph concept, consisting of nodes (intersections) and edges (roads) as SUMO does, but only of roads and connections between them. This difference makes import of VISSIM networks very complicated and the results must often be edited by hand after an initial conversion.

Because SUMO only allows importing VISSIM networks stored in German language – VISSIM uses a man-machine language for network description – the supported networks had to be translated from English into German, first. The manual validation step shown in this figure was done by comparing the network with images from Google Earth [4] and Google Maps [5], and with the supported junction telemetries. While the number of lanes was correct for all edges within the VISSIM networks, manual corrections were done on the connections between lanes over intersections.

2.5 Demand Evaluation

The city of Bologna has 636 detectors which are measuring the traffic flow. The measured values from three days are used for analysing the real traffic flow. Unfortunately, the detectors are measuring only the amount of vehicles which are passing the detectors within 5 Minutes there are no other values like speed or vehicle type available.

Generally, the measured traffic flow looks similar to the detector values which are shown in Figure 2-. There is only a small amount of vehicles driving during nights. In the morning hours the traffic flow is rising until there is a morning peak between 8 a.m. and 9 a.m. Afterwards, the traffic flow decreases a little bit and remains at a certain level. In the evening the traffic flow is rising to another peak.

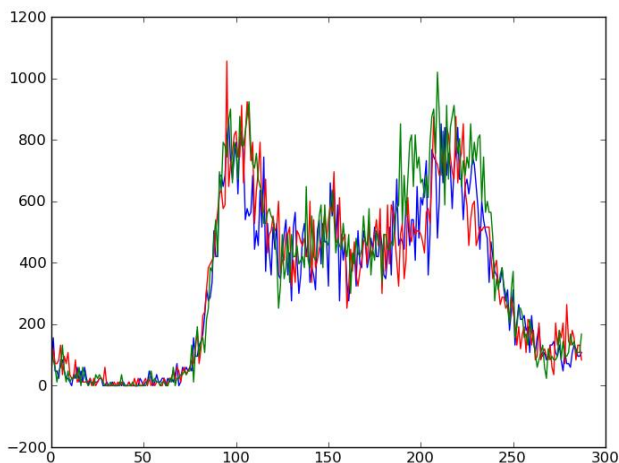
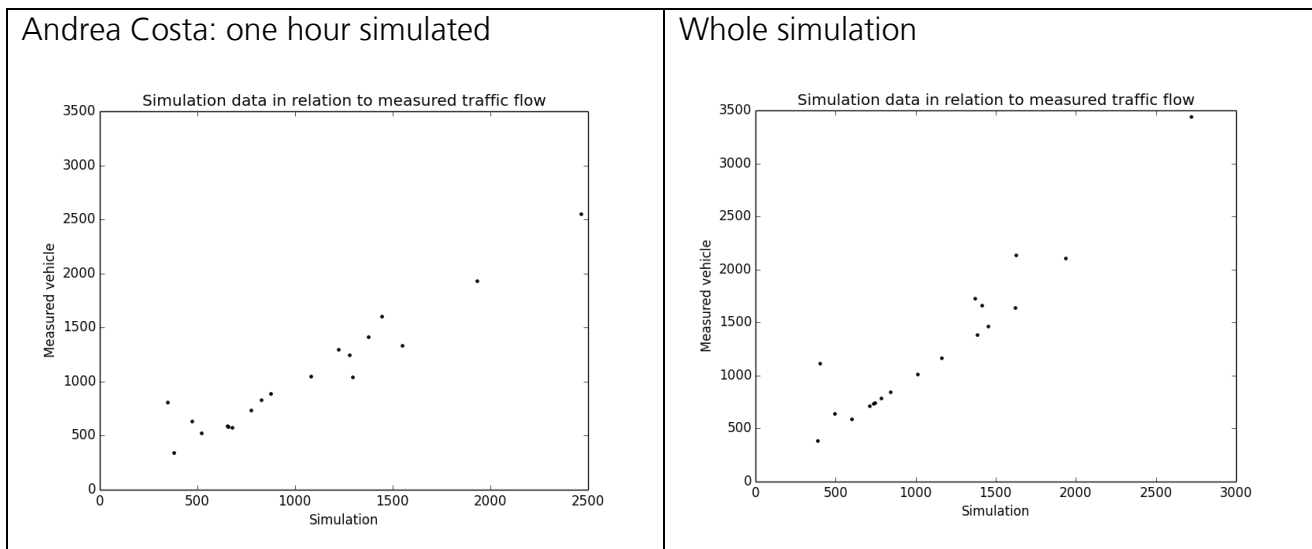


Figure 2-2: Example traffic flow of three days; blue: 11.11., red 12.11., green: 13.11.

For the validation of the simulation the real world measurements were compared to the results of the simulation. The imported networks were compared against the supplied induction loop values. Here, only the flow over the detectors were given, and were used for the evaluation. In the following, comparisons of the demands imported from VISSIM files against measures from the real world.

Table 2-4: Comparison of the simulated traffic flow compare to the measured traffic flow from the detector data. (Top to bottom: Andrea Costa, Pasubio, Joined; Left: Simulation of one hour, Right: whole simulation run).



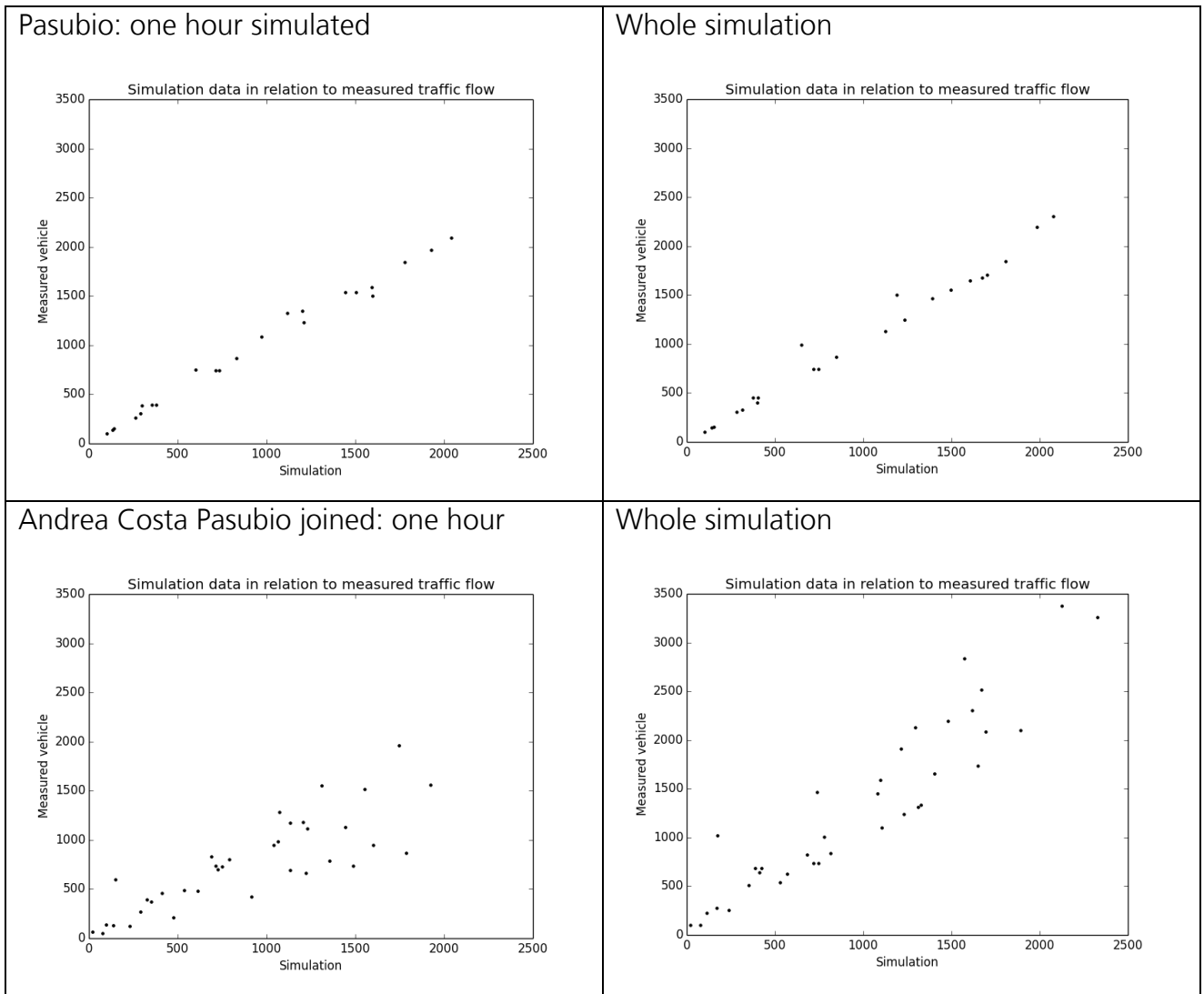


Table 2-4 shows the comparison of the average values of real measures from all detectors within the according area for the time between 8:00 and 9:00 against the ones obtained from the simulation using the demands imported from VISSIM. Two comparisons are given for each scenario: the left one shows the comparison between the real flow and the simulation results for the vehicles which passed the simulated detectors within the same time. Within the right comparison, the real world measures are put against all simulated vehicles (until the simulation ends). For both, the optimal results would be a bisectrix.

From all results the following can be seen: The overall number of vehicles which are simulated in SUMO are relatively well (right). Also the simulation results for one hour are not bad, but it can be seen that the simulation has problems to simulate the traffic demand in the joined scenario within one hour. The reason is very straight forward: with a growing areal size of the scenario, the vehicles need a longer time to populate it and cross the available induction loops. As a conclusion, it should be stated that a certain amount of simulation time is needed to fill the scenario with vehicles, before the real network state is reached. This simulation "warm-up" is a known need within traffic simulations. Normally, double of the maximum travel time through the network is used.

2.6 User guidelines

The Bologna scenarios are a good way to start traffic simulation based research with. The provided network, traffic demand and additional infrastructure data is broad and a lot of work was done on improve the simulation quality. For first simulations with real world data the scenarios can easily be used with less effort. But the use of the scenarios is also limited. For one hand side, the given traffic demand is only for one hour. Considering also a warming up and cooling down phase of the simulation, there is only approximately 30 minutes of simulation which can be used. Furthermore, the network size of the scenarios are relatively small so the route choice in the scenarios are very limeded. So rerouting algorithms should better not be simulated. Examples for traffic management strategies which can be simulated with these scenarios can be found in [6] and [7].

2.7 Further research

The Bologna scenarios provide traffic networks, traffic demands and representations about the traffic infrastructure to simulate a real world scenario in SUMO. But still there are open issues which should be improved. The multi-lane roundabouts produce unrealistic traffic jams. Pedestrians are currently not included into the simulation, this issue will be handled during the COLOMBO project [8].

On the one hand side, by making the scenario available to the public researcher can use these scenarios for their purposes and on the other hand side they can improve the quality of the scenarios by their corrections and enhancements.

2.8 References

- [1] Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.: Recent Development and Applications of SUMO - Simulation of Urban MObility. International Journal On Advances in Systems and Measurements, 5 (3&4):128-138, December 2012.
- [2] Simulation of Urban MObility, www.sumo-sim.org
- [3] iTETRIS Consortium (publisher): iTETRIS Deliverable 3.2 –Traffic Modelling: ITS Algorithms, April 2010.
- [4] Google, Google Earth web site, <http://earth.google.com>.
- [5] Google, Google Maps, <http://maps.google.com/>
- [6] Bieker, Laura und Krajzewicz, Daniel (2011) Evaluation of opening Bus Lanes for private Traffic triggered via V2X Communication. First Forum on Integrated and Sustainable Transportation Systems (FISTS), 29. Jun- 1. Jul. 2011, Wien, Österreich.
- [7] Bieker, Laura (2011) Emergency Vehicle Prioritization using Vehicle-To-Vehicle Communication. Young Researchers Seminar, 8.-10. Jun. 2011, Kopenhagen, Dänemark.
- [8] COLOMBO consortium (publisher): COLOMBO project's Deliverable D2.1: "Policy Definition and dynamic Policy Selection Algorithms", November 2013.

3 Interface between proprietary Controllers and SUMO

Robbin Blokpoel

*Senior researcher at Imtech Traffic & Infra, PO box 2542 3800 GB Amersfoort,
The Netherlands
robbin.blokpoel@imtech.com*

3.1 Abstract

Over the past years the open source traffic simulator SUMO has been improved and extended a lot. One of the most important factors to simulate urban traffic is the traffic light control. Currently available are various control methods like embedded fixed time and actuated control, but also external controllers that use the extensive TraCI interface of SUMO that allows reading and changing many parameters in the simulation. However, this interface has not yet been used to couple proprietary controllers, which would enable to use the simulator for accurate studies for governments who consider changing the traffic light controllers.

This paper will describe how Imtech controllers were coupled to SUMO. It will consider the topics of architecture, detection, signal group control and simulation speed optimization. In the last section results of the SUMO simulation will be compared to those of the commercial Vissim simulator for the exact same scenario.

3.2 Introduction

Over the past years the open source traffic simulator SUMO has been improved and extended a lot with at the time of writing already a 19th version available. With a large community involved and a history of more than 10 years, the simulator can be considered a serious alternative to available commercial solutions. The open source nature and easy access to almost all parameters during runtime make the simulator suitable for research projects. Therefore, the European funded project COLOMBO [1] chose to use SUMO for traffic simulations.

The COLOMBO project works on traffic surveillance algorithms for low penetration cooperative systems [2], in which both vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) communications are modelled using the ns-3 [3] communications simulator. The output of these traffic surveillance algorithms is used by new traffic control algorithms to control signalized intersections.

Currently SUMO supports various kinds of traffic control; fixed time and vehicle actuated are fully supported by SUMO itself. For other types of control and variations on the embedded vehicle actuated method, external controllers can take over control through TraCI (Traffic Control Interface). Currently, these external controllers are stand-alone applications specifically made for connection to SUMO, like the example Python program that comes with SUMO. However, for good comparison between traffic systems currently running on the

street and results of research projects, it is important to use the same scenario and simulation environment. Therefore, an interface between SUMO and a real-world controller would be a very useful tool for COLOMBO to compare its solutions with what is currently available on the market.

Fixed time and vehicle actuated controllers can still be approximated by either SUMO's internal traffic control options or external applications. City specific rules about pre-starts, not early cutoff and variable safety margins according to detection information can make this a very complicated task that would favour using the real world controllers instead. This holds even stronger for traffic adaptive control, like Imflow [4], which is too complicated and differences between competing products are too large to simulate their behavior with a different external application.

For these reasons it was decided to create an interface between Imtech's real world controllers and SUMO. This paper will describe the architecture, detection, signal groups, speeding up the simulation and a comparison between Vissim and SUMO. This is done for the scenario of Assen-Noord, a small network in the north of the Dutch city Assen.

3.3 Architecture

The architecture of the interface and all involved components is described in the picture below.

The TLC (Traffic Light Controller) blocks in the diagram are the real-world traffic light control

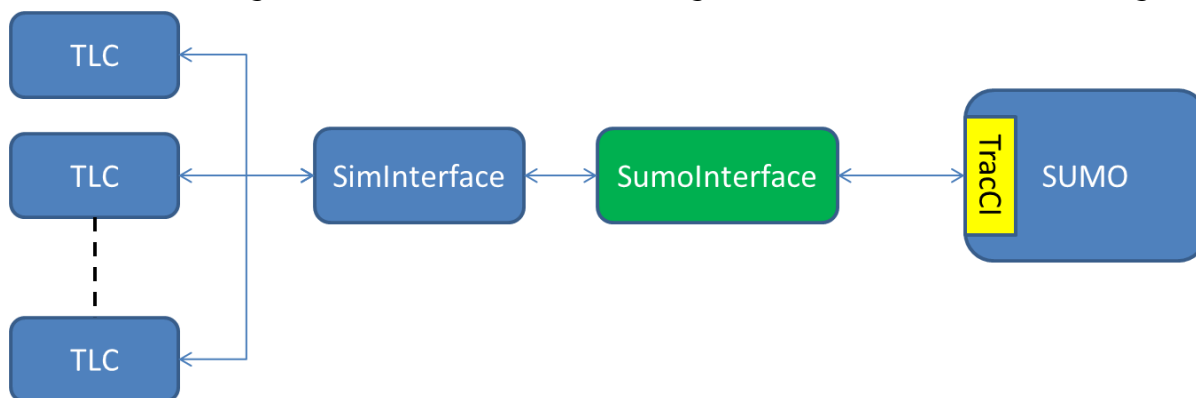


Figure 3-1: Interface architecture

executables. They accept either detection cards and lamp control units or a connection to the SimInterface to work properly. The SimInterface is a C++ dynamic link library (dll) that can maintain connection to multiple TLCs at once. On the other side it can connect to any external application that supports the dll. This has been used to connect to the commercial simulators Vissim, Paramics and Aimsun. For SUMO an intermediate block, the SumoInterface, has been created in Java that supports both the dll and can talk to TraCI to get information from SUMO. The flow of information consists of 2 main flows, detection information going from SUMO to the TLCs and signal group status from the TLCs to SUMO.

The execution flow of the interface starts with setting up a TraCI connection and connection to the SimInterface dll. When this is done all available detectors are requested through TraCI and linked to the correct intersection. Then the main execution loop is entered that checks

detector status and signal group status every 100ms. Changes to signal group status are written to SUMO, while detector status is sent directly to the dll. Finally, SUMO is ordered to execute another simulation step through TraCI.

3.4 Detection

Detection is a key element for a controller, because without it only fixed time control is possible. Therefore, having proper detection functionality is vital for accurate simulation. SUMO supports 3 different kinds of detectors, inductive loop, lane area and multi-entry multi-exit detectors. Current traffic control is mostly based on detectors that cover an area in a lane, this can be an inductive loop, but also a marked area in a video detector. Therefore, the original inductive loop detector of SUMO is actually not sufficient for traffic control simulation, since it's an infinitely small detector that doesn't cover an area in a lane, but just a point on the lane. Even real world inductive loops cover larger areas, so a real inductive loop cannot be modelled accurately with a SUMO inductive loop. Many vehicle actuated strategies use long area detectors of up to 30 meters to cut off the green phase at an efficient moment.



Figure 3-2: Typical detection field for vehicle actuated control

Figure 3-2 shows a loop close to the stopline and a longer loop at around 15 meters behind it that has a length of 20 meters. When a vehicle leaves this loop, the front of the vehicle is maximally 12 meters behind the stopline. Turning the light to amber at this moment will not make the vehicle stop, since the distance is less than 1 second. This enables the controller to utilize part of the amber time by letting the last vehicle of the platoon pass through during amber. This technique of detecting the end of the platoon would also work at the stopline, but then the amber time cannot be utilized. The reason for using a long loop of 20 meters is to deal with small gaps in platoons due to different acceleration rates. If the loop would be shorter, a threshold gap time would have to be introduced that would make usage of the amber time impossible. The most usable alternative would be a small loop at $15+20 = 35$ meters from the stopline, but its working would be based on a presumed fixed vehicle speed, which is inaccurate close to an intersection. Additionally, this work focusses on simulating real world controllers and the original controllers expect long area loops. Connecting different loops in the simulator will give different behavior unless parameters inside the controllers are changed.

Interfacing with the detectors through SUMO is quite straightforward. During the development of the interface a small extension to Traci was made to be able to access

occupancy of lane area detectors. This extension is available in version 0.20. This is done using command 0x8E (get LaneAreaDetector Variable) and variable 0x10, the number of vehicles on the loop. In the dll this is fed back as a list of detectors that can be occupied (1) or not occupied (0). Main challenge in this is the configuration, since the dll does not use detector IDs. The order of the detectors has to be the same as it is configured in the controller executable. This problem was previously solved for Vissim simulations by a naming convention, detector numbers have an ID number specified as follows: intersection ID * 1000 + detector sequence number. So the first detector for intersection 37 has an ID of 37000, the second 37001, etc. The network conversion tool of [5] automatically uses the correct naming conventions when the original network has the correct numbering as well.

As described in the architecture section, the update time for detectors is 100ms. This is done in order to never miss any detection. Motorcycles can be as short as 2 meters and on the highway, the speed can be over 30 meters per second. This means they occupy a detector for only 100 milliseconds. When vehicles are shorter and drive faster, a shorter update time will be required. In urban environments with longer vehicles the simulation may speed up when speeds are lower by checking the detectors less frequent. For 4 meter vehicles at 15 m/s, it is only required to check every 300ms.

Another important aspect to consider is the stopping distance in front of a red light. This is shown in the figure below.

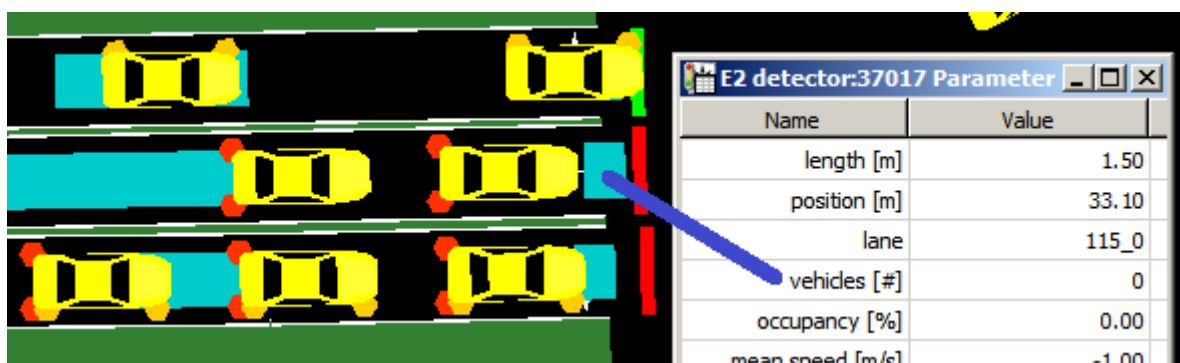


Figure 3- 3: Problem with detector location and stopping distance

The loop indicated by the blue line is not occupied because vehicles always stop at 2.5m before the signal head. Therefore, the request is not registered at the traffic light controller and the signal group will never become green. In SUMO 0.20.0 this stopping distance was decreased to 1.0m.

3.5 Signal Groups

Sumo uses a different kind of numbering for the signal groups than is usual in traffic light controllers. Vissim has one signal head per lane and a signal group can contain multiple signal heads, which happens for example when there are two lanes for a certain direction. SUMO, on the other hands defines connections, which can be considered signal heads, in the .net.xml. There is one connection per turn direction per lane. So when there is one lane from which a right turn, through direction, left turn and u-turn are possible, it will have 4 signal heads as opposed to only 1 in Vissim. Therefore a translation XML file is used by the interface

to convert TLC signal group number to a SUMO identification number. An example of a translation file is shown below:

```
<intersection id="1">
    <signalgroup id="1000" sumoSGs="2,3,4"/>
    <signalgroup id="1001" sumoSGs="5,6,7"/>
    <signalgroup id="1002" sumoSGs="1,8"/>
</intersection>
```

The translation is made as an add-on to the software of [5] during the network conversion process. Per edge the convertor knows which Imflow signal group number belongs to it, while the list of connectors per edge in the .net.xml is also known. The conversion file simply contains per signal group ID, the list of linkIndices. During operation of the interface the traffic light status is translated according to the file. Suppose the controller wants signal group 1000 green and the rest red, the SUMO translation is as follows: rGGrrrr.

Again in the dll there is no ID registration, the order is always the same and therefore it is important that the translation file has the signal groups numbered according to the order in which they are configured in the controller executable. Also, there are more states defined than in SUMO: undefined, green, red, off, red+amber, amber, amber flashing, red flashing, green flashing, red+green flashing and green+amber. Some of these states don't exist in SUMO and are converted to simpler states, like red+amber is functionally red, so it will just show red in SUMO, since the driver model wouldn't take this into account. Similarly, green+amber is just shown as green. Most flashing states are implemented to show "O" for half a second and "Y" or "G" for the other half second. Note that the symbols have to be capital otherwise vehicles may decelerate unnecessarily. For red flashing it is slightly different, it will just show continuous red to prevent vehicles from entering the intersection while the light is temporally off as part of the flashing. When no external controller is connected to the dll, the state is automatically set to amber flashing. During operation in every 100ms the software checks whether the status has changed and if so sends a "Change Traffic Lights State" 0xC2 with a new state tuple (0x20) String. The reason to choose for new state tuples is because the traffic light can show many combinations of some lights being yellow while others are still green during stage transitions. Putting all these possible combinations into either a program (0x2C) or predefine them in a SUMO configuration file and selecting the right phase index during operation (0x22) would be a lot of work.

3.6 Simulation speed

It was noticed that the network used for testing this interface was running much slower after the detectors were connected. Although the number of detectors is high, with 168 over 5 intersections, the delay was much larger than expected. An implementation that sends separate TraCI commands for each detector requires up to 30ms per intersection per simulation step of 100ms. This meant that the simulation ran approximately at the same speed as real-time speed (on a 2.53 GHz core 2 duo). So each second of simulation took one second on the clock. Without detection this speed was 50x realtime. A hypothesis that the large number of Traci calls caused this led to combining all detector requests of 1 intersection in one call. This led to an increase in simulation speed to almost 2x realtime speed, which is an improvement with respect to the first implementation, but still not acceptable. It appears there is an internal SUMO problem with TraCI causing the large delays. Subscriptions are also

not going to solve this problem, because reducing the amount of requests from 168 per timestep to 5 did only marginally decrease the delay. A further reduction from 5 to 0 would not reduce the delay significantly. Further investigation in cooperation with the SUMO development team is required to investigate this issue.

3.7 Comparison between Vissim and SUMO

This is done for the scenario of Assen-Noord, a small network in the north of the Dutch city Assen. The network only has pedestrian and bicycle crossings at the middle left intersection. All other intersections have just vehicles. The larger traffic streams (up to 1500 vehicles per hour) are going north-south on both sides of the network and the major bottleneck is the bottom intersection where the two north-south streams come together.

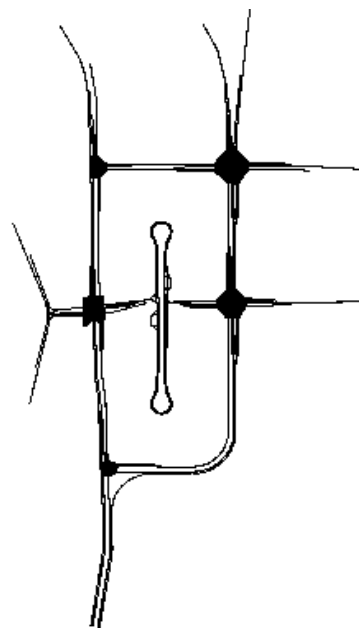


Figure 3-4: Simulation network of Assen-Noord

When watching the simulations in both Vissim and SUMO, no clear differences could be noted, except the uniformity of SUMO's vehicle injection and that all vehicles have the same acceleration at the stopline. SUMO was used in a standard way creating the routes with a trip file that would inject vehicles with a constant time period in the resulting .rou.xml. Evaluation in Vissim was done by putting a travel time section for each signal group and in SUMO a multi-entry multi-exit detector.

When evaluating the results it was found that the Vehicle count in SUMO was off, sometimes only 35% of the actual volume was measured. It appears to occur mostly when there is a high density on the multi entry multi exit detector, since signal groups with low volume were counted correct. The delay time could be acquired directly in Vissim, but in SUMO a run with all signal heads switched to "O" was done to acquire the free flow travel time, which was subtracted from the measured travel time to get the delay time. From this it could be noticed that on average the delay for pedestrians and bicycles was 2.0 seconds higher for SUMO than

for Vissim. On the other hand, for normal vehicles this delay was 1.3 seconds lower for SUMO.

These results were obtained using as much standard settings as possible. However, SUMO has many options for car following models and different vehicle models with other acceleration parameters for vehicles, bicycles and pedestrians. Tooling also exists for more random vehicle injections with normal or Poisson distributions. Using these options will make it possible to have the results closer to Vissim simulation results.

3.8 Conclusion

The paper has shown a method of coupling Imtech proprietary controllers to SUMO. The architecture used the same dll as other simulators use to couple to these controllers and therefore enables a user to freely select the preferred simulation software. For the interface different methods of assigning IDs to signal groups, signal heads and detectors between controllers and SUMO were overcome with a translation xml and a naming convention. On the SUMO side some extra variables were added to the TraCI interface to be able to access lane area detectors as well. A test network that was implemented both in Vissim and SUMO showed that the results do not differ more than could be explained by different vehicle model configuration parameters.

Open issues identified during the work were problems with counting vehicles on the multi-entry multi-exit detectors and slow response to detector status requests through TraCI. Both issues will be taken up with the SUMO development team.

3.9 References

- [1] Krajzewicz, D., et al., *COLOMBO: Investigating the Potential of V2X for Traffic Management Purposes assuming low penetration Rates*, ITS Europe congress, Dublin, Ireland, 4th of June 2013.
- [2] Koenders, E., in 't Velt, R., *Cooperative technology deployed*, ITS Europe, Lyon, France, 2011.
- [3] ns-3 (2012). ns-3 project web-pages, on-line: <http://www.nsnam.org/>. Last visited on 2014-02-11.
- [4] Van Vliet, K., Turksma, S., *ImFlow: Policy-based adaptive urban traffic control First field experience*, ITS Europe, Dublin, Ireland, 2013.
- [5] Blokpoel, R.J., *Network Conversion for SUMO Integration*, 2nd SUMO conference, Berlin, Germany, 2014.

4 Network Conversion for SUMO Integration

Robbin Blokpoel

*Senior researcher at Imtech Traffic & Infra, PO box 2542 3800 GB Amersfoort,
The Netherlands
robbin.blokpoel@imtech.com*

4.1 Abstract

Over the past years the open source traffic simulator SUMO has been improved and extended a lot. To improve compatibility with other traffic related software, network conversion tools are available. However, due to the different modelling architecture of these tools, network conversion proves to be difficult. Most tools choose a road as a starting point for the modelling, while SUMO uses nodes and connects between these nodes with streets/edges. This paper gives an overview of the problems that can be encountered while converting networks to SUMO format. A possible solution is given for each problem by using example networks from the Imflow traffic control configurator which uses a format similar to the commercial traffic simulator Vissim. The end result is a conversion tool from Imflow configurator format to SUMO of which the lessons learned can be used in future conversion tool development.

4.2 Introduction

Over the past years the open source traffic simulator SUMO has been improved and extended a lot with at the time of writing already a 19th version available. With a large community involved and a history of more than 10 years, the simulator can be considered a serious alternative to available commercial solutions. The open source nature and easy access to almost all parameters during runtime make the simulator suitable for research projects. Therefore, the European funded project COLOMBO [1] chose to use SUMO for traffic simulations.

The COLOMBO project works on traffic surveillance algorithms for low penetration cooperative systems [2], in which both vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) communications are modelled using the ns-3 [3] communications simulator. The output of these traffic surveillance algorithms is used by new traffic control algorithms to control signalized intersections.

In order to evaluate the results of the project accurately, test scenarios were required. These scenarios were already available from different projects, but only in Vissim and Imflow [4] configurator format. Since some of the baseline scenarios would use Imflow and also because parts of the project results will be integrated in Imflow as well, it was decided to make a convertor between Imflow configurations and SUMO. The main advantage of this conversion is that signal group and detector mappings can also be derived during the conversion process. The paper will describe the differences between the two network models and explain the conversion process. Topics discussed are how to convert edges with connectors to nodes and edges, intersection areas, signal group and detector locations and right of way rules.

4.3 Basic Layout: from Streets and Connectors to Nodes and Edges

The Vissim and Imflow model use streets, which can be connected with connectors. Creation of a start and end node for each street and making edges that connect between these nodes results in the following straightforward solution:

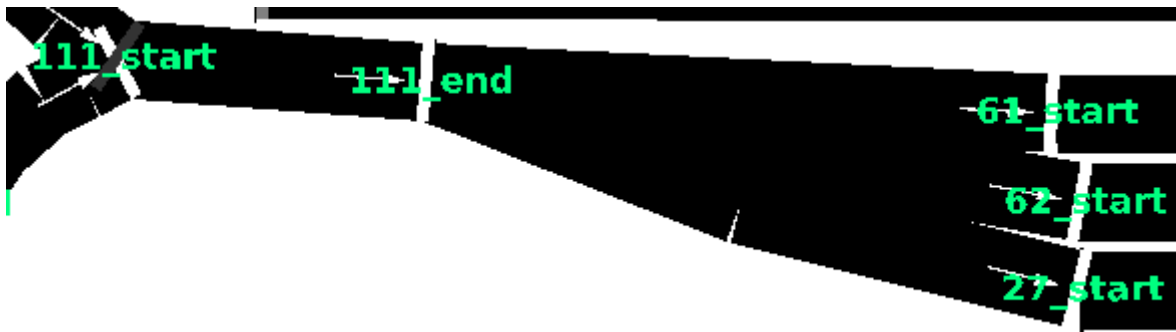


Figure 4-1: Streets to nodes and edges

The original street 111 is now connecting nodes 111_start and 111_end. The connector from 111 to 61 is connecting between node 111_end and 61_start. In the file format of Vissim, the street 111 looks as follows:

```
LINK    111 NAME "" LABEL 0.00 0.00
        BEHAVIORTYPE 1 DISPLAYTYPE 1
        LENGTH 12.659 LANES 1 LANE_WIDTH 3.50 GRADIENT 0.00000 COST 0.00000
        FROM 2196.640 926.585
        TO 2209.263 925.632
```

Between "FROM" and "TO" optional "OVER" fields can be added to describe a curvature. In Imflow format this is very similar, but there is no distinction between from, to and over, the fields are simply processed in the order of appearance, with the first and last being the from and to fields of the Vissim format:

```
<VISSIMLINK>
  <id>111</id>
  <points>
    <p>
      <x>2196.640</x>
      <y>926.585</y>
    </p>
    <p>
      <x>2209.263</x>
      <y>925.632</y>
    </p>
  </points>
```

For connectors the information is similar, but the streets that are connected should be specified. In Vissim this looks as follows:

```
FROM LINK 111 LANES 1 AT 9.043
TO LINK 27 LANES 1 AT 4.126
```

In Imflow a few XML elements are added:

```
<from>
  <link>111</link>
  <lane>1</lane>
```

```

    <location>9.043</location>
  </from>
  <to>
    <link>27</link>
    <lane>1</lane>
    <location>4.126</location>
  </to>

```

The main advantage of the Imflow format is the XML structure that makes it easier to read. However, the information regarding streets and connectors is exactly the same and conversion between the two would only be a change of format.

For the conversion to SUMO several steps are taken. First all the information about the streets and connectors is read and stored in a series of *Link* objects that contain per street or connector the following data: id, name, number of lanes, shape points and whether it is a connector or not. The name is optional and will not be used in SUMO.

The following step is to calculate where the nodes and edges should be. This process is quite straight forward, all *Link* objects that are not a connector will have a node at the location where the *Link* starts and one where it ends. When this is done a .nod.xml file for SUMO can be generated, for the example link 111, this looks as follows:

```

<node id="111_start" x="2196.64" y="926.585" />
<node id="111_end" x="2209.263" y="925.632" />

```

The .edg.xml can be created as well. Regular links go from their x_start node to their x_end node. Connectors are just like other edges in SUMO, but go from an x_end to an x_start node. For the example of link 111 and connector 111 to 27 this looks as follows:

```

<edge id="111" from="111_start" to="111_end" spreadType="center" />
<edge id="10167" from="111_end" to="27_start" shape="2209.263,925.632
    •          2221.886,920.63 2235.889,917.116 " spreadType="center"
/>

```

The shape field is optional and used when a curve was described in the original format as well. The spreadType is set to center, since just like in the original network format the coordinates indicate the center of the lane. Another optional field is numLanes="x", used when the edge has more than one lane.

The following step is to generate the .net.xml file from the .nod.xml and .edg.xml, which is done by using netconvert with the options `-offset.disable-normalization`, `--no-turnarounds` and `-no-internal-links`. The offset normalization is disabled to keep following the coordinates supplied as precise as possible. No-turnarounds and no internal links are used because there is a specific logic used for the controlled intersections, which is described later in this paper. The resulting .net.xml is loaded again by the conversion software to apply corrections.

The first correction is applied to the resulting edges. This is related to the exact positioning of the lanes, which is for some reason not entirely accurate in SUMO. For example in the Assen network, node 113_start has the following node specification:

```

<node id="113_start" x="2058.636" y="982.16" />

```

And the edge is simply specified to go from 113_start to 113_end, but still netconvert results:

```

<...shape="2058.64,981.91 2060.13,937.22" .../>

```

For some reason the Y-coordinate is shifted by 0.25. Larger differences do occur though, with up to 12 meters observed, significantly changing the topology specified in the .nod.xml file. The figure below shows a part of the network that was affected mostly by the changes of

netconvert. It is a bus-stop construction where busses can also make a U-turn after leaving and another parallel street nearby. Both have a 1 lane in each direction.

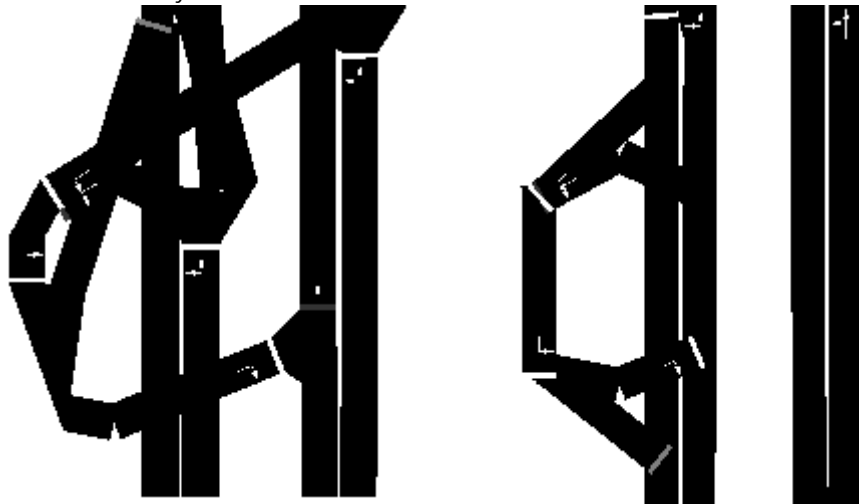


Figure 4-2: Edge movement by netconvert (left) and corrected version (right)

It appears like some edges even connect to the other street, but that is only visual. The shapes are therefore recalculated according to the node definition that was created earlier. The length of the lanes is also recalculated to keep the definition consistent.

Connectors in Vissim or Imflow do not necessarily need to connect at the end of a street, which would not be possible in SUMO with just start and end nodes. Using an extra mid-node solves this problem, but it is recommended not to use this in the editors of the original programs, since the SUMO network structure will become more complicated and thus more complex for manual editing.

4.4 Intersection areas

In Vissim and Imflow intersections are no separate entities, they are a set of signal groups and right of way rules that apply to a set of streets and connectors that may cross each other. In SUMO, however, the intersection is a separate area that has its own shape. The original streets and connectors can be used as internal lanes, but the shape of the intersection needs to be defined. The Imflow configurator has a special function for this, the intersection area polygon as shown in Figure 4- by the green area. The conversion process uses this area to identify all connectors within as internal lanes to the intersection. The shape of the junction is determined using the end points of the streets that enter the intersection area and the starting points of the streets that leave the area.

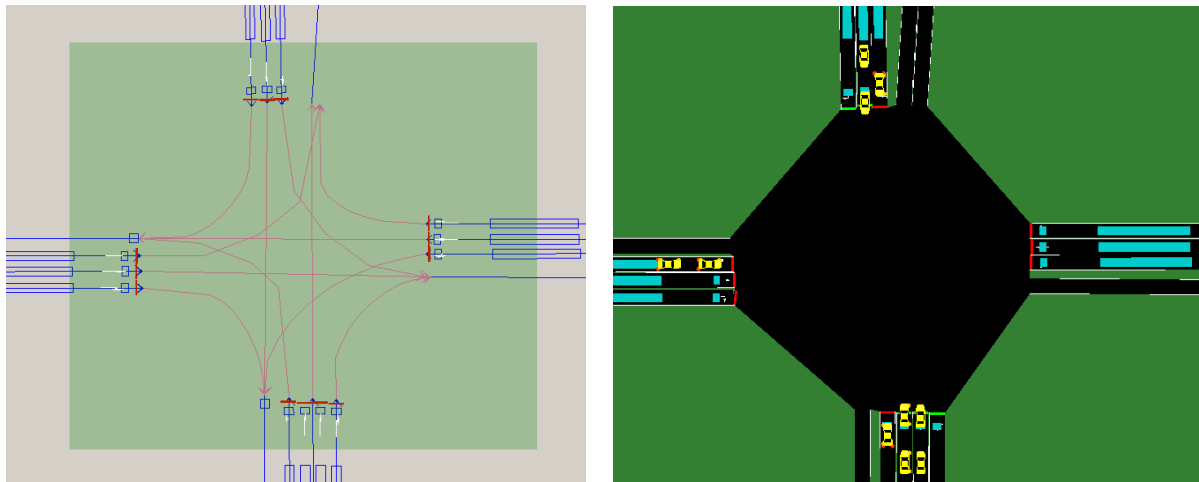


Figure 4-3: Intersection area marker (left) and result in SUMO (right)

The conversion process builds on the information acquired for converting the regular streets and connectors to nodes and edges. A distinction is made between controlled and non-controlled intersections. The latter are regular junctions in SUMO that should be made as small as possible to resemble the original network topology the best. They need small corrections for the same reason as the edges. The main goal of these small junctions is to make the network look better and prevent situations like these:

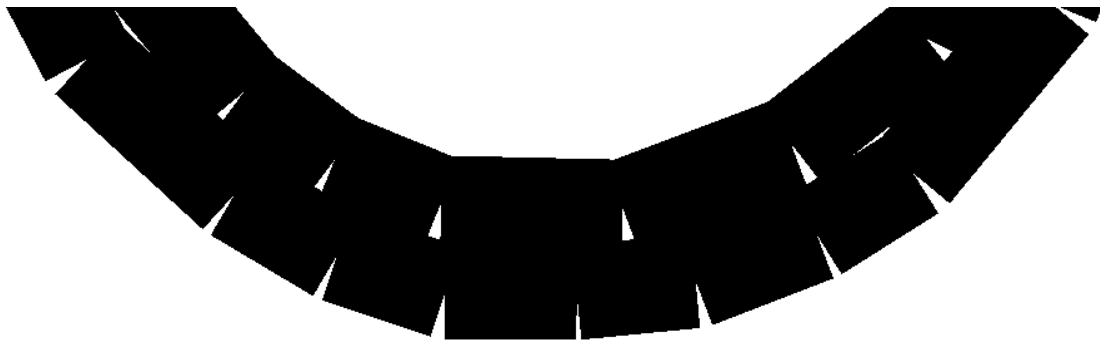


Figure 4-4: Lane "openings" in a curve

The figure shows a number of segments of an edge in a corner, there are openings in between them on the outer side of the curvature because they are displayed as pure rectangles. Between two edges this can be prevented by defining the junction as a triangle that fills this gap. Similar effects occur when multiple lanes are connected and the junction should form a shape that fills the gap between the lanes smoothly.

For controlled intersections the situation is different. A much bigger area has to be converted in a junction, which is done using the intersection rectangle of the *Imflow* configurator. Additional information is also required about the signal groups, this can be found in the "lane" element in the *Imflow* file format or in a separate *Signal Controllers* section in the *Vissim* file format. So as a first step these intersection areas and signal groups are read. This results new *intersection* objects being created and the signal groups being added to the *link* objects (which resemble edges in SUMO) in the conversion software.

When all this information has been read, the *.net.xml* file can be changed. All junctions inside the intersection rectangle are removed and all edges that were originally a connector in

Vissim/Imflow are converted to internal edges (adding a ":" to the beginning of the edge name according to SUMO standards). Then the connection elements of the generated .net.xml are checked if they are inside the intersection rectangle. If this is the case, the *state*, *linkIndex*, *via* and *tl* variables will be added to the connection element.

The latter is a more complicated procedure, the original "from" and "to" elements of the connection are read, for example:

```
<connection from="1" to="10001" fromLane="0" toLane="0" dir="s" state="M"/>
```

It is then checked if the end node of 1 is inside an intersection rectangle. In this example that is the case, so the logic proceeds. The "to" field is changed into "via" and the new "to" is looked up, which in this case is edge 16. This is because originally there were two junctions (1 to 10001 and 10001 to 16), but the whole area is merged into one large junction with 10001 an internal edge. When there are multiple lanes on 16 it is possible that there are two connectors between 1 and 16. The direction is calculated by comparing the headings of the last two points of the incoming lane and the first two of the outgoing lane. The turn directions are divided in classes, from 0 to $\pi/8$ is a through direction, $\pi/8$ to $\pi/4$ is a slight right/left, $\pi/4$ to $7\pi/8$ is a normal right/left and $7\pi/8$ to π is a u-turn. These thresholds could, however, be chosen differently when this makes the arrows on the street in SUMO look better. This implementation has few cases that will show slight right/left, its threshold could be changed to $\pi/8 - 3\pi/8$ for instance. The *tl* variable is set to the intersection number and the *linkIndex* is put in the order of appearance of the original .net.xml generated by netconvert. The *state* is put to "o" (off) by default, as during operation this should be changed by either a manually created tl-program or an external controller that connects to the simulation. The resulting line for this connection is then written to the new .net.xml as follows:

```
<connection dir="l" from="1" fromLane="0" linkIndex="0" state="o" tl="45"
to="16" toLane="0" via="10001_0"/>
```

Later the connection between 10001 and 16 is also found by the connection correction logic and a small modification is made to add the ":" to the 10001.

The next step in the conversion process is to create TL logic entries for each intersection. This entry is basically only put there because it is required to be a valid .net.xml file and contains no intelligent program. It creates a new xml *tlLogic* section for each intersection with program ID 0 and a static program consisting of two phases. These two phases both have 60 seconds duration and show all lights as off. The only intelligence in the software at this point is that the correct number of "o"'s should be shown in the phase definition. This number was acquired while reviewing the separate connections; the last "linkIndex" is the total number of characters in the phase definition. For intersection 45 it looks as follows:

```
<tlLogic id="45" offset="0" programID="0" type="static">
<phase duration="60" state="oooooooo"/>
<phase duration="60" state="oooooooo"/>
```

The last step for the intersection conversion process is the intersection itself. The signal group order and a dummy traffic light plan are also defined already. To define the shape of the intersection area all incoming and outgoing edges are ordered clockwise, like shown in the figure below:

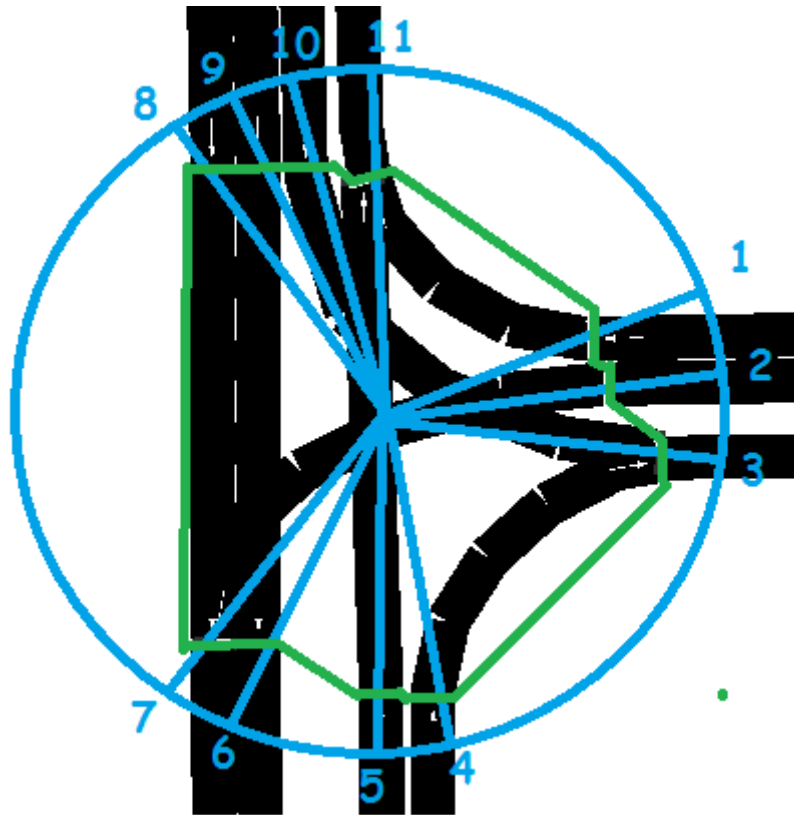


Figure 4-5: Ordering of incoming and outgoing lanes of an intersection

The intersection polygon shape is then defined by first taking the two points of end of the incoming edge marked with 1 in the figure. This process is shown by the green line in the figure and continues the same way for the 2nd lane, drawing a line between the last point of link 1 and the first of link 2. For link 3 it is a bit different because the two points are the beginning of the outgoing edge. The last line is drawn between edge 11 and edge 1 to complete the polygon. The same procedure is followed for each intersection and an example result can be seen in Figure 4-3. In the .net.xml file this results in a long series of 22 coordinates for the *shape* variable of the junction.

To complete the junction definition, some lines to define the right of way rules are required. These lines have this format:

```
<request cont="0" foes="0000000" index="0" response="0000000"/>
```

The number of these lines and the number of characters in the *foes* and *response* variables are again defined by the number of *connection* entries there are for the intersection. Currently there are no right of way rules defined automatically, since this depends on local regulations and sometimes even on the traffic light state. Therefore the entries have all zeroes and the user should define it manually when right of way functionality is required.

4.5 Signal groups and detectors

Signal groups are by definition in the end of an edge in SUMO and cannot be in the middle. In the previous section the signal groups in SUMO were already defined in the form of the connection entries per junction. This may be a problem when an intersection has a more complicated shape, with signal groups in the middle of the rectangular intersection area, like shown in the figure below:

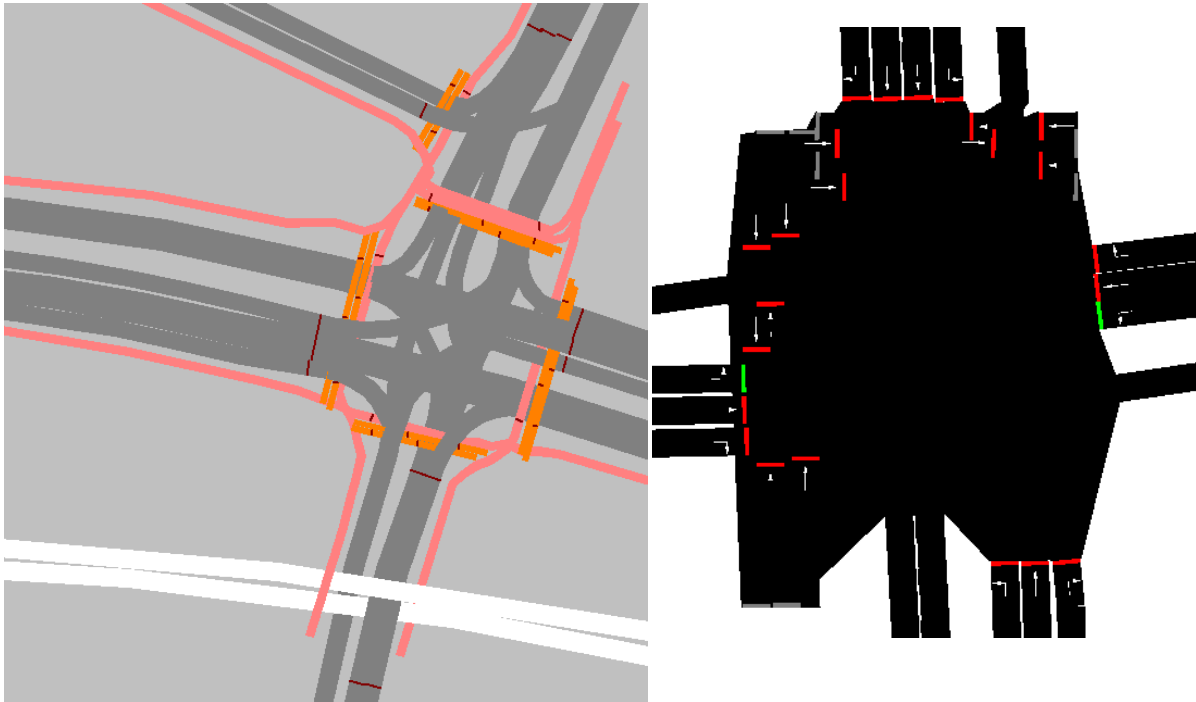


Figure 4-6: Complicated intersections for conversion

The situation of the Vissim network on the left has many pedestrian crossings inside the intersection area. On the south side of the intersection there are multiple signal heads before entering the intersection. This is due to the railroad crossing (white), which requires an extra signal head to prevent spillback on the crossing. As can be seen on the right, where only pedestrian crossings are present, this will result in a strangely shaped junction in which the pedestrian lanes are completely inside the junction shape and thus not visible as separate lanes. However, the network does function well and theoretically this is not a problem. Traditional SUMO networks solve this problem by defining multiple intersections controlled by the same traffic light controller. In theory all signal groups can be modelled like this with extra mid-nodes for a street, and a polygon for the intersection area instead of a rectangle. A less complicated solution would be possible when signal groups are allowed to be at a different position than the end of a lane. Because of the latter it is also recommended to put the signal heads in Vissim and Imflow at the very end of a street.

Another difference is the definition of a signal group in SUMO and in Vissim/Imflow. In the first it is basically a connection from an incoming lane to a via lane. This means there can be multiple signal groups per incoming lane. In the latter this is not possible, each lane has exactly one signal head and a signal group can contain multiple signal heads. A conversion between the two definitions was required to connect an external controller to SUMO and is described in [5].

Detectors in SUMO have the option of being anywhere on an edge. There are three different kinds, inductionloop, lane area and multy-entry multi-exit detectors. The first is, despite the name not suitable to model inductive loops, as they cannot cover an area in a lane and many control strategies rely on presence in an area. The figure below shows a typical detection field used in the Netherlands:

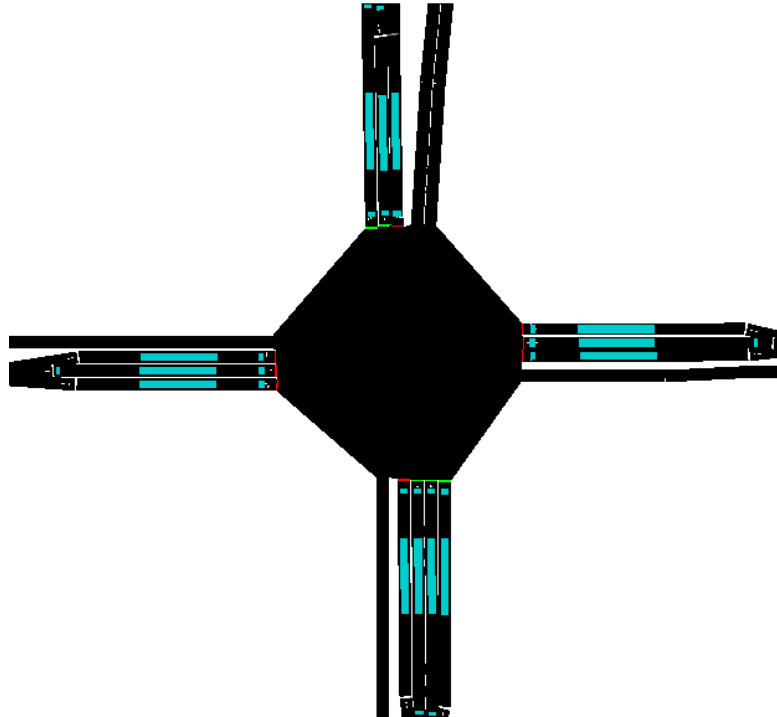


Figure 4-7: Detection field of a typical Dutch intersection

The loops close to the stopline are just 1 meter long and can still be modeled with a SUMO inductionloop, but the second loop is often longer than 25 meters and cannot be modelled with an inductionloop that has a length of 0. This would significantly change the behavior of the traffic light controller and therefore lane area detectors have been used as output of the convertor. More details on the importance of accurate detection can be found in [5]. For the rest the conversion is very straightforward, in Imflow and Vissim format the detectors are specified on which street or connector they are, at which distance from the beginning and their length. Those variables can simply be copied into the format of the .det.xml file for SUMO. Adding the file to the .sumo.cfg will then make the detectors available for the simulation.

4.6 Other network formats

The paper focused on Vissim and Imflow network formats. Other well-known network simulators are Aimsun and Paramics. Aimsun has a file format *.asn that is not human readable, since the simulator requires the same kind of data, conversion should be possible but not easy to achieve. An older publication [6] cites that Aimsun also has a separate network graphical editor (TEDI), which has a human readable format. For Paramics, the files are human readable, and in [6] it is described that the network has a more node-edge like structure like SUMO uses. This should enable easier conversion, although curves are described as an arc, part of a circle, which will require calculating the smaller segments to be used for SUMO edge shape modelling.

4.7 Conclusion

This paper demonstrated a conversion tool from Vissim/Imflow network format to SUMO. The tool has been used on several networks for the COLOMBO project resulting in directly usable SUMO networks without the need for further manual editing. The main challenges that were overcome during the conversion were correcting the output of netconvert and the definition of the intersection area.

Further developments can be done for more complex intersections that should be made up of multiple intersection areas, but still be controlled by the same traffic light controller. Additional functionality for right of way rules and traffic demand conversion is also recommended. A short review of the other popular traffic simulator network formats revealed that both Aimsun and Paramics network conversion should be possible, but both have their own specific challenges.

4.8 References

- [1] Krajzewicz, D., et al., *COLOMBO: Investigating the Potential of V2X for Traffic Management Purposes assuming low penetration Rates*, ITS Europe congress, Dublin, Ireland, 4th of June 2013.
- [2] Koenders, E., in 't Velt, R., *Cooperative technology deployed*, ITS Europe, Lyon, France, 2011.
- [3] ns-3 (2012). ns-3 project web-pages, on-line: <http://www.nsnam.org/>. Last visited on 2014-02-11.
- [4] Van Vliet, K., Turksma, S., *ImFlow: Policy-based adaptive urban traffic control First field experience*, ITS Europe, Dublin, Ireland, 2013.
- [5] Blokpoel, R.J., *Interface between proprietary controllers and SUMO*, 2nd SUMO conference, Berlin, Germany, 2014.
- [6] Cheu, R.L., Tan, Y., Lee, D., *Comparison of PARAMICS and GETRAM/AIMSUN Microscopic Traffic Simulation Tools*, 83rd annual meeting of the Transport Research Board, 2003.

5 Can Road Traffic Volume Information Improve Partitioning for Distributed SUMO?

Ulrich Dangel¹, Quentin Bragard¹, Patrick McDonagh², Anthony Ventresque¹ and Liam Murphy¹

¹Lero@UCD, School of Computer Science and Informatics, University College Dublin, Ireland. {ulrich.dangel,quentin.bragard}@ucdconnect.ie, {anthony.ventresque,liam.murphy}@ucd.ie

²Lero@DCU, School of Electronic Engineering, Dublin City University, Ireland. patrick.mcdonagh@dcu.ie

5.1 Abstract

Microscopic vehicular simulations can be computationally intensive due to the sheer size of the road network and number of vehicles. One solution is to parallelize the simulation through distribution and concurrent execution of the scenario being simulated. To enable distributed simulation of an area, the partitioning of the map into different areas for parallel execution on different nodes is required. How the map is partitioned is also a critical factor for distributed simulation, as a poor partitioning can lead to a communication overhead and/or an imbalance of workload among the computing nodes.

In this paper, we ask: Can traffic volume information improve the classical structural partitioning algorithms? In the context of improving distributed simulation in SUMO, we propose a modification to three existing mechanisms for road network partitioning, SParTSim, Smart Quadrees and Quadrees, with the aim of creating more balanced partitions (in terms of workload) derived from traffic volume data.

Keywords: Distributed Simulation, Road Partitioning, Graph Partitioning, SUMO

5.2 Introduction

Urban populations are growing dramatically: for instance, the aggregated annual population increase of six major developing-country cities is already higher than Europe's total population [1]. With the increase in the size of cities, traffic simulation requires more computation time in order to simulate more individual vehicles. This is particularly the case for microscopic traffic simulation, which can offer interesting insights to its users, but has a high computation time. Microscopic traffic simulation can accurately model urban traffic patterns and evaluate different scenarios and their impact on traffic, e.g. placement of additional bus stops at a route, traffic light sequencing, etc. By using microscopic simulation, stakeholders can directly observe the impact of their potential decisions on the traffic. As stated above, microscopic simulation models are generally slow, as they need to process a large number of elements (e.g., individual cars). A standard solution to reduce the overall required computation time is to parallelize and distribute the simulation.

Classically, parallel or distributed systems split the problem space into different partitions, i.e., sub problems for concurrent execution, this may involve synchronisation between nodes if

data from one partition needs to be moved to another partition. For vehicular simulation, these partitions are typically based on the road network or the spatial map - we call this style of partitioning, *structural*. The partitioning algorithms are evaluated using two main metrics [2]: (i) the balancing of computational workload across the nodes that run the partitions; (ii) the communication overhead generated by the distribution.

Distributed simulations are currently an active area of research interest within the SUMO community. There has been recent work to provide a multi-agent system on top of SUMO [3] by combining it with an existing multi-agent development framework [4]. Another approach for distributed SUMO simulation is dSUMO [5], a framework that interconnects SUMO instances, each running separate, but spatially connected areas of a map. Both solutions require mechanisms to divide the road-network into different areas for parallel processing on their respective nodes.

In this paper, we propose an enhancement for distributed simulation using SUMO by using traffic volume data to improve the load balancing of the individual partitions and minimizing the communication overhead, in order to reduce the overall required computation time of the distributed simulation. We evaluate this idea by comparing results against those obtained for SParTSim [6], Quadtrees [7] and Smart Quadtrees [8].

5.3 Related Work

Partitioning in general is a key concept in distributed and parallel computing. In MapReduce [9] the mapping is a partitioning which is responsible for distributing the input data to different processes. This partitioning step enables the distributed and parallel execution of the work.

Other, more domain-specific partitioning schemes, provide guidance how to select and choose appropriate partitioning algorithms.

Space partitioning, for example, is often used in computer graphics [10-12] and visualisation. An overview about different space partitioning algorithms was provided by the authors in [13]. Here the authors discussed Quadtrees, unconstrained k-d trees, constrained k-d trees and region growing with region growing performing best for their simulation. Space partitioning is widely used in distributed or parallel computation, such as Massively Multiplayer Online Role-Playing Games (MMORPG) or Raytracing [14]. Employing a binary space partitioning mechanism, such as Quadtrees, will lead to the creation of a spatial hierarchy. This hierarchy can be used to divide a city, and assign pieces of it (partitions) to different nodes. Another approach for the space partitioning of cities is to reuse existing boundaries such as postal districts. The problem with both approaches is that they typically do not use the road-network for the partitioning but only spatial information. With regards to a distributed vehicular simulation, this can lead to uneven distribution of workload. This in turn, will lead to decreased simulation performance as a result of uneven processing times for simulation steps, resulting in some nodes waiting for others when synchronisation is required.

Graph-partitioning on the other hand, does not consider the space but uses the graph-structure of the problem. Graph partitioning has been used to parallelize clustering of documents [15], parallel factorisation of sparse matrices [16] as well as workload distribution [17]. Graph partitioning has been originally implemented with heuristics [18] and was later extended to utilize genetic-algorithms [19]. Graph-growing, is a refinement and extension [2] of classical graph partitioning and expands individual partitions in each step. Region growing, similar to graph-growing, has been shown to be best solution for crowd simulations [13].

Graph partitioning is widely used [20, 21] in different domains such as workload distribution, task scheduling and in the VLSI [22] domain. Using graph partitioning for vehicular simulations solves multiple issues encountered with space partitioning, such as uneven distribution of roads in a partition as graph partitioning works on the street level and not on the map. Taking road properties into account can further refine graph partitioning, i.e. edges provide attributes about the significance of a particular street. By using additional attributes of street-data, a graph partitioning targeted for road networks can be derived, such as SParTSim.

5.4 Experimental Evaluation

In order to use input data for the different partitioning algorithms, we have to extract volume data to provide the partitioning. In real-world scenarios, such data can be extracted from existing Traffic Management Systems, such as SCATS [23] or IRIS³. In this work, we use the dataset provided by TAPAS Cologne [24] with SUMO to extract the volume data. Below, we describe the formula used for providing a weight for nodes in the road graph based on traffic data, as well as modifications to the existing algorithms.

5.4.1 Volume extraction

As some of the algorithms used are graph based, we provide a weight per node instead of a weight per edge. This allows us to use the same weighting for all algorithms, whether they are graph or space-based. We use a weighted sum as shown in (1), to calculate the weight of a node, N_w , with c_t being the total number of cars present at step t , c_{tn} the number of cars at node n at step t .

$$N_w = \sum w_t \frac{c_{tn}}{c_t}. \quad (1)$$

Where the weight w_t is defined as the number of cars in this step over the maximum number of observed cars (in any one step), as shown in (2).

$$w_t = \frac{c_{tn}}{c_{max}}. \quad (2)$$

By using (2) we ensure that steps with a low traffic volume have a lower impact on the overall weight of a node.

5.4.2 Modification of Quadtrees

Quadtrees are a space-dividing partitioning method, often used to divide two-dimensional spaces. Quadtrees divide a space recursively into sub-regions, until a specific stop condition is met, e.g., the space is divided evenly or, into the required number of partitions.

The original version of the used Quadtree algorithm uses the sum of the street size (street length * number of lanes) to select the partition to divide. We modify Quadtrees to not use the sum of the street size but the sum of the volume data from Section 5.4.1 oben, in order to select the partition to divide further. By using the sum of the volume data for each partition, we choose the area with the highest weight to divide further.

³ <http://iris.dot.state.mn.us/>

5.4.3 Modification of Smart Quadrees

Smart Quadrees, also referred as grid based partitioning, are an extension to Quadrees where the map is initially divided into small, independent grids. These grids are then merged together according to some heuristic, based on the value of an individual region. This differs to Quadrees as Quadtree divides a map into 4 similar regions, while Smart Quadrees divides a map into small grids and merges them until all grids are merged.

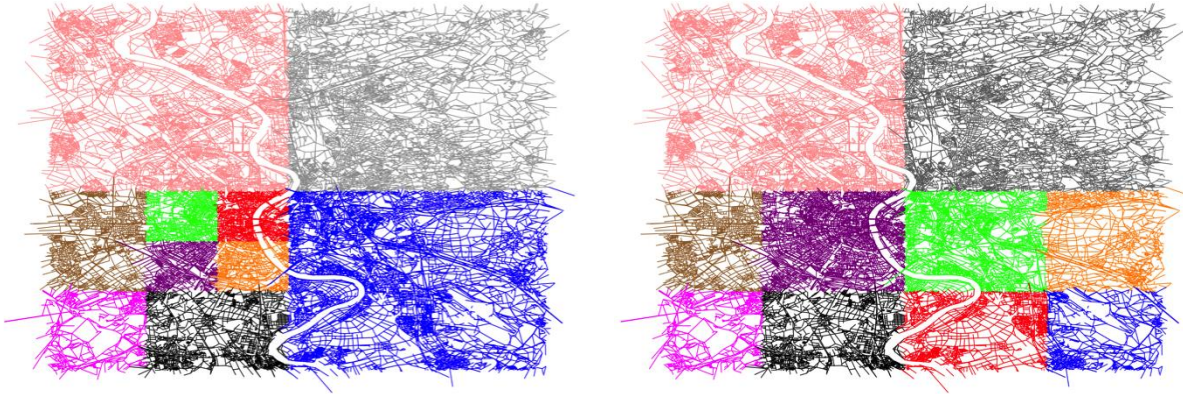


Figure 5-1: Output of the modified Quadtree algorithm (left) and unmodified Quadtree (right) with ten partitions or three divisions.

The unmodified version of the Smart Quadtree implementation uses the street size (street lengths * number of lanes) as a heuristic. We modify Smart Quadrees by changing the heuristic to use the sum of the volume data, as described in Section 5.4.1, for each grid. The difference between the two implementations is shown in Figure 5-2.

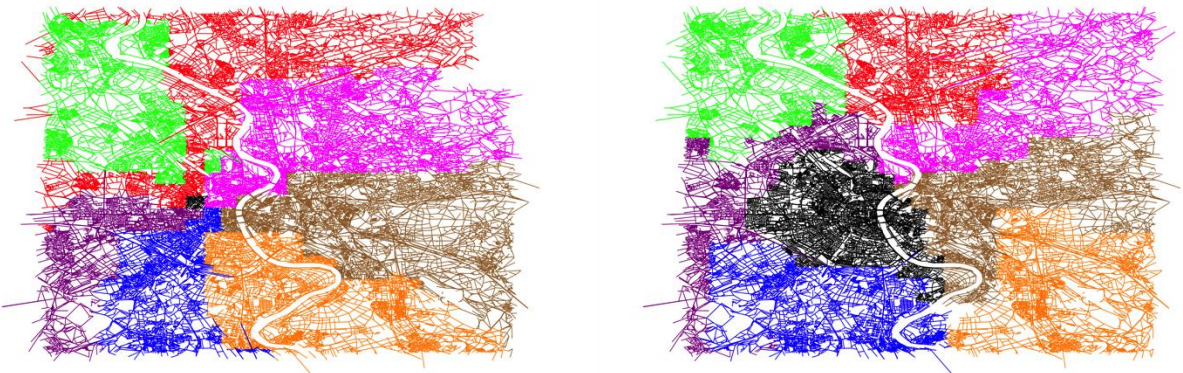


Figure 5-2: Output of the modified Smart Quadtree algorithm (left) and unmodified Smart Quadtree (right) with eight partitions.

5.4.4 Modification of SParTSim

SParTSim uses the concept of creating a domain-specific partitioning algorithm for road networks by combining space partitioning (region-growing) with graph partitioning. By utilizing both space-, and graph-partitioning methodologies, SParTSim aims to produce better partitions for vehicular distributed simulations. SParTSim determines the starting point of each partition by choosing the node with the highest degree. After the starting point for the individual partitions is selected, each partition grows, starting from the starting point. SParTSim grows the partitions based on road-network attributes, such as number of lanes.

As unmodified version of SParTSim determines the starting point of a partition by choosing the nodes with the highest degrees, the starting point of a partition impacts the shape of an individual partition as the partition starts to grow around this point until it can't grow any more as a result of all areas now belonging to other partitions, i.e. all areas on the map are

covered. SParTSim then trades road segments between partitions to minimize the road cuts between partitions and to achieve load-balanced partitions. The SParTSim algorithm only uses static graph properties to achieve evenness of road topology between partitions. In order to do this it uses a heuristic to determine the workload for the individual partitions. However, SParTSim considers that if the road topology is balanced between partitions, then the workload will be similar, irrespective of the actual traffic volume. Therefore, we modified the starting point selection in SParTSim to use the nodes with the highest traffic volume (as determined in Section 5.4.1) instead of using the nodes with the highest degree. Figure 5-3 shows the partitioning result of both the unmodified and modified version of SParTSim.

5.5 Evaluation

In this section, we evaluate the use of traffic volume data for Quadtrees, Smart Quadtrees and SParTSim. We divide the city for both Smart Quadtrees and SParTSim into four and eight partitions while we use for Quadtrees four and ten partitions. The visual partitioning outputs for Quadtrees with 10 partitions is shown in Figure 5-1, with the outputs for Smart Quadtrees with eight partitions is shown in Figure 5-2 and those for SParTSim are shown in Figure 5-3.

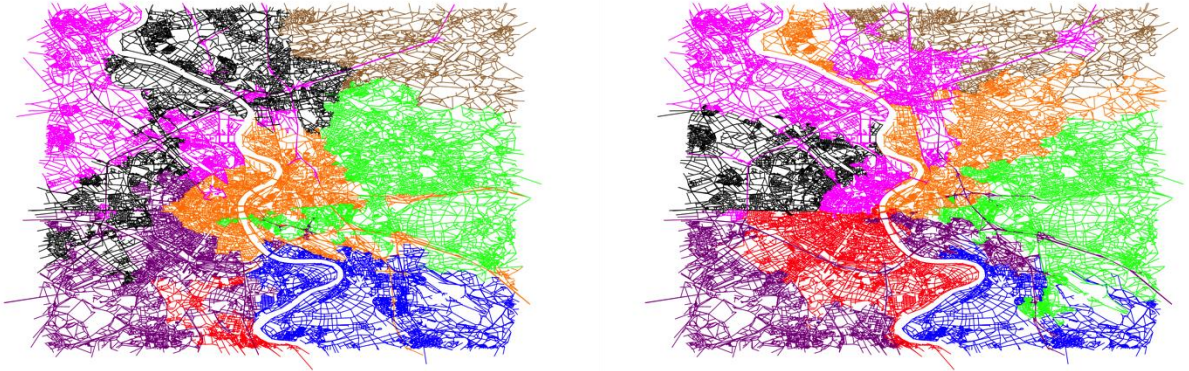


Figure 5-3: Output of the modified SParTSim algorithm (left) and unmodified SParTSim (right) with eight partitions.

5.5.1 Metrics

In our evaluation we focus on two metrics, communication overhead and workload balance. For communication overhead, we calculate the number of messages sent between partitions in each step. These messages represent the movement of a vehicle on a road segment, which is divided across partitions. We can calculate this with SUMO by extracting the position of each vehicle in each step. If a street intersects or touches a partition border, it is part of multiple partitions. This ensures that states are shared between different nodes. If a vehicle is on a road segment, which is divided across partitions, a message has to be sent to the neighbouring partition to transfer the state of the vehicle across to the new partition. As each message has to be communicated and processed by dSUMO, the lower the number of messages, the better. The results for this metric are provided below.

To evaluate the workload balance between partitions, we calculate the Simpson Diversity Index [25], as shown in (3), with C_p being the cars in partition p , c_t the total number of cars in step t and P the number of partitions. The result is between 0 and 1 with 1 being a perfectly load balanced system and 0 being the opposite for unbalanced workloads between partitions.

$$D_t = \frac{1}{\sum_{p=0}^{P-1} (C_p/C_t)^2 P}. \quad (3)$$

5.5.2 Results

We use the TAPAS Cologne [24] 0.17 scenario to evaluate our result. TAPAS Cologne is a simulation describing the traffic of Cologne on a workday between 06:00 and 08:00 am. The data was captured as part of the TAPAS project [26] and has been refined multiple times. The scenario consists of 7200 steps, with one step representing one second in real-time. TAPAS Cologne contains more than 250,000 vehicles traces for the two-hour period.

Figures 5-4 and 5-5 display the number of messages between partitions per simulation step for Quadtrees, Smart Quadtree and SPaRTSim. We don't distinguish between the modified and unmodified Quadtree for four partitions, as both results are exactly the same.

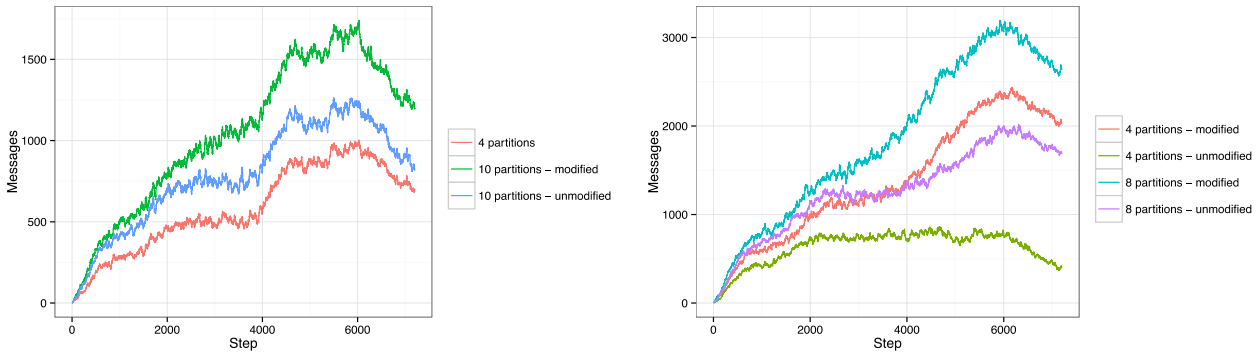


Figure 5-4: Number of messages sent per simulation step for Quadtrees (left) and Smart Quadtrees (right). Modified and unmodified versions are both shown.

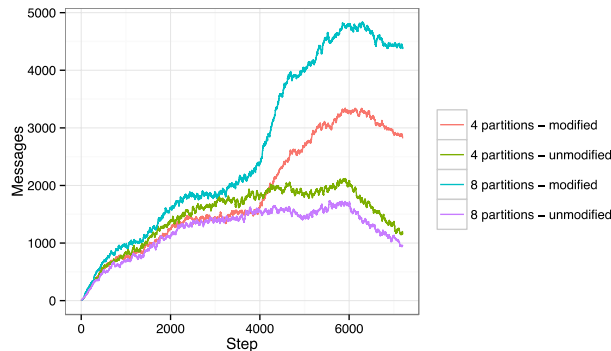


Figure 5-5: Number of messages sent per simulation step for SPaRTSim, both modified and unmodified for 4 and 8 partitions

Due to the regular, rectangular shape of the partitions the Quadtree shows the best communication properties. In all cases, though, the modified versions of the algorithms show increased levels of communication, compared to the unmodified versions. The modified Quadtree algorithm selects the city centre (Figure 5-1) for further partitioning, resulting in additional communication overhead. For the Smart Quadtree algorithm, our modified version created some small partitions (Figure 5-2), causing additional communication overhead. Our modified algorithms show higher communication overhead compared to the unmodified versions. This is expected as our modifications focus on load balanced partitions and does not optimize with regard to communication. However, as can be seen below (in terms of workload balance) our algorithms achieve a higher level of balancing between partitions, which should provide higher utilisation across all compute nodes as delays incurred by waiting for simulation should be decreased.

SPaRTSim has a trading phase, which aims to reduce the communication overhead. This behaviour can be observed in Figure 5-5 for the unmodified versions, which perform better than the Smart Quadtree. Our modified initial starting point selection for SPaRTSim caused the

increased communication overhead. This shows, that even though SParTSim has a trading mechanism to reduce the communication overhead, the initial point selection has a large impact on the resulting partition.

For the case of workload balance between partitions, Table 5-1 shows the properties of the Simpson diversity index (the higher the number, the better) over the complete simulation for all 3 algorithms. For both Quadtree and Smart Quadtree our modification provides better load-balanced partitions compared to the unmodified versions of the same algorithm, e.g. for the Smart Quadtree our modifications are twice as good as the unmodified versions. Our modifications to SParTSim on the other hand, provide slightly worse results compared to the unmodified algorithms. This is due to the trading phase of SParTSim, as we did not adjust the trading phase but only the initial starting point selection.

Table 5-1: Simpson diversity index for the different partitioning algorithms over the simulation.

Name		Min	Median	Mean	Max
Quadtree	4 partitions	0.3680	0.7190	0.7318	0.8540
	10 partitions – modified	0.3570	0.6070	0.6021	0.6330
	10 – unmodified	0.2270	0.3530	0.3674	0.5540
Smart Quadtree	4 partitions - modified	0.5610	0.9000	0.9157	0.9940
	4 partitions - unmodified	0.358	0.431	0.427	0.568
	8 partitions – modified	0.4460	0.7760	0.7798	0.8550
	8 partitions – unmodified	0.2840	0.3890	0.3889	0.4940
SParTSim	4 partitions – modified	0.4810	0.6650	0.6718	0.7340
	4 partitions – unmodified	0.7210	0.7920	0.7854	0.8450
	8 partitions - modified	0.4060	0.4540	0.4642	0.6430
	8 partitions – unmodified	0.4710	0.6850	0.6573	0.7780

Comparing the different algorithms to each other shows that our modified Smart Quadtree produces more even partitions than the other partitioning algorithm. Our modified version of the Quadtree took 1h13min to compute 10 partitions, Smart Quadtree took 1h22min to compute eight partitions while SParTSim took 5h14min for eight partitions on a 4 Core I7-2600 with 16GB of memory. As shown in [27, 28] load balanced simulations are a required to optimize the overall computation time.

The modified Smart Quadtree provides more balanced partitions compared to the other algorithms. Furthermore, the modification to Quadtree and Smart Quadtree provide more balanced partitions compared to unmodified versions of their algorithm. In addition to providing more balanced partitions, we can observe that for Smart Quadtree, the time taken to compute these partitions is significantly lower, compared to SParTSim. In the case of Quadtree, the time taken to compute the partitions is significantly lower than SParTSim (and Smart Quadtree) but at the expense of workload balance. Our modification to SParTSim on the other hand, did not provide better results, due to the unmodified trading phase. We expect that by modifying the trading phase, the result for SParTSim will improve as well.

5.6 Conclusion

In this paper we propose the use of volume data to improve road partitioning for distributed simulations using SUMO. We modify three existing partitioning algorithms to take volume data into account. In general, the volume data can be extracted by a Transportation Management System for a city or by examining results from previous simulations. We show the impact of volume data on the individual partitioning algorithms for the partition topology, as well as the impact on the distributed simulation by comparing communication overhead and workload balance between the different algorithms.

We show that partition algorithms have a large impact for distributed simulation, either providing workload balanced partitions or reducing the overall communication overhead. SParTSim, the algorithm trying to optimize for both cases, has a long runtime making it impractical for dynamic load balancing. By using traffic volume, we can improve the workload balance of simple spatial partitioning algorithms, which could make them useful for dynamic repartitioning of large simulations. This means that in order to be able to scale and distribute large-scale simulations with dSUMO, the focus for dSUMO should be on the communication overhead with external systems, as balanced partitioning has been shown reduces the overall computation time.

5.7 Acknowledgement

This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie)

5.8 References

- [1] J. Abbott, "State of the world's cities 2012/2013: prosperity of cities," *Australian Planner*, pp. 1-2, 2013.
- [2] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on scientific Computing*, vol. 20, pp. 359-392, 1998.
- [3] G. Soares, J. Macedo, Z. Kokkinogonis, and R. J. Rossetti, "An Integrated Framework for Multi-Agent Traffic Simulation using SUMO and JADE," in *SUMO2013, The first SUMO User Conference, May 15-17, 2013 - Berlin-Adlershof, Germany, 2013*, pp. 125-131.
- [4] F. Bellifemine, F. Bergenti, G. Caire, and A. Poggi, "JADE—a java agent development framework," in *Multi-Agent Programming*, ed: Springer, 2005, pp. 125-147.
- [5] Q. Bragard, A. Ventresque, and L. Murphy, "dSUMO: towards a distributed SUMO," 2013.
- [6] A. Ventresque, Q. Bragard, E. S. Liu, D. Nowak, L. Murphy, G. Theodoropoulos, *et al.*, "SParTSim: A Space Partitioning Guided by Road Network for Distributed Traffic Simulations," in *Proceedings of the 2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications, 2012*, pp. 202-209.
- [7] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta informatica*, vol. 4, pp. 1-9, 1974.
- [8] Y. Wang, M. Lees, and W. Cai, "Grid-based partitioning for large-scale distributed agent-based crowd simulation," in *Proceedings of the Winter Simulation Conference, 2012*, p. 241.
- [9] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, pp. 107-113, 2008.

- [10] J. Amanatides and A. Woo, "A fast voxel traversal algorithm for ray tracing," in *Proceedings of EUROGRAPHICS*, 1987, pp. 3-10.
- [11] H. Radha, M. Vetterli, and R. Leonardi, "Image compression using binary space partitioning trees," *Image Processing, IEEE Transactions on*, vol. 5, pp. 1610-1624, 1996.
- [12] E. Torres, "Optimization of the binary space partition algorithm (BSP) for the visualization of dynamic scenes," in *Eurographics*, 1990, pp. 507-518.
- [13] A. Steed and R. Abou-Haidar, "Partitioning crowded virtual environments," in *Proceedings of the ACM symposium on Virtual reality software and technology*, 2003, pp. 7-14.
- [14] B. Freisleben, D. Hartmann, and T. Kielmann, "Parallel raytracing: a case study on partitioning and scheduling on workstation clusters," in *System Sciences, 1997, Proceedings of the Thirtieth Hawaii International Conference on*, 1997, pp. 596-605.
- [15] I. S. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001, pp. 269-274.
- [16] A. Pothen, H. D. Simon, and K.-P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM Journal on Matrix Analysis and Applications*, vol. 11, pp. 430-452, 1990.
- [17] B. Hendrickson and R. Leland, "An improved spectral graph partitioning algorithm for mapping parallel computations," *SIAM Journal on Scientific Computing*, vol. 16, pp. 452-469, 1995.
- [18] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell system technical journal*, vol. 49, pp. 291-307, 1970.
- [19] P.-O. Fjällström, "Algorithms for graph partitioning: A survey," *Linköping electronic articles in computer and information science*, vol. 3, 1998.
- [20] B. Hendrickson and R. W. Leland, "A Multi-Level Algorithm For Partitioning Graphs," *SC*, vol. 95, p. 28, 1995.
- [21] K. Andreev and H. Racke, "Balanced graph partitioning," *Theory of Computing Systems*, vol. 39, pp. 929-939, 2006.
- [22] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: applications in VLSI domain," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 7, pp. 69-79, 1999.
- [23] P. Lowrie, "Scats, sydney co-ordinated adaptive traffic system: A traffic responsive method of controlling urban traffic," 1990.
- [24] SUMO. (2014). *TAPAS-Cologne dataset*. Available: <http://sourceforge.net/apps/mediawiki/sumo/index.php?title=Data/Scenarios/TAPASCologne>
- [25] E. H. Simpson, "Measurement of diversity," *Nature*, 1949.
- [26] C. Varschen and P. Wagner, "Mikroskopische Modellierung der Personenverkehrsnachfrage auf Basis von Zeitverwendungstagebüchern," *Stadt Region Land*, vol. 81, pp. 63-69, 2006.
- [27] A. Boukerche and S. K. Das, "Dynamic load balancing strategies for conservative parallel simulations," in *Parallel and Distributed Simulation, 1997., Proceedings, 11th Workshop on*, 1997, pp. 20-28.
- [28] K. D. Devine, E. G. Boman, R. T. Heaphy, B. A. Hendrickson, J. D. Teresco, J. Faik, et al., "New challenges in dynamic load balancing," *Applied Numerical Mathematics*, vol. 52, pp. -152, 2005.

6 Models enabling Simulations of V2X Applications regarding Emergency Vehicles in Urban Environment

Michael Düring, Mark Gonter, Falco Thiel, and Florian Weinert

*Volkswagen AG, Berliner Ring 2, 38440 Wolfsburg, Germany
{Michael.Duering, Florian.Weinert}@Volkswagen.de*

6.1 Abstract

Statistically, emergency vehicles (EVs) encounter a higher risk of getting involved in accidents during their missions than other road users. The successful completion of these missions can be facilitated by new applications. As it is not possible to test new applications in a real traffic system, suitable simulations can address that issue. Simulation of Urban Mobility (SUMO) is one possible tool to conduct simulations within real traffic systems. However, SUMO is not capable of modelling a realistic behavior of EVs, new types of infrastructure, and individual vehicles (IVs) concerning EVs by a predefined function. We propose models for each of the missing pieces towards an integrated approach to simulate EVs in an urban environment. The models are adjusted with a video analysis, simulated and assessed. They are a reference for testing new applications. The example applications are a traffic light preemption via V2I and an automated cooperative formation of a rescue lane via V2V. The models and the two applications are assessed regarding travelling time.

Keywords: Simulation of emergency vehicle, Simulation of EV, Urban, Intersection, Real EV behavior, Rescue Lane, Intelligent Transportation System, Intelligent Traffic light, V2X, Preemption, Automated Formation of a Rescue Lane, Travelling time

6.2 Introduction

Statistics about missions of rescue services in Germany indicate over 14 million missions a year [12]. This corresponds to several ten thousand missions of emergency vehicles (EVs) a day. Each mission is carried out under enormous time pressure as regional response time regulates the maximal time difference between the incoming call and the arrival of the rescue team [17]. The travelling time of an EV may be influenced by any incident on the road. Especially in urban environment, red traffic lights are a serious threat for reaching the destination in time [4, 5, 13, 14]. A red traffic light has two effects on the trip. First, the red light itself which indicates possible crossing traffic and second the obstruction by other road users waiting in front of the red light. This leads to a reduced speed as well as a higher risk of getting involved in an accident [11, 8]. A study examined the likelihood of having an accident by comparing accidents per kilometer of EVs and individual vehicles (IVs). According to the study, the risk of being killed is four times higher, being severely injured is eight times higher, and having a material damage is seventeen times higher while being on a mission in an EV [11].

To support EVs in urban situations, research and development presented many systems [5, 7, 13, 14]. A new set of applications for EVs may further enhance the safety and efficiency of

rescue services. These applications may require new types of traffic infrastructure and communication among vehicles (V2V) and vehicles and infrastructure (V2I). In short this communication is called V2X communication. To evaluate the potential of new applications, prototype systems need to be deployed in a real traffic environment and analyzed over a long time period. This is a severe alteration of the traffic system and it is hardly imaginable that local authorities allow such a procedure. However, simulations are a suitable tool to perform the necessary potential analysis.

6.3 Problem Statement

An applicable simulation framework allows us to conduct research concerning effects of new applications for EVs on the traffic system. We decided to use the simulation tool Simulation of Urban MObility (SUMO) as its strength is to simulate V2X applications improving traffic efficiency [9]. However, simulating special situations – e.g. situations comprising EVs – are not covered. EVs are allowed to not obey general traffic rules. They can drive faster, may drive through red lights, and are allowed to use their siren and light bar to inform others about their arrival and their right of way. Thus, the EV has an effect on the behavior of individual vehicles (IVs). Research community does not agree whether these effects need to be modelled in order to evaluate new applications e.g. preemption systems. Driving through red lights and the behavior of IVs may be neglected because only the difference in travelling time with and without the application is significant [10]. Others argue that by neglecting these effects the potential of new applications may be overestimated and thus needs to be implemented [18]. Bieker [1] does not implement a driving through red lights because the EV coincidental arrives during the green phase. She points out that a model to overcome the red light issue needs to be investigated. Additionally, the study implements the behavior of IVs as stopping when an EV is approaching.

The effects mentioned above issue a challenge for SUMO. Within this paper, we want to present models enabling SUMO to simulate V2X applications improving traffic efficiency involving EVs and conduct simulations of two applications, namely a preemption and an automated cooperative formation of a rescue lane by IVs. This paper is organized as follows. Section 6.4 describes the simulative environment with all boundary conditions and input parameters. Section 6.5 explains the different implementations. Section 6.6 deals with the calibration of the proposed models. Section 6.7 describes two example applications as well as the simulation and assessment of the models and the example applications. Section 6.8 completes the paper by giving a conclusion and outlook.

6.4 Simulative environment

Material provided by OpenStreetMap is the basis for the traffic system used in this paper. It is shown in Figure 6-1 and includes three urban intersections in Braunschweig⁴, Germany. Apart from this realistic traffic system, a real traffic signal timing plan and a collected traffic census data is the basis for an approximated real traffic flow. Figure 6-2 shows the underlying data

⁴ Intersections from west to east: Rebenring / Pockelsstraße, Rebenring / Hagenring, and Hans-Sommer-Straße / Langer Kamp

of the traffic census. Straight arrows and the corresponding numbers indicate straight traffic whereas angled arrows and corresponding numbers indicate turning traffic (left or right). The percentage share of trucks is 3% with a distribution of semi-trailer trucks (Truck1) and short trucks (Truck2) in a ratio of 1 : 1. The remaining road users are passenger cars divided into three groups in a ratio of 1 : 2 : 1 (Car1 : Car2 : Car3). They differ in vehicle dimensions, maximal speeds, reaction time of the driver, and driver attention. Values for type Car1 are comparable to the vehicles of the A00 segment. Type Car2 represents the A segment, type Car3 equals the B segment, type Truck1 comprises of semitrailer trucks, type Truck2 is a smaller truck, and type EV a fire truck. Values for the maximal acceleration and maximal deceleration consider a comfortable acceleration and are not equal to the maximal physical values. Table 6-1 shows vehicle related parameters and used driver models (minGap, Sigma and Impatience). The table also contains data about the parameters used for the EV.

Table 6-1: Vehicle parameters and driver behaviors

Type	Max. Speed [m/s]	Speed-factor [-]	Max. Accel. [m/s ²]	Max Decel. [m/s ²]	Length [m]	minGap [m]	Sigma [-]	Impatience [-]
Car1	40	0.8	1.9	3.0	3.5	2.00	0.6	0.3
Car2	50	0.95	2.6	3.5	4.2	1.20	0.8	0.5
Car3	60	1.0	3.1	4.0	4.7	0.65	0.8	0.8
Truck1	22	1.0	0.8	3.5	18.4	0.75	0.9	0.7
Truck2	22	1.0	0.8	3.5	12.4	0.75	0.9	0.5
EV	30	1.2	2.5	7.0	12.4	0.5	1	1



Figure 6-1: The simulated traffic system

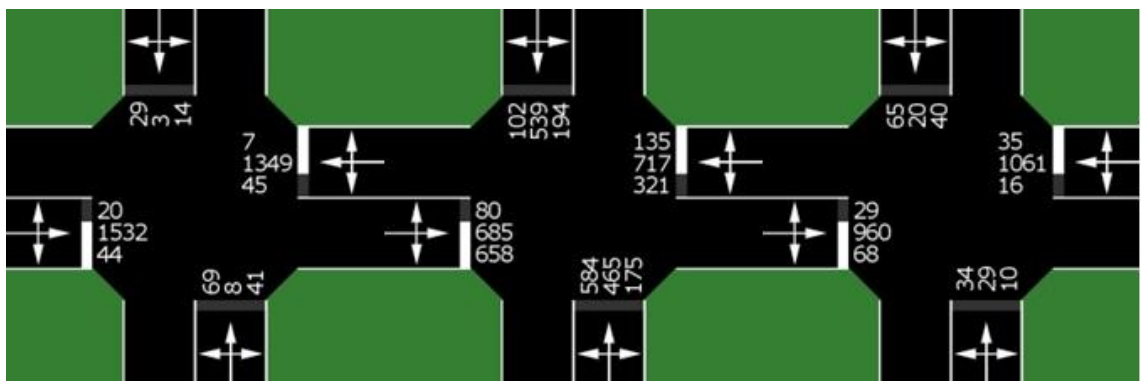


Figure 6-2: Collected data of a traffic census at the relevant intersections during the peak hour

6.5 Models

6.5.1 EV behavior

EVs are allowed to not obey general traffic rules. SUMO is not able to model the necessary behavior of EVs with a predefined internal function. Our implementation concerning the EV's behavior considers speeding and the abilities to drive through red lights. The usage of a siren and a light bar is not visualized within the simulation. However, the effect on the IV is described in Section 6.5.3.

Speeding:

The EV may override speed restrictions by using the implemented speed factor. Table 6-1 shows the maximum speed of the EV (30 m/s) and the speed factor (1.2). By setting the speed factor to a value greater than 1.0 (=100%), the related vehicle is allowed to drive faster than the speed limit. The speed limit is set to 13.8 m/s (equals 50 km/h), as the traffic system is located in an urban environment. Thus, the EV could drive 16.56 m/s ($1.2 \cdot 13.8 \text{ m/s} \approx 16.56 \text{ m/s}$) within the traffic system, if the maximum speed of the EV (30 m/s) is not exceeded.

Drive through red lights:

The TraCI (Traffic Control Interface) enables an enhanced alteration of the EV's behavior. Using this interface, the EV may cross an intersection while having a red light. This is possible by using our method which works as follow. The EV is approaching a red traffic light with full speed. As implemented in the driver model, the EV starts to brake in order to not violate traffic rules. Even if no vehicle is congesting the intersection, the EV will wait until the traffic light switches to green. Figure 6-3 shows a flowchart to of the implemented algorithm which allows an EV to drive through red lights even with obstructing vehicles. First, the algorithm determines the speed and the lane of the EV as well as the signal state of the intersection. Additionally, a minimal and maximal speed value is read from a configuration file which allows modeling a realistic approaching behavior (see Section 6.6). Second, it checks whether the EV is in front of

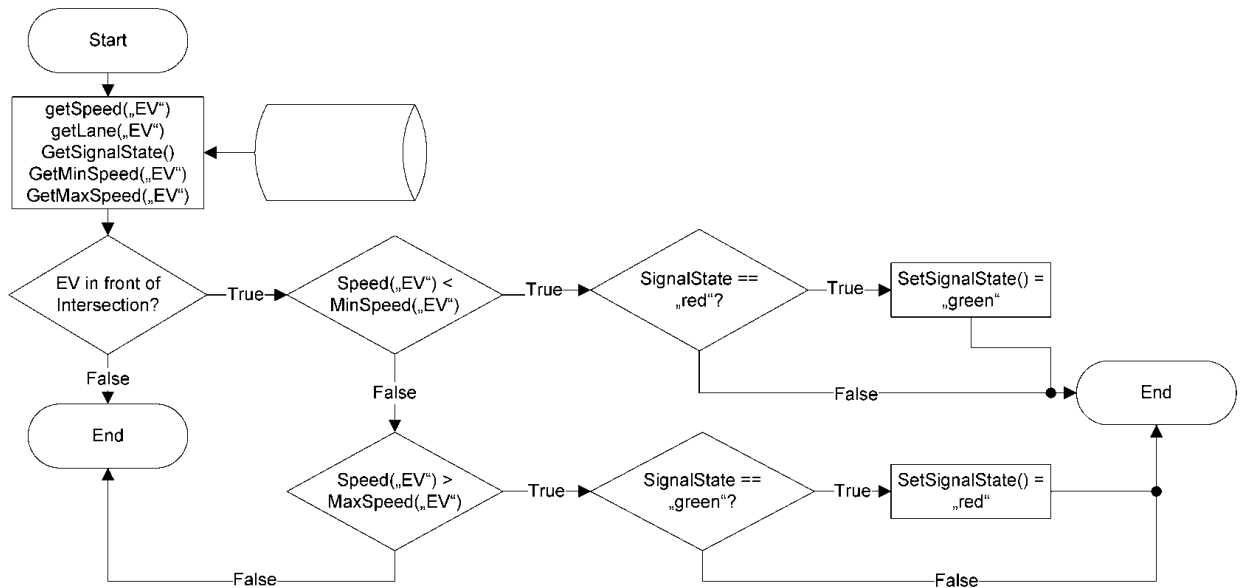


Figure 6-3: Flowchart showing one EV model

an intersection. As a third step, the EV's speed is checked against the minimal speed value and the maximal speed value. If the EV is driving slower than the lower threshold the signal is hold or switched to green. If the EV is driving faster than the upper threshold, the signal is hold or switched to red. This leads to an averaged approaching behavior of the EV which can be observed in real situations with EV approaching intersections. The algorithm is executed every time step in the simulation.

6.5.2 Intelligent infrastructure

New applications require a novel type of infrastructure. Characteristics of a new infrastructure, for instance accessibility by special road users, influence the modulation. We propose a model to interact with traffic infrastructure using TraCI and inductive loops. Inductive loops are a trigger to start an application on the infrastructure, e.g. setting a new traffic signal timing plan. The implemented loops are adapted in a way that they only react to vehicles of the type EV. The distance between the loops and the intersection represents the V2X reception radius.

6.5.3 IV behavior

The implementation of the IVs' behavior is essential, especially in congested traffic situations. Road users respond in a certain way when perceiving a siren or blue light. The most favorable way is to respond in a cooperative manner as discussed in [6]. One possibility to behave cooperatively is described in the Road Traffic Regulations [16] as creating a rescue lane in order to let the EV drive through the congested area quickly. A method to implement such a behavior is presented. An example situation clarifies the functional principle of the method. Figure 6-4 (top) shows an oneway road with three lanes. Ten vehicles drive on that road as an EV s approaching on the middle lane from behind. In this example, the method clears the middle lane by forcing the obstructing vehicles to change the lane and the other vehicles to stay on their lane. It induces a lane change maneuver using the SUMO internal ChangeLane()-function based on the SUMO vehicle dynamics. Figure 6-4 (middle) shows the turn signals

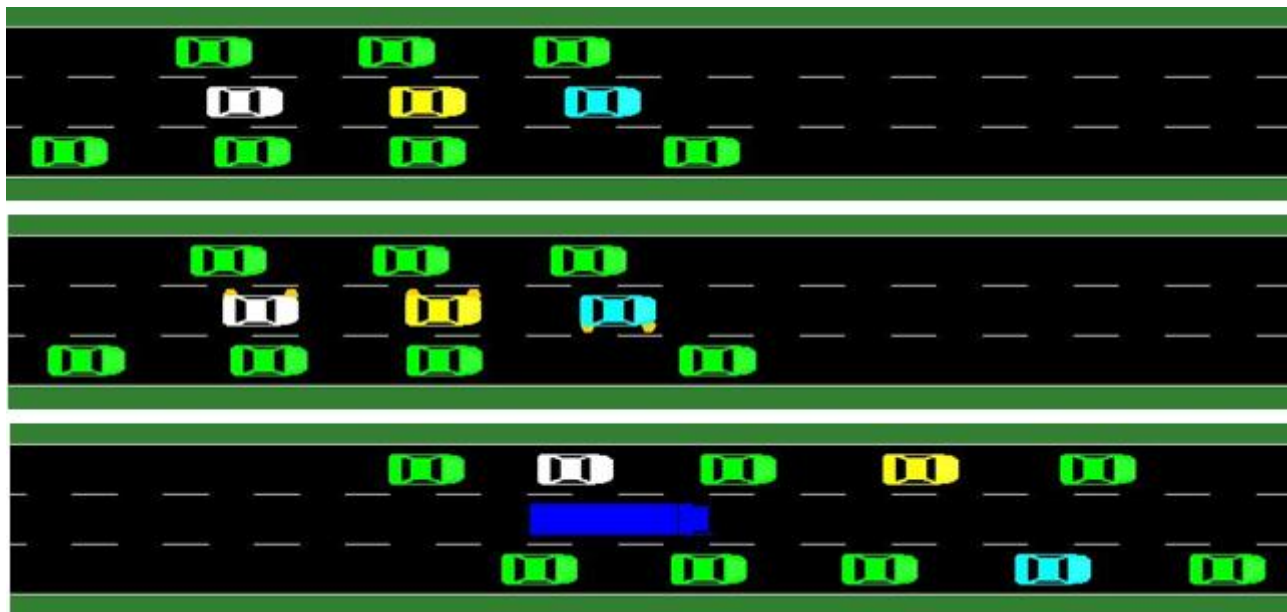


Figure 6-4: Example situation for the IV behavior

indicating a lane change of the obstructing vehicles. The direction of the lane change is random but may be defined. Figure 6-4 (bottom) shows the final rescue lane created by the method. The flowchart in Figure 6-5 shows the algorithm. The algorithm determines the number of vehicles on the EV's lane (amount) and their identification number. After that, a procedure is applied for each vehicle. The speed of the vehicle is determined. Afterwards, a check clarifies if the vehicle entered the EV lane within the last simulation step. If so, a reacting distance is calculated in which the vehicle reacts on the EV's presence (see Section 6.6 for the sub function). If not, the old values are used. The algorithm calculates the distance between the vehicle and the EV. The calculated distance to the EV needs to be lower than the reacting distance and the vehicles speed needs to be higher than 3 m/s to induce a lane change. This procedure is repeated for each vehicle and each time step in the simulation.

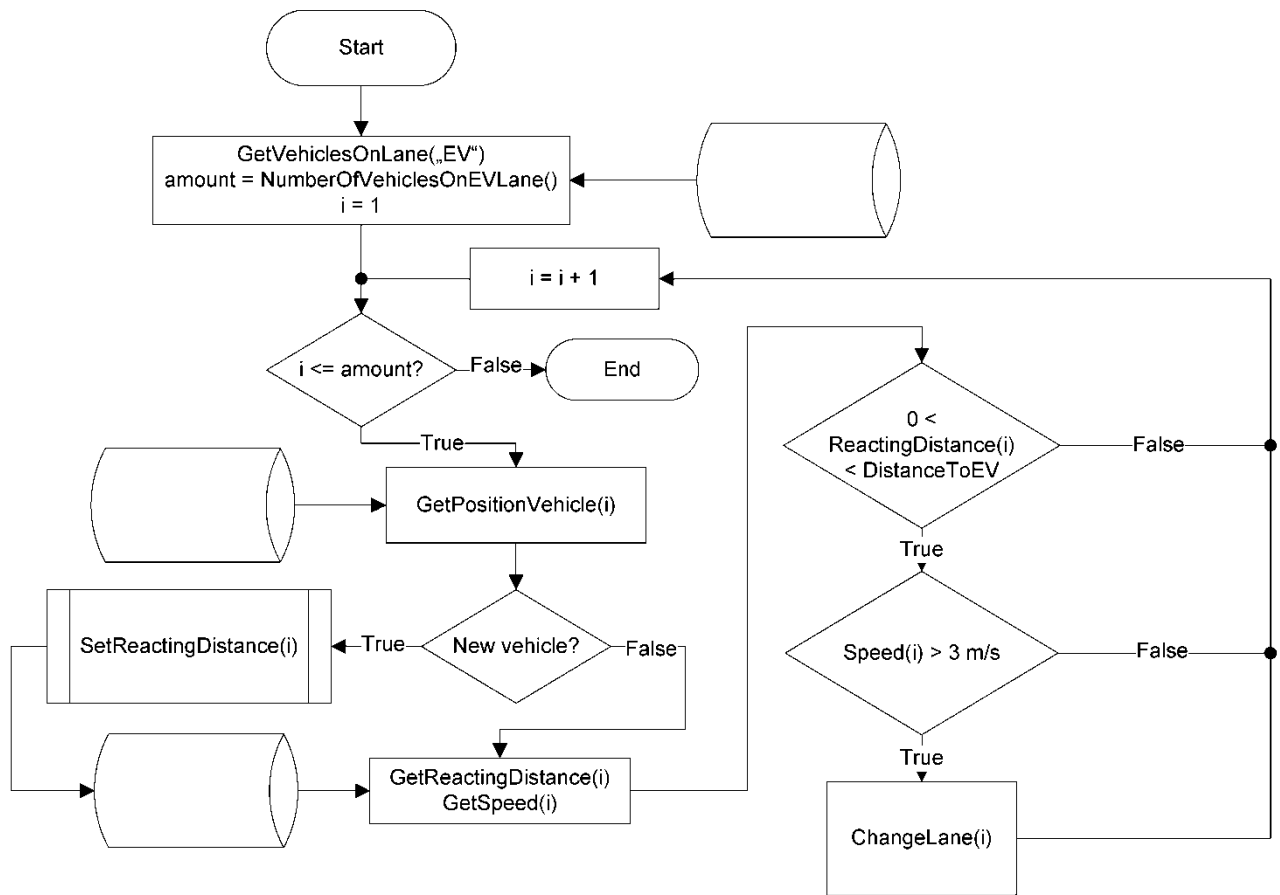


Figure 6-5: Flowchart showing the model “IV behavior”

6.6 Calibration

The calibration of our models implementing EV and IV behavior aims to resemble a realistic behavior. Therefore, different methods are conceivable. For instance, assessments of traffic census comprising missions of EV’s indicate the effect on the EV’s travelling time. Using this data, it is possible to estimate an average time loss. We forego using such a method. First, a traffic census in required dimensions involving EVs does not exist. Second, and more important, an average time loss may not be representative to the scene and ineligible to calibrate the models in required detail. Some intersections and urban roads may cause only little time loss whereas others are a major issue for EV’s travelling time. That is why we focus on a real data analysis using videos at congested urban roads and intersections. The videos reveal the behavior and retarding effects in real situations. This analysis gives several example situations to adjust parameters of the models.

Assessing videos to gain insights of interactions between EVs and IV was also done in Buchenscheit et al. [3]. They mounted a camera on the dashboard of an EV and recorded 21 typical emergency response trips with a total length of 147 minutes. They came to the conclusion that dangerous and/or retarding factors can be condensed to a late perception of the approaching emergency vehicle and a non-optimal switching of traffic lights. Red traffic lights, which occur in 50% of the trips, cause a delay of 15-30 s each. Moreover, on average 2.5 drivers are misbehaving which leads to a loss of 1 minute in average for each trip. As we want to calibrate the proposed models, we need a more detailed analysis. However, we seize

the idea of assessing recorded EV missions. Because data protection laws require a certain protection for people, we decide to assess already declassified videos available of different rescue services in Germany. The selection of video files is based on the following factors:

- The regional rescue service declassified a couple of videos (not only a few). This reduces the risk of extracting unique environmental/traffic impacts and come to flawed conclusions
- The database comprises of different videos of different rescue services. This reduces the risk of adjusting the models according to a regional instruction of EV drivers
- The videos do not include exceptional situations (e.g. missions during natural disasters, educational films).

The video database consists of urban and suburban/rural missions. Necessary parameters such as distances between road users and speeds are either recorded or estimated. The overall length of the video material is 90 minutes with 116 traffic light controlled intersections and a variety of numbers and composition of vehicles and environments. Concerning the traffic lights, 56 times the traffic light was red at the moment of passing whereas it was green in 60 instances. This indicates that the EVs had red in 48% of the times an EV passes a traffic light controlled intersection. Although the EV drivers reduced their speed in these instances dramatically (on average 20 km/h) misbehavior of crossing IV almost leads to two accidents. Additionally, 36 instances showed heedless behavior of IV which leads to critical situations caused by wrong perception of the situation. Concerning the analysis of distance for the noticeable first reaction regarding the EV, the reacting distance is divided into the three clusters: "50 m and more", "50m – 20 m", and "20 m and less". Depending on the environment and the perception of the EV's presence, circa 25% of the IVs react in a distance of 50 m and more. Around 50% of the IV's drivers react in a distance between 50 m and 20 m, whereas 25% of the drivers react in a distance of 20 m and less. However, the time to form a rescue lane is strongly depended on the traffic density. This is why the model is adjusted concerning the distance for first reaction and not the time for successfully creating a rescue lane. With this analysis, the models can be calibrated. The average speed for the EV is about 20 km/h. Hence, the upper speed limit is set to 7 m/s whereas the lower speed limit is set to 4 m/s (see Figure 6-3). Thus, the average speed equals 5.5 m/s (=19.8 km/h).

The IV reaction model is calibrated according to the estimated values which are used as shown in the flowchart in Figure 6-6. The discovered distribution over the obtained reacting distances is modelled by a random, uniformly distributed float generator.

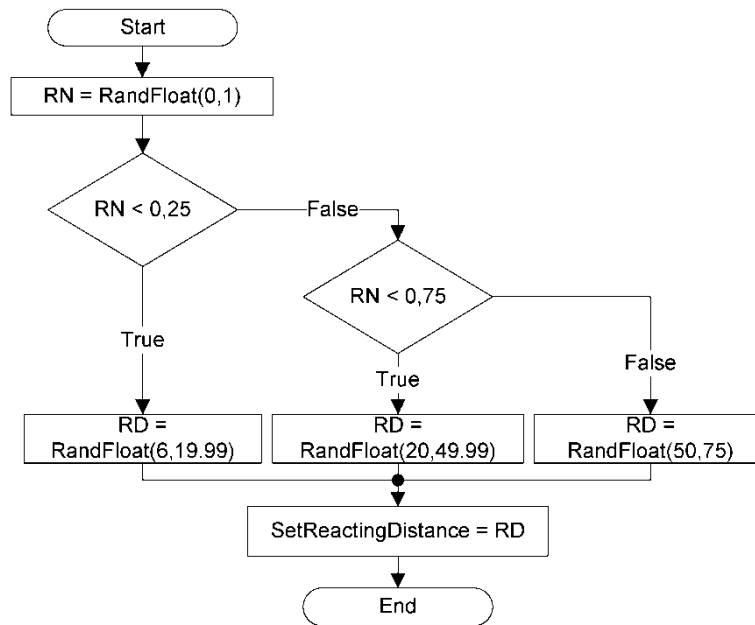


Figure 6-6: Flowchart to calibrate the IV behavior model (SetReactingDistance)

6.7 Simulation of the Models and V2X-Applications

We present an intelligent traffic infrastructure and near realistic behavior of IV and EV to enable tests of different V2X applications related to EVs. We conduct a simulation with the normal SUMO models and a simulation with the proposed models to show the differences. Afterwards, we conduct simulations for two applications: preemption system and an automated formation of a rescue lane. The next two subsections describe the functionality of the apps. Afterwards, the simulation procedure and the assessment are presented.

6.7.1 Preemption

Preemption is a technical system that enables an EV to register its arrival at a traffic light regulated intersection. A special infrastructure at the intersection runs the necessary application. This application switches to a special phase program that allows the EV to pass while having green. The approaching EV triggers the system by sending a V2I message. The principle is shown in Figure 6-7. The algorithm determines if an EV-Preemption program is active. If not, it checks for a request at the starting induction loop. When an EV triggers the loop, the signal program and signal phase is determined. Depending on the current program and phase, the algorithm chooses a suitable, German Guidelines for Traffic Signals (RiLSA) [15] conform, predefined EV-Preemption program and starts a timer. If an EV-Preemption is active, the algorithm checks whether the ending induction loop is triggered by the EV or the maximal timespan has expired. There are two factors that influence the success of the preemption system.

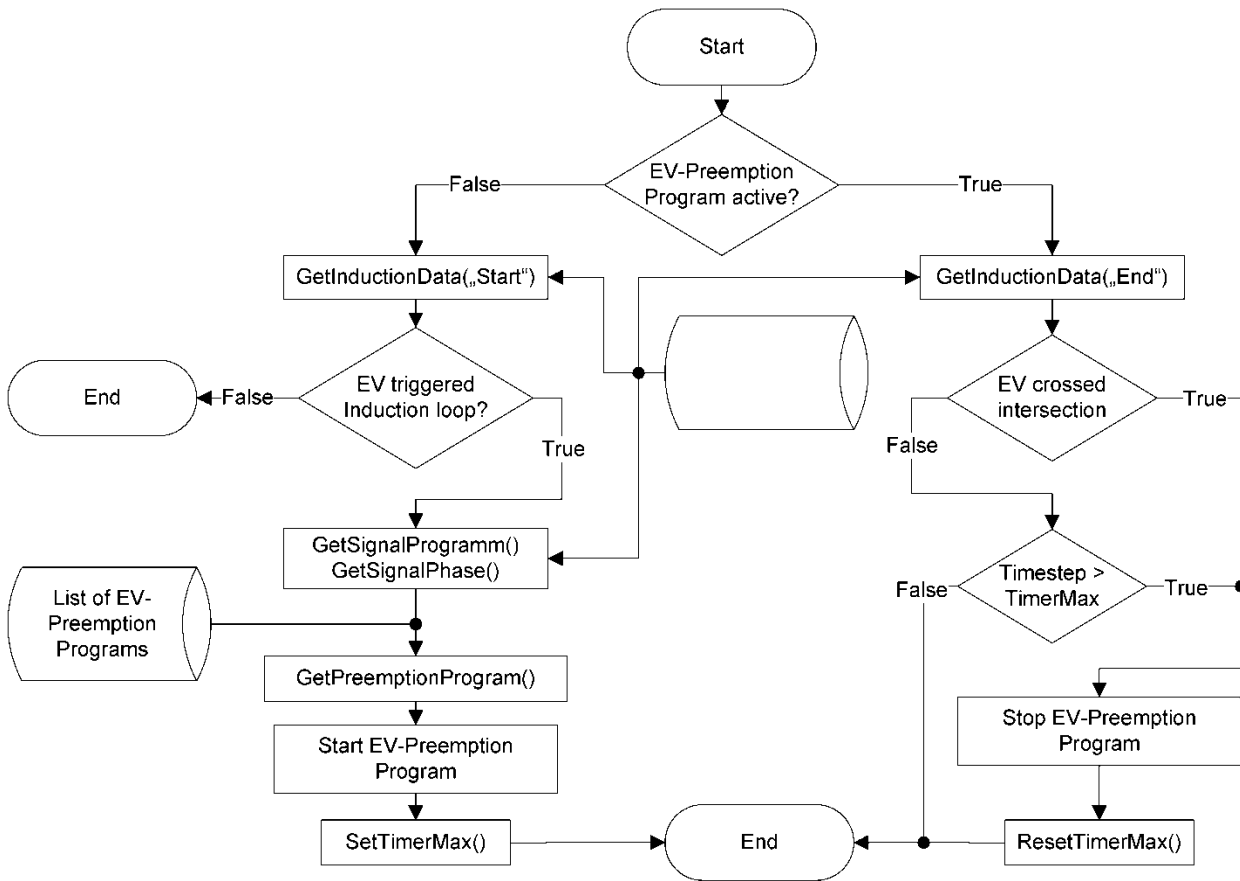


Figure 6-7: Flowchart of the traffic light preemption

First, the moment of registration at the infrastructure influences the possibilities to switch the signal phase according to the RiLSA. Second, the communication distance depending on the intersection's topology and environmental message signal attenuation. In general, there are two initial states when the preemption request is sent: the EVs traffic light shows green or yellow/red. While having green, the RiLSA defines that a green light may be held as long as the other directions do not have a red light for more than 3 minutes. When the traffic light is yellow or red, the phase program is shifted to a special phase program at the next possible moment. This also leads to the maximum requirement for the communication distance. When the EV is having a red light, under certain circumstances, the RiLSA standards require a secure time to shift the phase program. The distance between registration and the intersection must be great enough to allow the EV driving as fast as possible while the phase shift takes place. To ensure a lawful traffic light program switching behavior, our method obeys restrictions based on the RiLSA. By observing the guidelines, our method does consider pedestrians implicitly. By simulating this application, the dependency between communication distance, the signal phase shift and the success of the preemption for the investigated intersection may be found. Therefore, the two effects are varied. The communication distance can be varied. In this paper, it is set to a distance of (west to east) 165 m, 450 m, 165 m and realized by using induction loops. The cycle second in which the EV is approaching, is also varied over the whole phase time. The travelling time of the EV is then assessed and compared. The goal is to minimize the travelling time of the EV in urban environment. A secondary goal is to minimize the impact on the IV. Therefore, different special phase plans may help solving traffic jams and congestions while and after a preemption system is active.

6.7.2 Automated Formation of a Rescue Lane

The second application is a system that supports drivers to automatically form a rescue lane. By doing so, it makes the vehicles behavior cooperatively according to an operationalization of cooperative behavior as shown in [6]. To apply the aforementioned concept of cooperative behavior, both the EV and the IV require a cost function. In this elementary assessment, the EV wants to pass through this area as quickly as possible, meaning that the cost increases when the travelling time increases. The IV wants to let the EV pass by, which results in a cost function that also increases when the EV has to wait longer. This means that every reaction of the IV improving the travelling time of the EV is a cooperative behavior. A conceptual system assisting drivers forming a rescue lane by providing additional information can be found in the literature [3]. Depending on the system, it can give additional information and thus assists the driver, give advice or induce maneuvers itself. V2X communication enables sharing necessary information. The information itself needs to meet two requirements: First, the obstructing vehicles know that they are blocking the EV and get helpful information on how to solve that issue. Second, the information needs to be consistent among different IVs. Determining a cooperative coordination among IVs can be obtained by using different methods. In this application, a rule based approach is employed. Figure 6-5 shows the implemented algorithm to model IV behavior. The automated formation of a rescue lane is based on this flowchart, but the SetReactionDistance function is set to a constant value of 150 m. The application has one master and several slaves. The master with the implemented rules runs on the EV, determining what the IVs have to do. The slave instances are running on the IVs which send ego information (i.e. own lane, own position) to the master. Additionally it is assumed that the slaves execute the commands sent by the master without sending an acknowledgement.

6.7.3 Simulation Procedure

The simulation procedure describes configurations of executed simulations. Table 6-3 shows employed models and applications for different runs of the simulation. Each test is performed 50 times to take randomized effects into account and ends when the EV reaches a specific point at the end of the simulation. Models are implementations as discussed in section 6.5. The two applications are employed as shown in section 6.7.1 and 6.7.2.

Table 6-2: Setup of the different tests

No	Speeding	Driving through red lights	IV reaction	Preemption	Autom. Rescue Lane
M 1	X				
M 2	X	X	X		
A 1	X	X	X	X	
A 2	X			X	
A 3	X	X	X		X
A 4	X	X	X	X	X

The simulation runs for 500 seconds without modification. After that, the EV gets in the simulation. The moment the EV enters the simulation is varied from the 500th second in steps

of 1 second to the 585th second. The timespan of 85 seconds matches the timespan of one phase shift for all three intersections. The first test has only the speeding model activated to show the travelling time of an EV as implemented in SUMO. The second test includes the other proposed models to show the difference in travelling time. These simulations are assessed to decide which setup is the reference for the applications. Afterwards, the applications are simulated and evaluated. Test A1 includes the first application, which is a traffic light preemption system. Test A2 shows the effects on the travelling time when the models driving through red lights and IV reaction are not activated. The last two tests include the other application, the automated formation of a rescue lane. A3 simulates the application alone whereas A4 includes both applications. The figures introduced in the following discussion have the following properties. The x-axis denotes the introduction second in which the EV entered the simulation whereas the y-axis indicates the travelling time of the EV through the traffic system. The gray area marks the range of values obtained during the 50 simulations. The dashed line represents the median of travelling times in order to classify the resulting range of the travelling time. The solid line is a trendline to illustrate the general course of the travelling times over the introduction second.

6.7.4 Assessment of models

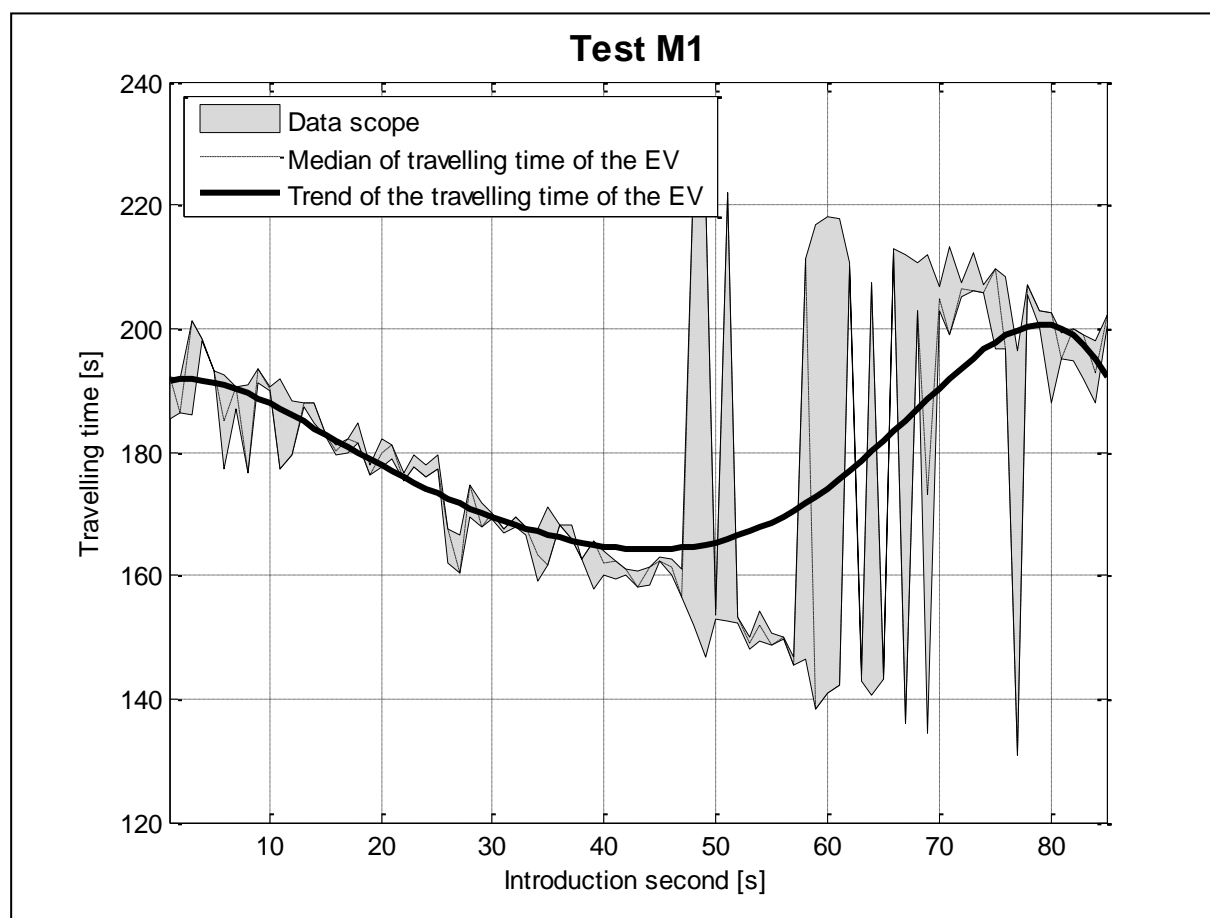


Figure 6-8: Travelling time in test M1

Figure 6-8 shows the travelling time of an EV within simulation M1. None of the presented models are activated, so that the EV behaves as a normal road user. Only the speeding

method is activated as this can be modeled by SUMO itself. The travelling time is not constant over the introduction second. This variance is caused by randomized behavior of IVs. This leads to direct interference (e.g. changing the lane very late) and indirect interference (e.g. that the EV is slowed down so that it does not arrive within the green phase at the next traffic light). At small introduction seconds, the first traffic light is red and switches to green. This leads to a delay of the EV due to waiting vehicles and causes a travelling time around 190 s. These vehicles have more time to start with increasing introduction time of the EV. Around introduction second 43, a green wave is established with a travelling time of around 165 s. After that introduction second, the travelling time is increasing and has its maximum value (about 200 s) around introduction second 78. Considering the data scope, noticeable differences around introduction second 50, 60, and 70 can be observed. These introduction seconds are mainly affected by waiting vehicles as the green wave breaks down and the EV has to wait for one complete cycle to cross at least one of the intersections.

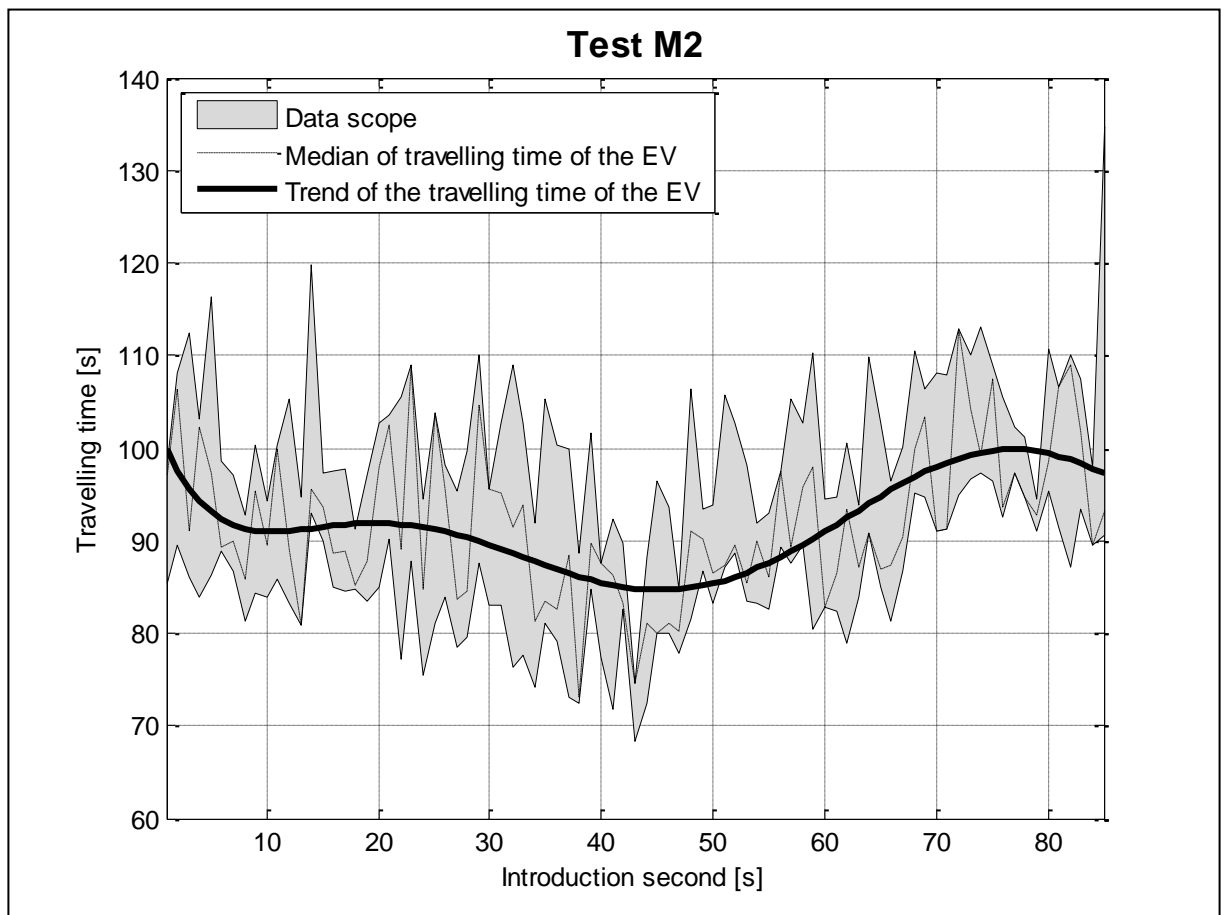


Figure 6-9: Travelling time in test M2

Figure 6-9 shows the simulation M2 with all three models activated. It shows that the travelling time depends on the introduction second. Moreover, there are different values for one introduction second. This distribution of values is also caused by randomized behavior of IVs. The trendline reveals four distinct areas that can be interpreted as follows. At introduction second 8 and travelling time around 100 s, the EV approaches the first intersection while having a green light. The vehicles in line are already moving and are slowly clearing the path. In comparison with introduction seconds smaller than 8, the EV is approaching in an advantageous moment, because the waiting vehicles have some time to start. The second

traffic light is red and some vehicles are waiting in line. If trucks are waiting on the EV's lane, the travelling time is severely affected (as indicated by the gray area). In introduction second 8, the vehicles waiting in front of the second traffic light can clear the lane quick enough so that the EV reaches the third traffic light at green. However, around introduction second 23, the IV interferes the movements of the EV in a way that it reaches the third traffic light while having red. This explains the local maximum of about 90 s around introduction second 23. The global minimum of 85 s around introduction second 43 can be explained as the optimal entrance second to catch the green wave. This introduction second is not influenced by variations of IV behavior as the data scope is relatively small. This can be explained by the circumstance that vehicles which are located at the intersections may start early enough to clear the route when the EV arrives. After that introduction second, the green wave gets interrupted easily by obstructing vehicles. Considering the median graph, introduction seconds 23, 28, 72, 75 and 82 have a relatively high travelling time (about 100 s) for the EV. At least one traffic light is red with waiting and obstructing vehicles. This leads to a delay so that at least one other traffic light after it changed red to green and thus also obstructs the EV.

The comparison of the two simulations M1 and M2 shows that the travelling time of the EV is reduced by half by using the EV and IV models. Using the first simulation results as a reference, applications improving the EV's travelling time would be overestimated as the behavior of the IV and EV is neglected. Additionally, the peaks at introduction seconds 50, 60 and 70 can be considerably reduced. This is mainly caused by the EV model that allows the EV to drive through red lights. Therefore, it does not have to wait for one cycle in order to pass the intersection. In the following, the results of test M2 are used as a near realistic reference to estimate the potential of EV applications.

6.7.5 Assessment of the applications

Figure 6-10 shows the travelling time over the introduction second for test A1. The course of the trendline drops from a value of around 85 seconds at introduction second 1 to a minimum of 70 seconds around introduction second 45. Afterwards it rises to a travelling time around 95 seconds at introduction second 78. Then it decreases again. The maximum at small introduction seconds can be explained by the late preemption at the first traffic light. The congested intersection cannot be cleared in time so that the EV has to wait. Between introduction seconds 13 to 55, the preemption comes in time to either hold the green phase or change the red/yellow traffic light to green. The variances are caused by IVs randomly merging into the EVs lane causing its delay. Taking the median values into account, introduction seconds 28, 31, 43, 64, and 65 are an advantageous moment for the preemption so that the travelling time is less than 63 seconds. Introduction second 65 shows that the randomized IV behavior does not necessarily have an effect of the EV's travelling time as the maximum and minimum values of the 50 runs are in a range of 2 seconds.

Overall, the travelling time can be reduced by using a traffic light preemption system. The trendline indicates that the EV is faster for all introduction seconds compared to results of test M2. The difference in travelling time is about 15 seconds for all introduction seconds. Additionally, a more detailed calibration of the traffic light preemption can take place. The influences of parameters such as i.e. distance between the induction loops and the intersections, congestion in front of the intersection, or speed of participants need to be

analyzed to adjust a better system behavior. This study is out of scope of this paper but will be addressed in further work.

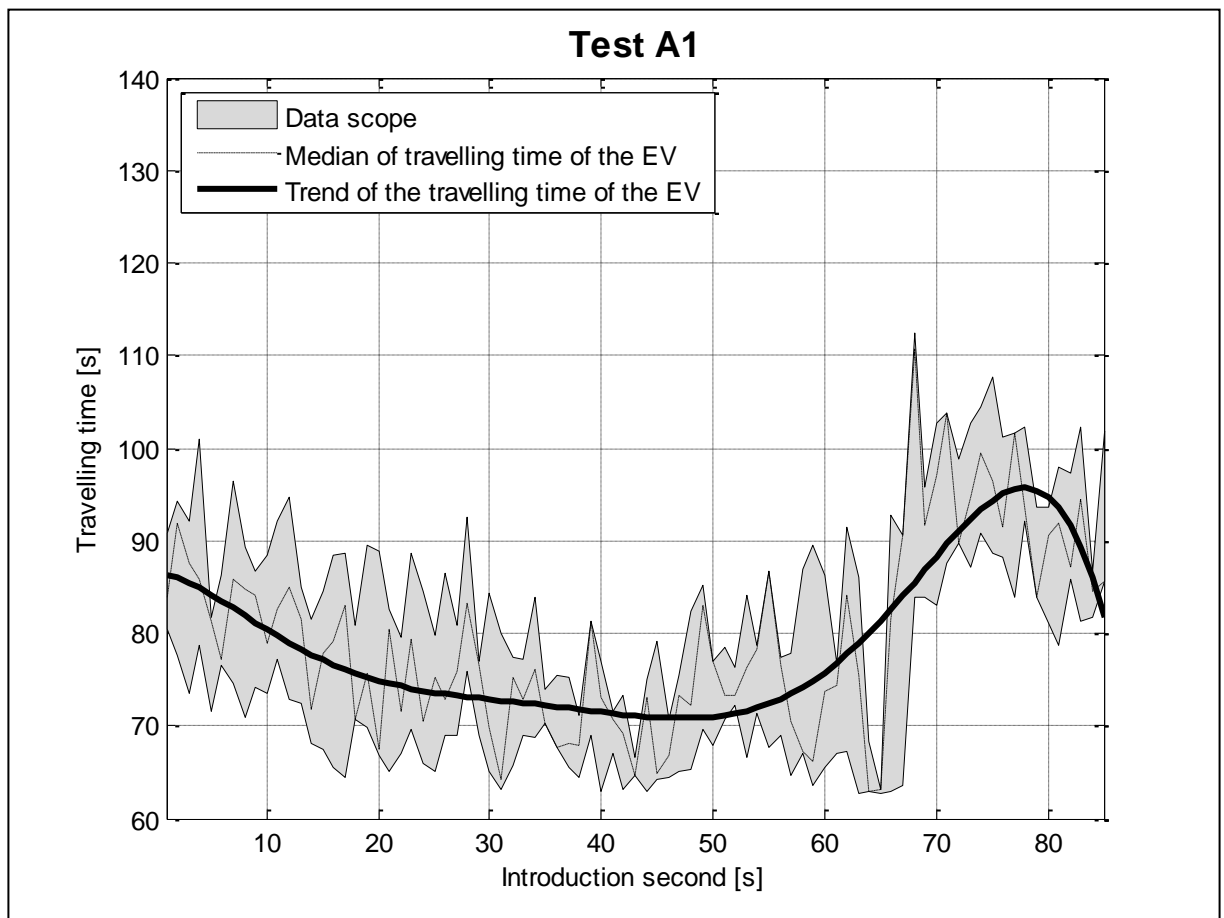


Figure 6-10: Travelling time in test A1

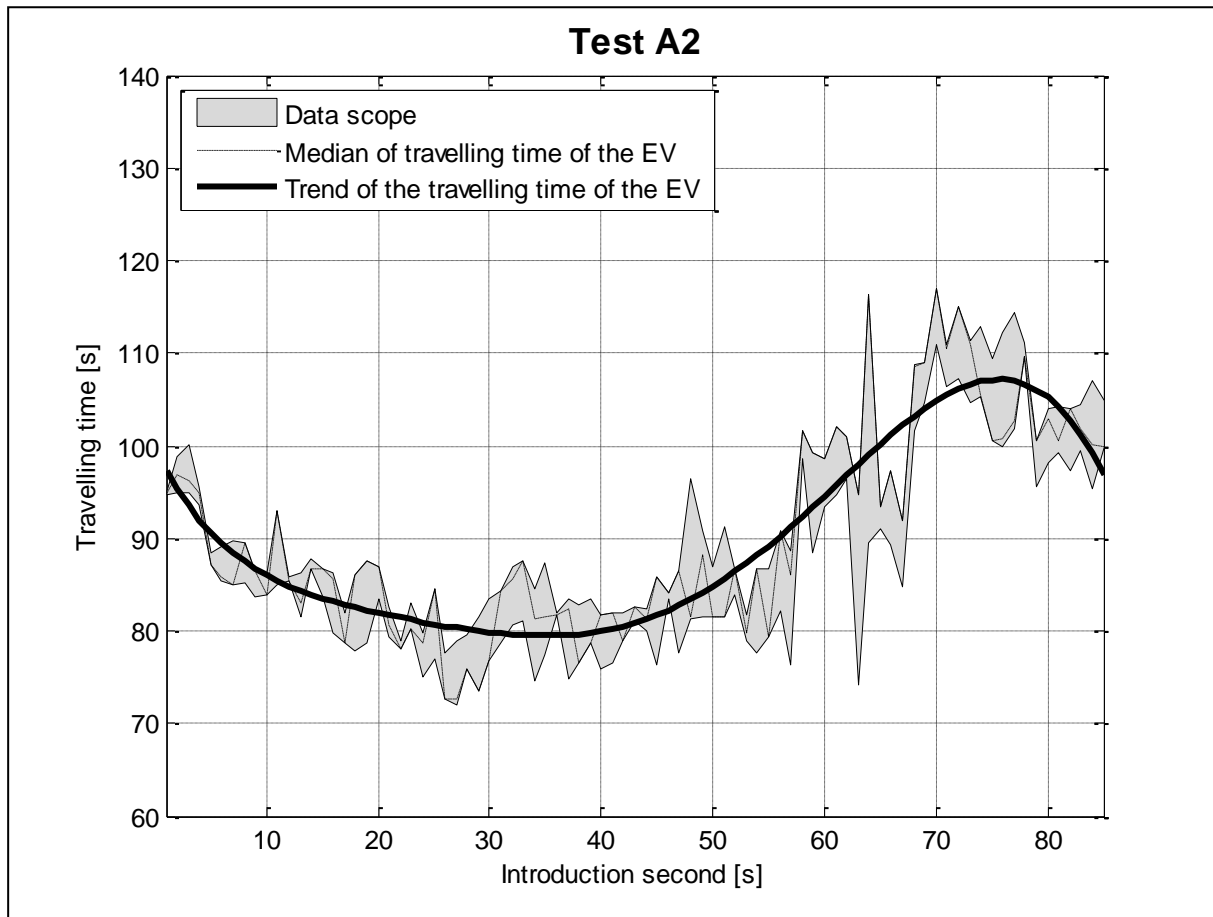


Figure 6-11: Travelling time in test A2

The travelling time of test A2 is represented by the graph in Figure 6-11. The gray area is not very distinct as the IV behavior and the drive through red lights models are deactivated. Thus, it is comparable to the simulation M1 with the difference that the preemption system is activated. The course of the trendline starts at a travelling time of about 95 seconds, then declines to a minimum around 80 seconds at introduction second 36 and eventually rises to a maximum of 105 seconds around introduction second 75. The dropping course of the trendline can be explainable by the EV encountering delays due to waiting vehicles at the intersections. The preemption allows vehicles to clear the intersection earlier. For later introduction seconds, the preemption gets more efficient by holding the green phases. After a minimum around introduction second 35, the EV reaches the first traffic light while it changes to red. The preemption takes some time and does not switch to green fast enough which causes the EV to slow down. As none of the two models IV reaction and drive through red light is active, IVs do not clear the lane while waiting at a red traffic light to let the EV drive through. Additionally, they stay in the EV's lane and obstruct its mission until they voluntary change the lane. The maxima at introduction second 64 and 68-79 are caused by this effect at different or multiple intersections.

The results indicate that the travelling time is dramatically reduced compared to test M1. This indicates that studies comparing a preemption system to a reference situation without considering IV reaction and the EV behavior, vastly overestimate the potential of a preemption system.

Figure 6-12 shows the travelling time with the second application, namely the automated formation of a rescue lane. Taking a look at the trendline, the graph drops until introduction 70

second 8, then increases until introduction second 23, declines until second 43, rises until second 71 and decreases again. Starting from introduction second 1 to 8, the first traffic light switches green, but vehicles have slightly more time to start and clear the lane for the EV. On the EV's route between the first and the second traffic light, all vehicles clear the lane for the EV. However, these change maneuvers are the reason for the high travelling times around introduction second 23. Vehicles are forced to move to the neighboring lane despite their desire to turn at the next intersection. This leads to situations in which vehicles in a distance of e.g. 140 m receive the command to leave the EV's lane although they would not interfere with the EV. Eventually, to reach their destination, the area in front of the second intersection becomes a highly congested area. This congestion also affects the EV's travelling time. Simulations at introduction second 38 and 43 show that by avoiding this effect, the travelling time can be significantly reduced. The first traffic light is green, but depending on the congestion and phase in front of the second traffic light, the EV may reach the third traffic light while having green as well. This green wave allows the EV to drive through the traffic system very fast. The maximum starting around introduction second 64 is a consequence of the red light at the intersections. The EV enters the simulation and reaches the first traffic light while having red. The waiting vehicles start but they also need to change the lane in order to let the EV drive in the direction of the second intersection. The second traffic light is also red, which means that waiting vehicles also obstruct the EV's way. After the maximum, the first traffic light switches to green again. This state is comparable with the first introduction seconds.

Comparing these results with the reference M2, the plots are very similar. However, the fact that some lane change commands lead to a congested second intersection reduce the potential of the application. A constant value to apply an automated lane change may affect the travelling time of the EV negatively. That means that a rule based application needs smart rules. Using a distance alone is not enough. A dynamic set of rules is a possible enhancement. For instance, the IV can be forced to slow down to reach their individual goals after the EV drove away. Additionally, the lane change of other vehicles to the EV's lane should be prohibited. Even if the vehicle tries to immediately leave the EV's lane after it changed to it, the EV still needs to slow down.

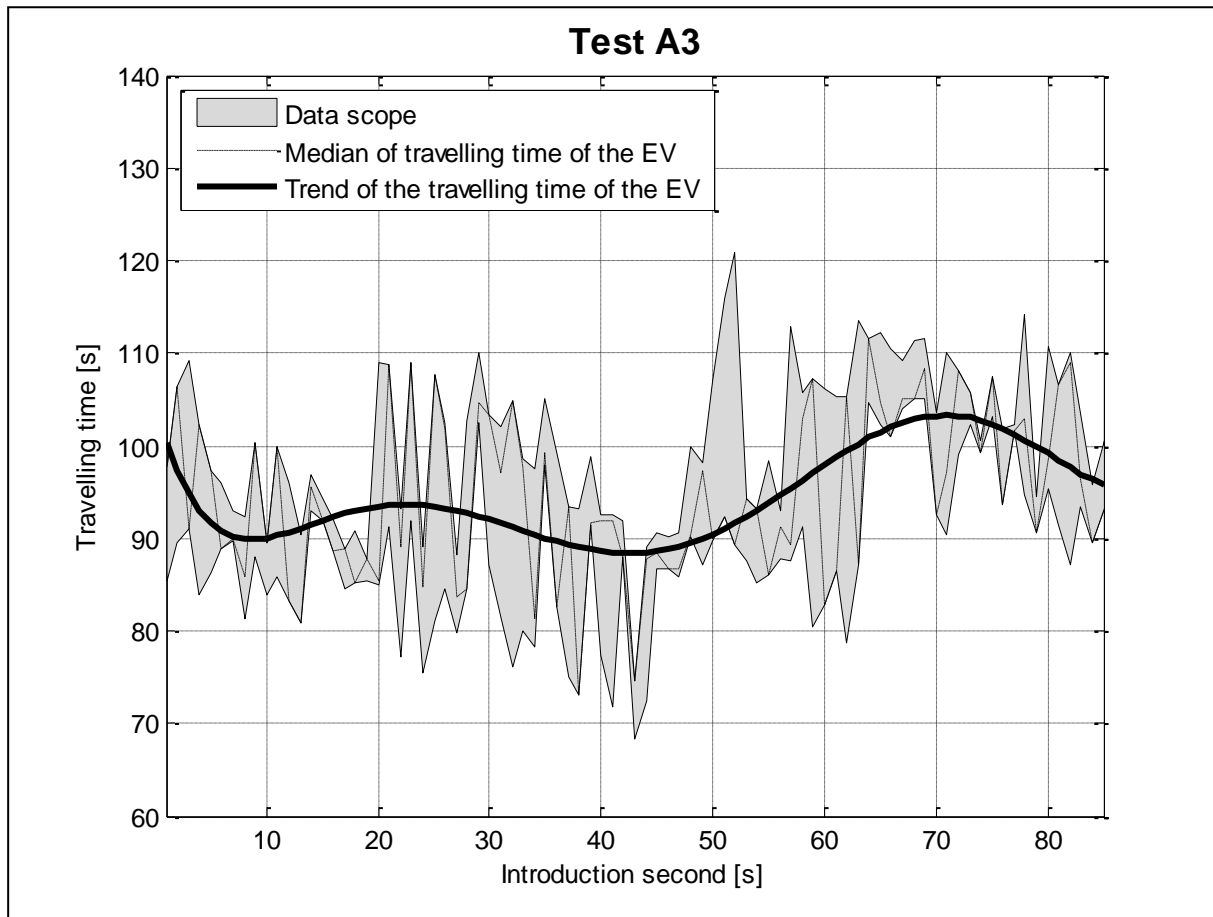


Figure 6-12: Travelling time in test A3

Figure 6-13 shows the travelling time of the EV in test A4. The trendline declines from a travelling time of 90 seconds at the first introduction second to a local minimum of 74 seconds between introduction seconds 22 and 36. After that, the travelling time decreases again to a minimum of less than 70 seconds around introduction second 50. The course of the trendline rises rapidly to a maximum of 95 seconds around introduction second 78 and then declines to 80 seconds of travelling time at introduction second 85. The reason for high travelling times at small introduction seconds is that the preemption is started too late at the first intersection. The vehicles already waiting in front of the traffic light cannot be moved by the automated application. They need to wait for the preemption and clear the lane for the EV. With rising introduction seconds, the effect's impact is decreasing, because the vehicles have more time to start and thus clear the lane. After introduction second 13, the first traffic light is green and the IVs have enough time to start with stopping the EV at all. The comparatively high travelling time at introduction second 21 is due to misbehavior of IV as they change to the EV's lane and thus obstructing it in front of the traffic light. This behavior is caused by the constant reaction distance influencing the IVs since the algorithm does not obey individual goals. The vehicles are forced to change to e.g. the left lane although they want to make a turn to the right at the next intersection. Minimum values of the solid line can be found at various introduction seconds. For instance introduction second 24, 27, 30, 38, 43 and 65 show a short travelling time. These introduction seconds are advantageous for the EV, because green phases may be hold by the preemption system and IVs have enough time to clear the lane without congesting any of the three intersections. The high travelling times at introduction seconds 68-83 due to congestion which is not solved with the

preemption. The distance between the induction loops and the intersections are too short to make the vehicles disappear, especially at the first and the third intersection. At the second intersection, the preemption system has a range of 450 m which is enough for that intersection.

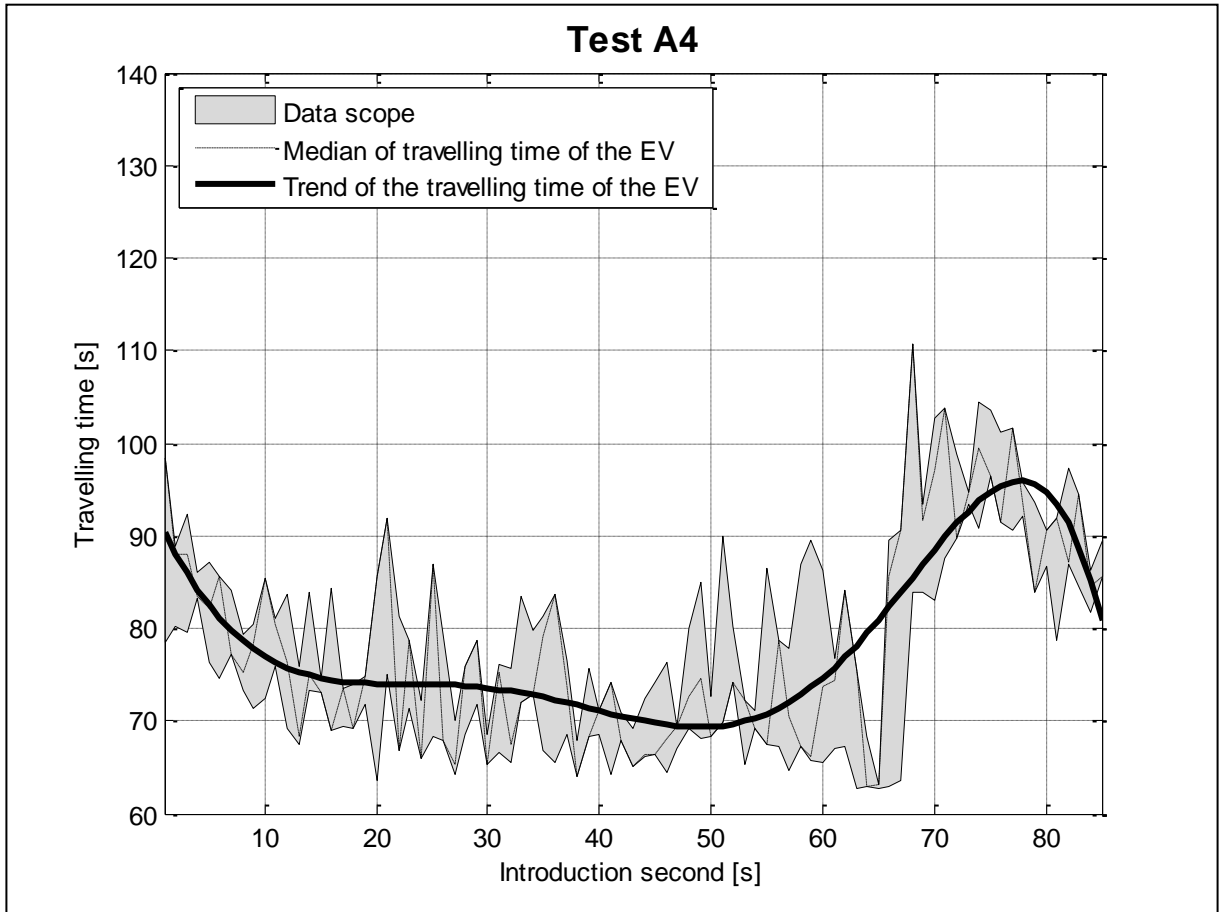


Figure 6-13: Travelling time in test A4

Comparing these results to the reference M2, the trendline of this test is about 10-15 seconds better. The two applications allow the EV to quickly pass through the simulation. The gray area, describing the range of travelling times, is smaller as the IV behavior is controlled by the automated lane change. Comparing to the traffic light preemption only, the results of this application indicates a smaller gray area. Comparing these results the with automated lane change only, the combination of both applications is much better. This indicates that the IVs are obstructing the EV mostly at intersections with slow speeds in which the automated formation of a rescue lane application does not work. A combination of a traffic light preemption system and an automated formation of a lane change integrate the advantages of both systems to support the EV reaching its destination as fast as possible.

6.8 Conclusion and Outlook

Within this paper, we showed that emergency vehicles (EVs) encounter a higher risk of getting involved in accidents during their missions. To support EVs, new applications may be developed and tested with the issue that such prototypes cannot be tested in real life traffic

systems. Simulations however, are a suitable tool to do so. We decided to use SUMO as it is applicable to simulate V2X applications improving traffic efficiency. Yet, SUMO does not feature necessary models such as a realistic EV behavior, enhanced infrastructure, and realistic individual vehicle (IV) behavior responding to EVs. In addition, research community disagrees whether these effects have to be modelled to assess applications regarding EVs. We created a traffic system based on a real traffic system in Braunschweig, Germany. The traffic flow within this net was based on traffic census data during peak hour. We presented models regarding the three road users EV, infrastructure, and IV. The EV model implements speeding and driving through red lights. The infrastructure model consists of an interface for access by special road users (in this case EVs) to initiate infrastructure based applications. The IV model implements a response behavior to the EV. A calibration of these models took place. We showed that a realistic reference scenario is needed to not overestimate the potential of new applications. Finally, we simulated two applications, namely a traffic light preemption system and an automated formation of a rescue lane via V2X. By assessing the simulations, we showed that neglecting aspects of EV or IV behavior leads to different travelling times of the EVs. Concerning the applications it can be stated that a preemption system reduces the travelling times of the EV compared to a reference travelling time. The automated formation of a rescue lane does not necessarily reduce the travelling time as a consequence of various factors. However, a combination of both applications has the potential to support an EV on its mission best by allowing the EV to pass through congested and traffic light controlled urban environments quickly.

As the field of simulating EVs in urban environments is very important but only little investigated, further work needs to be done. As shown, realistic models are necessary to estimate the potential of EV applications. Hence, a wider calibration and validation for the proposed models may take place, e.g. by driver studies or suitable traffic data. We also want to investigate additional aspects that are not yet covered by our models. Some areas to mention are: realistic delays in road users' starting behavior (e.g. shown in [2]), IV behavior receiving multiple requests of EVs to form a rescue lane, occurrences of critical situations while forming a lane, and misbehavior and the consequences for the travelling time. Concerning the early stage of exemplary V2X applications, further research needs to be conducted as well. For the preemption system, a study concerning parameters such as communication distance, phase program, and communication requirements and their effect on the travelling time is necessary. The automated formation of a rescue lane needs to be further investigated as this application is just in an early stage of development. Challenges regarding penetration rate, communication requirements, and security need to be addressed as well as more enhanced methods to determine intelligent cooperative maneuver combinations for the IVs. Instead of static using rules, other maneuver planning methods can calculate a better behavior of the IVs by considering the IVs' individual goals.

6.9 References

- [1] L. Bieker. Emergency Vehicle Prioritization using Vehicle-to-Infrastructure Communication. Tech. rep. German Aerospace Center - Institute of Transportation Systems, 2011.

- [2] D. Bosserhoff. "Handbuch für Verkehrssicherheit und Verkehrstechnik der Hessischen Straßen- und Verkehrsverwaltung". In: ed. by J. Follmann. Forschungsgesellschaft für Straßen- und Verkehrswesen (FGSV), 2010. Chap. Knotenpunkte mit Lichtsignalanlagen, pp. 4.5–1 –4.5.131.
- [3] A. Buchenscheit et al. "A VANET-based emergency vehicle warning system". In: Vehicular Networking Conference (VNC), 2009 IEEE. 2009, pp. 1–8.
- [4] J. Bycraft. Green Light Pre-emption of Traffic Signals for Emergency Vehicles. Tech. rep. City of Richmond Traffic Signal Control System, 2000.
- [5] M. Cetin and C.A. Jordan. "Making way for emergency vehicles at oversaturated signals under vehicle-to-vehicle communications". In: Vehicular Electronics and Safety (ICVES), 2012 IEEE International Conference on. 2012, pp. 279–284.
- [6] M. Düring and P. Pascheka. "Cooperative Decentralized Decision Making for Conflict Resolution among Autonomous Agents". In: IEEE International Symposium on Innovations in Intelligent Systems and Applications. 2014.
- [7] A.S. Eltayeb, H.O. Almubarak, and T.A. Attia. "A GPS based traffic light pre-emption control system for emergency vehicles". In: Computing, Electrical and Electronics Engineering (ICCEEE), 2013 International Conference on. 2013, pp. 724–729.
- [8] C. Kahn, R. Pirrallo, and E. Kuhn. "Characteristics of Fatal Ambulance Crashes in the United States: An 11-Year retrospective Analysis". In: Prehospital Emergency Care 5 (2001), pp. 261–269.
- [9] D. Krajzewicz et al. "Recent Development and Applications of SUMO - Simulation of Urban MObility". In: International Journal On Advances in Systems and Measurements. International Journal On Advances in Systems and Measurements 5.3&4 (2012), pp. 128–138.
- [10] G. McHale and J. Collura. Improving Emergency Vehicle Traffic Signal Priority System Assessment Methodologies. Tech. rep. Federal Highway Administration and Virginia Polytechnic Institute & State University, 2002.
- [11] D. Müller. Typische Gefahren bei Einsatzfahrten des Rettungsdienstes. Tech. rep. Institut für Verkehrsrecht und Verkehrsverhalten Bautzen, 2007.
- [12] Bundesanstalt für Strassenwesen. "Leistungen des Rettungsdienstes 2008/09". In: Berichte der Bundesanstalt für Straßenwesen - Mensch und Sicherheit 217 (2011).
- [13] Traffic Signal Preemption for Emergency Vehicles. Tech. rep. U.S. Department of Transportation, 2006.
- [14] Traffic Signal Priority Control for emergency vehicle preemption. Tech. rep. Global Traffic Technologies, LLC, 2011.
- [15] Arbeitsgruppe Verkehrsmanagement, ed. Richtlinien für Lichtsignalanlagen. Forschungsgesellschaft für Straßen- und Verkehrswesen (FGSV) Verlag, 2010.
- [16] Bundesanzeiger Verlag. Verordnung zur Neufassung der Straßenverkehrsordnung (StVO). Bundesministerium der Justiz, 2013.
- [17] "Verordnung über die Bemessung des Bedarfs an Einrichtungen des Rettungsdienstes (BedarfVO-RettD)." In: Niedersachs. GVBl. (1993), p. 1.
- [18] L. Zhang et al. Simulation Modeling and Application with Emergency Vehicle Presence in CORSIM. Tech. rep. Mississippi State University and Turner- Fair Bank Highway Research Center, 2010.

7 Lane-Changing Model in SUMO

Jakob Erdmann

German Aerospace Center , Rutherfordstraße 2, 12489 Berlin, Germany
Jakob.Erdmann@DLR.de

7.1 Abstract

SUMO is an open source microscopic traffic simulation. A major component of modelling microscopic vehicle behavior is the lane-changing behavior on multi-lane roads. We describe a new model which uses a 4-layered hierarchy of motivations to determine the vehicle behavior during every simulation step and motivate in which ways it improves the current lane-changing model.

Keywords: microscopic simulation, lane changing.

7.2 Introduction

The SUMO application suite [1, 2] provides tools for the *Simulation Of Urban MObility*. It consists of a microscopic simulator for multimodal road traffic and a host of applications for preparing simulation input data (network import and modification, traffic import, routing) and for working with simulation outputs. The microscopic driving dynamics of road vehicles are determined by the interplay of several models briefly listed below:

- **Car-following model:** determines the speed of a vehicle in relation to the vehicle ahead of it.
- **Intersection model:** determines the behavior of vehicles at different types of intersections in regard to right-of-way rules, gap acceptance and avoiding junction blockage.
- **Lane-changing model:** determines lane choice on multi-lane roads and speed adjustments related to lane changing.

When simulating traffic on complex road networks with multi-lane roads, most routes which a vehicle might use require changing lanes. Even where there are no such hard necessities, lane-changing behavior is often a major determinant for traffic efficiency which underscores the importance of the respective model.

The lane-changing model in SUMO has been under continuous development since the start of the project in 2001 and will certainly undergo changes in the future. Due to a large number of improvements in 2013 we see the need to report on the current state of the model. These changes were prompted by problems and visibly implausible behavior in some of our simulation scenarios.

- Motorway traffic which requires many vehicles to change lanes at a point where the motorway splits exhibited heavy jamming contrary to real-world measurements (A92 scenario).
- Heavy jamming where motorway traffic in the main direction came to a stop because of vehicles merging at on-ramps (*Braunschweig* scenario).

- Jamming because vehicles did not change to their respective turn lanes in time and thus blocked the flow (*Braunschweig* scenario).
- Jamming because vehicles only used the outer lanes of a two-lane roundabout (*ACOSTA* scenario)

The model changes which were undertaken to alleviate these problems are tightly interwoven with the previous model which makes it impractical to discuss them in isolation. Instead we will describe the new model fully in the following sections and then describe areas of improvement relating to the above scenarios in section 7.10. Where the new model differs significantly from the previous model this is indicated by the flag **(new)** within the descriptions.

The lane-changing model described herein fulfills two main purposes: It computes the change decision of a vehicle for a single simulation step based on the route of the vehicle and the current and historical traffic conditions in the vehicles surroundings. Furthermore, it computes changes in the velocity for the vehicle itself and for obstructing vehicles which promote the successful execution of the desired lane change maneuver.

In comparison to other microscopic lane-changing models, this model explicitly discriminates between four different motivations for lane-changing. After discussing the general architecture of lane-changing within the simulation, the handling of these four motivations will be discussed in detail. The complete formulas and decision trees used in the implementation cannot be given due to lack of space. For those wishing to re-implement or modify these models, this paper should serve as a useful guide when reading the source files of the implementation in SUMO [3].

7.3 Architecture

Road traffic simulation in SUMO represents the road network in terms of **edges** which are unidirectional street segments between intersections and remain constant in their number of lanes, and their maximum speed (among other attributes). An edge consists of one or more parallel **lanes** which correspond to the (mostly marked) lanes found in European road networks. These lanes are **indexed** from right to left starting at 0. The route of a vehicle is stated in terms of the edges it needs to follow but during the simulation it moves along the lanes with mostly free choice of lane usage (except where lane usage restrictions are explicitly defined). Connectivity in the road network is defined on the level of lanes, with each lane having 0 or more successor lanes. If the lane on which a vehicle drives does not have a successor lane which belongs to the next edge along this vehicles route, the vehicles must change its lane in order to continue.

The speed of a vehicle is mainly determined by the next vehicle in front of it called the **leader**, which may be on the same lane or on the preferred successor lane after the current lane. This preference is discussed in section 7.4.1. The speed for following the leader is defined by the car-following model which is not discussed in this paper [5]. A vehicle may only change its lane if there is enough physical space on the **target lane** and if it neither comes too close to the leader on the target lane nor to its immediate **follower** on the target lane (*too close* being defined by the car-following model). If either of these conditions is not met, the vehicle is said to have a **blocking leader** or a **blocking follower**. To distinguish the vehicle currently under consideration from its leaders and followers we will refer to it as the **ego** vehicle. A vehicle that advances to a lane on the next edge is said to **advance** the lane, whereas a vehicle that changes to a parallel lane on the same edge is said to **change** lane.

During each simulation step, the following sub-steps are executed in order for every vehicle:

- 1) Computation of preferred successor lanes (called **bestLanes**)
- 2) Computation of safe velocities under the assumption of staying on the current lane and integration with lane-changing related speed requests from the previous simulation step
- 3) Lane-changing model computes change request (left, right, stay)
- 4) Either execute lane-changing maneuver or compute speed request for the next simulation step (involves planning ahead for multiple steps). Whether speed changes are requested depends on the urgency of the lane-changing request.

The sub-steps 3 and 4 are handled by a customizable software component the *laneChangingModel*. This gives a high amount of configurability within the bounds of the architecture. The *laneChangingModel* described in this paper is can be swapped against the previous model by setting user-configurable parameters. In the following, the four motivations for lane-changing are discussed in the order of their priority beginning with the most important. In section 7.9 we explain how conflicts between these motivations are resolved.

7.4 Strategic lane changing

Whenever a vehicle must change its lane in order to be able to reach the next edge on its route, we call this type of lane changing *strategic*. This happens whenever the current lane of the vehicle has no connection to the next edge of the route. In this case we say that the vehicle is on a **dead** lane. Note that such a lane does not have to be a dead-end in the common sense. A left-only turn lane is dead from the perspective of a vehicle that wants to go straight. A vehicle may perform a strategically motivated lane change well in advance before reaching the dead lane if no other motivation prevents it. This topic is discussed in the next two sections.

7.4.1 Evaluating subsequent lanes

Vehicles (or rather their assumed drivers) need to decide a sequence of lanes to follow along their route of edges. In this they have some degree of restriction (because some lanes are dead-ends) and they have some degree of freedom because there are multiple lane sequences available. In SUMO a data structure is computed which allows retrieving the following information necessary for subsequent computations:

- a) For every lane on the current edge, a sequence of lanes that can be followed without lane changing up to the next dead-end or to a maximum distance (**bestLanes**).
- b) For every lane on the current edge, the traffic density along the bestLanes (**occupation**)
- c) For every lane on the current edge, the offset in lane index to the lane which is strategically advisable (**bestLaneOffset**)

Note, that multiple lanes may have a bestLaneOffset of zero. In this case, the bestLaneOffset of other lanes points to the closest best lane. Most parts of this data-structure are only updated whenever the vehicle advances to the next lane. The algorithm for computing this data structure is discussed in [3]. The strategically advisable direction is not part of the customizable architecture because it is rather unambiguous (being based on maximizing the drivable distance and minimizing the number of necessary lane changes). Nevertheless, improvements in this part of the lane-changing model were made by fixing long-standing implementation bugs during the work on improving the lane-changing model in SUMO.

7.4.2 Determining Urgency

While approaching a dead-end lane, a vehicle has some amount of freedom to pursue the strategically advisable lane (which may involve changing or staying) or to follow conflicting motivations. The urgency for following the strategic necessities (i.e. changing to the left if $\text{bestLaneOffset} < 0$ and changing right if $\text{bestLaneOffset} > 0$) correlates with the following factors:

- a) remaining distance to the dead-end
- b) the presumed speed while approaching the end of the dead lane (**lookAheadSpeed**)
- c) magnitude of the bestLaneOffset
- d) occupation on the ultimate target lane (lane with $\text{bestLaneOffset} = 0$)
- e) occupation of the intermediate target lane (next target in direction of the bestLaneOffset)

A strategic change is deemed urgent if the following relation holds true:

$$d - o < \text{lookAheadSpeed} \times \text{abs}(\text{bestLaneOffset}) \times f \quad (1)$$

Where d is the distance to the end of the dead lane, o is a discount due to occupation and f is a factor that encodes the time typically needed to perform a successful change maneuver set to 10 for changing to the left and 20 for changing to the right.

Notably, if there are multiple lanes in between the current lane and the ultimate target lane, all their occupations should also matter, but are not currently evaluated. The lookAheadSpeed depends on the current and historical speed of the vehicle. This is necessary to avoid vehicles which temporarily have to slow down from losing all sense of urgency (new). The expected number of seconds until reaching the end of the dead lane (**remainingSeconds**) is used in subsequent computations. Currently, urgency is only considered for strategic lane changes but we discuss how it could apply to other motivations in section 7.11.

7.4.3 Speed adjustment to support lane-changing

Whenever a desired lane change cannot be executed due to blocking vehicles, a vehicle may adjust its speed to allow the lane change to succeed in later steps. Furthermore a vehicle may exert an influence on the speed of blocking vehicles (in reality this typically happens as a reaction to observing the turn signals of the ego vehicle). Due to the importance of completing strategic lane changes, it is assumed that the ego vehicle will take careful adjustments to enable the change. Basically, vehicles are assumed to drive at the maximum safe speed, so speeds can only ever adjusted downwards. However, as a part of the car-following model, vehicles may have a stochastic component which prevents them from using their maximum possible acceleration. Preventing this stochasticity (called **dawdling**) is a way of increasing vehicle speeds somewhat.

To compute the desirable speed adjustments, the following hierarchy of situations is distinguished by comparing ego speed, blocker speed, gaps and remainingSeconds (new):

(1) Has blocking leader

- a. **Will be able to overtake leader:** request leader to refrain from speeding up, prevent dawdling, (prevent overtaking on the right where forbidden by law)
- b. **needs to stay behind the leader**
 - i. slow down to stay behind the leader
 - ii. keep speed since the leader is faster anyway

(2) leader is not blocking: set a maximum speed to ensure that the distance to the leader remains sufficiently high

(3) there is no leader: drive with the maximum safe speed

The above decision tree results in a **plannedSpeed** with regard to a blocking leader. Using this value,

(4) has blocking follower

a. will be able to cut in before follower

i. fast enough to do so with current speeds: request follower to refrain from speeding up, prevent dawdling

ii. follower decelerating once is sufficient to open a gap: request follower to decelerate as much as needed, prevent dawdling

b. needs to be overtaken by follower

i. follower should slow down a bit to increase the chance that subsequent followers will be slow enough: request follower to decelerate a bit, slow down to be overtaken fast enough

ii. follower should overtake quickly: prevent follower from dawdling, slow down to be overtaken fast enough

Speeds, computed in this way are integrated with the maximum safe speed (**vSafe**) as computed by the car-following model by using the minimum of vSafe and all requested speeds.

The distinction between cases (4)b.i and (4)b.ii warrants further explanation. Whenever a vehicle tries to change from an on-ramp onto the motorway it has to yield to vehicles already on the motorway. These vehicles may slow down slightly to help merging vehicles, but they must not cause the flow on the motorway to break down. For this reason, vehicles that try to change to the left only cause blocking followers to slow down if their own speed exceeds a threshold value (currently 27m/s).

7.4.4 Preventing deadlock

If a vehicle needs to stop on a dead lane because changing to a continuing lane did not succeed it creates an undesirable impediment to traffic flow. The measures in the previous section help to prevent this situation from occurring too often (and it does occur in reality as well). However, if two vehicles on adjacent lanes both need to change to the lane occupied by the other vehicle (**counterLaneChange**) and both vehicles reach the end of a dead lane, a deadlock occurs. Neither vehicle has the option of driving any further nor can either vehicle get the space it needs to execute the strategic lane change (vehicles in SUMO cannot go backwards). This situation blocks the flow of traffic on both lanes and is highly undesirable. Currently, it can only be resolved by moving vehicles in a non-standard way (teleporting) after a time threshold is elapsed.

To prevent this type of deadlock, special care is taken whenever two vehicles are in a counterLaneChange relation. We refer to the vehicle which is closer to the end of the dead lane as the **blocking leader** and the other vehicle as **the blocking follower**. Note that this relation may change from one simulation step to the next. Generally, the blocking follower slows down when approaching the dead-end to ensure that the blocking leader has enough space to complete its lane change. In some cases the blocking follower is too fast or the blocking leader is too long. In this case the blocking leader must slow down to leave enough space for the follower before the dead-end (new).

Unfortunately, dead-lock situations can still arise if vehicles need to perform strategic lane changes across multiple lanes. In this case, a counterLaneChange situation can arise at a time

where both vehicles have already reached the dead-end and are unable to move. To prevent this, vehicles reserve additional space in front of the dead-end whenever they have to change across more than one lane (new). Currently, additional space of 20m is reserved for vehicles which need to change to the right across and 40m for vehicles which need to change to the left. The asymmetry is necessary to prevent yet another type of deadlock. The values were selected because they were found to perform well in preventing deadlock. Eventually they should be made configurable and be subject to rigorous calibration.

7.5 Cooperative lane-changing

In some real-world situations vehicles (or rather their drivers) perform lane-changing maneuvers with the sole purpose of helping another vehicle with lane-changing towards their lane. In the current model, vehicles are informed by other vehicles about being a blocking follower (the reason being that the turn-signals of the vehicle being blocked are always visible to the follower, whereas being a blocking leader is less obvious). If there are no strategic reasons against changing the lane, the ego vehicle may change in either possible direction to clear a gap for the blocked vehicle. Contrary to expectation, this was found to have a beneficial impact on traffic flow in some scenarios even if the ego vehicle attempts to change towards the blocked vehicle. This effect is not yet understood and warrants further investigation.

Vehicles which cannot perform a cooperative lane change adjust their own speeds slightly to increase the success probability for subsequent simulation steps. However, they do not request speed changes if they are blocked.

A special case for cooperative behavior arises at multi-lane roundabouts. Typically, all vehicles enter the roundabout at the outermost lane and also need to leave again at the outermost lane. Due to the short distances involved, this means they should always remain on the outermost lane for strategic reasons. However, this effectively turns all multi-lane roundabouts into one-lane roundabouts and thus degrades throughput. For this reason, the lane-changing model compels vehicles which are not yet on their final roundabout edge to change towards the inner lane (new). While this ignorance of strategic motivations sometimes results in stranded vehicles it has a beneficial impact on roundabout performance (see results for the ACOSTA scenario).

7.6 Tactical lane-changing

Tactical lane-changing refers to maneuvers where a vehicle attempts to avoid following a slow leader. It requires balancing the expected speed gains from lane changing against the effort of lane-changing (which is arguably a very driver-subjective value). The expected speed gains must also be balanced against the obligation for keeping the overtaking lane free. Failure to do so results in situations where slow vehicles with minor speed differences become major impediments to traffic flow. This part of the model is left unchanged from the old model [4]. Each vehicle maintains a signed variable **speedGainProbability** which by its sign indicates the beneficial change direction and by its magnitude the expected benefit. If the magnitude exceeds a threshold value, a tactical lane change is attempted. The variable `speedGainProbability` is modified incrementally during each simulation step and reset upon lane changing to prevent oscillations. If the lane to the left is faster than the current lane the value is updated according to the following formula:

$$\text{speedGainProbability} := \text{speedGainProbability} + \frac{v - u}{v} \quad (2)$$

Where v is the expected speed on the left lane and u is the expected speed on the current lane. If the left lane is slower, `speedGainProbability` is divided by 2 instead. If the right lane is faster than the current lane by at least 5 km/h the value is updated as:

$$\text{speedGainProbability} := \text{speedGainProbability} - \frac{v - u}{v} \quad (3)$$

Where v is the expected speed on the right lane and u is the expected speed on the current lane. If the right lane is slower, `speedGainProbability` is again divided by 2 instead. The asymmetry in handling changes to the left and to the right and the size of the thresholds (-2.0 for changing to the right, 0.2 for changing to the left) serve to prevent oscillations in lane usage.

Currently, whenever a slower vehicle cannot be overtaken because the ego vehicle is on the right side of it (and overtaking on the right is prevented in SUMO in accordance with German driving regulations), an impulse to move towards the left lane is generated. This should allow the blocking vehicle to move to the right and be overtaken on the left. This prohibition and the resulting behavior can be disabled by setting the simulation option `--lanechange.overtake-right`⁵.

7.7 Obligation to clear the overtaking lane

The compulsion to clear the overtaking lane could be framed as cooperative behavior because it helps other faster moving vehicles. However, contrary to the cooperative lane-changing behavior described in section 7.5 which is optional, the behavior described in this section is mandated by traffic laws. In the current lane-changing model, each vehicle maintains a variable `keepRightProbability` which is decremented over time and triggers a lane change to the right once a lower threshold value of -2 is exceeded (negative values are used in allusion to the variable `speedGainProbability`).

In reality, (German) drivers are obliged to change to the right unless they plan to overtake another vehicle. This anticipation of future behavior is modeled in the following way (new): If the current lane is not faster than the right lane by at least 5 km/h the ego vehicle determines the time t it could expect to drive with its desired maximum speed on the right lane before having to perform an overtaking maneuver depending on the gap and speed difference to the lead vehicle on the right. This is used to update the value of `keepRightProbability` p :

$$p := p - \frac{t * L}{V * v * T} \quad (4)$$

Where L is the legal speed limit on the current road, V is the maximum desired speed of the ego vehicle, v is its current speed and T is a calibration parameter currently set to 5. Thus, vehicles only change to the right if the expectation that they may stay there for some time is confirmed repeatedly. Vehicles which desire a lower speed are more eager to change to the right.

7.8 Remote controlled lane changing (TraCI)

Running SUMO simulations can be controlled by external programs using an interface called TraCI (**Traffic Control Interface**). Among the things that can be controlled is the lane usage of the vehicles. Remote requests to change to a target lane or keep the current lane must be

⁵ Available in the current source repository and in version 0.21.0 which is expected to be released at the time of this publication.

integrated with the “intrinsic” requests computed by the lane-changing model. This is accomplished by letting the user determine the urgency and the priority of remote requests by setting appropriate flags (new).

As an example, the interface allows the remote program to specify that a given vehicle should try to change to the left lane with urgency (i.e. with speed adjustments to itself and to blockers), unless there are urgent strategic reason against changing to the left and that the vehicle should ignore all other requests by the lane-changing model.

7.9 A hierarchy of lane changing motivations

The four motivations discussed above are considered in a hierarchical fashion as described by the following decision schema. The first statement which applies determines the vehicles change request. In every simulation step, each vehicle first considers changing to the right, and if no change to the right is performed, a change to the left is considered as well. Accordingly, the currently considered direction d is either right (-1) or left (1) according to resulting offset in lane index.

1. **Urgent strategic change to d needed:** change
2. **Change to d would create an urgent situation:** stay
3. **Vehicle is a blocking follower for another vehicle with urgent strategic change request:** change
4. **speedGainProbability above threshold and its sign matches d :** change
5. **non-urgent strategic change to d needed:** change (new)

An important aspect of preventing stopping at a dead lane is avoiding detrimental lane changes. Generally speaking, the fewer lane change maneuvers vehicles have to perform, the less chance they have to become stuck. One change that was found to be quite beneficial was the avoidance of changing to a dead lane which continues elsewhere. This is most often the case for turn-lanes at an intersection. A vehicle which intends to go straight should not use the left-only turn lane to get ahead because it will find it difficult to go back onto the required lane (new). In reality these turn-lanes often have directional markings at their start and there are rules which prohibit their use by vehicles which follow another direction.

7.10 Improvements over earlier Model

For a quantitative evaluation of the improvements, the following metrics were computed for a selection of benchmarking scenarios.

- **avgWaitingTime:** the average time each vehicle spent with speed below 0.1m/s
- **wrongLaneTeleports:** the count of vehicles which had to be moved artificially (teleported) because they could not complete a strategic lane change (after a threshold time t)
- **jamTeleports:** the count of vehicles which had to be moved artificially (teleported) because the successor lane was occupied (after a threshold time t)

The scenario *Braunschweig* contains the urban area of the German city Braunschweig (Brunswick) and the surrounding area with sections of motorway. The scenario spans one day and contains 650000 vehicle movements. The threshold time for teleporting was set to 120 seconds.

The scenario *A92* consists of a motorway section in southern Germany with a length of 20km. It contains 63000 vehicle movements over the course of one day. The threshold time for teleporting was set to 300 seconds.

The scenario *ACOSTA* is comprised of a section of the Italian city of Bologna and contains 9000 vehicles over the course of 1 hour. It is notable for containing a 2-lane roundabout. The threshold time for teleporting was set to 300 seconds.

The algorithms *old* and *new* correspond to the SUMO vType parameters `laneChangeModel="DK2008"` and `laneChangeModel="JE2013"`. As can be seen in Table 7-1, the new algorithm brings a significant improvement in all considered scenarios. Additional topics for future improvement are discussed in the next section.

Scenario/Algorithm	avgWaitingTime	wrongLaneTeleports	jamTeleports
Braunschweig/old	89.73	845	464
Braunschweig/new	46.66	7	9
A92/old	17.16	21	1
A92/new	0.02	0	0
ACOSTA/old	144.59	0	7
ACOSTA/new	76.69	0	0

Table 7-1 Performance metrics for old and new lane-changing model and different scenarios.

Compared to the old lane-changing model described in [4], the new model shows improvements in the following areas:

- Fine grained control over speed adjustments to ego vehicle and blockers lead to higher fulfillment rate of change request. In the old model, vehicles always reacted to blocking leaders by slowing down and they always slowed down when being a blocking follower (improved all metrics and all scenarios).
- Extrapolation of dynamics over multiple steps allows better choices between overtaking blockers and allowing to be overtaken (also improves the success rate and thus improves all metrics).
- Improved checking for deadlock-prone situations avoids deadlocks in more cases. In the old model, some cases of deadlock were avoided by allowing neighboring vehicles with opposite change requests to swap their positions instantly. This oversimplification is no longer necessary (primary impact on wrongLaneTeleports but secondary effect for the other metrics).
- Asymmetrical behavior when helping other vehicles with lane-changing (depending on the direction of change) prevents the main flow from breaking down at busy highway on-ramps (improved avgWaitingTime especially in *Braunschweig*).
- Special behavior within multi-lane roundabouts ensures that all lanes are used whereas in the earlier model only the outer lane was ever used (improved avgWaitingTime and jamTeleports, only *ACOSTA*).
- The explicit discrimination between the 4 different motivations for lane changing allows fine grained control for integrating model dynamics with external change requests (TraCI). This was necessary to successfully complete a project which simulated automated platooning (not discussed here).

Note, that a large number of model changes were tested in isolation using the above metrics. To simplify the presentation of our results we only show the effect of all combined model changes. It can be seen that all metrics improved for all scenarios except for the few cases where they had the best possible value to begin with, thus validating the usefulness of the new lane changing model. The A92 scenario is based on fine grained detector measurements which showed no jamming in the real world data. Also, the *Braunschweig* scenario exhibited deadlocks and jamming with a frequency that was utterly implausible for the demand model of a normal working day. Although improved traffic flow is not generally a sign of a more realistic model (after all, jams are a fact of life), for the above scenarios an increase in realism can be posited.

7.11 Outlook

The focus of the recent improvements of the lane-changing model was on deadlock-prevention and success rate of lane-changing. Since these dynamics are qualitative rather than quantitative and the failure cases are visibly obvious, simulation behavior could be tuned by inspection. The next goal is to calibrate the model to reproduce empirical lane-usage data.

Current measurements are promising but indicate room for improvement. Figure 7-5 and Figure 7- show the evolution of simulated traffic measurements with the leftmost data points being taken directly from traffic measurements on the German motorway A3 whereas data points to the right are measured at successive points on an otherwise featureless 3 lane road. It can be seen that vehicles more or less maintain the lane distribution and speeds which were measured in reality. Numbers are averaged over a simulated period of 24 hours.

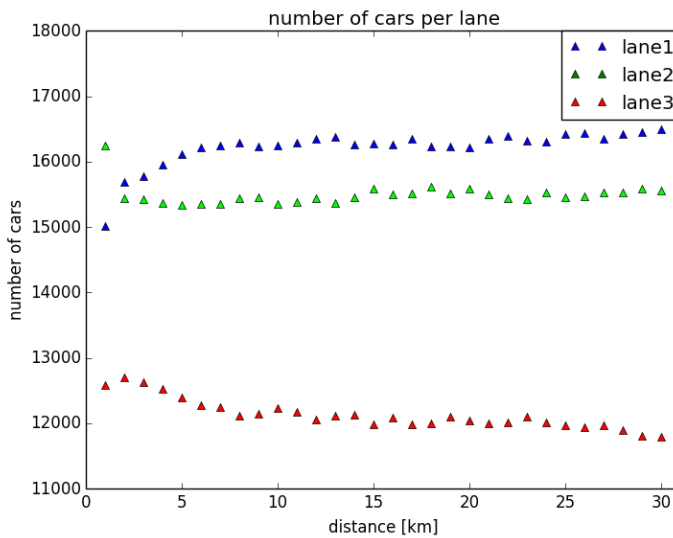


Figure 7-5 Evolution of lane usage over space for a 3-lane motorway section

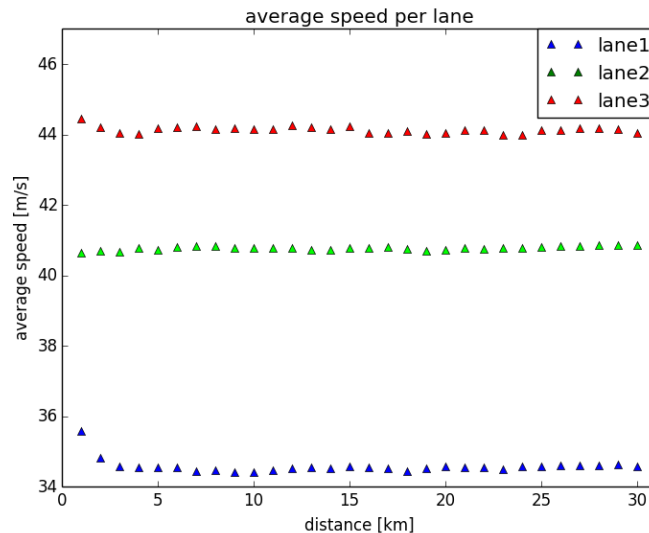


Figure 7-2 Evolution of speeds over space for a 3-lane motorway section

To allow model calibration, several hard-coded model parameters shall be exposed to the end user in a future release. There should at the very least be one parameter for each of the four motivations:

- Urgency of strategic changes
- Tradeoff between altruistic and egoistical behavior
- Eagerness to realize speed-gains
- Eagerness to clear the overtaking lane

Using these parameters the model will be calibrated and validated using real world measurements.

Cooperative lane-changing has not been extensively looked at and is a probable candidate for model improvements. Currently, only blocking followers change cooperatively whereas real-world situations are conceivable in which blocking leaders change as well. A typical situation which is not yet considered by the lane-changing model is the coercion to change to the right because a faster vehicle is approaching from the rear on the same lane. Another point is the usage of multi-lane roundabouts where currently, some vehicles become stuck on a dead-end inside lane. Additional checks should be done to prevent these situations from arising. It would also be helpful to know the degree in which inner lanes of roundabouts are used in reality. The curious fact that cooperative lane changes towards the vehicle being benefit simulation performance should also be investigated.

Some of the above issues might be resolved by extending the concept of *urgency* to all four motivations. A cooperative lane change is more urgent if the supported vehicle is about to suffer a bigger speed loss unless it receives help. Likewise a tactical lane change is more urgent if the ego vehicle is about to suffer a bigger speed loss (due to a slow leader on its lane). Changes with the intent of clearing the overtaking lane are more urgent if the follower on this lane is about to suffer a bigger speed loss as well.

7.12 References

- [1] Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D.: SUMO – Simulation of Urban MObility: An Overview. In: SIMUL 2011, The Third International Conference on Advances in System Simulation (2011)
- [2] DLR and contributors: SUMO homepage. <http://sumo.sourceforge.net/> (2013)
- [3] SUMO source code corresponding to this document. http://sumo-sim.org/trac.wsgi/browser/tags/v0_20_0/sumo/src
- [4] Krajzewicz, D.: Traffic Simulation with SUMO – Simulation of Urban Mobility. In: Barceló, Jaume (Ed.): Fundamentals of Traffic Simulation, Series: International Series in Operations Research & Management Science, Vol. 145, Springer, ISBN 978-1-4419-6141-9 (2010)
- [5] Krauß, S., Wagner, P., Gawron, C.: Metastable states in a microscopic model of traffic flow. *Phys. Rev. E*, American Physical Society, 1997, 55, pp. 5597-5602, (1997)

8 Second Generation of Pollutant Emission Models for SUMO

Daniel Krajzewicz¹, Michael Behrisch¹, Peter Wagner¹, Stefan Hausberger², Mario Krumnow³

¹German Aerospace Center, Rutherfordstraße 2, 12489 Berlin, Germany
{Daniel.Krajzewicz, Michael.Behrisch, Peter.Wagner}@DLR.de

²3130 Institut für Verbrennungskraftmaschinen und Thermodynamik, Inffeldgasse 19//
(Emissionsforschung), 8010 Graz, Austria
hausberger@ivt.tugraz.at

³Technische Universität Dresden, Institut für Verkehrstelematik, 01062 Dresden, Germany
Mario.Krumnow@tu-dresden.de

8.1 Abstract

Traffic puts a high burden on the environment in means of emitted pollutants and consumed fuel. Different attempts exist for reducing these impacts, ranging from traffic management actions to in-vehicle ITS solutions. When equipped with a model for vehicular pollutant emissions, microscopic traffic simulations are assumed to be helpful in predicting the performance of such approaches. We report about the implementation of a second generation of pollutant emission models.

Keywords: emissions, modelling, environment, traffic management.

8.2 Introduction

Air pollution is a well-known problem that ranges from local air quality issues up to global effects the humanity is confronted to. Following the International Transport Forum ([1]), “[the] Transport-sector CO₂ emissions represent 23% (globally) and 30% (OECD) of overall CO₂ emissions from Fossil fuel combustion. The sector accounts for approximately 15% of overall greenhouse gas emissions.” Different actors are involved in reducing road traffic’s environmental impact and resource consumption, often forced to do so by law. In Europe, automobile manufacturers shall reduce their fleet emissions [2]. Cities try to keep the amounts of pollutant concentrations below the thresholds formulated in according regulations, such as [3]. Finally, pollutant generation is closely coupled with the consumption of fuel. As fuel price has increased in the past years the reduction of pollutants is also in the focus of end users – individuals as well as (e.g. logistics) companies.

The development of technical solutions for critical systems usually includes a step where the solution is modelled and simulated. This step allows to validate the assumptions about the solution’s functionality and to a-priori benchmark or proof its performance. Traffic simulations are an established tool used by both, consultants and researchers. Conventionally, microscopic models are applied to areas that cover few intersections, while macroscopic models are used to replicate areas at city level. Academic approaches, such as the traffic simulation SUMO [4, 5] that is discussed herein, attempt to simulate large, city-wide areas using microscopic models.

In any case, for evaluating a solution that was designed to reduce road traffic's impact on the air quality, the involved traffic simulation must be capable to compute the amount the emitted pollutants to reduce with a predictable error. A large variety of emission models is described in the scientific literature. They differ in the required input parameters, the covered pollutants, the coverage of the real-world vehicle population (regarding their emission behavior), and the aggregation of the results in time and area. Therefore, it is necessary to formulate the requirements before choosing a model to implement.

In the following, the recent work on vehicular emissions modelling in SUMO will be presented. This work has been performed within the projects "COLOMBO" [6, 7] and "AMITRAN" [8, 9]. The models implemented within these projects replace SUMO's initial emissions model that was developed within the project "iTETRIS" [10, 11]. All three project are or respectively were co-funded by the European Commission.

The remainder is structured as following. A discussion of SUMO's requirements to an emission model is given, first, followed by an overview about emissions modelling and available emission models. A description of the implemented emission models into SUMO is given afterwards. Then, using and extending the emission models embedded in SUMO is described. Some use cases are presented afterwards. The report ends with a summary.

8.3 SUMO's Requirements to an Emissions Model

Briefly said, the model to choose should be capable to be used as a further measurement within the applications the traffic simulation is usually used for. As SUMO's goal is to simulate real-world traffic in large areas, the model should cover the population of vehicles found on roads nowadays. This counts for passenger vehicles as well as for heavy duty vehicles, busses, motorcycles, etc. One should also take into regard that the deployment of currently developed ITS applications will be realized in the future. Therefore, the model should be capable to represent future compositions of vehicle population. Some types of investigations require a distinction of regulative emission classes, e.g. the EURO-norm. Such a classification also helps in representing the population of vehicles over time, as most statistics on past and current vehicle fleets are represented this way.

A second top-level requirement is that the emission model should match the resolution of the traffic simulation. It should be sensible to all vehicle (or traffic) state attributes that are available in the simulation. In case of a microscopic simulation, a vehicle's acceleration, speed, and the slope of the road beneath the vehicle are the major attributes to consider. On the contrary, the model is wanted to use only parameters that are offered by the traffic simulation model. Such a close connection to the traffic model implies the possibility to compute emission values for each simulated time step, usually having a length of 1 s. To achieve this, the emissions model must allow to compute emissions at such a time scale.

Not all available models cover all pollutants emitted by road traffic. Therefore the pollutants assumed to be needed should be defined. Within the iTETRIS project (see [12]), it was decided to model the emission of CO, CO₂, NO_x, PM_x, and HC, because these emissions are toxic (CO), cause cancer (PM_x), are responsible for ground-level ozone increase and smog generation (NO_x and HC) or are greenhouse gases (CO₂). Additionally, the fuel consumption should have been modelled.

The model has to fulfil some other, non-functional requirements. It should be portable matching SUMO's overall portability. It should be fast in execution for not prohibiting its usage in large-scale scenarios. And it should be directly embedded into the simulation to avoid additional interaction (e.g. socket-based) or file exchange.

SUMO's viral GPL license requires the implementation of the model to be made available under the same license. And, of course, the model should be easily usable. Summarizing, the following requirements are put on the model:

- Cover the complete vehicle population (in means of emission classes);
- Offer a classification of classes into EURO-norms;
- Compute certain pollutants (CO, CO₂, NO_x, PM_x, HC, and fuel consumption were chosen);
- (Be) sensible to microscopic parameters available in the simulation;
- Require only information that is available in the simulation;
- (Be) able to compute emissions in simulated time steps;
- (Be) easy to parameterize:
- (Be) portable, fast in execution, and directly embedded into the simulation;
- (Be) licensed under a GPL-compatible license.

8.4 Emission Models Overview

Most of nowadays vehicles burn petroleum-derived fuel for propulsion. When regarding small time scales⁶, fuel consumption depends on the vehicle's engine characteristics as well as on the current load on the engine. The load is dictated by the force a vehicle needs to overcome as well as by the chosen gear (see [13] for a good explanation). The majority of the fuel burns to CO₂ and water, but other, often poisonous gases are generated as well. Catalytic converters convert a major portion of some of these pollutants into non-poisonous gases. The performance of the catalytic converter mainly depends on the catalyst's temperature as well as on the engine's current operating point. Different other influences exist, such as drive train losses, the road's slope, or the air-fuel ratio at combustion to name a few. Additionally, long-term effects of certain driving styles may change a vehicle's emission behavior.

In summary, every vehicle has an individual emission behavior. But when investigating road traffic, many vehicles of different types have to be regarded. It is thereby necessary to find a tradeoff between the amount of vehicle emission classes a model covers and the details in modelling each single vehicle or emission class. The literature accordingly distinguishes the following major emission model classes:

- "inventory" emission models that include data for the major portion of the vehicle emission classes; their input usually includes a vehicle population composition and the amount of driven distance, optionally also the average speed or an abstract traffic state. Such models usually cover a large set of different pollutants.
- "instantaneous" (or "modal") emission models that simulate a single vehicle's emission, where [14] proposes a further distinction into emission maps, regression-based models, and load-based models. Trying to model the emissions for a single vehicle as exact as possible, these models usually regard a small number of vehicles only.

As one can see, the models differ in granularity and the input parameters they need as well as in the number of pollutants covered. One should note that some "instantaneous" models exist which databases were incrementally extended over the years to cover a large portion of real-world's vehicle emission classes. The model PHEM ("Passenger and Heavy Vehicles

⁶ Most instantaneous consumption models work with steps of 1 second

Emission Model”) [15, 16] which derivate was included in SUMO, see section 8.5.2, is one of such models. Its sub modules and their inter-dependencies are shown in Figure 8-1.

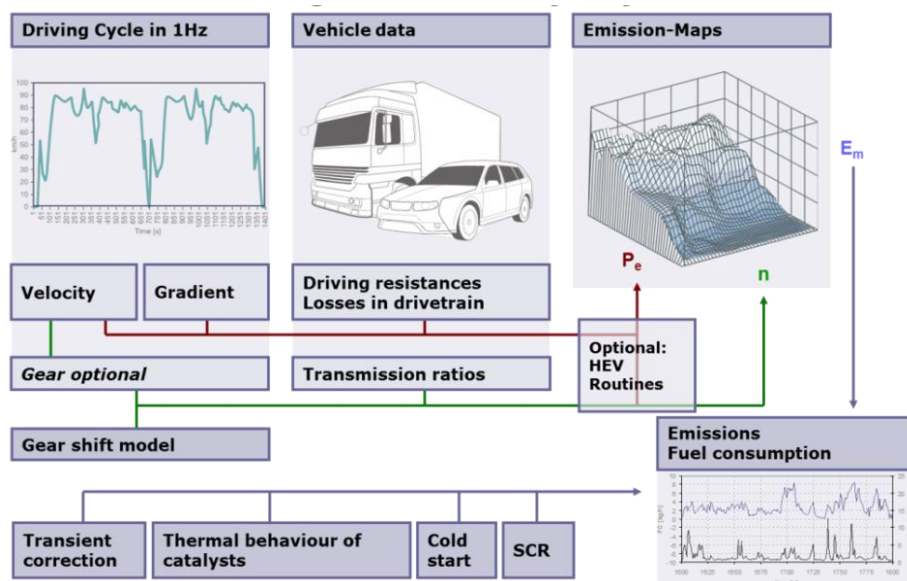


Figure 8-1: Schematic representation of the PHEM emission model [18].

Within the iTETRIS project, 15 non-commercial (freely available in means of data or a document that completely defines them) emission models have been investigated to determine candidates for being embedded into SUMO. Commercial models have not been included in this investigation. None of the 15 models fulfilled the posed requirements directly. The inventory models were found to be too coarse due to being insensitive to the vehicle’s acceleration. Most microscopic models either did not include methods for computing all of the desired pollutants or the needed input parameters were not completely given or would introduce a high number of additional parameter into SUMO’s vehicle type description.

8.5 Implemented Emission Models

As no emission model could be found that is instantaneous in means of regarding vehicle attributes used in a microscopic simulation but still covering a major part of the vehicle population, the decision to build an own model based on data from the HBEFA [17] database was taken. HBEFA, in version 2.1 at that time, is one of the investigated inventory models. This initial implementation of an emission model in SUMO will be described in the following sub-section. Two recently developed emission models will be presented afterwards: PHEMlight which is derived from PHEM and a new approach to reformulate the emissions inventory database HBEFA in its version 3.1. The models have been implemented in the projects “COLOMBO” and “AMITRAN”, respectively.

8.5.1 Initial HBEFA v2.1 Derivation

The model was implemented by extracting the data from HBEFA and fitting them to a continuous function that was obtained by simplifying the function of the power the vehicle engine must produce to overcome the driving resistance force. The simplified function is given as (1-1):

$$c_0 + c_1va + c_2va^2 + c_3v + c_4v^2 + c_5v^3 \quad (1-1)$$

The same functional form has been used for all emission types, only the parameters change per emission type and vehicle. HBEFA’s lack of a dependency on acceleration was

compensated by using the contained information about the dependency of the emissions on the road slope. But it should be noted that the only values up to $\pm 0.6 \text{ m/s}^2$ can be determined this way, the dependency on higher acceleration/deceleration was obtained by extrapolating given values. The used version 2.1 of HBEFA lacked data for rare vehicle classes (e.g. Euro-Norm-6 vehicles at that time). Both low as well as high velocities, the latter mainly for heavy duty vehicles, were missing for some emission classes as well. Therefore, the obtained curves did not match some basic emission properties, such as being always above zero or producing emissions at a velocity of 0 m/s. Emission classes that were recognized to be badly represented by the fitted function were removed.

The so obtained curves for the remaining vehicle classes were clustered into groups of similar behavior. The initial idea for performing this step was to reduce the number of emission classes to reduce the effort needed to set up a simulation scenario. In fact, this attempted simplification yielded in a set of badly usable emission classes. The lack of an explicitly given Euro-norm does not allow investigations of regulatory actions such as environmental zones that distinguish between vehicles of a certain emission class and the lack of a projection from the clusters back to the original classes makes setting up a realistic emission population complicated.

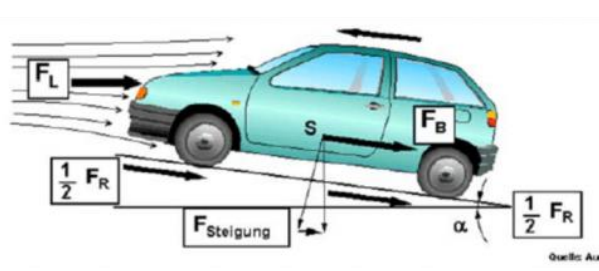
These issues were regarded during the implementation of the new HBEFA-based emission model described in section 8.5.3.

8.5.2 PHEMlight

PHEMlight is a simplified version of the emission model PHEM. PHEMlight has been designed and implemented by the Technical University of Graz within the COLOMBO project. PHEM itself provides basic emission factors for HBEFA 3 and COPERT and thus can be regarded as an de facto European reference.

PHEMlight uses characteristic emission curves which define the emission amount [g/h] as function of the actual engine power of the vehicle. These characteristic curves were computed using PHEM with representative dynamic real word driving cycles. The amounts of emissions produced by a vehicle (as well as the amount of consumed fuel) during a simulation step is determined by computing the power needed by the vehicle as shown in the following Figure 8-2, first. Then, this value is used to look up the characteristic emission curves.

$$P_e = (P_{\text{rolling resistance}} + P_{\text{air resistance}} + P_{\text{acceleration}} + P_{\text{road gradient}}) / \eta_{\text{gearbox}}$$



$$P_R = (m_{\text{Vehicle}} + m_{\text{Load}}) \times g \times (Fr_0 + Fr_1 \times v + Fr_4 \times v^4) \times v$$

$$P_{\text{Air}} = (Cd \times A \times \frac{\rho}{2}) \times v^3$$

$$P_a = (m_{\text{Vehicle}} + m_{\text{Rot}} + m_{\text{Load}}) \times a \times v$$

$$P_{\text{grad}} = (m_{\text{Vehicle}} + m_{\text{Load}}) \times \text{Gradient} \times 0.01 \times v$$

$$\eta_{\text{gearbox}} = 0.95 \text{ (average efficiency)}$$

Figure 8-2: Computation of the needed propulsion power in PHEMlight [18].

PHEMlight is available as a commercial add-on. The implementation itself is included in the usual SUMO version. But the major information is stored in the characteristic curves' and vehicle attribute files. Only two emission classes are included in SUMO's open source release: an Euro-4 passenger car with a gasoline engine and a passenger car with the same emission

class, but running on Diesel. The remaining emission classes have to be purchased from the Technical University of Graz.

8.5.3 HBEFA v3.1 Derivation

Given the lessons learned while implementing and using the initial HBEFA v2.1-based emission model and the availability of a new HBEFA version that includes new measures for modern Euro-Norm-6 vehicles, a new attempt to build an emission model was done in the scope of the AMITRAN project.

The basic procedure is similar to the one used for the initial HBEFA derivation: values included in HBEFA are extracted for each emission class and the function (1-1) is fitted against them. Again, the slope information given in HBEFA was used to take the part of the missing dependency on acceleration. The restrictions concerning available acceleration values remain as in the initial implementation.

Fitting the values to the given function is a linear problem, since only the linear coefficients c_0 to c_5 need to be evaluated. The fitting was performed using a linear model estimation algorithm from Python's "statsmodels" package. Since a linear fit usually does not lead to a clear answer whether or not a coefficient is zero, a couple of slightly different models is tested in each case (one emission class and one vehicle class), where some of the coefficients of (1-1) are set to zero and are not estimated in the fit. By comparing these candidate functions, the best one (based on RMS and t-value) is used as the final result, i.e. a set of fitting parameters for this case at hand. This works quite well in most of the cases, the remaining challenges are that not all emissions seem well represented by (1-1).

In principle, emission curves could be fit to all emission classes included in HBEFA's version 3.1 resulting in coefficients for some hundred different emission classes in SUMO. It was but decided to use only the most common emission classes.

8.5.4 Comparisons

In a first step, fulfilling the requirements formulated in section 8.3 by the models is presented. It should be mentioned that all models compute the desired pollutants' emissions (CO , CO_2 , NO_x , PM_x , HC , and fuel consumption). Table 8-1 shows a summary of other named requirements.

Table 8-1: A comparison of features for the three implemented models.

Requirement	HBEFA 2.1-based	HBEFA 3.1-based	PHEMlight
No. of emission classes	56*2(+1)	45(+1)	112(+1)
coverage	no modern (Euro 6) and seldom classes	Major passenger, heavy duty, and bus classes	almost complete
Euro-Norms	-	x	x
Covers chosen polutants	x	x	x
Uses speed	x	x	x
Uses acceleration	x	x	x
Uses slope	-	-	x
Needs further attributes	-	-	- (are included)
Step-size resolution	x	x	x
Easy parameterization	x	x	x

The number of respectively covered emission classes requires an explanation. The “+1” denotes that each model includes an emission class that does not produce emissions (“zero_emissions”). The initial model derived from HBEFA v2.1 duplicates all vehicle classes, where the second set ignores the current acceleration. These acceleration-free models were used within the investigations on emission-optimal routing (see section 8.7.2). The model does not include 56 distinct emission classes, but rather classification scheme clusters of such. Passenger vehicles can be chosen from three sets that include 3, 6, and 12 emission classes, respectively. Clusters with 7 and 14 emission types can be used for modelling heavy duty vehicles. As mentioned in section 8.5.3, the number of emission classes in the HBEFA 3.1-based model could be increased when necessary.

As discussed, it is hardly possible to give a comparison of the models against real-world data as such are not available for a complete vehicular population. Currently under investigation are comparisons against a single Euro 4 passenger car emissions as well as comparisons against PHEM.

8.6 Working with SUMO’s Emission Models

8.6.1 User Interaction

The implementation tries to give the user the highest grade of flexibility by allowing him to compose the vehicle fleet using the implemented emission classes. Each vehicle type can be assigned to a dedicated emission class. A vehicle type can be shared by an arbitrary number of vehicles. Emission computation is performed as soon as the user a) asks for an according output, b) asks to visualize the emissions, and/or c) asks for a vehicle’s current emissions via TraCI. All these interfaces cover all of the modelled emissions and – ignoring the visualization of emissions – are available in both, the command line and the graphical version of the simulation. The available outputs include:

- aggregation of emissions per lane with variable interval time spans
- aggregation of emissions per edge with variable interval time spans
- aggregation of emissions for each simulated vehicle
- non-aggregated (step-wise) vehicle emissions
- a vehicular trajectory file as defined in AMITRAN

In addition, SUMO’s on-line interaction interface “TraCI” allows to retrieve the emissions a single vehicle “produced” in the last simulation step, as well as emissions produced on edges or lanes. The visualization allows to color lanes and/or vehicles by the amount of pollutants emitted on them or generated by them, respectively.

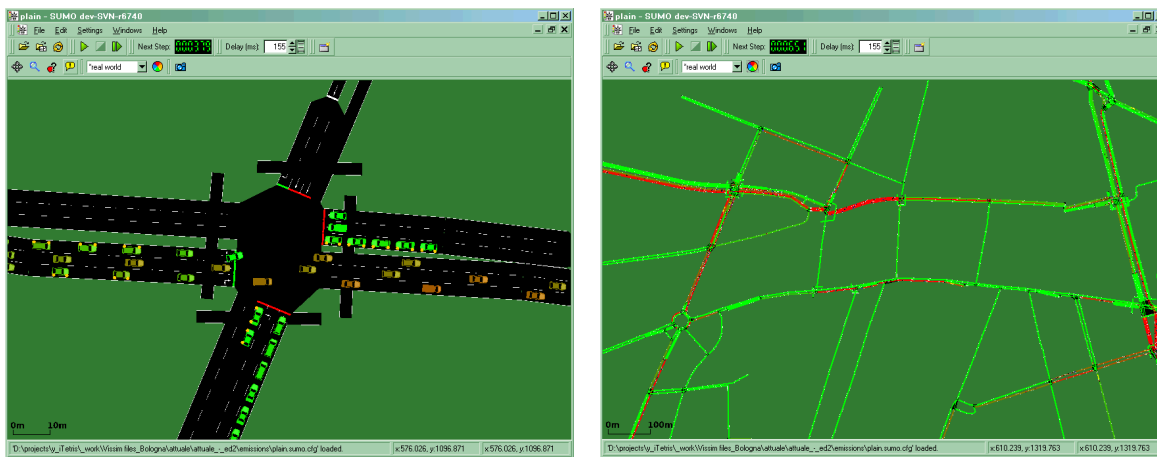


Figure 8-3: Examples of emissions visualization in SUMO; left: CO₂ emitted in the last time step by individual vehicles, right: lanes colored by the amount of fuel consumed in the last step by vehicles that were on the respective lane [12].

The AMITRAN trajectory file is an intermediate data exchange file that may be converted into inputs for emission models such as VERSIT+, PHEM, and HBEFA. It is interchangeably usable among different traffic simulation ecosystems such as SUMO, VISSIM and TNO ITS Modeler. A similar approach was used to generate input files for the PHEM emission model: a converter script was set up that obtains an “fcd-output” as generated by SUMO and converts it to files that resemble the vehicle fleet, the road network, and the trajectories as read by PHEM.

SUMO's user documentation includes a description of the output functionalities and was as well extended by a chapter on emissions modelling.

8.6.2 Router Support

Besides enabling the traffic simulation to compute pollutant emissions, the route computation applications included in the SUMO suite were extended. The wish was to perform route computation based on the amount of emitted pollutants instead of the conventionally used travel time. To achieve this purpose, the shortest-path router was extended to read time lines of vehicular emissions as weights of edges of the road network graph. The implementations of the shortest-path algorithms were extended to use these read values as edge weights, but additionally keep track of the travel time to obtain these weights from the correct time slice of the loaded time line. This extension has already been used for different purposes, see section 8.7.2.

8.6.3 Tools

Several additional tools support the development and usage of emission models in SUMO context.

“*emissionsDrivingCycle*” takes trajectories consisting of speed, acceleration (optional), and slope (optional) for each time step of a virtually driven drive cycle for one or multiple vehicles and computes the according emissions. The obtained emission time lines can be visualized using additionally available scripts. The tool as well reads trajectories in the AMITRAN format mentioned above and can thus be employed to use SUMO's emission models with trajectories from other simulation tools.

“*emissionsMap*” computes a matrix that contains the emission amounts of modeled pollutants in dependence to a driven speed, acceleration, and slope for a named emission class. Additional visualization script allow to investigate the so obtained matrices.

8.6.4 Embedding new Emission Models into SUMO

The co-existence of different emission models was realized by deriving a common “interface”. The interface is kept very simple. For each known pollutant, a method exists that returns its computed emission amount in mg/s (ml/s for fuel). The method obtains the vehicle’s emission class, its speed, acceleration, and the slope of the road it drives at. The model to use is encoded in the internal representation of the emission class which uses a 32 bit integer where the upper 16 bits are used to encode the used models while the lower 15 bits define a single emission class within this model. Bit 15 (the 16th bit) denotes whether the regarded emission type is a heavy duty or a light (passenger) vehicle. This information is needed to compute the vehicles’ noise emissions using the embedded Harmonoise noise emission model [19].

When being asked to compute the amount of a pollutant emissions, the interface determines the model to use based on the upper 16 bits, first. Then it asks the model implementation for computing the emission amount, passing all given values.

Besides giving access to the emission computation, the interface holds several further methods, mainly for computing parameters needed for file exchange between AMITRAN tools. As SUMO does not force emission models to fulfill a common view on emission classes, these methods derive information such as the fuel type, the Euro Norm, or the type of the vehicle based on the information known to the emission model implementations only.

The interface offers a clean access to the implemented models, but it should be noted that currently only models that rely on the selected parameters – emission class, speed, acceleration, and slope – can be implemented. As soon as other parameter have to be taken into account, the interface would have to be extended.

8.7 Use Cases

Being available for several years, the emission models have been already used in a large variety of investigations of which some are outlined in the following.

8.7.1 Investigating Environment Impacts of ITS Solutions

The major application is surely to measure changes in produced emissions when investigating new methods that influence traffic. In such cases, the computed emissions are used as a performance indicator besides the commonly used traffic efficiency measures, such as travel time or waiting times. Given SUMO’s output capabilities, such measurements can be easily obtained.

As increasing traffic efficiency usually reduces pollutant emission, often no new insight despite obtaining numbers can be gained from such evaluations. But it is interesting to note that in some cases, the deployment of a new ITS solution may increase the amounts of produced emissions. This was shown for the GLOSA (Green Light Optimal Speed Advisory) application [20] where, when assuming long communication ranges of more than 500 m, a vehicle may be advised to run at a low velocity (below 25 km/h) for a long time, yielding in emissions above the non-equipped situation. Therefore, it is advised to include environmental performance indicators when evaluating a new method or system.

8.7.2 Emission-optimal Routing

Usually, route computation is performed using travel times as weights for the edges of a road network. But what if one would use the emitted pollutants instead? Would their emission be reduced? The first investigations on this topic were performed using a real-world network

[21]. To gain a deeper understanding about the dynamics of the processes, later investigations ([22]) were performed using synthetic scenarios. Neither a singular user nor a singular system optimum are assumed to be computable using currently available methods. Besides new assignment methods. But the research is as well supposed to increase the understanding about the inter-dependencies between traffic flow and pollutant emissions.

8.7.3 Evaluation of real Traffic Management Actions

European authorities are forced by the “Directive 2008/50/EC of the European Parliament and of the Council” to assure certain air quality. Traffic management, usually operated by local authorities, has the duty to perform corrective actions that reduce road traffic’s impact, if needed. A proof-of-concept for simulating such actions that used SUMO and the HBEFA 2.1-derivation is presented in [23] where three speed limit changes were investigated – 30 km/h and 60 km/h for urban areas and 80 km/h for highways.

In his Master thesis ([24]), Tomàs Josep Vergés investigates the MARLIS [25] database that lists actions performed by traffic management authorities, first, to evaluate which of the actions can be simulated using only a microscopic traffic simulation. The evaluation shows that most actions target at a change in the population’s mobility behavior, mainly for changing to a more environment-friendly transport mode. This can only be simulated using an according population model that was not available within his research. The resulting selection of actions to be implemented in SUMO consists of a) a reduction of the allowed velocity in inhabited areas to 30 km/h, b) a restrictive environmental zone, and c) a permissive environmental zone. These actions are modeled, simulated using PHEMlight, and discussed within the thesis.

8.7.4 Emissions-related COLOMBO Solutions

The major objective of the COLOMBO project is the development of traffic management solutions that use data from vehicular communications, assuming only a small number of vehicles to be equipped with this technology. Here, not only the impacts of solutions on the environment are investigated. Instead, some parts of the work are directly concerned with the development of solutions and methods that measure or influence emission behavior. Among the targeted topics one may find a “local emissions monitoring” system that uses trajectories obtained from vehicular communications and inductive loop measures to compute the amount of emitted pollutants. A further methodology under development is the “emission optimal driver behavior” that given a vehicle and a traffic situation determines the emission-optimal longitudinal behavior of the vehicle. COLOMBO will use the PHEMlight emission model, verifying the results against the original PHEM model.

8.8 Summary

Recent steps in modelling and using emissions within the open source traffic simulation SUMO were presented. As shown, the inclusion of emission models into microscopic road traffic simulations allows to gain insights into the effects of evaluated solutions on the environment. Even though in most cases the induction “smoother traffic -> less emissions” holds, evaluating pollutant emission behavior may offer some surprises, as named for the GLOSA example in section 8.7.1. In addition, embedding emission models into a traffic simulation allows new investigations besides measuring the impacts of ITS solutions or regulatory actions on the environment.

Three emission models that are currently implemented in SUMO were outlined. Issues regarding the first HBEFA implementation were recognized and named, and a recently

implemented model that tries to solve them was described. In addition, the extension of SUMO by a commercial emission model, PHEMlight, was presented.

The presented extensions cover the work defined for the projects "COLOMBO" and "AMITRAN" well. Nonetheless, several possible extensions were already identified that may be targeted in the future. As discussed, all currently implemented models rely on the vehicle's speed, acceleration, and the slope of the road it is located at. When looking at possible future applications, the vehicle's mass comes into play. Incorporating this attribute would improve the simulation's usability for simulating fleet management and other logistics approaches. Further model improvements could be expected if the gear choice and cold-start emissions would be incorporated, but both would need new, respective models and additional input data that the user would have to supply.

But given the currently implemented models, the next steps towards a further improvements of the results should be performed on increasing the simulation's representation of single vehicles' longitudinal behaviour; it is known that nowadays car-following models neither replicate approaches to an intersections correctly nor their acceleration. As acceleration has a major influence on pollutant emissions, its correct representation should be attempted.

8.9 Acknowledgements

The authors want to thank the European Commission for co-funding the work in the context of the projects "iTETRIS", "COLOMBO", and "AMITRAN".

8.10 References

- [1] International Transport Forum: REDUCING TRANSPORT GREENHOUSE GAS EMISSIONS: Trends & Data, OECD, 2010
- [2] European Parliament and the Council of the European Union, Regulation (EC) No 443/2009 setting emission performance standards for new passenger cars as part of the Community's integrated approach to reduce CO₂ emissions from light-duty vehicles, 2009
- [3] European Parliament and the Council of the European Union, Directive 2008/50/EC on ambient air quality and cleaner air for Europe, 2008
- [4] Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.: Recent Development and Applications of SUMO - Simulation of Urban MObility, International Journal on Advances in Systems and Measurements, 5 (3&4), pp.128-138, ISSN 1942-261x, 2012
- [5] DLR and contributors: SUMO homepage, <http://sumo.sourceforge.net/>, 2013
- [6] Krajzewicz, D., Heinrich, M., Milano, M., Bellavista, P., Stütze, T., Härrig, J., Spyropoulos, T., Blokpoel, R., Hausberger, S., Fellendorf, M.: COLOMBO: Investigating the Potential of V2X for Traffic Management Purposes assuming low penetration Rates, In: ITS Europe 2013, Dublin
- [7] COLOMBO consortium (2013): COLOMBO web pages, <http://colombo-fp7.eu/>, last visited on 10.04.2014
- [8] Jonkers, E., Klunder, G., Mahmood, M., Benz, T.: Methodology and Framework Architecture for the Evaluation of Effects of ICT Measures on CO₂ Emissions. In: Proceedings of the 20th ITS World Congress, Tokyo, Japan, 2013

- [9] AMITRAN consortium (2013): AMITRAN web pages, <http://www.amitran.eu/>, last visited on 10.04.2014
- [10] Rondinone, M., Maneros, J., Krajzewicz, D., Bauza, R., Cataldi, P., Hrizi, F., Gozalvez, J., Kumar, V., Röckl, M., Lin, L., Lazaro, O., Leguay, J., Härrri, J., Vaz, S., Lopez, Y., Sepulcre, M., Wetterwald, M., Blokpoel, R., Cartolano, F.: ITETRIS: a modular simulation platform for the large scale evaluation of cooperative ITS applications. In: *Simulation Modelling Practice and Theory*. Elsevier. DOI: 10.1016/j.simpat.2013.01.007. ISSN 1569-190X, 2013
- [11] iTETRIS consortium: iTETRIS web site, <http://www.ict-itetris.eu/>, 2011, last visited on 8th of January 2014
- [12] Krajzewicz, D., Blokpoel, R., Cataldi, P., Bieker, L., Ringel, J., Leguay, J., Lopez, Y.: iTETRIS Deliverable 3.2 – Traffic Modelling: ITS Algorithms, public deliverable, 2010
- [13] Treiber, M., Kesting, A., Thiemann, C.: How Much does Traffic Congestion Increase Fuel Consumption and Emissions? Applying a Fuel Consumption Model to the NGSIM Trajectory Data, Presentation No. 08-2715 at the Annual Meeting of the Transportation Research Board, January 13-17, 2008, Washington, DC.
- [14] Cappiello, A., Chabini, I., Nam, E.K., Lue, A., Abou Zeid, M.: A statistical model of vehicle emissions and fuel consumption, *Intelligent Transportation Systems, Proceedings of the IEEE 5th International Conference on*, vol., no., pp.801,809, doi: 10.1109/ITSC.2002.1041322, 2002
- [15] Hausberger, S., Rexeis, M., Zallinger, M., Luz, R.(2009): Emission Factors from the Model PHEM for the HBEFA Version 3, Report Nr. I-20/2009 Haus-Em 33/08/679
- [16] Technical University of Graz: pages of the Institute for Internal Combustion Engines and Thermodynamics (IVT), 2014, last visited on 10.04.2014
- [17] INFRAS (2013): Handbuch für Emissionsfaktoren, <http://www.hbefa.net/>, last visited on 06.02.2014
- [18] Hausberger, S., Krajzewicz, D.; Deliverable 4.2 - Extended Simulation Tool PHEM coupled to SUMO with User Guide, public project report, available at http://www.colombo-fp7.eu/results_deliverables.php, 2014
- [19] Nota, R., Barelds, R., van Leeuwen, H.: Harmonoise WP 3 - Engineering method for road traffic and railway noise after validation and fine-tuning, Harmonoise Technical Report HAR32TR-040922-DGMR20 (Deliverable 18)
- [20] Krajzewicz, D., Bieker, L., Erdmann, J.: Preparing Simulative Evaluation of the GLOSA Application. In: *Proceedings CD ROM 19th ITS World Congress 2012*, Paper ID: EU-00630. ITS World Congress 2012, 22.-26. Okt. 2012, Wien, Austria
- [21] Krajzewicz, D., Wagner, P.: Large-scale Vehicle Routing Scenarios based on Pollutant Emission. In: G. Meyer and J. Valldorf (Eds.). *Advanced Microsystems for Automotive Applications 2011*, AMAA 2011, Springer, 2011, pp. 237-246
- [22] Flötteröd, Y.-P., Wagner, P., Behrisch, M., Krajzewicz, D.: Simulated-based Validity Analysis of Ecological User Equilibrium. In: *Winter Simulation Conference Archive, 2012 Winter Simulation Conference*, 2012
- [23] Krajzewicz, D., Flötteröd, Y.-P.: Simulative Untersuchung abstrakter und realer Verkehrsmanagementansätze zur Emissionsreduktion. In: *Kolloquium Luftqualität an Straßen 2013*, pp. 42-57. Bundesanstalt für Straßenwesen. 2013.

- [24] Josep Vergés, T: Analysis and simulation of traffic management actions for traffic emission reduction. TU Berlin. 2013.
- [25] BAST (2012): MARLIS - Datenbank mit Maßnahmen zur Reinhaltung der Luft in Bezug auf Immissionen an Straßen, Version 3.1, http://www.bast.de/nn_42544/DE/Publikationen/Datenbanken/MARLIS/MARLIS.html, 06.02.2014

9 Modelling Bluetooth Inquiry for SUMO

Michael Behrisch, Gaby Gurczik

*German Aerospace Center, Rutherfordstr. 2, 12489 Berlin, Germany
{Michael.Behrisch,Gaby.Gurczik}@DLR.de*

9.1 Abstract

SUMO provides an interface for the implementation of arbitrary additional vehicle devices. This paper describes how this interface was used to implement Bluetooth devices with a special focus on the inquiry process and how its modelling relates to real world measurements and a simple analytic model.

Keywords: Traffic simulation, Bluetooth, Inquiry Modelling

9.2 Introduction

Bluetooth [5] is a short-range, low-power, IEEE open standard for implementing wireless personal area networks. Bluetooth operates in the globally unlicensed 2.4GHz short-range radio frequency spectrum. Since there is a potential problem of interference from other devices using this frequency band, Bluetooth uses a Frequency-Hopping Spread Spectrum (FHSS) scheme, where devices alternate rapidly among the 79 available frequencies to transmit data. To set up an actual connection to exchange the necessary information between two Bluetooth devices, the so called inquiry process is designed to scan for other devices within range and thereby to discover each other. During the inquiry (discovery) process, one Bluetooth device (the master) enters the inquiry substate, whereas the other Bluetooth device (the slave) enters the inquiry scan substate. In the inquiry process the 48-bits unique MAC address and the internal clock-offset are exchanged in order to set up a lasting connection [5, 3, 8].

Bluetooth devices are available in a number of vehicles and depict an easy way of detecting motions of persons and goods. Since every device is uniquely identifiable via its MAC address the devices can also be used to redetect vehicles over long ranges giving way to new applications of traffic monitoring. Since every Bluetooth device can be detected the data is ubiquitously available from headsets and navigational devices and also from in vehicle detectors such as tire pressure measurements. It is also easy to equip small devices such as smartphones to act as a detector making Bluetooth a universally accessible data source.

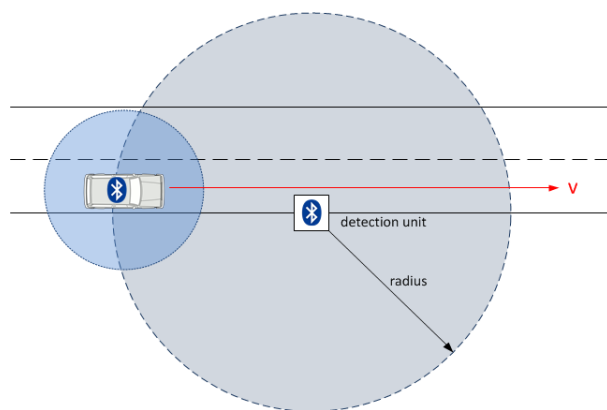


Figure 9-1: The Bluetooth detection principle

The German Aerospace Center (DLR) developed a traffic monitoring approach, called DYNAMIC [1, 7], which combines the advantages of Floating Car Data (FCD) and Floating Observer Data (FOD) principles. DYNAMIC is based on detections which are made by floating traffic observers using wireless radio-based technologies such as Bluetooth while passing other traffic objects (vehicles, cyclists, pedestrians). For the evaluation of the performance of DYNAMIC it is crucial to know how likely it is that a detectable traffic object (i.e. with Bluetooth device on board) within the detection range will be monitored. The major point to answer this question is the inquiry process which sets up the connection between Bluetooth devices and which can take up to several seconds. Given the possibly high speed of the vehicles and the relatively small detection range this poses a major problem to this detection mechanism. This paper focusses on a simple model for the inquiry process, describes its outcomings and the implementation of the process in the Bluetooth model of SUMO [2, 6] and compares it to real world measurements. The first section will focus on the analytical part, the second will describe the implementation in SUMO and the scenario used for evaluation and finally we will compare the theoretical and the simulative results with real world measurements.

9.3 State of the Art

In this paper we deal with the modeling of the inquiry process performance due to integrate the model in SUMO so that we can simulate Bluetooth detection behavior for stationary as well as mobile Bluetooth traffic monitoring systems. Since empirical analyses are complex and costly, a benchmarking implement is of particular importance. Unfortunately, researches in terms of evaluating Bluetooth traffic monitoring take Bluetooth performance mostly for granted. Therefore, they consider only frame conditions like distance between detector location and street, detection range and vehicle speed [10, 12]. The inquiry process and with it the Bluetooth technology in itself is no object of research.

Thus, we had a closer look at related works from special field computer engineering where several researches deal with formal analysis or empirical approaches to model the inquiry process performance. For example in [4] a formal analysis using probabilistic model checking is developed to compute the expected time required for a master device to successfully receive replies from listening slave devices. On the basis of two different empirical approaches (first model using observation windows, second model using FHS interval times), [3] try to find

out whether the number of inquirer and inquiry scanners has an effect on the discovery time. In [8] a detailed analysis of the interaction between Bluetooth devices in the inquiry and inquiry scan substates is given to analytically derive the inquiry time probability density function. Nevertheless, they state that precise inquiry time characterization is difficult due to the complex temporal and spectral interactions between two devices (for details see [8, 9, 11]).

Difficulties in using these models occur since that work is in the majority of cases older research of the time when Bluetooth was introduced as short-range communication technology between electronic devices. Therefore, these researches typically refer to Bluetooth specification version 1.1 which is important to know since the main difference in terms of the protocol is that, in version 1.1, the receiver only sends replies to every second message received. Hence, a device has to be discovered twice before it is actually considered to be discovered [4]. Furthermore, most researches are based on the assumption of an ideal, error-free environment, where messages never get lost [8]. This is, especially in our special field of studies, not lifelike.

For this reason, we investigate a simple model for the inquiry process in this paper, which fulfils our purpose while at the same time considering the specific behavior of the Bluetooth inquiry process.

9.4 Analytical Modelling of the Inquiry

The inquiry will be modelled as a frequency scanning process which lets the detector determine the frequency the vehicle device is using for the communication. Since the detector may change the order in which it scans for every pass and the device may change its frequency as well and we have no a priori knowledge about the distribution we assume every frequency and every order is equally likely.

The real inquiry process as described in [5] is much more complicated, it involves two trains of frequencies which change after every scan while the device to be detected only shows up in regular intervals. We will make use of these properties in our modelling later on.

Assuming a length of the scanning interval of l then the target frequency is one point in each of the successive intervals. The task is now to calculate the probability that another interval of length t (modelling the travel time in the detection range) contains at least one of the points.

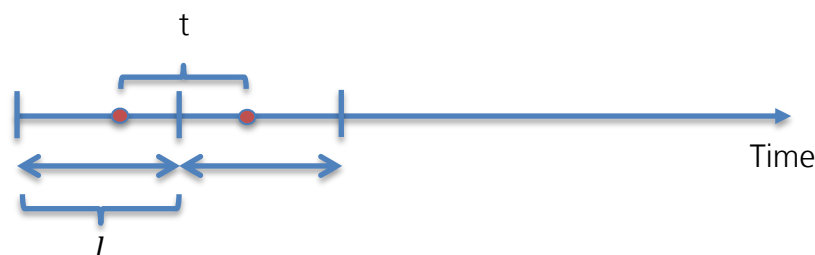


Figure 9-2: Model of the inquiry process with two scanning intervals of length l

We distinguish three cases depending on the relation of travel time and scanning interval:

1. $t < l$
2. $l \leq t < 2l$
3. $t \geq 2l$: The detection probability is obviously 1

We solve case 1 and 2 by integrating over the position of the starting point of the travel interval in the first scanning interval and then dividing by the length of the interval. The integration always needs to be split into the cases where t lies completely in l and where t can be divided into a part a in l and a part outside l :

$t < l$:

$$\begin{aligned}
 P_2(t, l) &= \frac{\int_0^l p(t, l, x) dx}{l} \\
 &= \frac{1}{l} \int_0^{l-t} \frac{t}{l} dx + \frac{1}{l} \int_{l-t}^l 1 - \left(1 - \frac{l-x}{l}\right) \left(1 - \frac{t-(l-x)}{l}\right) dx \\
 &= \frac{t}{l} - \frac{t^3}{6l^3}
 \end{aligned} \tag{9-1a}$$

$l \leq t < 2l$:

$$\begin{aligned}
 P_2(t, l) &= \frac{\int_0^l p(t, l, x) dx}{l} \\
 &= \frac{1}{l} \int_0^{2l-t} 1 - \left(1 - \frac{l-x}{l}\right) \left(1 - \frac{t-(l-x)}{l}\right) dx + \frac{1}{l} \int_{2l-t}^l 1 dx \\
 &= 1 - \frac{(2l-t)^3}{6l^3}
 \end{aligned} \tag{9-1b}$$

The resulting function depicting the probability depending on the travel time ratio is shown in Figure 9-3. For l we can assume the length of the scanning interval which is about 2.56s.

During the evaluation of the theoretical result we found a simpler exponential model to fit the data even better. The major drawback of the first approach is that the detection is assumed to be for sure if the interval is larger than $2l$, so there is no possibility of a miss right after this point. To handle this case more gracefully and also get closer to the real world functions presented below, we assume that we have a fixed detection probability p_d whenever the detector happens to be online simultaneously with the device to be detected. We assume this probability to be close to 0.5, because the detector as described above may be in the wrong train when the device appears and so it may scan the wrong frequencies. On the other hand the device is long enough online that it is principle possible that (provided the train is correct) every frequency is detected. The number of tries for a detection is calculated by the ratio of the travel time t and the interval between two online events b of the device (which is about 0.64s). We assume that there are on average t/b detection tries, so the simple resulting formula is:

$$P_1(t, p_d, b) = 1 - (1 - p_d)^{\frac{t}{b}} \tag{9-2}$$

A third function was taken into account when evaluating the first practical results which also resembles the fact that the detector might need an additional amount of time to recover after

each detection and thus may take a longer period before detecting all of a number of available devices. The function is of a similar general shape as the binomial form above, but to get a better fit an additional exponent was introduced. Fitting to the data resulted in:

$$P_3(t) = 1 - e^{-.24*t^{2.68}} \quad (9-3)$$

A comparison of the three function shows the picture below

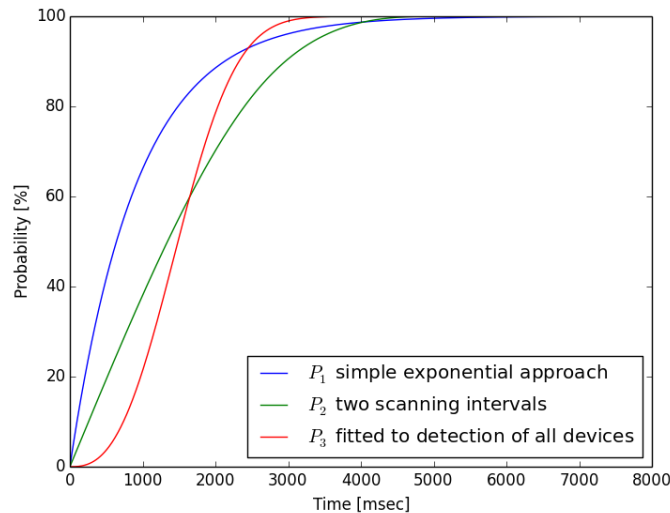


Figure 9-3: Comparison of modelling functions

9.5 The Simulation Implementation

To evaluate our analytical results SUMO was extended by the functionality to specify whether a traffic object works as Bluetooth transmitter (BTsender) or Bluetooth receiver (BTreceiver). BTsender are all the vehicles which can be detected by the BTreceivers. In practice that means that these vehicles have a Bluetooth device on board. Furthermore they have no additional functionality. The vehicles which are defined to be BTreceivers are our Floating Traffic Observers which are used for traffic monitoring. Every simulated vehicle can be a BTsender or a BTreceiver, it can also have both properties or none of them (i.e. being a simple traffic object with no additional Bluetooth features). To control the Bluetooth detection in SUMO global parameters like equipment rates for BTsender and/or BTreceiver or the detection range can be stated using the command line options. The mentioned functionalities were implemented for SUMO version 0.19.0.

The implemented detection process in SUMO calculates the time the BTsender is in the detection range of the BTreceiver and determines the probability whether a detection took place purely based on this time. The first implementation also available in SUMO 0.19.0 used the function P_3 above but was found to have two major drawbacks compared to real world data as well as analytical evaluation: The relatively slow incline at the start and later increase of the first derivative in the process. There is no delay to be expected in the detection of the first device so the new detections should become less and less in the course of the process as it happens with P_1 and P_2 .

When choosing between P_1 and P_2 there is (beside the property of not being fixed to 1 after a certain amount of time mentioned above) an additional benefit of P_1 related to the implementation. Since the simulation determines in every simulation step anew whether a detection took place, the probabilities should be additive, that is, it should be easy to

calculate the probability that there was a detection in the joined interval $t_1 + t_2$ from the individual probabilities that there were detections either in t_1 or t_2 . As it turns out this can be easily achieved with the exponential distribution above.

$$\begin{aligned} P_1(t_1 + t_2, p_d, b) &= 1 - (1 - p_d)^{\frac{t_1+t_2}{b}} = 1 - (1 - p_d)^{\frac{t_1}{b}} (1 - p_d)^{\frac{t_2}{b}} \\ &= 1 - (1 - P_1(t_1, p_d, b))(1 - P_1(t_2, p_d, b)) \end{aligned} \quad (9-4)$$

Where the last term denotes exactly the probability of two independent throws in successive intervals. This combination of probabilities is not possible with the other approaches.

9.6 The Simulation Scenario

The underlying network for our simulation scenario is a representation of the DLR test track, the Ernst-Ruska-Ufer (abbreviated ERU in the figures below) in Berlin-Adlershof. It includes a total track length of about 4 km with one major road (1.4 km with two directions) and several incoming and outgoing minor roads. We simulated a whole day with the demand and the route choices being calculated directly from induction loop data for the 11.1.2011.

There is a total demand of about 30000 vehicles including about 4% trucks and busses. In a first step the scenario was calibrated to the detector data such that network effects as a major traffic jam in the late afternoon on the eastbound direction are correctly reflected in the simulation.

The Bluetooth related parameters are the following:

- one fixed BTreceiver in each direction (see the green spots)
- fixed BTsender equipement rate of 30%
- detection range 100m

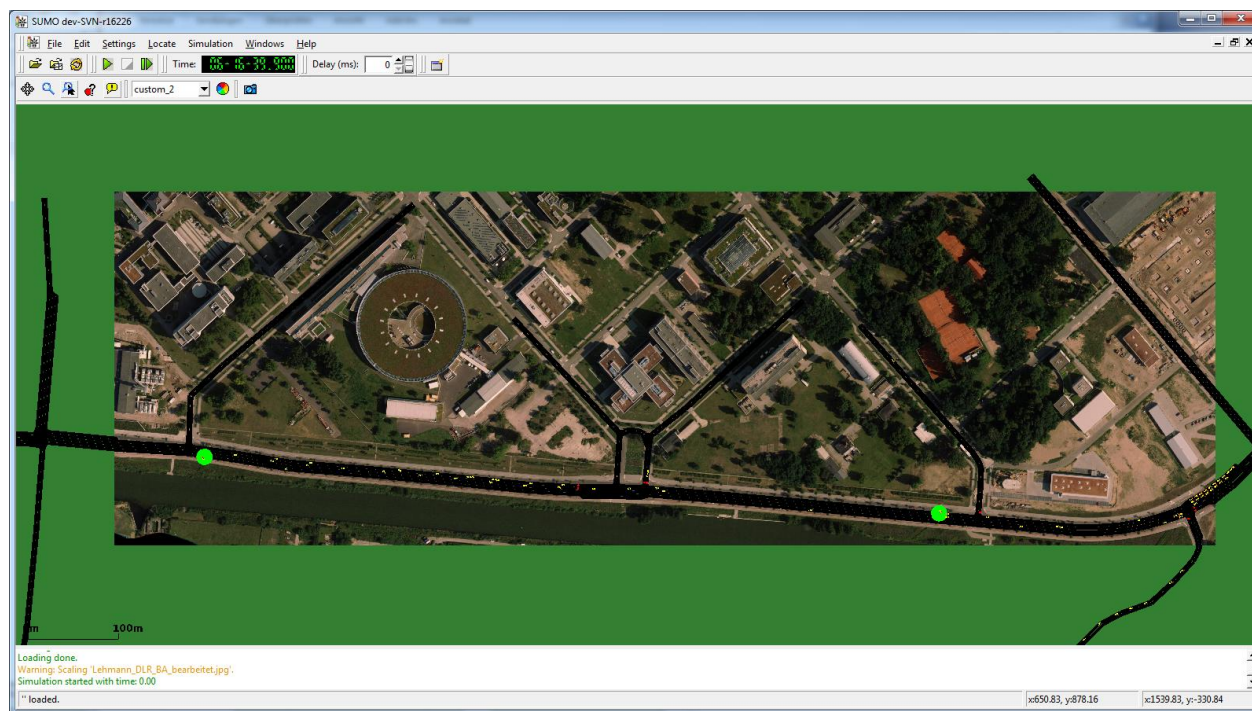


Figure 9-4: Test track scenario in the final SUMO simulation (green points denote the bluetooth detectors)

9.7 Comparison to Real World Measurements

In order to derive the exponential function mentioned above, laboratory as well as field experiments with Bluetooth receivers and senders were conducted to measure detection rates as a function of inquiry times.

In the laboratory test 1 (respectively 2) BTreceivers were stationary installed to find 1, 2, 4 or 6 BTsender within the detection range. The BTsenders were transmitting their signal continuously, whereas the BTreceiver(s) were periodically restarted after 10 seconds of being in inquiry mode. Every time, one of the BTsenders was detected, a data set including timestamp, BTsender-ID and signal strength value was stored to a log file.

Figure 9-5 illustrates the results from the laboratory test. For the varying number of BTreceivers and/or BTsenders the probability density is given. There you can see that more than 80% of all detections are realised within a time interval of 1 second (1000 milliseconds). For the probability density we looked at the intertimes. The intertimes are the time differences between a detection of a BTsender and the starting time of the inquiry mode of the BTreceiver respectively a previous detection time of that specific BTsender. That means the intertimes are exactly that times it took a BTreceiver to detect a BTsender provided it is in the detection range. In the laboratory test nearly 100% of all detectable devices were detected after 3 seconds independent of the number of BTreceivers and BTsenders.

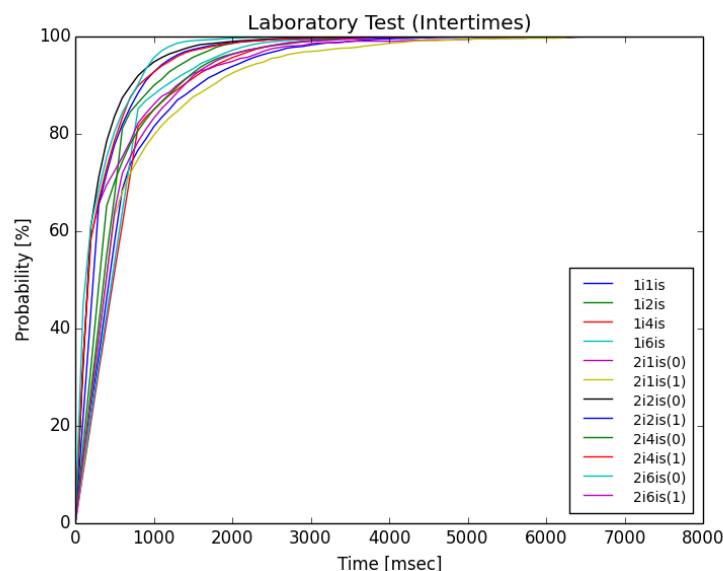


Figure 9-5: Probability density for the laboratory tests

In the laboratory test almost perfect conditions were given for the BTreceiver to detect BTsenders. The reality looks somewhat different – besides several error sources (e.g. Bluetooth signal reflections or shading effects) the to be detected BTsenders are moving objects which makes detection less likely since the BTsenders are not permanently within detection range. Furthermore, the speed factor reduces the time the BTsender is within detection range additionally.

To evaluate how the inquiry time process is influenced from real environment a two-hour field test which took place on August 20th 2013 between 6 a.m. and 8 a.m. was conducted. In that field test 4 BTreceiver objects (observer) in form of cars moving along the street Ernst-Ruska-Ufer (two lanes per direction; approximately 1.4km) on the so called WISTA area in Berlin-Adlershof were used (see Figure 9-6). Contemporaneously, these objects were considered as BTsender (i.e. the traffic participants), which should be detected by the other observers. Within the observer cars our prototyped Bluetooth monitoring systems (called Bluetooth-Box, shortened "BluB") was installed. The observers moved freely according to

their desired speed respectively to local feasibility and under consideration of the German Road Traffic Act (StVO).

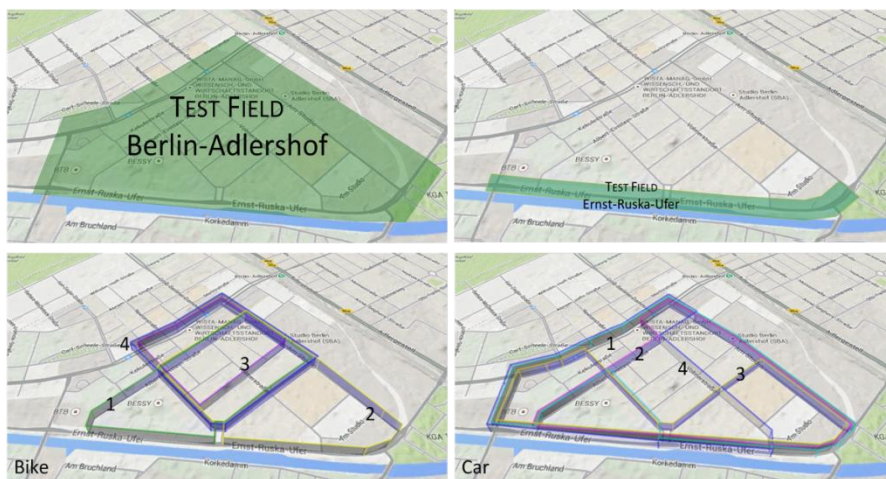


Figure 9-6: Defined field test observer routes (map source: Google): Ernst-Ruska-Ufer (upper right), car and cyclist routes (lower right and left) in second field test

The Ernst-Ruska-Ufer is both at once, public place and our DLR test track where additionally traffic monitoring infrastructure is installed to observe real-time traffic situations. Therefore, we could benefit from reference data collected from stationary Bluetooth detectors so that during the two hours two different types of data were collected. On the one hand, we monitored the traffic via stationary Bluetooth detectors. On the other hand, a mobile detection was done by our four moving observer vehicles. For both data types, the same data sets as in the laboratory test were stored containing timestamp, BTsender-ID and signal strength value.

The results of the stationary Bluetooth measurements are given in Figure 9-7. The left figure shows the results from the specific field test day (August 20th 2013). For higher reliability we permanently installed our BluBs at two points of the Ernst-Ruska-Ufer for several months so that we could benefit from long-term measurements. The right figure shows the results from the long-term measurements. It is obvious that the probability density is quite similar. Due to still undefined explicit error values the increase is less sharp in comparison to the laboratory data. Especially between 1 and 6 seconds the course of the function is smoother than that under laboratory conditions. Still unclear are effects where no inclination is observable within longer time slots (e.g. from 1500ms to 5000ms in the left figure and from 2000ms to nearly 4000ms in the right figure). It would mean that there were no inquiry times which are that long. Since there are even longer time periods in the data it might have a methodical reason.

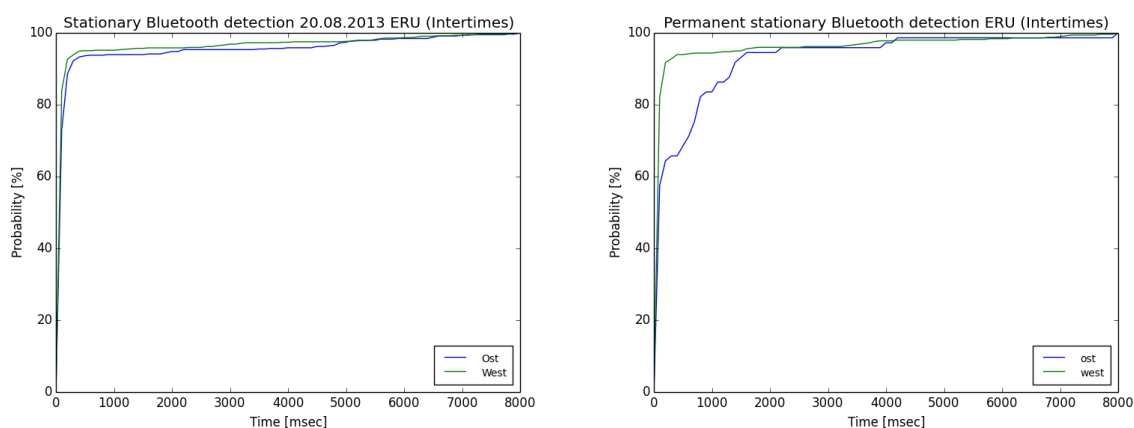


Figure 9-7: Probability density for field test results on Ernst-Ruska-Ufer (left: two-hour test on August 20th 2013; right: permanent stationary Bluetooth detection)

The results collected from moving Bluetooth observer vehicles are illustrated in Figure 9-8. The course of the functions is similar to that of the stationary Bluetooth measurements for all four observers even if that of observer car 1 seems to be more consistent. A reason might be the amount of collected data which was the biggest from car 1. Interesting is that the same effect of time slots without inclination is observable in that case as well. It seems to occur always between approximately 2000ms and 7000ms.

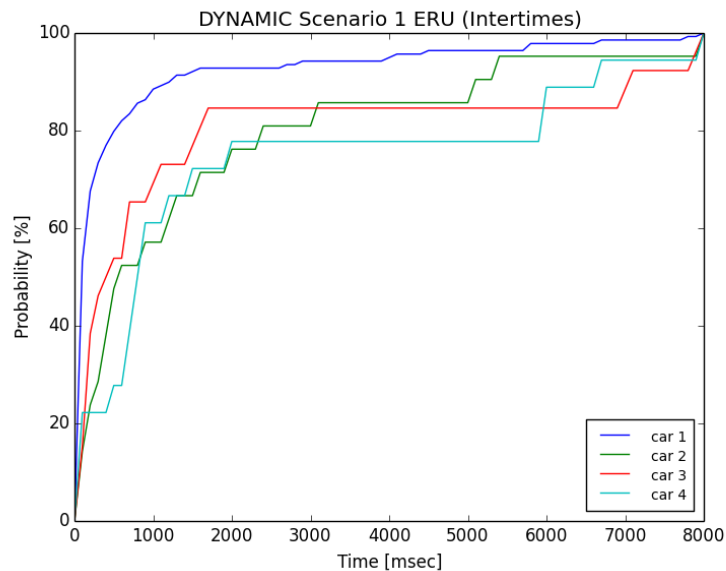


Figure 9-8: Probability density for field test results on Ernst-Ruska-Ufer using moving observer (cars)

In addition to the field test on Ernst-Ruska-Ufer, several test runs with 8 moving observers using multimodal BTreceiver objects (i.e. cars and cyclists) were conducted on other routes on the WISTA area (Figure 6) to see whether the results affirm our conclusion. These additional field tests took even place on August 20th 2013, but from 9 a.m. to 10 a.m. and 1 p.m. to 3 p.m. Note that in these field tests observer car 3 (red line) had some major problems in collecting data. Nevertheless the results from these area wide measurements show better accordance with the results derived from laboratory tests.

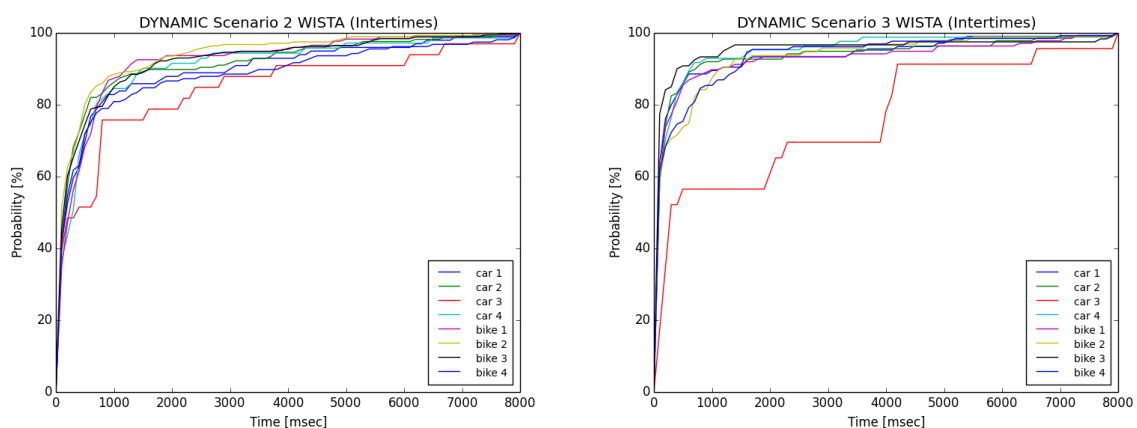


Figure 9-9: Probability density for additional field test results on WISTA area using cars and cyclists as moving observers

Figure 9-10 shows the results from the simulation scenario. For both simulated stationary Bluetooth detection units (modelling the East and West measuring bridges on the Ernst-Ruska-Ufer), the probability density to detect vehicles with Bluetooth devices on board within a specific time interval (in seconds) is given. One can see that in more than 80% the equipped traffic objects are discovered in a time interval less than one second. The results differ

between the two monitoring positions. That is possibly due to the jam occurring in the eastern part of the scenario which leads to far more (re-)detections of waiting vehicles in shorter time intervals. All in all, the density probabilities look very similar especially to the results from the real world stationary Bluetooth measurements (see Figure 9-7, cf. curve 'ERU_ost' and 'ERU_west' in the right figure).

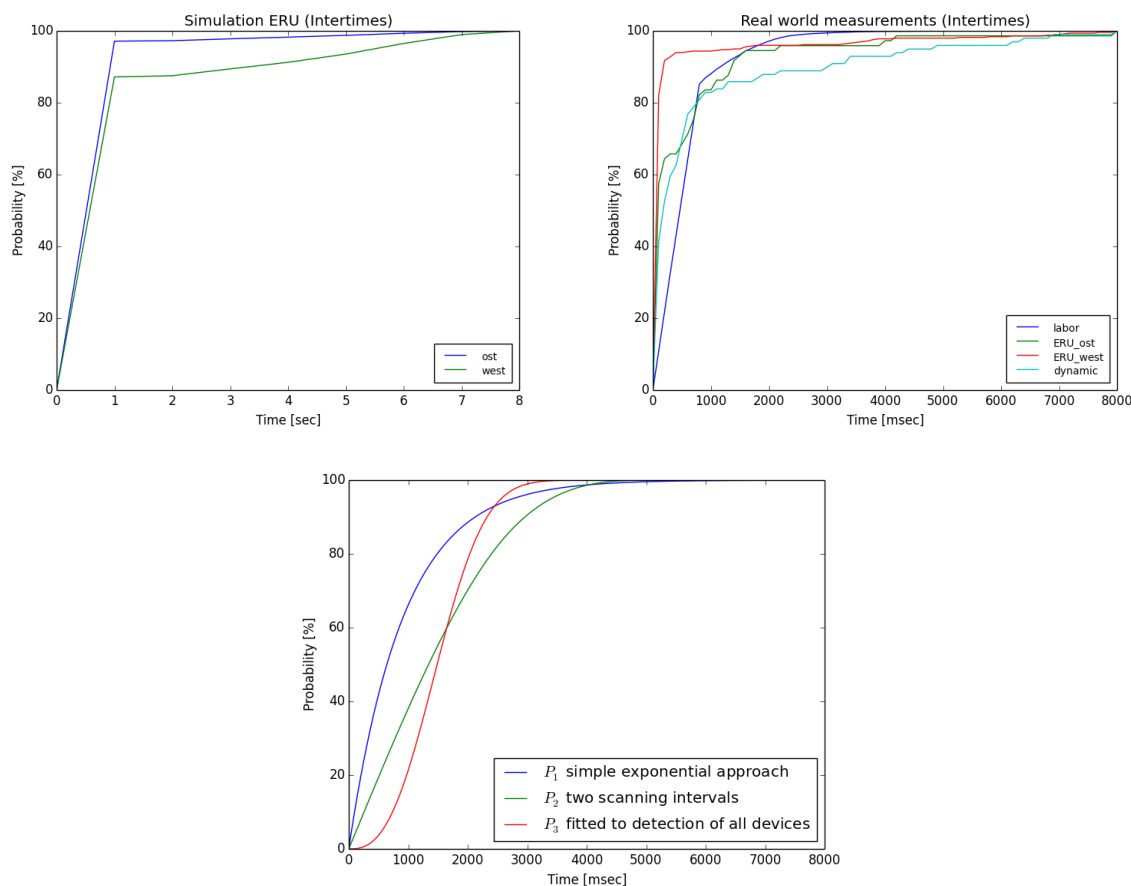


Figure 9-10: Probability density for simulation results (top left) compared to a summary of most important results from real world measurements (top right) and the theoretical results (bottom)

What we learn from this comparison is that the probability density seems to be best fitted by an exponential distribution which makes sense since the number of detections based on Bluetooth is a sequence of n independent seen/not seen trials each of which occurs with probability p . This follows from the assumption that the number of vehicles equipped with Bluetooth devices and the number of observer vehicles within the network is small, so that the chances to encounter are statistically independent events. Therefore, the existences of those encounter respectively detection events can be described using an exponential distribution.

9.8 Conclusions and Discussions

To sum up the following results can be stated from the experiments:

- The probability density seems to be best fitted by an exponential distribution.
- Within 1 second more than 90% of all detections are done under laboratory condition; in real environment conditions at least 80%. That means most of the detectable BTsenders in detection range are found within the first second.

- In case of moving observers field test results show better accordance with laboratory results than the stationary Bluetooth measurements. It has to be kept in mind that laboratory results reflect perfection.
- Simulation results fit the stationary Bluetooth monitoring results quite well.
- The simulations carried out are in good agreement with the empirical data as well as the theoretical model.

One weakness of our approach is that we can not detect the inquiry time directly but can only detect the interval between two successful inquiries, so in the case of small traffic densities we will need different measurements to validate our data. This will be a subject to further research. Additionally we need to investigate further the unusual plateau behavior in the dynamic cases (see Figure 9-8) where we often had no additional detections between second 2 and second 7.

9.9 Acknowledgements

We thank Daniel Krajzewicz for the initial implementation of the Bluetooth detection mechanism in SUMO and Andreas Lubert for performing the laboratory experiments and providing their data.

9.10 References

- [1] Ruppe, S., Junghans, M., Haberjahn, M., Troppenz, C.: Augmenting the Floating Car Data Approach by Dynamic Indirect Traffic Detection. In Proceedings of Transport Research Arena – Europe 2012, Athens, Greece (2012)
- [2] Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.: Recent Development and Applications of SUMO - Simulation of Urban MObility. International Journal On Advances in Systems and Measurements, 5 (3&4):128-138 (2012)
- [3] Franssens, A.: Impact of multiple inquirers on the Bluetooth discovery process – And its application to localization. University of Twente (2010)
- [4] Duflot, M., Kwiatkowska, M., Norman, G., Parker, D.: A Formal Analysis of Bluetooth Device Discovery. International Journal on Software Tools for Technology Transfer, Volume 8, Issue 6, pp 621-632 (2006)
- [5] Specification of the Bluetooth system, version 4.0, Bluetooth SIG, 2010, <http://www.bluetooth.com>.
- [6] SUMO - Simulation of Urban Mobility, available at <http://sumo-sim.org/>
- [7] Gurczik, G., Junghans, M., Ruppe, S.: *Conceptual Approach for Determining Penetration Rates for Dynamic Indirect Traffic Detection*. ITS World Congress 2012, Vienna, Austria (2012)
- [8] Peterson, B.S., Baldwin, R.O., Kharoufeh, J.P.: Bluetooth Inquiry Time Characterization and Selection. In IEEE Transactions on mobile computing, Volume 5, Issue 9 (2006)
- [9] Peterson, B.S., Baldwin, R.O., Kharoufeh, J.P.: A Specification-Compatible Bluetooth Inquiry Simplification. In Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS'04), Waikoloa, Hawaii (2004)
- [10] Hoyer R., Leitzke C.: Verfahrenstechnische Bedingungen für die Reisezeitbestimmung mittels Bluetooth-Technologie. In Proceedings of Heureka 2011, Kassel, Germany (2011)

- [11] Kasten, O., Langheinrich, M.: First Experiences with Bluetooth in the Smart-ITS Distributed Sensor Network. In Proceedings of 2001 International Conference on Parallel Architectures and Compilation Techniques (PACT'01), Barcelona, Spain (2001)
- [12] Wasson J. S., Sturdevant J. R., Bullock D.M.: Real-Time Travel Time Estimates Using Media Access Control Address Matching. Institute of Transportation Engineers Journal (ITE Journal), Volume 78, Issue 6, pp 20-23, June (2008)

10 Stochastic Optimization of Signal Plan and Coordination using Parallelized Traffic Simulation

Junchen Jin and Xiaoliang Ma¹

ITS Lab, Traffic and Logistics, KTH Royal Institute of Technology, Sweden

¹liang@kth.se

10.1 Abstract

Stochastic optimization algorithms are extensively used, together with traffic simulation model, to facilitate traffic signal planning for different policies goals, or objectives. However, the algorithms often require lengthy computation because so many simulation runs are needed for achieving statistically significant solution during optimization. This paper presents such a simulation-based framework that integrates the SUMO traffic model with an evolutionary optimizer and external emission estimator. It is used to evaluate optimal signal plans with respects not only to mobility measures but also indexes for sustainability, in terms of emission and fuel efficiency. Parallelization of simulation runs is implemented to reduce computational time during the optimization process. A case study is carried out and optimizes, using the simulation-based approach, signal control at a two-intersection network in Stockholm, where simple fixed-time logic is used to approximate real LHORVA control. The scenarios with and without coordination are analyzed when different policy goals in signal optimization are applied.

Keywords: Traffic signal planning; signal coordination; stochastic optimization; parallelized traffic simulation.

10.2 Introduction

Urban intersection is normally signalized in order to permit conflicting traffic movements to proceed safely and efficiently through space. Fixed-time (FT) control is basic and widely used control logic. Signal parameters in FT control are predetermined based on historical traffic data. The duration and order of phases are kept fixed in this logic. In practice, more advanced control logic has been applied such as vehicle actuated (VA) and adaptive signal control. For example, LHORVA is one most widely used approach in Sweden that belongs to the VA control. In reality, a platoon of vehicles, released from a signalized intersection, is often not completely dispersed before it arrives at the next intersection. Leading as many platoons of vehicles as possible to meet the green wave at next intersections, signal coordination has the capacity to further improve performance of traffic system [2].

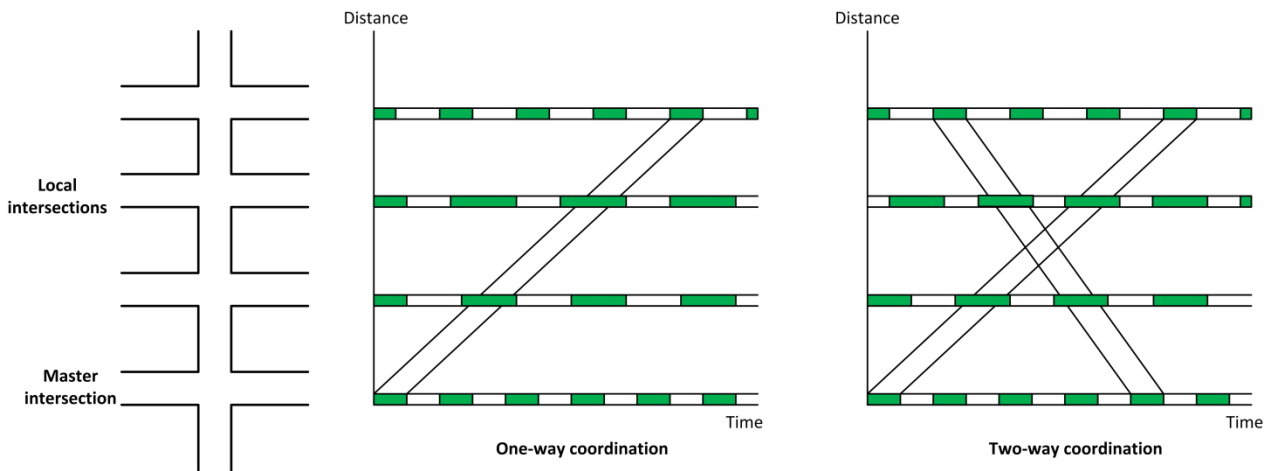


Figure 10-1: Configuration of traffic signal coordination.

Signal coordination is usually classified into two types: one-way and two-way coordination. Figure 10-1 illustrates the time-space diagrams of four coordinated intersections. The through-band is the strip bordered by black, indicating the length of time available for vehicles driving with a certain speed to proceed through a continuous series of green lights. In one-way coordination, through-band only exists under the situation that vehicle drive from master intersections towards other local intersections whereas dual-directional through-bands are observed in two-way coordination system. Nevertheless, it is more difficult to make specifications for two-way coordination than coordinating signals in one direction. Sometimes, only the direction with the higher flow is coordinated in practice. For example, the inbound directions can be coordinated in the morning while the outbound flows are coordinated during the evening.

One way to improve traffic mobility and sustainability in practice is to optimize signal parameters. Signal optimization approach was first proposed by Webster, who introduced a simple analytical model for minimizing travel delay experienced by drivers [3]. With the fast development of computer power and traffic modeling tools, micro-simulation based optimization becomes a favorable solution in practice of traffic planning and management. In the mean time, different planning strategies, in terms of policy goals, have to be considered by decision makers. Robertson proposed, in 1980, the concept of optimizing signal plans to reduce environmental impacts [4]. Recent years have seen many useful tools for calculating fuel consumption and emissions either at aggregate or detailed level. This provides opportunity to integrate traffic models with emission models to analyze environmental impacts according to activities of individual vehicle or fleet.

Microscopic traffic models are applied to describe complex traffic system when individual vehicle activities are of interest. Such a model represents detailed driving characteristics and interaction amongst different objects, described by instantaneous speeds, accelerations and decelerations, which are essential for modeling energy use and emissions of air pollutants. SUMO is a discrete-time based microscopic traffic model developed by Institute of Transportation Systems, German Aerospace Center [5]. It is not only an open source microscopic simulator but also a suite of applications that facilitate preparing and performing traffic simulation. SUMO is also highly portable and flexible for manipulating simulation via

socket connection interface (TraCI). It becomes popular in research community with the strong interoperability features.

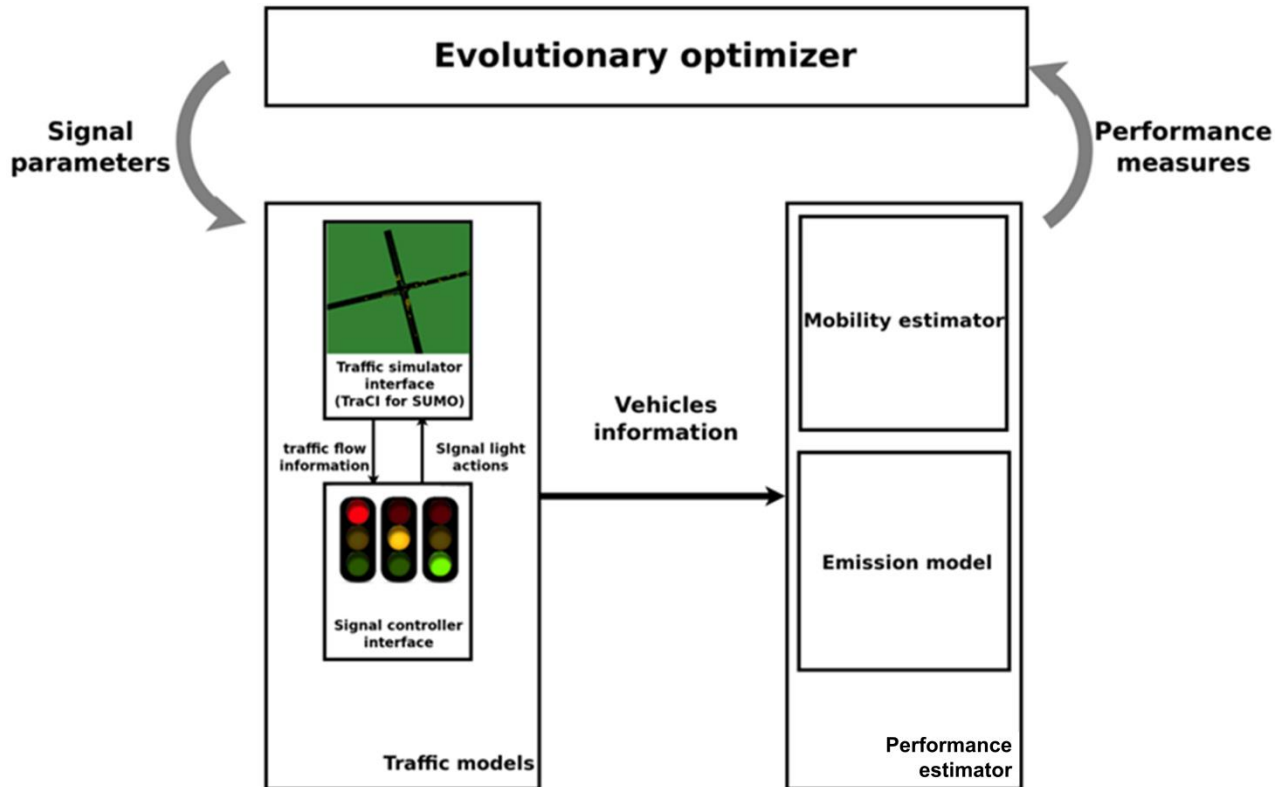


Figure 10-2: Computational engine to optimize signal parameters.

Emission models are developed to estimate the levels of vehicle exhaust emissions (e.g. CO_x , NO_x , HC etc.) and fuel consumption. In general, Comprehensive Modal Emission Model (CMEM) is a power-based model developed to estimate second-by-second emissions of vehicles using inputs of driving cycle [8]. Previous studies show that the microscopic CMEM model performs superior than other detailed emission models [6]. The comparison with field measurements found that CMEM exhibits better capacity in capturing HC and CO trends [7]. Hence, CMEM is applied in this study. Detailed driving characteristics, represented by, for example, vehicle category, instantaneous speed and acceleration, are considered as independent variables to calculate tailpipe emissions and fuel consumption.

Previous study has, for example, integrated the VISSIM traffic simulator, CMEM and a stochastic optimizer to optimize signal timings with respect to mobility performance and fuel consumption [9]. As reported, such stochastic optimization process takes a long time to converge to optimal solutions [10]. The computational requirement hence limits the application of the simulation-based optimization in traffic signal planning in practice. This study presents a model-based engine to generate optimal signal plans, but using SUMO as the traffic simulation engine. In particular, the flexibility of SUMO makes it possible to carry out parallel traffic simulation during signal optimization. This contributes to overcome the limitation of computational power in application.

10.3 Simulation-based signal optimization

A software framework is developed to integrate traffic models and evaluation estimator while applying an evolutionary optimizer to achieve a pre-defined signal control objective. Figure 10-2 shows that the optimization process starts with the evolutionary optimizer, randomly generating initial population of signal parameters. The signal plans are then sent to traffic simulation engine. In fact, implementation of signal control is embedded in traffic models in this framework. The basic idea behind is to give traffic simulator access, through an interface, to interact with signal controller online even when simulation is running. As SUMO is regarded as the simulation engine, the logic of signal control, including phase definition and duration definition, can be implemented using the Python language. During the simulation period, relevant second-by-second data for each vehicle is registered in memory. This dynamic vehicles information is then treated as the inputs for estimating performance indexes when the execution of traffic simulation is done. According to the predetermined optimization objective, performance indexes in mobility, emissions and energy are calculated and then summarized using the received vehicles information data. The estimated performance measures will be returned to the signal optimizer so as to generate new signal parameters for further optimization. The whole process is repeated until the termination criteria are finally met.

In parallel to gradient-based local optimization methods, evolutionary algorithms are a group of global search methods based on a metaphor of natural evolution process [11]. Genetic algorithm (GA) is one of the most widely used evolutionary optimizer in engineering field. This algorithm starts from a population of feasible solutions, moving towards the global optimum, while successive generations adapt from previous ones with parents of less fitness being selectively eliminated. GA performs the standard steps in selection, crossover, and mutation. In practice, GA might converge towards the local optimum or even an arbitrary point rather than the global optimum. The likelihood of such occurrence depends on the size of the space being explored. Srinivas [12] demonstrated that values of crossover probability and mutation probability play a significant role on the size of searching space. Usually, the higher the mutation probability is the bigger area is covered by the search space. Therefore, adaptive mutation probability is often applied in order to achieve better performance in real application. This strategy is also implemented in our application.

Pseudocode of the GA algorithm implemented in this study is summarized in Table 10-1. GA iteratively updates a population of individuals until a predefined number for evolutionary generations is finally reached. Fitness of each individual is estimated according to the performance indexes and defined integration. Thereafter, elite members (individuals with best fitness values) are kept directly to the new generation, preventing better organism from being eliminated. Binary encoding transforms signal parameters, from integer to bits string, before the rest of the genetic algorithm can be put to work. Relative better fitting individuals are selected to carry out crossover through selection. Tournament selection is applied in this study, involving running several competitions among a few randomly chosen individuals and winner of each tournament is picked out [13]. Portions of two parents from the current generation are combined to create two offsprings when crossover, such as uniform crossover, is carried out. Each part of the father's bit string is possible (with a fixed possibility) to be swapped with the counterpart of the mother's bit string [14]. Furthermore, bit-flip mutation operator goes through chromosome and randomly chooses bit to invert its value [15]. After producing new generations (crossover and mutation), chromosomes are inversely decoded from bit string to integer.

Table 1-1: GA-based evolutionary algorithm for signal plan optimization.

```

// Initialization:
 $n$  := number of parents per generation;
 $m$  := total number of generations;
 $k$  := generation id;
 $P_0$  := the initial signal parameter population (generation 0) randomly generated;
 $n_e$  := number of elitism;
 $n_c$  := number of crossover;
repeat
  // Evaluate:
  Run traffic simulation with signal parameters in population  $P_k$ ;
  Calculate fitness of each individual  $f(p_k)$ ;
  Sort parents by corresponding fitness values;
  // Elitism:
  Select best  $n_e$  parents of  $P_k$  and add to the next population  $P_{k+1}$ ;
  // Binary encoding:
  Convert parameters in the rest of  $P_k$  from integer to bit string;
  // Tournament selection:
  Select  $n_c/2$  pairs of parents in  $P_k$  to conduct crossover;
  // Uniform crossover:
  Produce offspring and add the offspring to  $P_{k+1}$ ;
  // Adaptive bit-flip mutation:
  Calculate adaptive mutation probability based on fitness values;
  Select members of  $P_{k+1}$ ;
  Invert a bit by flipping with the mutation probability;
  // Decoding:
  Decode the parameters from bit string to integer;
  // Increment:
   $k := k + 1$ ;
until  $k = m$ ;
return the fittest individual from  $P_m$ ;

```

The following GA operator parameters are implemented in our simulation driven optimization:

- The generation number is 90;
- Population size is 20;
- Elite number is 2;
- The tournament pool size is 4;
- Crossover number is 18;
- Uniform crossover probability is 0.50;
- The maximum mutation probability is 0.50 whereas minimum is 0.05.

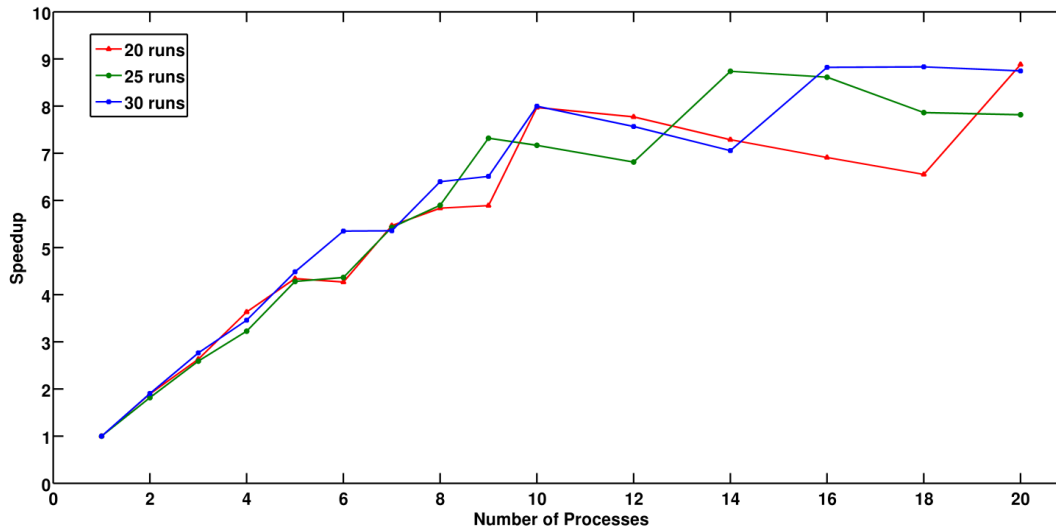


Figure 10-3: Scalability of parallel traffic simulations in the shared memory architecture.

10.4 Parallel traffic simulation

During the optimization process, traffic simulation runs have to be carried out for each individual signal parameter in each generation. What's more, multiple simulation runs are needed to make simulation-based evaluation statistically significant. For example, thirty simulation runs are required in order to make the t-test satisfying 95% confidence level. The optimization process hence becomes so expensive since the computational time increases dramatically when the number of parents and/or number of simulation runs increase. More importantly, traffic simulation consumes the majority of computational power because a large number of vehicles are modeled and simulated step by step. Besides, the I/O operations, when calling the CMEM emission software, require lots of computational resource.

Traffic simulation, the most computationally intensive part of this application, has the opportunity to be executed in parallel, as a single simulation does not depend upon other simulation runs. The application has the capacity to allocate multiple traffic simulation runs to different processes in systems with multiple processors or cores (SMP) while the emission calculation can be also distributed to other machines [16].

Figure 10-3 illustrates how multiple simulation runs are executed simultaneously in SMP. Multiple processes can operate independently but share the same memory resources. Specifically, one process handles with one simulation run with a particular random seed, used as the port id to connect TraCI server in SUMO. Corresponding performance measures are estimated after simulation is done for each process. The performance measures from simulation runs are finally summarized by a reduction operator e.g. taking average or worst result.

The study implements parallel simulations with shared memory architecture, a Windows server with two Xeon 2.40GHz quad-core processors and 8GB memory. Simulation runs are split up into evenly sized chunks while every process operates on a specific chunk. Figure 10-4 shows the speedup results from a comparative experiment study on parallel simulation, demonstrating how much faster parallel computation can bring over sequential simulation. The speedup curves grow in proportion to a logarithm of the number of processes (p) for small p in all three experiments. A speedup value of more than 8 times is achieved. This turns

out that the computational time is largely reduced by parallelized simulation in comparison to the execution time of the similar stochastic optimization process done early [17]. However, the speedup is still limited by the number of available processing units in machine when the value of p increases. For example, considering the case that 25 simulation runs are carried out, the speedup performance meets its threshold when the number of processes is larger than 14.

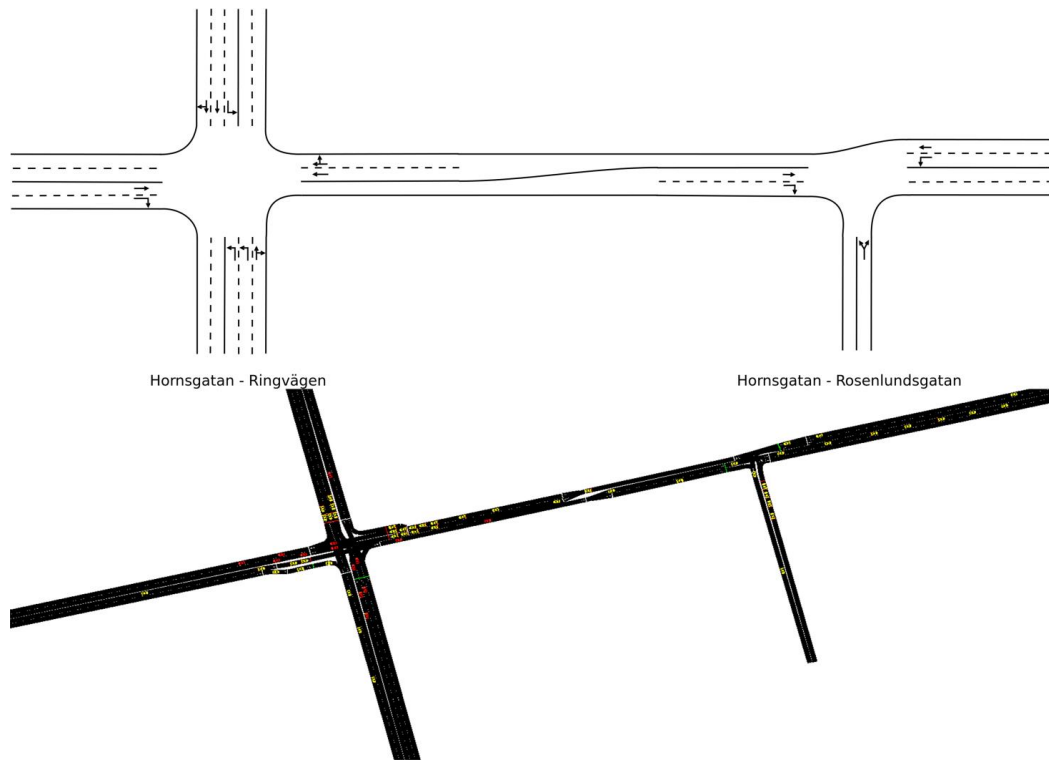


Figure 10-4: Layout of study intersections and its simulation model in SUMO.

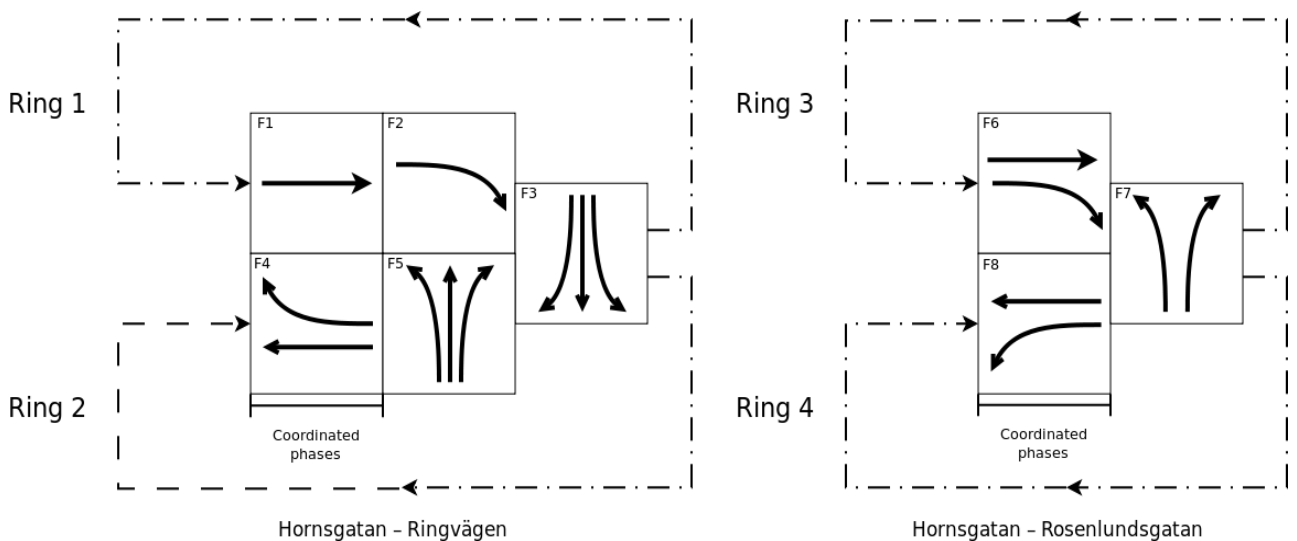


Figure 10-5: The phase rings for signal controls at the intersections in Stockholm.

Table 10-2: Notation for the mathematical formulation in signal optimization.

Parameters	Notations
P	penalty function, such as average delay, average fuel consumption etc.
\mathcal{X}	a vector of other inputs to traffic model, such as traffic demand
ρ	a vector of random seeds
y	a vector of yellow times
r	a vector of all-red times
ϕ	a vector of green times in non-coordinated control
φ	a vector of green times in one-way coordinated control
θ	offset
C	cycle length
$\phi_{s,j}$	duration of the green time for phase s at intersection j (no coordination)
$\phi_{s,j,-}$	minimum duration of the green time for phase s at intersection j (no coordination)
$\phi_{s,j,+}$	maximum duration of the green time for phase s at intersection j (no coordination)
$\varphi_{s,j}$	duration of the green time for phase s at intersection j (one-way coordination, $j=1$ is the master)
$\varphi_{s,j,-}$	minimum duration of the green time for stage s at intersection j (one-way coordination)
$\varphi_{s,j,+}$	maximum duration of the green time for stage s at intersection j (one-way coordination)
$\gamma_{s,j}$	duration of yellow time for stage s at intersection j
$r_{s,j}$	duration of all-red time for stage s at intersection j
S_j	total number of stages at intersection j
J	total number of intersections in traffic network

10.5 Case study

In order to demonstrate the stochastic optimization approach, this section presents a case study on signal planning of a pair of connected intersections in Stockholm. Since the LHORVA control logic applied is highly complex for optimization, FT control is used to approximate real signal timing in our study. Layout of these two intersections and corresponding simulation model in SUMO are illustrated in figure 10-4. In the figure, Hornsgatan is the connected arterial street between the studied intersections. Ringvägen is the road located left while the right side street is named as Rosenlundsgatan. The distance between the two intersection is around 270 meters. In addition, the Hornsgatan – Ringvägen intersection is defined as the master intersection in signal coordination system.

Real signal timing and phase sequence were observed, and then approximated by a FT control as a baseline case (see figure 10-5). The green times for Hornsgatan – Ringvägen intersection is approximated as 32, 18 and 15 seconds while green times for Hornsgatan – Rosenlundsgatan intersection is 56 and 14 seconds. The amber time is 3 seconds equally for both intersections. All-red times for clearance of vehicles between two stages are set equally as 2 seconds. Signals in the observed baseline case are coordinated, though not optimized, since the cycle lengths are identical for both intersections, and the offset is around 19 seconds.

When the framework is applied for signal optimization, all simulations are based on 60 minutes of traffic simulations, with additional 15 minutes for initial vehicle loading. Optimization process follows the GA-based evolutionary computation while different policy goals can be set in advance. Once the simulation-based optimization is completed, the best signal plan is evaluated through 30 randomly seeded SUMO simulations. Average statistics of all related performance measures are computed for comparison purpose.

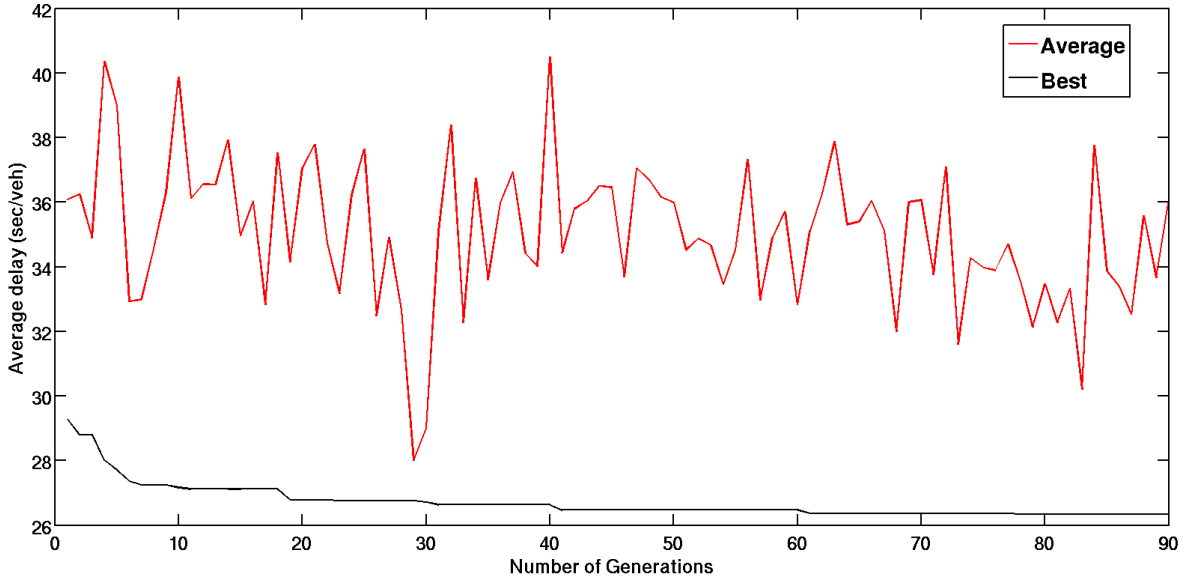


Figure 10-6: Optimization results of average delay.

10.5.1 Optimization problems

When signal plans for two intersections are not coordinated, the lengths of green phases are the variables to optimize. The amber times and all-red times are kept constants. The stochastic optimization problem can be simply represented by

$$\begin{aligned} & \min_{\phi} P(\chi, \phi, y, r, \rho) \\ \text{s.t.} \quad & \phi_{s,j,-} \leq \phi_{s,j} \leq \phi_{s,j,+} \end{aligned} \quad (2)$$

The notation is explained Table 10-2. In the case of coordination, two main requirements have to be met. The first one is that all traffic signals have to operate with the same cycle length. The other one is that the offset parameters, representing the beginning of green time relative to master intersection, should be identified in advance. In practice, the offset is often assigned to an ideal value (e.g. 25 seconds in our case) calculated by the distance between adjacent intersections divide by the average desired speed [18]. The optimization problem is therefore formulated by

$$\begin{aligned} & \min_{\phi} P(\chi, \phi, y, r, \rho, \theta) \\ \text{s.t.} \quad & j_{s,l,-} \leq j_{s,l} \leq j_{s,l,+} \\ & j_{s,j',-} \leq j_{s,j'} \leq j_{s,j',+} \\ & C = \hat{a}_{s-1}^{s_1} j_{s,1} + \hat{a}_{s-1}^{s_1} y_{s,1} + \hat{a}_{s-1}^{s_1} r_{s,1} \\ & j_{1,j} = C - \hat{a}_{s-2}^{s_j} j_{s,j} - \hat{a}_{s'}^{s_j} r_{s',j} - \hat{a}_{s'}^{s_j} y_{s',j} \\ & s \in [2, S_j], j \in [1, J] \end{aligned} \quad (3)$$

where green times at the master ($j = 1$) and slave nodes ($j = j'$) are all bounded; the cycle length is determined by the green time of different phases whereas the green time of different phases in the slave node is distributed based on the same cycle length.

Table 10-3: Results of optimal signal plans without coordination.

Objectives	Performances measures				
	Avg. delay (sec/vehicle)	Avg. Fuel (g/km)	Avg. CO (g/km)	Avg. HC (g/km)	Avg. NOx (g/km)
Baseline	29.425	90.868	4.257	0.143	0.205
Avg. delay	26.370	88.483	4.185	0.141	0.203
Avg. fuel	27.737	87.675	4.048	0.136	0.195

Table 10-3: Results of optimal signal plans with one-way coordination (offset 25 seconds).

Objectives	Performances measures				
	Avg. delay (sec/vehicle)	Avg. Fuel (g/km)	Avg. CO (g/km)	Avg. HC (g/km)	Avg. NOx (g/km)
Baseline	29.425	90.868	4.257	0.143	0.205
Avg. delay	25.651	88.216	4.253	0.143	0.204
Avg. fuel	27.135	86.627	4.034	0.136	0.195

10.5.2 Analysis of computational results

Figure 10-6 demonstrates how the best fitness value and average of fitness value evolve during a stochastic optimization process. Similar trend of convergence is also observed for the best fitness in other optimization runs. Table 10-2 summarizes performances measures for optimal signal plans when two intersections are not coordinated. The objectives in average travel delay and fuel economy are considered respectively. Performance indexes in travel delay, fuel economy, and emission factors are compared for optimal signal settings with different goals. The optimal signal plans show significant improvement over the baseline case with respects to all performance indexes. When comparing the performance results corresponding to optimal fuel economy and minimum average travel delay, average fuel consumption is about 1% higher when average delay is the objective. However, when average fuel consumption is regarded as an objective, average travel delay is about 5% longer than when it is minimized. The emission factors (g/km) are obviously reduced when fuel economy is optimized in comparison to the case of minimum travel delay. While it is relatively more difficult to improve fuel economy, the trade-off exists between travel delay and indexes for fuel and emissions. This indicates that goals of improving traffic system from both two perspectives at the same time are difficult to achieve.

The second scenario in the case study considers one-way signal coordination between the two intersections. Similarly, results for minimum travel delay and optimal fuel economy are analyzed. In dependent of whether travel delay or fuel economy is minimized, signal plan with coordination (offset 25 seconds) outperforms, though slightly, the case without coordination concerning both mobility and sustainability indexes. For example, a gain of 2.7% reduction in average travel delay can be achieved by coordination. The fuel efficiency improvement is about 1.2%.

Table 10-4 shows the impacts of coordination offset on optimal signal plans. When travel delay is set as objective, smaller offset will lead to reduced travel delay, but higher average fuel consumption. The same trend is observed for the case when fuel economy is minimized. This indicates that the tradeoff between average travel delay and fuel economy exists for, not only local signal settings but also coordination parameters.

Table 10-4: Impacts of offset on signal optimization.

Performances measures	Objectives					
	Avg. delay			Avg. fuel		
	20 sec	25 sec	30 sec	20 sec	25 sec	30 sec
Avg. delay (sec/vehicle)	25.091	25.651	25.830	26.477	27.135	27.307
Avg. fuel (g/km)	87.937	88.216	87.559	86.357	86.627	85.987

10.6 Conclusions

While SUMO becomes more and more accepted in the community of transport scientists, this study integrates it in a computational framework for optimal traffic signal planning. Parallelized traffic simulation is implemented to assess signal parameters, therefore accelerating the optimization process driven by genetic algorithm. The methodology is applied to evaluate signal control in a two-intersection network in Stockholm. The FT control strategies, with and without coordination, are analyzed for objectives of travel delay and fuel economy. The final conclusions are:

- Optimal signal plans for different goals outperform the baseline case with respect to performance indexes in mobility and sustainability;
- One-way coordination brings benefits in reduction of travel delay and fuel saving, though the gain is small for the case of FT control;
- Trade-off between average travel delay and fuel economy, also emissions, are apparent in the analysis not only for parameter settings at single intersection but also coordination between nodes;
- Parallelized SUMO simulation accelerates the stochastic signal optimization dramatically with symmetrical multi-processor computing (SMP). For this application, other high-performance computing technologies, such as graphics processing unit (GPU) computing, can be applied in the future.

10.7 References

- [1] U.S. Department of Transportation.: Traffic Signal Timing Manual. Federal Highway Administration (2008)
- [2] Gartner, N., Little, J., Gabbay, H.: Optimization of Traffic Signal Settings by Mixed-Integer Linear Programming: Part I: The Network Coordination Problem. *Transportation Science*, 9, 321--343 (1975)
- [3] Webster, F.: Traffic Signal Settings. Road research technical paper, Great Britain Road Research Laboratory (1958)
- [4] Robertson, D.I., Lucas, C.F., Baker, R.T.: Coordinating Traffic Signals to Reduce Fuel Consumption. TRL report LR934. Transport Research Laboratory, Crowthorn, Berkshire, United Kingdom (1980)
- [5] Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.: Recent Development and Applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5 (3&4), 128--138 (2012)

- [6] Ma, X., Lei, W., Andréasson, I., Chen, H.: An Evaluation of Microscopic Emission Models for Traffic Pollution Simulation Using On-board Measurement. *Environmental Modeling & Assessment*, 17(4), 375--387 (2012)
- [7] Nam, E.K., Gierczak, C.A., Butler, J.W.: Comparison of Real-World and Modeled Emissions Under Conditions of Variable Driver Aggressiveness. In: 82nd Annual Meeting of the Transportation Research Board, Washington, D.C. (2003)
- [8] Scora, G., Barth, M.: Comprehensive Modal Emission Model (CMEM) Version 3.01 User's Guide. University of California, Riverside (2006)
- [9] Stevanovic, A., Stevanovic, J., Zhang, K., Batterman, S.: Optimizing Traffic Control to Reduce Fuel Consumption and Vehicular Emissions Integrated Approach with VISSIM, CMEM, and VISGAOST. In: 88th Annual Meeting of the Transportation Research Board. Washington D.C. (2009)
- [10] Liu, Z.: A Survey of Intelligence Methods in Urban Traffic Signal Control. *International Journal of Computer Science and Network Security*. 7, 105--112 (2007)
- [11] Spall, J. C.: Introduction to Stochastic Search and Optimization: Estimation, Simulation and Control, 1st ed., New Jersey, US, John Wiley & Sons Inc (2003)
- [12] Srinivas, M. and Patnaik, Lalit M.: Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4), 656—667 (1994)
- [13] Miller, B.L., Goldberg, D.E.: Genetic Algorithms, Tournament Selection, and the Effects of Noise. *Complex Systems*, 9, 193-212 (1995)
- [14] Falkenauer, E.: The Worth of the Uniform. *Proc. Congr. Evolutionary Computation*, 1, 776–782 (1999)
- [15] Hinterding, R., Gielewski, H., Peachey, T. C.: The Nature of Mutation in Genetic Algorithms. In: Sixth International Conference on Genetic Algorithms (1995)
- [16] Ma, X., Huang, Z., Koutsopoulos, H. N.: Integrated Traffic and Emission Simulation: A Model Calibration Approach using Aggregate Information. *Environmental Modeling & Assessment*, in press (2014)
- [17] Ma, X., Jin, J. Lei, W., Robels, D.: Multi-Criteria Analysis Of Optimal Signal Plans Using Microscopic Traffic Models. — submitted for publication (2014)
- [18] Roess, R.P., William R.M., and Elena S.P.: *Traffic Engineering*, 2nd Ed. Prentice Hall, Upper Saddle River, New Jersey (1998)

11 DFROUTER – Route estimate method based on detector data

TeRon V. Nguyen¹, Daniel Krajzewicz², Matthew Fullerton¹, Son T. Mai³

¹Institute of Transportation, Technische Universität München, Arcisstraße 21, 80333 München
teron.nguyen@tum.de; matthew.fullerton@vt.bv.tum.de

²German Aerospace Center, Rutherfordstraße 2, 12489 Berlin, Germany
Daniel.Krajzewicz@DLR.de

³University of Munich, Oettingenstraße 67, 80538 München
mtson@dbb.fwi.lmu.de

11.1 Abstract

The contribution analyses, evaluates and improves the open-source “DFROUTER” tool contained as part of the SUMO traffic microsimulation suite. DFROUTER uses traffic volume detector values to calculate complete routes for vehicles through simulation networks. This approach is designed for highway corridors. The study analyzes DFROUTER’s current functionality and compares it with other similar approaches. The tool is also tested for different network types and data availability. Suggestions for the improvement of DFROUTER are made and tested to calculate routes more accurately. In addition, future areas for research are identified, such as flow computation in urban areas with two-lane ways.

Keywords: DFROUTER, route/demand generation, detector data, SUMO, inductive loop, OD matrix.

11.2 Introduction

Traffic congestion is a big problem challenging transport planners and traffic engineers worldwide. A wide range of measures are implemented to tackle this problem, in particular Intelligent Transport Systems utilizing state-of-the-art technologies and updated information for traffic control and management. Once the traffic data is collected, controls can be used in order to regulate traffic flow, for instance traffic rerouting, speed harmonization or incident warning, etc. These methods are designed to improve the safety, security, quality and efficiency of the transport systems as well as ensure a more optimal use of natural resources.

Traffic demand estimation is the key input for transportation system operation, design, analysis and planning [1]. Origin-destination matrices contain information about the spatial and temporal distribution of activities in different traffic zones in an area. Various methods have been developed for generating traffic demand like household survey, roadside interviews, license plate recognition and returnable-post card interviews [2]. However they are all expensive and obtaining the data is cumbersome [3]. Another method uses traffic counts to estimate the OD matrix given its great economic advantages. This data often comes from

inductive loop detectors, installed on the road surface to collect information such as vehicle type, volume, speed, and occupancy. In addition to establishing the traffic demand, the goal is to derive travel routes and an OD matrix estimation from detector data.

OD matrix estimation methods based on traffic counts have been developed over the last 30 years. There are two types of OD matrix estimation: the static method assumes OD flows are constant over time for determining an average OD demand for long-time transport planning and design purposes; whereas the dynamic method considers OD flows with time variation for short-term strategic traffic control and management [4]. It could be said that dynamic methods are an extension of static methods considering the time varying dimension. Furthermore, due to congestion effects, the OD matrix estimation can use proportional assignment (uncongested) or equilibrium assignment (congested), resulting in four basic cases of OD estimation [1].

We analyse and evaluate the “DFROUTER” tool contained as part of the SUMO traffic micro-simulation suite. DFROUTER uses detector values to calculate complete routes for vehicles through (primarily motorway/ corridor) simulation networks [5]. This tool also generates a demand for the traffic simulation in which each detector (inductive loop) location is considered as an observation point. Detectors are classified as either “source” (origin), “sink” (destination) or “between” (mere observation point between source and sink). The overall goal is that after running the simulation, similar values should be observed at the detection points in the simulation compared to the detector data used to generate the traffic demand and routing for the simulation. To this end, DFROUTER generates validation detectors for the SUMO simulation. Of course, a subsequent calibration of the driving behaviour model also plays a role in ensuring a good match between simulation and reality, this issue being particularly relevant in congested situations.

This paper provides a detailed (and until now, absent) description of the tool and compares it with other similar approaches that do not necessarily place restrictions on the network type. Special attention is given to the accuracy, performance, convergence and usability of the tool with diverse network sizes and forms. The flow probabilities are the decisive indicator for comparison.

In contrast to urban road networks where one origin-destination pair could involve several different routes in between; an OD pair in a highway corridor consisting of an on- and off-ramp has only one possible route. This less complicated characteristic facilitates the estimation of vehicle routes and traffic demand based on detector data. DFROUTER focuses mainly on highway networks.

11.3 Literature review

11.3.1 OD matrix estimation

Specifically for highways, the problem of determining an OD matrix from traffic counts can be formulated as follows:

$$\sum_i b_{ij} Q_i = O_j \quad (1)$$

$$\sum_j b_{ij} = 1 \quad (2)$$

Where: b_{ij} = proportion of trip from i to j ; Q_i = on-ramp counts (origin flows); O_j = off-ramp counts (destination flows).

Considering a sample highway section that illustrates the OD matrix estimation problem, Q_i and O_j can be seen in *Figure 11-1*.

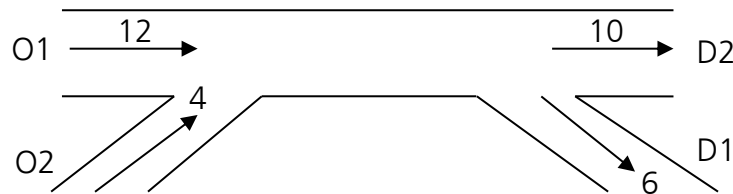


Figure 11-1: Sample highway segment

Some possible OD matrices could be:

Table 11-1: Possible OD matrices

	D1	D2	Sum
O1	8	4	12
O2	2	2	4
Sum	10	6	16

	D1	D2	Sum
O1	10	2	12
O2	0	4	4
Sum	10	6	16

	D1	D2	Sum
O1	6	6	12
O2	4	0	4
Sum	10	6	16

As seen in *Table 11-1*, several results could satisfy the requirements due to the under-specification problem: there are fewer equations than variables. The problem does not have a unique solution.

This trip demand estimation problem is therefore resolved by efficiently combining traffic count based data and all other available information [6]. There are huge number of OD matrix estimation techniques and the most popular static methods could be named as Information minimization (IM) and entropy maximization (EM) [2], Maximum likelihood (ML) [6], Generalized Least Squared (GLS) [3], Bayesian Inference approach [7], etc. Regarding dynamic methods, there are Cross-correlation matrices, Constrained optimization, Recursive estimation, Kalman filtering [8], Recursive least square [9], a Neural-network approach [10], and Combined estimators [11], etc.

11.3.2 DFROUTER

The DFROUTER routing module has been used since version 0.9.5 of the SUMO-package [12]. This is the tool designed specifically for highway scenarios based on the idea that most highways are well equipped with inductive loops, measuring each of the highways' entering and leaving flows. From this information regarding vehicle types, flows and speeds, DFROUTER is able to rebuild vehicle amounts and routes. DFROUTER was initially set-up as a minor tool (script) used within a larger system for generating routes for calibration purposes. The calibration was done by adding/removing vehicles to/from the simulation at the measurement points so as to match the real counts. Several relevant projects are the Weltjugendtag 2005 / 2006 World Cup in the city of Cologne or the VABENE project in Munich, etc.

"This approach is quite successful when applied to highway scenarios where the road network does not contain rings and the highway entries and exits are completely

covered by detectors. It fails on inner-city networks with rings and if the coverage with inductive loops is low" [13].

DFROUTER works by following several steps, being mainly:

- 1) Computing detector types
- 2) Computing routes
- 3) Computing flows
- 4) Saving flows and other values

The study mainly focuses on analyzing the computing flows step to understand its mechanism in delivering travel demand. The algorithm has not been documented before.

11.4 Methodology

The research methods focus on three key areas:

- 1) A more formal description of DFROUTER
- 2) How does the algorithm compare to similar approaches?
- 3) How could the algorithm be improved in order to estimate routes more accurately?

Analyzing DFROUTER

In order to analyze the algorithm, several abstract highway networks and simulated data sets ranging from simple to complex will be used. The four factors to be considered are network type, number of detectors, vehicle flows and routes. These elements will be altered to test the generated results based on the idea that the algorithm works well in simple cases but may have problems for the more complicated ones.

Beginning with 2 on- and off-ramps, the initial network is then developed to more complicated scenarios with extra ramps and lanes as well. The number of entrances and exits (origins and destinations) are also changed so as to test the number of routes generated. Basically there is one main highway line connected to several on- and off-ramps installed with detectors.

With regard to detectors, each type of detector will be omitted in order to test route and demand generation in case of a lack of detectors, from sink to in-between and source detector. The number of flows and routes used as input are also varied.

DFROUTER will generate routes/demand based on the same network as the SUMO run; therefore routes and vehicle flows are the main indicators when evaluating the tool. In general, the flows/routes/detectors generated by DFROUTER (2) must be identical to the initial input for SUMO simulation (1) as they are derived from one origin. The general framework for analyzing DFROUTER is shown in *Figure 11-2*.

Comparison with different approaches

Similar approaches are described and compared with DFROUTER algorithm. In order to do that, the same networks will be created for route estimation with different approaches that

do not necessarily place restrictions on the network type. The main indicator for performing comparisons is route probability.

Improvement of the algorithm

In this step, changes to the algorithm will be proposed and implemented. The adjustment is expected to produce better results compared to the current DFROUTER.

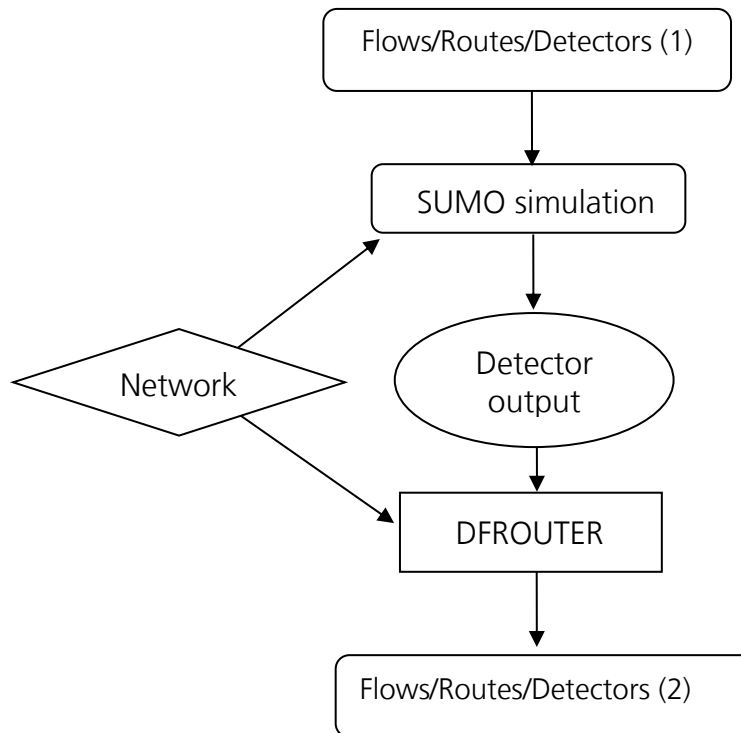


Figure 11-2. Framework to analyze DFROUTER tool

11.5 Results

This part describes results from the testing of abstract highway corridors, comparison with similar approaches as well as ideas for algorithm improvement.

11.5.1 Scenario testing

Three abstract highway corridors are selected for examination, ranging from very simple to complicated.

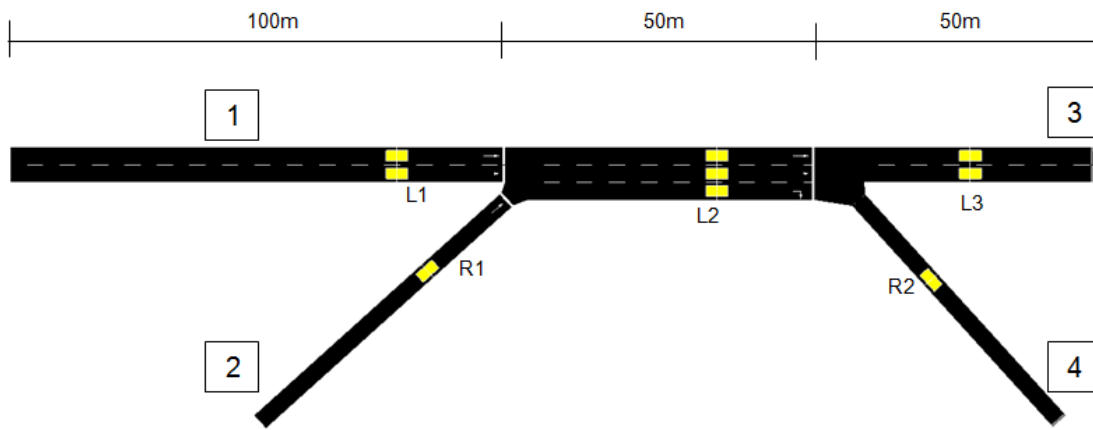


Figure 11-3: CASE 1 – 2 origins, 2 destinations

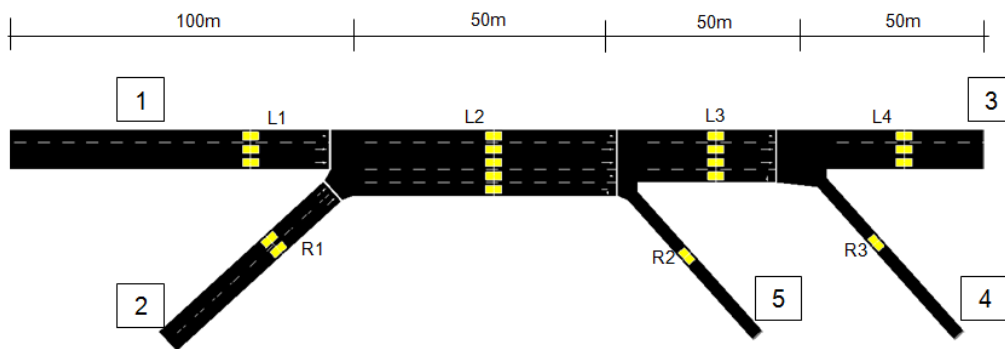


Figure 11-4: CASE 2 – 2 origins, 3 destinations

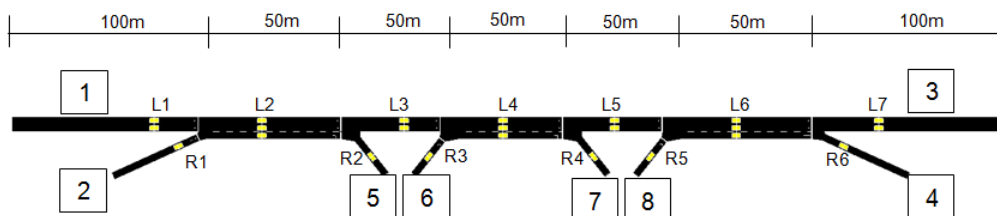


Figure 11-5: CASE 3 – 3 origins, 3 destinations

Observing the output generated from DFROUTER in the 3 cases when compared to initial input has indicated that:

- The algorithm works well whenever the network is fully covered with detectors and generates routes comprising all OD pairs. The algorithm could not detect that some routes were absent; e.g. in one scenario of CASE 2 there were only 4 routes but DFROUTER created 6 routes, which consist of all possible connections.
- Missing of in-between detectors in 3 cases does not cause a big estimation problem as long as the source and sink detectors are present. This shows that the in-between detectors do not play an important role in the probability estimation procedure.
- Basically the estimated probabilities are identical to flow proportions at destinations, therefore sink detectors are the decisive elements in flow computation.

Additional analysis of DFROUTER's C++ code has shown the main steps of the algorithm performing OD matrix estimation are:

- *Step 1:* For all routes starting from source or between detectors to sink detectors → determining split edges (the legs go out of a junction) having detectors on them.
- *Step 2:* Calculating the proportion of flow on split edges using detector data → each split edge contains a different probability, the others have probability = 1.0 as default.
- *Step 3:* For routes starting from source detectors → calculating destination distribution by multiplying all flow probabilities on all edges constructing that route.

From this, pros and cons of the algorithm could be described as follows:

Pros: the algorithm is simple for route calculation that uses only data from detectors laid on split edges by taking the destination proportion as route probability. If all sink detectors are supplied, the flows should be replicated correctly. Testing results in a highway corridor have shown that the algorithm therefore requires basically only source and sink detector data to generate relative route distribution without considering in-between detector data before the edge splitting. The in-between detectors which lie on split edges play an important role in determining split probability. Therefore another kind of in-between detector (not on split edges) could be missing, which does not affect the calculation procedure. This algorithm works perfectly in a limited condition when there is only one origin and the network is fully covered by detectors. However this is a rather rare case as only one inflow or origin is uncommon in reality.

Cons: this simple algorithm could not work successfully in the case of missing detectors, especially detector data on split edges (in-between or sink detector) as it is not able to guess the missing data. It does not work if not all "ends" are covered with detectors. There are actually existing cases in which missing data can be generated from the available ones, for example the absent flow from only one detector could be deduced by subtraction of all inflows to all outflows. In this case the probability of missing flow would not be overestimated to 1.0 as default.

In fact, many OD matrices can be produced from one set of traffic counts in this case as it is an under-specified problem, meaning the number of OD variables is greater than the number of independent constraints. Additional information like a priority matrix or a specific route assignment is required to compute routes more appropriately.

11.5.2 Comparison with similar approaches

DFROUTER generates route/demand data based merely on proportions of flows on split edges. The destination distribution is an average result of different *timeOffset* calculations using a default interval of 60 seconds. Congestion effects and travel time between origin and destination are also not considered. This method is most likely to work for the static OD estimation method mentioned above (a workaround would be to run DFROUTER multiple times with data split into intervals for which routes are desired, e.g. 15 or 60 minutes); however the algorithm considers only constraints between link flows (sum of all link proportions equal to 1.0 in case of full detector coverage) but not any optimization function (e.g. minimization differences between estimated and observed link flows).

In order to compare DFROUTER algorithm with similar approaches, the same highway corridor is examined and outflow probabilities are compared. The test case in *Figure 11-6* comprises detector data and highway network CASE 2 (*Figure 11-4*) as shown in Table 11-2.

Table 11-2: Detector data on Test case

Item	Value
Section length	100, 50, 50, 50
On-ramp counts	280, 180
Off-ramp counts	70, 120, 270
Mainline counts	280, 460, 390, 270

The **DFROUTER** algorithm calculates flow probability for each of the split edges by examining the outflows of each junction considering off-ramp counts and mainline counts, e.g 70/460, 390/460, 120/390, 270/390 (equal to 0.15, 0.85, 0.31 and 0.69 respectively).

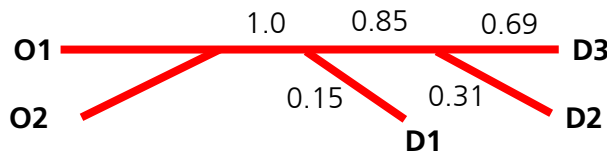


Figure 11-6. Test case configuration

The destination distribution can be obtained by multiplying the available probabilities on each route departing from a source detector as shown in Table 11-3.

Table 11-3: The DFROUTER OD matrix

O\D	D1	D2	D3
O1	$=1.0 * 0.15$ $=0.15$	$=1.0 * 0.85 * 0.31$ $=0.26$	$=1.0 * 0.85 * 0.69$ $=0.59$
O2	$=1.0 * 0.15$ $=0.15$	$=1.0 * 0.85 * 0.31$ $=0.26$	$=1.0 * 0.85 * 0.69$ $=0.59$

This OD matrix estimation method can be compared to similar approaches, which generate traffic demand without taking into account an optimization function, such as the equally split OD matrix, proportional OD matrix, iterative method, the gravity model and turning percentages.

1) The Equally Split OD matrix

This is the simplest method for seed generation. As the name suggests, an equal proportion is assigned to all destinations. In the Test case (*Figure 11-6*) with 3 destinations, the method concludes that D1, D2 and D3 are equally likely for trips from origin O1 and O2, so the proportion will be 1/3 (33.3%).

Table 11-4: The equally split OD matrix

O/D	D1	D2	D3
O1	1/3	1/3	1/3
O2	1/3	1/3	1/3

2) Proportional OD matrix

This is the common and oldest method to estimate an OD matrix [14]. It is based on the concept that the attraction of any destination is the function of the number of trips that end at that destination. In other words, higher attraction is at a destination having higher flow proportion and vice versa. The origin flow will hence be distributed according to destination flows.

Considering the Test case (Figure 11-6) with destination flows collected at D1, D2, D3 are 70, 120, 270 veh respectively, the proportional OD matrix can be computed manually as follows, which is identical to DFROUTER calculation.

Table 11-5: The proportional OD matrix

O/D	D1	D2	D3
O1	$=70/(270+120+70)$ =0.15	$=120/(270+120+70)$ =0.26	$=270/(270+120+70)$ =0.59
O2	$=70/(270+120+70)$ =0.15	$=120/(270+120+70)$ =0.26	$=270/(270+120+70)$ =0.59

3) Iterative method

This is considered a hybrid proportional assignment technique that balances both inflows and outflows [14], adopted from Wills and May (1981) based on an iterative fitting algorithm. The algorithm computes each OD cell iteratively until convergence is reached. The algorithm steps are given below.

Step 0 Set $k = 0$

$$T_{ij}^{(0)} = \begin{cases} 1 & \text{for all possible interchanges} \\ 0 & \text{for all impossible interchanges} \end{cases}$$

$$\text{Step 1} \quad \text{Set } T_{ij}^{(2k+1)} = \frac{O'_i}{\sum_j T_{ij}^{(2k)}} T_{ij}^{(2k)} \quad \text{for all } i, j \quad (3)$$

Where O'_i is the observed volume at point i adjusted for all known demands from i .

$$\text{Step 2} \quad \text{Set } T_{ij}^{(2k+2)} = \frac{D'_j}{\sum_j T_{ij}^{(2k)}} T_{ij}^{(2k)} \quad \text{for all } i, j \quad (4)$$

Where D'_j is the observed exit volume at point j adjusted for all known trips that end at j .

Step 3 If $T_{ij}^{(2k+2)} - T_{ij}^{(2k)} < \delta$ for all i, j then STOP

Else set $k = k + 1$ and go to Step 1

Using these algorithm steps to compute the OD matrix for the test case (Figure 11-6) yields the results shown in Table 11-6. The algorithm produced a converged output after 2 iterations.

Table 11-6: The iterative OD matrix estimate

O/D	D1	D2	D3
O1	0.15	0.26	0.59
O2	0.15	0.26	0.59

The final iterative OD matrix estimation, however, contains the same values as that of the proportional OD matrix estimation. This could be because the method computes OD elements iteratively but does not consider any constraint such as distance or travel time as a deterrence function. The following gravity model will take these parameters into account.

4) The Gravity model

The gravity model is one of the oldest trip distribution methods widely used in macroscopic modelling. This model is also extended to estimate the trip proportion between ramps, in which the impedance function is the main parameter of this model, which was proposed by Nancy Nihan[15]. It is related to the concept that the probability of very long and very short trips is low on the freeway. The model is based on the Gamma distribution as follows:

$$F_{ij} = \frac{\beta^\alpha}{\Gamma(\alpha)} d_{ij}^{(\alpha-1)} e^{-\beta d_{ij}} \quad (5)$$

Where F_{ij} is the travel propensity factor between ramp i and j ; α = shape factor $\cong 3.0$ for the highway; β = size parameter = $\alpha/\text{avg. trip length}$; d_{ij} = distance between pair (i, j) ; avg. trip length = $(1/T) * \sum(\text{Link length}) * (\text{Link volume})$; T = sum of all trips generated

The cell entries in the OD matrix are defined as

$$T_{ij} = \frac{b_j F_{ij}}{\sum_j b_j F_{ij}} O_i \quad (6)$$

Where T_{ij} = trip interchange between pair (i,j) ; b_j = balance factor from iterations; O_i = production at i ; D_j = attraction at j

Subject to the constraint: $\sum_i T_{ij} = D_j$

In the implementation of the algorithm, the balancing factor was ignored for the first iteration. The average trip length from the geometry = $(100*280+50*460+50*390+50*270)/(280+180) = 183$. Then parameter $\beta = 3/183 = 0.016$. Using these parameters and the distance matrix, the OD matrix results are calculated accordingly.

Table 11-7: The Gravity model OD matrix

O/D	D1	D2	D3
O1	0.4312	0.3371	0.2317
O2	0.2222	0.3909	0.3869

The resulting OD matrix is different from both that of the proportional method and DFROUTER as it is only the first iteration in the case of ignoring balance factor b_j .

5) The Turning Percentage

This is the most intuitive method of estimating an OD matrix for a freeway section, which is based on turning percentages. Similar to the equally split and proportional OD estimate, it is assumed that turning percentages at any given off-ramp are independent of the trip origin [14]. Therefore the OD matrix is derived by tracking the turning percentages in each section. Using the example corridor Test case (Figure 11-6), there are 4 sections, each between on-ramp and off-ramp, with turning percentages as follows: 0, 15.2, 30.8 and 100 (0, 70/460, 120/390, 270/270 respectively). The resulting OD matrix is shown in Table 11-8.

Table 11-8: Turning percentage OD matrix

O/D	D1	D2	D3
O1	$= 1 - 0.15 - 0.26$ $= 0.59$	$= 0.31 * (1 - 0.15)$ $= 0.26$	0.15
O2	$= 1 - 0.15 - 0.26$ $= 0.59$	$= 0.31 * (1 - 0.15)$ $= 0.26$	0.15

In summary

The equally split OD matrix method did not generate a plausible result.

Due to the missing balance factor b_j , the gravity model has not been examined thoroughly and produced rather incomplete output in the first iterative calculation. However the accuracy of the estimation procedure depends heavily on the treatment of external stations [15], as it is related to the distance between different OD pairs. This approach is not suitable for application to DFROUTER, as the tool generates routes arbitrarily depending on network topology. Therefore the route lengths are variant.

Similar OD matrices were achieved from various approaches: DFROUTER, proportional OD matrix, iterative method and turning percentage. The comparison results also indicate that DFROUTER is working most similarly to the turning percentage approach as it takes each flow proportion at each split edge into consideration.

Concerning the iterative method, it does not take distance, time or any deterrence parameter into account, but only performs iteration based on the number of origin and destination counts. The results therefore are proportional to these counts.

Furthermore, DFROUTER and the proportional OD matrix also have similar working mechanisms. Considering a tree graph as follows including 1 origin and 7 destinations where a, b, c, d, e, f are the relative detector data on edges.

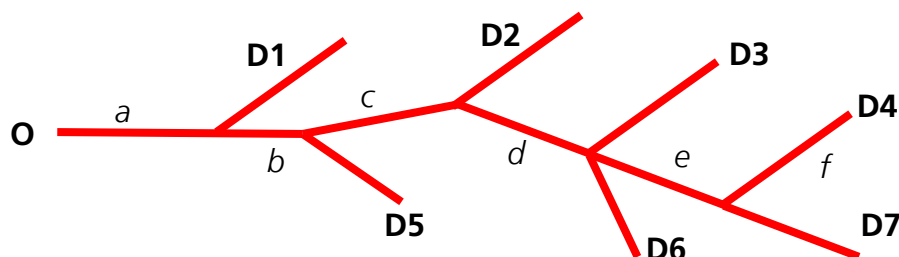


Figure 11-7: A tree graph

Then the flow probability at each destination, e.g D4, is computed as:

- DFROUTER: $pro = \frac{b}{a} \times \frac{c}{b} \times \frac{d}{c} \times \frac{e}{d} \times \frac{f}{e} = \frac{f}{a}$
- Proportional OD matrix: $pro = \frac{f}{\sum_{j=1}^n D_j} = \frac{f}{\sum_{j=1}^n O_i} = \frac{f}{a}$

From above, it could be said that for the case of one origin, DFROUTER and the proportional OD matrix are basically the same in their approach. The proportional OD matrix works more simply than DFROUTER as it does not take into account in-between detectors or split edges; only the data at sink detectors are required for calculation. There is another computer model named SYNOD having been developed to synthesize the required OD matrix based on proportional OD matrix approach. This simple proportionality scheme on the other hand is considered as a crude approximation that has the problem of over-predicting the number of very short and very long trips with 20-30% level of error as in[15].

Due to the drawback of these methods, they are often used to generate a starting solution (seed or target, a priori matrix) for the OD estimation problem to solve the minimization function of difference between estimated and observed link flows or OD matrix [14].

11.5.3 Suggestions for DFROUTER improvement

From the analysis of DFROUTER, especially its pros and cons, there are several improvements to the algorithm that could be considered:

- Guessing missing data based on existing detector flows. This could be done by considering the relationship between all inflows and outflows at a certain junction.
- Improving DFROUTER's operation for the case of highway rings or a fully covered urban intersection.

The most promising improvement is to guess the missing data on one of 2 (or several) split edges. By doing this, DFROUTER can perform well in case of not all "ends" being covered with detectors and the overestimation problem of the current DFROUTER that assigns probability = 1,0 as default for missing detector data can be eliminated.

1) Calculating missing data

At this moment the current algorithm only takes the split edges which have a detector on them into the calculating list and omits the ones without a detector. This problem could be solved by the following proposed algorithm:

Step 1: Calculate the flow value on each edge of the highway network using backward or forward recursion .



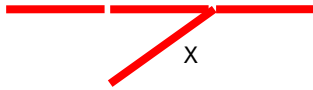

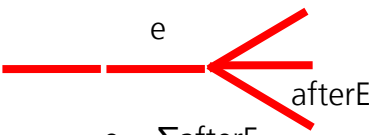
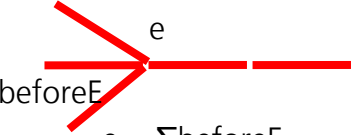
Step 2: For all routes starting from source or between detectors to sink detectors → determining split edges after a junction.

Step 3: Calculating flow proportion of split edges based on computed flow → each split edge contains different probability.

Step 4: For only routes starting from source detectors → calculating destination distribution by multiplying all flow probabilities on all edges constructing that route.

Of all steps above, the first step is the most challenging as there are many scenarios to consider. Edge e is the one that needs have a value computes. Forward recursion will be performed when there is no detector before e and backward recursion works in the opposite way.

Table 11-9: Cases to consider in the recursion algorithm

	Recursion forward	Recursion backward
1	$\text{e} \quad \text{afterE}$  $\text{e} = \text{afterE}$	$\text{beforeE} \quad \text{e}$  $\text{e} = \text{beforeE}$
2	$\text{e} \quad \text{afterE}$  $\text{e} = \text{afterE} - x$	$\text{beforeE} \quad \text{e}$  $\text{e} = \text{beforeE} - x$
3	e  $\text{e} = \sum \text{afterE}$	beforeE  $\text{e} = \sum \text{beforeE}$

If the algorithm could not figure out the value after a certain number of recursions, its probability will be returned to 1.0.

2) Application in an abstract network

A hypothetical highway network was developed so as to test the improved algorithm, therefore it should contain all cases as required in *Table 11-9*. There are only 7 detectors at the location of L1, L9, R1, R2, R5, R66, R7 and the rest are missing, including some in-between and sink detectors. The input probabilities will be used to compare with DFROUTER's output.

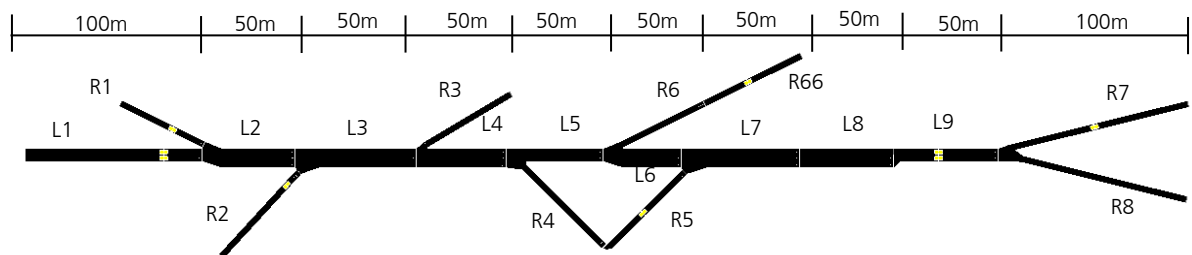


Figure 11-8: Abstract network with missing detectors

In order to calculate flow at a certain edge, the recursion function will be used, be it forward or backward. For instance, R3 will be computed as follows:

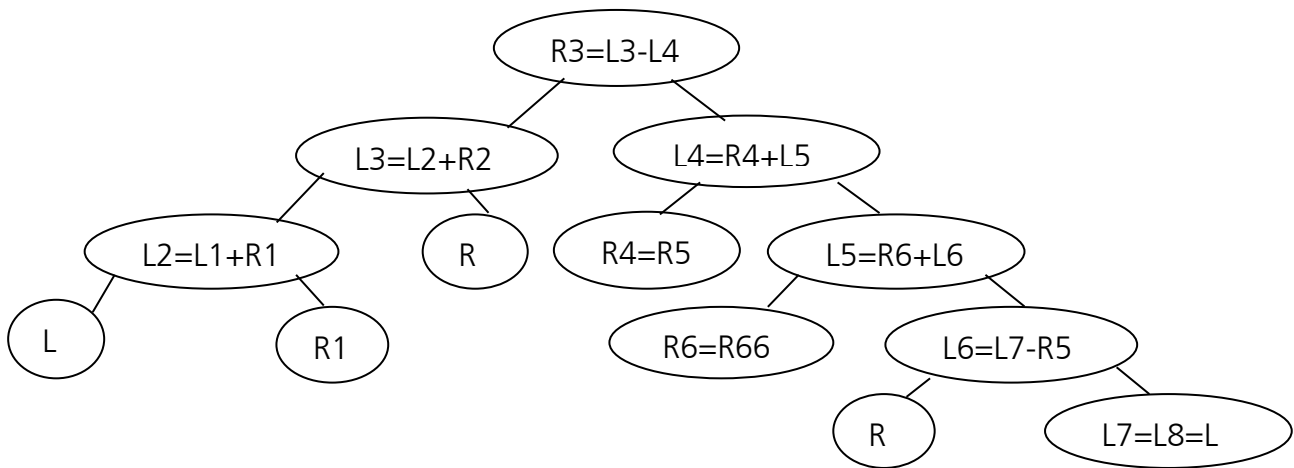


Figure 11-9: Example of calculating R3

Calculated results and their comparisons are shown below regarding both probabilities at destinations (all “ends”) and the original input as well. The differences are evident and significant as the original DFROUTER does not consider missing data at destinations. The probabilities generated by the improved DFROUTER are approximate to the destination probabilities and also more accurate compared to that of the original DFROUTER.

Table 11-10: Comparison of the destination probabilities of the original and the improved algorithm

Trip	Des- counts	Des- Pro	Probability		Relative error	
			DFROUTER	Improved DFROUTER	DFROUTER	Improved DFROUTER
From L1/R1/R2 to R3	900	0.24	1	0.23	3.22	-0.03
From L1/R1/R2 to R66	500	0.13	0.14	0.13	0.06	-0.01
From L1/R1/R2 to R7	1100	0.29	0.69	0.28	1.38	-0.03
From L1/R1/R2 to R8	700	0.18	0.69	0.20	2.75	0.09
From L1/R1/R2 to R7_1	300	0.08	0.69	0.09	7.74	0.14
From L1/R1/R2 to R8_1	300	0.08	0.69	0.06	7.74	-0.24

3) Application in a larger network

The improved algorithm was tested successfully in the case of an abstract network in Figure 11-8 which generates exactly the same results as the corridor covered fully with detectors. In this step, a larger network containing 3 main interchanges in Nuremberg was converted from OpenStreetMap data.

Each interchange is equipped with different numbers of detectors:

- Interchange 1: fully covered with detectors and there are 5 routes as an input to SUMO
- Interchange 2: only detectors in main corridor, only 1 route toward interchange 1
- Interchange 3: only detectors in main corridor, only 1 route toward interchange 1

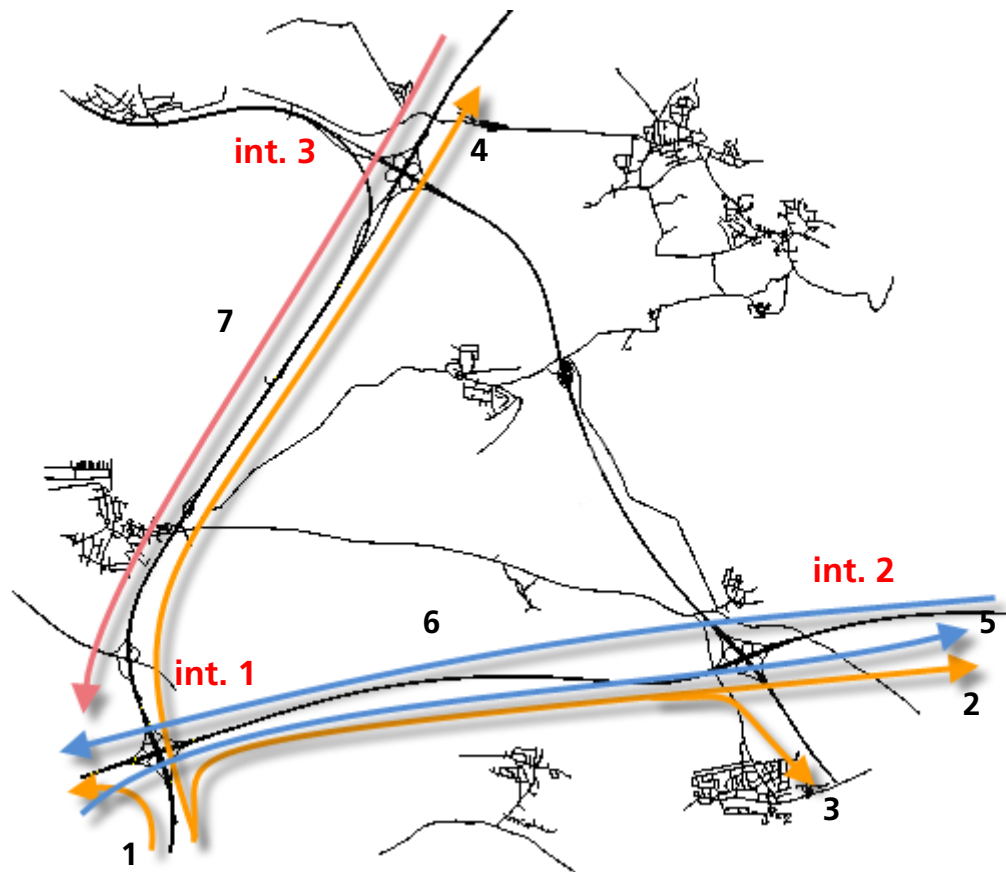


Figure 11-10: Nuremberg highway network

Table 11-11: Comparison of DFROUTER and improved DFROUTER results with input probabilities

No	Trip	Colour	Input probability	Probability		Relative error	
				DFROUTER	Improved DFROUTER	DFROUTER	Improved DFROUTER
1	From 1 to 1 left	Yellow	0.16	0.16	0.16	0.00	0.00
2	From 1 to 2 straight1	Yellow	0.13	0.06	0.18	-0.54	0.38
3	From 1 to 2 right	Yellow	0.09	0.22	0.04	1.44	-0.56
4	From 1 to 3	Yellow	0.63	0.24	0.62	-0.62	-0.02
5	From 1 to 2 straight2	Yellow	1	0.28	0.82	-0.72	-0.18
6	From 2 to 1	Blue	1	1	1	0.00	0.00
7	From 3 to 1	Pink	1	0.4	0.86	-0.60	-0.14

For this complicated scenario, it takes around 5 minutes for the tool to run and perform recursion (on a laptop with an 2GHz CPU and 4.00GB of RAM). The flow probabilities produced by the improved DFROUTER are different from those computed by the original DFROUTER as shown in the *Table 11-11*. More reliable and accurate results from the improved DFROUTER are achieved as expected.

11.6 Conclusion

The study has been conducted to analyze the DFROUTER tool within the SUMO suite by examining several typical highway corridors as well as investigating the nature of DFROUTER, especially demand generation. The algorithm has been also compared with other similar approaches based on the same abstract highway corridor resulting in more understanding of its relationship to these relevant algorithms. The study has also focused on proposing an improved algorithm, which generates traffic demand/flow probabilities more accurately. The study sought to answer these questions:

- 1) How can DFROUTER be formally described?
- 2) What are the differences to other approaches?
- 3) How could the algorithm be improved in order to estimate routes/demand more accurately?

The literature review has indicated two main groups of OD estimation: static and dynamic, which have been developed over the last 30 years. These methods are fundamentally different from the DFROUTER algorithm, as they require an additional *a-priori* OD matrix, are more complicated in calculation, need more time to compute and are mostly applied to small areas. Therefore they are not going to compare with DFROUTER in that they do not necessarily place restrictions on the network type. DFROUTER's approach of dividing incoming flow proportionally to off-ramp counts makes it simple and fast to calculate respective flows. Similar approaches also lead to similar results when examining a testing highway corridor.

The improved algorithm has been applied successfully to a large highway network and produced reliable results using recursion to guess missing data. Each edge after a junction will contain a certain traffic count and relative probability. The method of multiplying individual probabilities is left unchanged. The problem of missing detectors at destinations as well as the problem of rings that occur in highway networks is solved partly. More testing in practical highway scenarios should be performed to complete the improved algorithm.

The improved algorithm, however, is applied only to highway corridors (one way street). Future research is needed to extend the demand calculation for urban areas where route computation is complicated as each edge contains two ways, resulting in ambiguities while performing recursions over the network.

11.7 References

- [28] Bert, E., Dynamic urban origin-destination matrix estimation methodology. 2009, EPFL.
- [29] Van Zuylen, H.J. and L.G. Willumsen, The most likely trip matrix estimated from traffic counts. *Transportation Research Part B: Methodological*, 1980. 14(3): p. 281-293.
- [30] Cascetta, E., Estimation of trip matrices from traffic counts and survey data: A generalized least squares estimator. *Transportation Research Part B: Methodological*, 1984. 18(4-5): p. 289-299.
- [31] Kattan, L. and B. Abdulhai, Traffic Origin-Destination Estimation in *Handbook of Transportation Engineering, Volume II: Applications and Technologies, Second Edition*. 2011, McGraw Hill Professional, Access Engineering.

- [32] Krajzewicz, D., et al., Recent Development and Applications of SUMO – Simulation of Urban MObility. International Journal on Advances in Systems and Measurements, 2012. vol 5, no 3 & 4.
- [33] Cascetta, E. and S. Nguyen, A unified framework for estimating or updating origin/destination matrices from traffic counts. Transportation Research Part B: Methodological, 1988. 22(6): p. 437-455.
- [34] Maher, M.J., Inferences on trip matrices from observations on link volumes: A Bayesian statistical approach. Transportation Research Part B: Methodological, 1983. 17(6): p. 435-447.
- [35] Cremer, M. and H. Keller, A new class of dynamic methods for the identification of origin-destination flows. Transportation Research Part B: Methodological, 1987. 21(2): p. 117-132.
- [36] Nihan, N.L. and G.A. Davis, Recursive estimation of origin-destination matrices from input/output counts. Transportation Research Part B: Methodological, 1987. 21(2): p. 149-163.
- [37] Yang, H., et al., Estimation of origin-destination matrices from link traffic counts on congested networks. Transportation Research Part B: Methodological, 1992. 26(6): p. 417-434.
- [38] Bell, M.G.H., The real time estimation of origin-destination flows in the presence of platoon dispersion. Transportation Research Part B: Methodological, 1991. 25(2–3): p. 115-125.
- [39] Krajzewicz, D. and M. Behrisch, SUMO - Simulation of Urban Mobility - User documentation.
- [40] Behrisch, M., et al., SUMO - Simulation of Urban MObility - an Overview. IMUL 2011, The Third International Conference on Advances in System Simulation, 2011, 2011: p. 63-68.
- [41] Muthuswamy, S., et al., Improving the Estimation of Travel Demand for Traffic Simulation: Part I, in Final Report 2005, Department of Civil Engineering, University of Minnesota
- [42] L.Nihan, N., Use of volume data to reproduce ramp-to-ramp freeway trip patterns - A pilot study. Technical report standard, Washington State Department of Transportation, 1979. WSDOT - 35.1.

12 TraCI4Matlab: Re-engineering the Python implementation of the TraCI interface

Andrés F. Acosta Gil¹, Jairo Espinosa¹, Jorge E. Espinosa²

*¹Facultad de Minas, Universidad Nacional de Colombia, Cra 80No 66-223, Medellín, Colombia
{afacostag, jespinov}@unal.edu.co,*

*²Politécnico Colombiano Jaime Isaza Cadavid, Cra 48 No 7-151, Medellín, Colombia
jeespinosa@elpoli.edu.co*

12.1 Abstract

SUMO (Simulation of Urban Mobility) has become one of the preferred open-source platforms for researchers to perform microscopic road traffic simulation. Thanks to the TraCI API (Traffic Control Interface), SUMO offers a high level of flexibility, allowing controlling the objects of the SUMO simulation through a TCP-IP client that can be developed in any programming language, or even offered as a service. Two important implementations of the TraCI interface have been released till now, namely TraCI-Python and TraCI4J (TraCI for Java). On the other hand Matlab is a software tool with a programming language with a broad user's community of researchers. Matlab is used in many tasks on simulation, control, optimization and it is a preferred tool for rapid prototyping. Both tools share strengths that are desirable for the development of control strategies for traffic. The desired of combining both strengths motivated the interest to develop a TraCI implementation for Matlab. In this article, the re-engineering process of the TraCI-Python API used as a basis for the creation of TraCI4Matlab (TraCI for Matlab) is described.

Keywords: SUMO, TraCI, Matlab, Traffic Simulation, Reverse Engineering, Re-engineering.

12.2 Introduction

SUMO is a set of tools to create and execute microscopic road traffic simulation scenarios[1]. These tools are grouped in three categories:

- Mapping tools. For creating the "map" (network), where the simulation will be performed, comprised by intersections, streets, traffic light definitions, polygons that represent buildings and other structures, and a variety of sensors for output delivery. The network can be created from scratch or imported from a wide range of sources;
- Demand modeling tools. For creating vehicle demands from several sources or even randomly, allowing to define vehicle types according to their physical characteristics and specify entry times, origins and destinations;
- Simulation tools. The sumo application itself that receives the network, the demand and some optional information as inputs to execute the simulation and output results

in XML format, a feature that demonstrates the high integration capacity of the simulator.

SUMO includes an external interface: TraCI (Traffic Control Interface), that simplifies the retrieval and modification of the SUMO objects, through an application protocol built on top of the TCP-IP stack, allowing applications like vehicular communications, dynamic routing and traffic light control algorithms[2].

The SUMO community has developed two remarkable TraCI clients: one made in Python by the SUMO developers, which we will call TraCI-Python; and TraCI4J, made in Java by researchers from Politecnico di Torino (Italy)[3].

Very often MATLAB has been used as a control and simulation platform by the control community. Since our research is oriented to dynamic traffic control, the need for a simulation environment across the two platforms came up, in order to take advantage of the control and optimization tools of MATLAB and the simulation capabilities of SUMO.

The open-source nature of the SUMO suite brings many advantages, including: (i) a high degree of flexibility, allowing to tailor the software to the specific context in which it will be used;(ii) no need to pay for the software; and (iii) educational advantages e.g.learning about algorithms and software engineering. These advantages make the open-source tools ideal for researchers and academics. However, open-source software also brings some disadvantages, perhaps the most known are the slow learning curve and, sometimes, lack of documentation or support [4]. Although SUMO has a complete reference for users and developers, documentation related to high level software architecture and design artifacts, and the dynamic behavior of the system, in terms of UML diagrams, is not available for the community. Here, reverse engineering plays an important role by allowing to identify the software components and understand how they collaborate to fulfill the required functionality. Reverse engineering is also an important step for re-engineering, which is "*the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form*" [5].Note that this process results to save costs by taking advantage of an available open-source implementation, compared to the case in which requirements are accomplished through the whole software engineering cycle. Furthermore, as stated by Ewer et. al., re-engineering has many advantages over a direct code translation either by hand or using semi-automated tools [6].

In this article, we describe the re-engineering process applied to the TraCI-Python API used to re-implement TraCI for Matlab (TraCI4Matlab).

This article is organized as follows: Section 12.3 makes a general explanation of the re-engineering process. Section 12.4 describes the reverse engineering sub-process of the TraCI-Python implementation and shows the extracted architectural and component models and descriptions. Section 12.5 describes the forward engineering sub-process resulting from the adaptation of the models obtained in the previous section to the constraints imposed by the Matlab language. Section 12.6 shows the results and discussion. Finally, section 12.7 shows the conclusions.

12.3 The re-engineering process

Unlike the traditional forward engineering process, in which the source code software project is developed starting from abstract representations based on the requirements of the stakeholders, the re-engineering process starts from the source code of an existing software to analyze it and extract representations in higher levels of abstraction (reverse-engineering), which can be modified (re-structuring) to make a new implementation that satisfies a new set of requirements (forward engineering). The re-engineering process is illustrated in figure 12-1.

Re-engineering is part of the software's lifecycle maintenance phase. Re-engineering improves the software to adapt it to the increasingly changing environment and to satisfy new requirements. Thus, software maintenance is one of the most important phases of the software lifecycle. It has been estimated that represents around 70% of the total software lifecycle cost[7]. In the case of the TraCI-Python API, the new requirement consists on adapting it to Matlab while trying to maintain the same syntax.

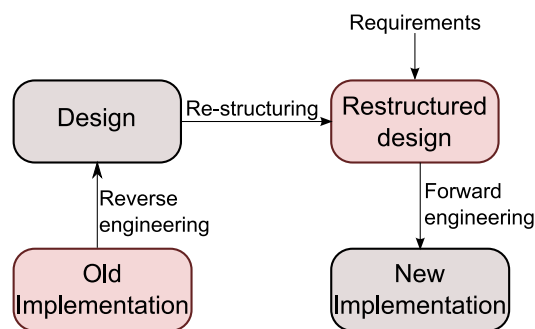


Figure 12-1: The re-engineering process

As stated by Chikofsky and Cross[5], frequently, software maintainers are not its developers. Therefore, the subject software should be initially understood through reverse engineering process to modify it properly. Furthermore, Demeyer et. al. [8] argued that a specific reason to use re-engineering was to "*extract the design as a first step to a new implementation*". They also defined a catalog of re-engineering patterns, grouped in clusters according to the re-engineering cycle. These design patterns are understood as best practices to address common problems found in a re-engineering process. Some of these patterns will be related to the activities carried out in the re-engineering process of TraCI-Python.

In the following section, the reverse-engineering sub-process of the TraCI-Python API is described.

12.4 Reverse-engineering the TraCI-Python implementation

The reengineering process in this project resorted to using some patterns described by Demeyer et. al. in [8], particularly there were used:

- Chat with the maintainers. According to Demeyer et. al. the intent of this pattern is to "*Learn about the historical and political context of your project through discussions with the people maintaining the system*". Since the interaction with the SUMO team through the mailing list system was important to install and use the simulator, the Chat with the maintainers intent was redefined more generally to "*Learn about the context and the technical aspects of your project through discussions with the people maintaining the system.*"

- Read all the code, whose intent is to "Assess the state of a software system by means of a brief, but intensive code review". It's important to note that, in this case, the focus was on understanding the structure of the TraCI-Python API, not on its quality.

Step through the execution, whose intent is to "understand how objects in the system collaborate by stepping through examples in a debugger". Two open-source tools were used to apply this pattern: Winpdb [9], which is a Python debugger, and StarUML [10], which is a program to draw UML diagrams. The first step to understand the TraCI-Python implementation was the debugging of the TraCI4Traffic Lights tutorial, provided with the SUMO installation, to find that TraCI-Python comprises three main components: The TraCI package, the modules representing the SUMO objects (edge, junction, lane and so on) and the TraCI constants definition. At this point, it's important to note that Python modules create namespaces, which can be represented, in terms of UML, as packages. Furthermore, as stated in [11]: "The import statement creates a new namespace and executes all the statements in the associated .py file within that namespace". In the case of packages, the `__init__.py` module is executed. Therefore, one can access the variables, classes and methods defined in Python modules through the dot operator once they're imported. Figure 12-2 shows two UML package diagrams, in which the namespaces created by the TraCI-Python modules are represented as packages. It also illustrates how are they deployed and the dependence relationships among them. Note, that the abstract package `sumo_object` was defined to generalize the modules representing the SUMO objects. Moreover, it was found that the sumo objects share some methods and attributes in common.

In the following subsections, the components of the TraCI-Python API are described.

12.4.1 The TraCI package

This is the top-level package. It contains the modules corresponding to the SUMO objects plus five public functions and others with, at most, package visibility. Through these functions, the functionalities of the TraCI package could be extracted, being: Initialize and close the connection to the SUMO server as well as switching among connections, allowing several SUMO instances to be controlled by the same client; perform a simulation step; populate the subscription results of each module; construct and send the outgoing messages according to the TraCI protocol; and read the responses from the SUMO server and check them for errors throwing the corresponding exceptions. In terms of UML, this package can be diagrammed using the utility stereotype class, as showed in figure 12-3.



Figure 12-2: The TraCI-Python API components: (a) deployment, (b) Dependency relationships.

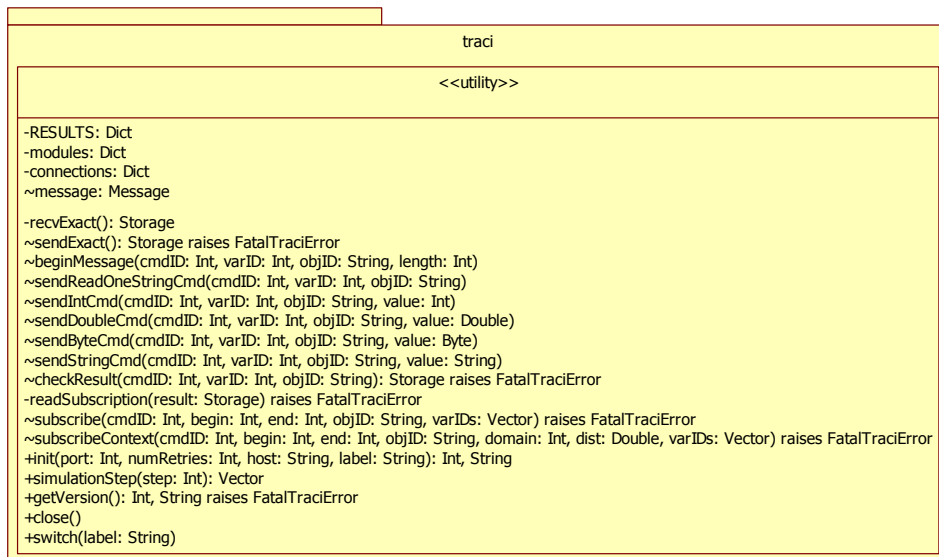


Figure 12-3: The TraCI package.

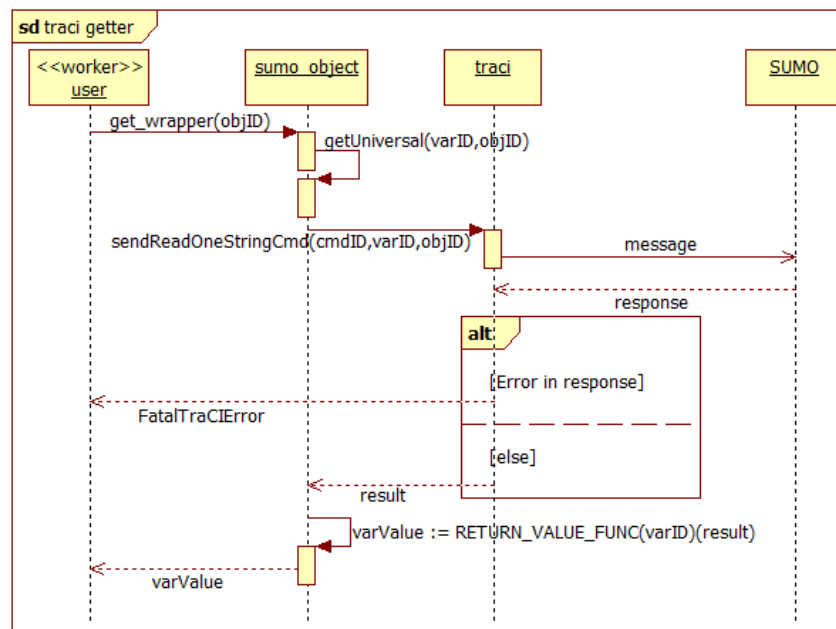


Figure 12-4: UML sequence diagram for the get process in the TraCI-Python API.

12.4.2 Packages corresponding to the SUMO objects

These modules can be briefly summarized through the so called getters and setters. The get and set processes follow a sequence of functions in the TraCI components that collaborate by appending the proper command, requested attribute, and desired value (in the set case) from the TraCI constants to build the outgoing get/set message according to the TraCI protocol. Here, the *get_wrapper* and *set_wrapper* abstract methods are defined to represent the set of public methods designed for the end user in such a way that he/she only needs to provide the ID of the SUMO object of interest and the desired attribute value (in the set case). Finally, the *sumo_object* packages include another four wrapper functions related to the TraCI subscriptions: two for subscribing to the desired object and variable, and other two for retrieving the subscription results. Figure 12-4 shows an UML sequence diagram, which is an example of the above process, in this case, the get process. Note how the different components collaborate: the end user calls the get wrapper which calls the universal getter of

the *sumo_object* module, which in turn calls the proper TraCI function to build the outgoing message, read the response from the SUMO server and check it for errors. Figure 12-5 shows the class diagram corresponding to the abstract class *sumo_object*. It is worth to notice, that this abstract class was not physically implemented, but serves as a way to explain the packages corresponding to the SUMO objects and their attributes and methods in common. Additionally, the simulation module is the only one that does not contain the `getIDList` method.

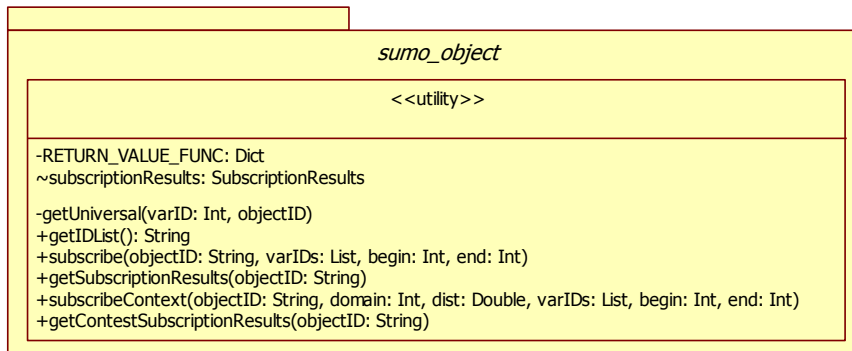


Figure 12-5: Abstract package *sumo_object* which represents the packages corresponding to the SUMO objects.

12.4.3 TraCI constants

This is a module containing the command, variable-type and data-type codes as constant attributes from the TraCI protocol specification. The other TraCI-Python modules use these constants as parameters for their functions. For example, referring to figure 12-4, the parameters `varID` and `cmdID` are taken from the TraCI constants module.

12.5 The forward engineering subprocess

The Matlab language specification has some limitations that makes the reverse-engineered design of TraCI-Python to be re-factored. The most important, is that Matlab imposes only one function definition per m-file, at most, including nested functions; the same holds for class definitions. Moreover, the Matlab's import statement allows adding only package-based functions and classes to the current import list.

From figure 12-3, it can be seen that TraCI-Python's namespaces have variables with the following properties:

1. They are not associated with a specific object instance
2. They can be imported
3. Their values can be changed by methods in other namespaces.

Hence, to achieve the same behavior in Matlab, three options were considered:

- Implement TraCI-Python's namespaces as classes with static members: This solution was discarded because, although Matlab allows to define constant attributes in a class, the same cannot be done for static ones, i.e. those that do not need the class to be instantiated and whose values can be changed[12]. Note that this option conflicts with properties 1 and 3.
- Execute m-files that load the required variables into the workspace: This solution would require the Matlab's package functions to access those variables. In the Matlab documentation, it has been stated that the best practice is to pass the variables as

arguments[13]. In this way, not only the workspace would be filled with variables that should be transparent to the user, but he/she would need to pass those variables as arguments, which results impractical. Another strategy listed in the Matlab documentation, is the use of persistent variables in a function. However, persistent variables can be changed only by the function that defined them, which conflicts with property 3. Finally, one could evaluate a given expression in another workspace, but it has limited flexibility in the sense that it doesn't allow the variable to contain indexes. For these reasons, this solution was discarded.

- Finally, the use of global variables was chosen because it can deal with properties 1 and 3. Global variables are defined in the methods that require them and can be accessed by any other method.

There were some special cases in which there was no need to use global variables. For example, It was found that some attributes were used only by one function. Therefore, those attributes were defined inside the functions that use them. Another case is related to the RETURN_VALUE_FUNC attribute of the *sumo_object* packages, which is a constant. Then, a corresponding new class with only constant attributes was defined. Finally, it was found that the *modules* attribute of the TraCI package only was used in two methods of the same package: *readSubscription* and *simulationStep*. The *modules* attribute is a dictionary that associates responses from the SUMO server to the corresponding *sumo_object* module, allowing to detect errors and populate the TraCI subscription results. In the *readSubscription* method, the *modules* attribute is used to populate the TraCI subscription results based on the response of the SUMO server. For this reason, a new dictionary called *subscriptionResults* was defined inside the *readSubscription* method. On the other hand, the *modules* attribute is used in the *simulationStep* module only to reset its values, i.e. the subscription results of each *sumo_object* module. Note that, in this case, it is not necessary to define a map. Therefore, a new array called *modules* was defined in the *readSubscription* method.

Figure 12-6 shows the re-structured architecture for the implementation of TraCI4Matlab, including the the addition of the new package of constants RETURN_VALUE_FUNC.

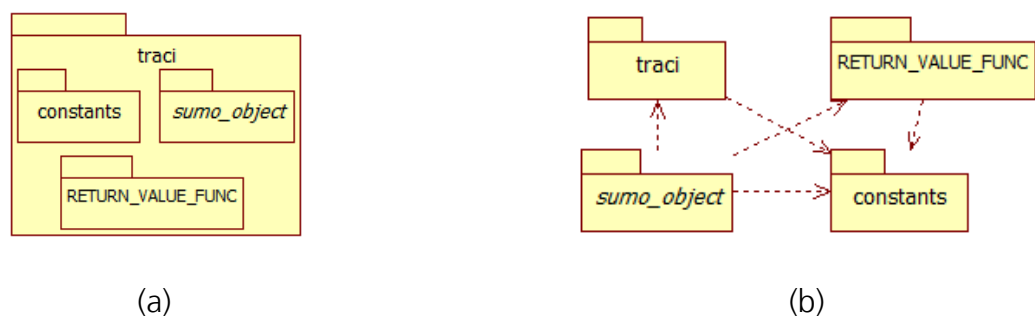


Figure 12-6: The TraCI4Matlab API components: (a) Deployment, (b) Dependency relationships.

Figure 12-7 shows the global variables used in the TraCI4Matlab implementation. Note that there are 14 global instances of the class *SubscriptionResults*, namely *edgeSubscriptionResults*, *guiSubscriptionResults* and so on (including the areal detector introduced in the version 7 of TraCI). If no subscription was made to a particular sumo object, Matlab sets the corresponding global variable to a null object by default. Recall that the rest of the attributes associated to namespaces are defined in the methods that use them, e.g. the RESULTS and *modules* attributes of the TraCI package.

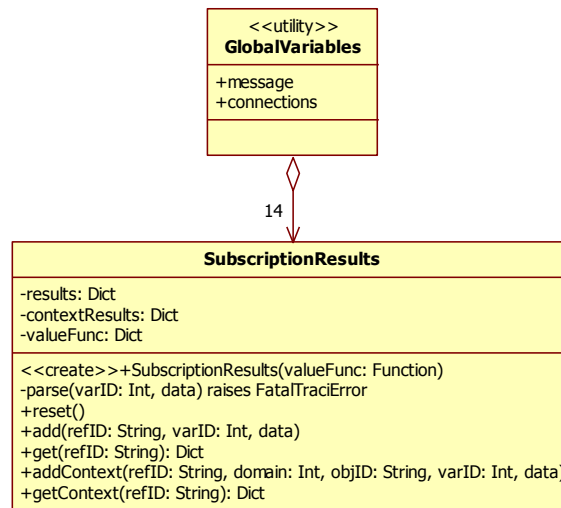


Figure 12-7: The global variables used in TraCI4Matlab.

The re-engineering patterns associated with the forward engineering process are explained, as follows:

- Testing patterns. The testing cluster materialized through the application of the "Grow your tests base incrementally" pattern. Once some components of TraCI4Matlab were developed, the corresponding test were performed. The implementation of TraCI4Matlab started with the *getVersion* and the *init* methods, which were the first tested methods.
- Involve the users. Regular meetings were carried out with the stakeholders to check that the satisfactory completion of the requirements.
- Migrate systems incrementally. Some components and functionalities were initially implemented and tested to perform demonstrations in the meetings. Then, according to the conclusions of those meetings, further components and functionalities were tested and implemented until the final product was finished.

12.6 Results and discussion

TraCI4Matlab was released on 24th December of 2013 under the BSD license. It is free software and is available for the community at Matlab Central[14], or as part of the SUMO contributed tools since SUMO 0.20.0.

Currently, TraCI4Matlab is being used in the project "Modelamiento y Control de tráfico urbano en la ciudad de Medellín: MOYCOT" (Modelling and Control of Urban Traffic in the City of Medellín-Colombia). One of the objectives of the MOYCOT project is to design a MPC (Model Predictive Control) traffic lights control system for the urban traffic network of the city of Medellín. Some parameters needed by this system include the length of the queues in vehicles on each signalized lane and the traffic flow in the edge. Thanks to TraCI4Matlab, preliminary results were obtained in a scenario consisting of a single intersection, showed in figure 12-8.

Using induction loops and lane area detectors, the number of vehicles entering the north-south as well as the length of the queues (jam length in TraCI) on each lane in vehicles were obtained, as shown in figure 12-9.

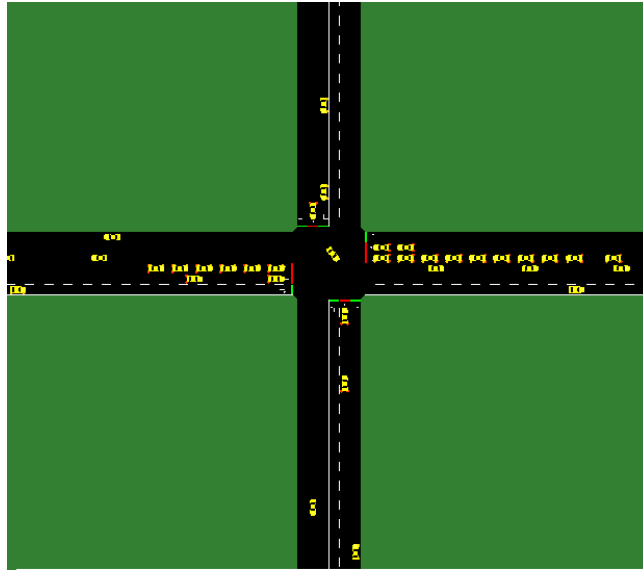
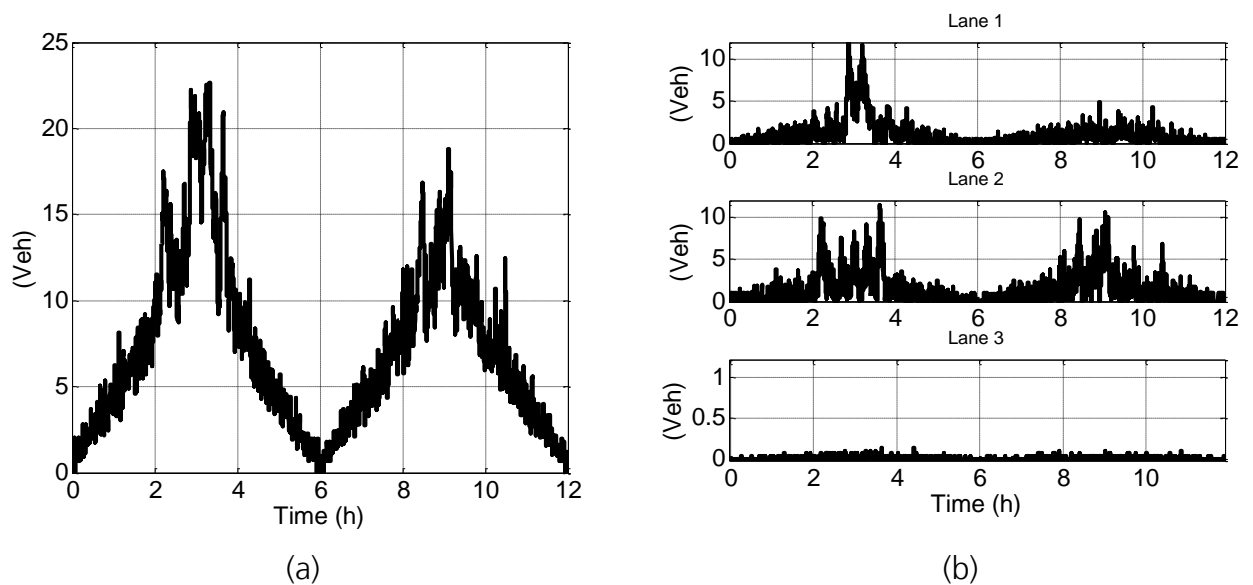


Figure 12-8: The single intersection scenario used in the MOYCOT project to obtain parameters needed for MPC traffic lights controller.



12.7 Conclusions

In this paper, the re-engineering process of the TraCI-Python API used to develop a Matlab implementation was presented. Static and dynamic models related to the architectural and component design were obtained. The authors consider that those models can be used to implement TraCI in any object-oriented programming language.

One of the requirements formulated for TraCI4Matlab was to preserve the same syntax of TraCI-Python. Although it could be accomplished through the approach described in the forward engineering process, performance implications were not considered, in fact, there were no requirements related to performance. As a result, it was found that the TraCI4Matlab performance was much lower than the TraCI-Python's. Following the do, do better, and do fast software re-engineering sequence[8], the authors propose to include as requirements for

future versions of TraCI4Matlab, improvements in code reuse and to detect the components that cause the low performance, exploring different solutions to optimize it.

Another issue that has to be addressed is related to the case in which the user launches the sumo server in GUI mode. In TraCI-Python, when the connection is established and the user closes the SUMO GUI, without closing the connection from the client, the client detects that the connection was closed from the server and an exception is thrown. In the case of TraCI4Matlab, Matlab cannot differentiate if the connection was closed or the user paused the simulation from the SUMO GUI, which can cause two undesired behaviors:

- If the socket was created with a fixed timeout, then when the user pauses the simulation from the GUI for a time longer than the timeout, TraCI4Matlab will throw an erroneous "Connection closed by SUMO" exception.
- If the socket was created with a timeout of infinity, then when the user closes the simulation from the GUI, TraCI4Matlab will enter in an infinite listening state.

Finally, the design obtained through reverse engineering suggests some private methods and some others with package visibility. For private methods, Matlab has developed the concept of "private functions". Unfortunately, Matlab has not defined, to date, a similar approach for the case of methods with package visibility.

12.8 Acknowledgments

This work was supported by Proyecto Colciencias 111856934640 contrato 941-2012: Modelamiento y Control de tráfico urbano en la ciudad de Medellín. Convocatoria 569.

12.9 References

- [1] "SUMO_User_Documentation - SUMO - Simulation of Urban Mobility." [Online]. Available: <http://sumo-sim.org/userdoc/>. [Accessed: 30-Jan-2014].
- [2] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent Development and Applications of SUMO - Simulation of Urban MObility," *Int. J. Adv. Syst. Meas.*, vol. 5, no. 3&4, pp. 128–138, Dec. 2012.
- [3] "egueli/TraCI4J · GitHub." [Online]. Available: <https://github.com/egueli/TraCI4J>. [Accessed: 01-Apr-2014].
- [4] K. Ven, J. Verelst, and H. Mannaert, "Should You Adopt Open Source Software?," *IEEE Softw.*, vol. 25, no. 3, pp. 54–59, May 2008.
- [5] E. J. Chikofsky and I. Cross, J.H., "Reverse engineering and design recovery: a taxonomy," *Softw. IEEE*, vol. 7, no. 1, pp. 13–17, 1990.
- [6] J. Ewer, B. Knight, and D. Cowell, "Case study: an incremental approach to re-engineering a legacy {FORTRAN} Computational Fluid Dynamics code in C++," *Adv. Eng. Softw.*, vol. 22, no. 3, pp. 153 – 168, 1995.
- [7] D. C. Tucker and D. M. Simmonds, "A Case Study in Software Reengineering," in *2010 Seventh International Conference on Information Technology: New Generations (ITNG)*, 2010, pp. 1107–1112.
- [8] S. Demeyer, S. Ducasse, and O. Nierstrasz, *Object-Oriented Reengineering Patterns*. San Francisco, CA, USA: Morgan Kaufmann, 2002.

- [9] "Winpdb - A Platform Independent Python Debugger." [Online]. Available: <http://winpdb.org/>. [Accessed: 30-Jan-2014].
- [10] "StarUML - The Open Source UML/MDA Platform." [Online]. Available: <http://staruml.sourceforge.net/en/>. [Accessed: 30-Jan-2014].
- [11] D. M. Beazley, Python Essential Reference, 4th ed. Addison-Wesley Professional, 2009.
- [12] "Comparing MATLAB with Other OO Languages - MATLAB & Simulink." [Online]. Available: http://www.mathworks.com/help/matlab/matlab_oop/matlab-vs-other-oo-languages.html. [Accessed: 02-Apr-2014].
- [13] "Share Data Between Workspaces - MATLAB & Simulink." [Online]. Available: http://www.mathworks.com/help/matlab/matlab_prog/share-data-between-workspaces.html. [Accessed: 02-Apr-2014].
- [14] "TraCI4Matlab - File Exchange - MATLAB Central." [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/file_infos/44805-traci4matlab. [Accessed: 01-Apr-2014].

13 TOMS – Traffic Online Monitoring System for ITS Austria Wests

Karl-Heinz Kastner and Petru Pau;

*RISC Software GmbH, Softwarepark 35, 4232 Hagenberg, Austria
{karl-heinz.kastner, petru.pau}@risc-software.at*

13.1 Abstract

ITS Austria West is a long-term Austrian project in which, among other tasks, real-time traffic data is continuously generated and made public. The traffic monitoring application integrates sensor data coming in real time from various data sources. Our system relies heavily on the open source package SUMO to generate and maintain the road network and to simulate the traffic in order to obtain estimates for traffic values on roads that are not covered by sensors. Due to inaccuracies in the demand model, a series of calibration steps are executed. The resulting demand model achieves an acceptable level of stability and conformity with the reality. A traffic simulation runs with this demand model in parallel with the traffic monitoring software and is continuously adjusted in order to comply with the current traffic situation, as reported by sensors.

Keywords: traffic monitoring, traffic simulation.

13.2 Introduction

In the frame of the project “ITS Austria West” we developed a system that monitors the traffic on Upper Austrian roads. The system integrates *real-time sensor data* with *traffic simulation results* in order to generate snapshots of the traffic situation. The road infrastructure of Upper Austria is modelled by a trimmed road network; this is used by the traffic simulation as well, together with a calibrated demand model for an average working day.

There are two categories of real-time data: floating car data (FCD) from sensors installed in roaming cars, and vehicle detection loops (VDL) installed at static position on a fixed number of roads. FCD are used to estimate the current average velocities on corresponding roads; VDL are basically vehicle counters that also provide velocity information.

Since real-time data do not cover all roads, a traffic simulation can be used to fill the gaps. Every few minutes, the results of the simulation are compared with the real-time data. Whenever flagrant discrepancies are observed between the simulated traffic and the real-time data, adjustments are computed and injected back into the simulation. Simulation results and real-time data are eventually aggregated into a snapshot of the traffic situation, which is subsequently published.

The simulation needs an accurate demand model for providing results that closely resemble the real-time development of traffic conditions. Since the original demand model, provided by the Upper Austrian authorities, does not correspond anymore to the reality, it needs to be improved. We perform a series of calibration steps so that, as much as possible, dramatic discrepancies between the reality and the simulation are removed.

In this paper, we give an overview of our system, with emphasis on aspects related to the use of SUMO concepts and components. In the section 13.3 we describe the system architecture, at a high level of abstraction. In section 13.4 we show how the static data, fundamental to all sub-systems, is generated. Section 13.5 contains the calibration process, executed as a preliminary step. In section 13.6 we explain how the static and real-time sensor data are processed and integrated. Section 0 describes TOMS, the main component of our system. Finally, our last section contains some conclusions and a few words regarding future work.

13.3 System Overview

Figure 13-1 contains the system architecture of ITS Austria West.

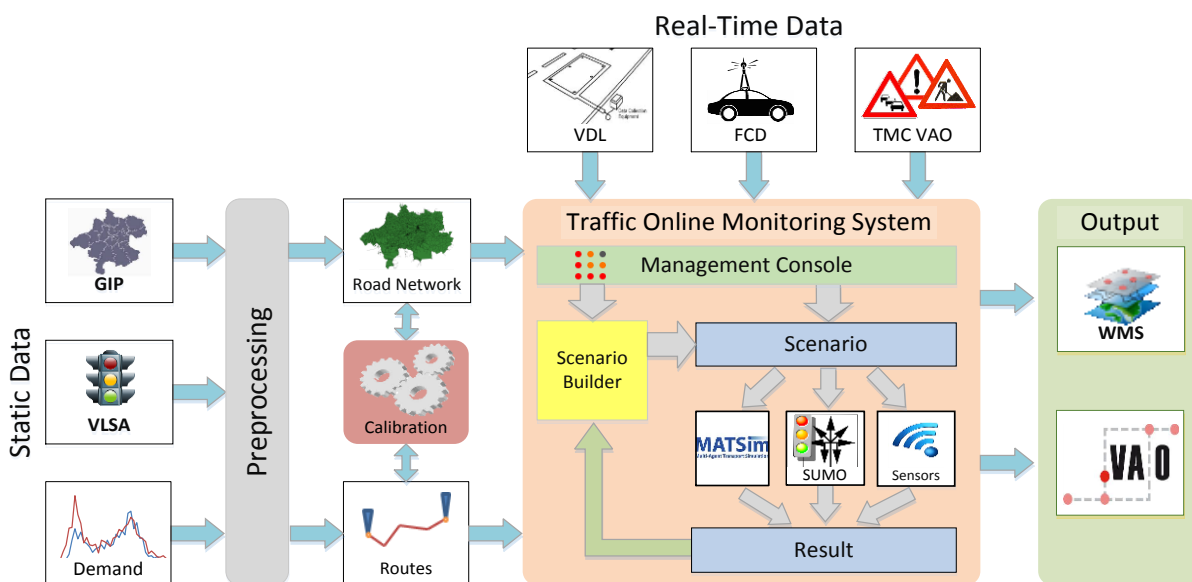


Figure 13-1: Architecture of ITS Austria West

- In the *preprocessing* phase, a road network file, as well as a routes file, are generated from the static incoming data (GIP – Graph Integration Platform, the main, most comprehensive database of Austrian roads; VLSA – traffic lights description file; the demand model – a file with all origin-destination relations). The generated files follow the SUMO formats.
- The *calibration* step is meant to balance the trips, so that eventually a routes file is computed, with which a traffic simulator (e.g. SUMO) produces a good approximation of the traffic situation. With the initial, unprocessed demand model, SUMO needs more than five days to simulate all trips – which in fact cover no more than one working day.

- The *Traffic Online Monitoring System* (TOMS) periodically collects real-time data (FCD and VDL) and aggregates them with simulation (SUMO, MATSim) results, in order to generate snapshots of the traffic situation.
- Currently, TOMS sends its *output* to VAO (“Verkehrsauskunft Österreich”, the Austrian traffic information system) and to a WMS (Web Mapping Service) layer that can be accessed by various applications. A nice, user-friendly, HTML5-based web page, as well as an app for Android and iPad integrate these WMS layer.

13.4 Preprocessing Static Data

In this phase, the internal data collections used by all components of ITS Austria West are generated from external data.

13.4.1 The Road Network

From the GIP database, information concerning relevant road segments is extracted and filtered, based on specific criteria:

- Geographically: The system monitors only roads from a rectangular area which encloses Upper Austria (Fig. 13-2).
- Functionally: Only roads with a certain level of significance are taken. The relevant significance levels vary according to road position (e.g., in urban or rural areas). Fig. 13-3 contains the roads in Linz city centre.

The extracted road segments are used to generate two files, containing “nodes” and “edges”, in SUMO-specific XML format. These two files are given as input to the program NETCONVERT, the component of SUMO which generates a road network file. The network can be generated in either SUMO or MATSim format.

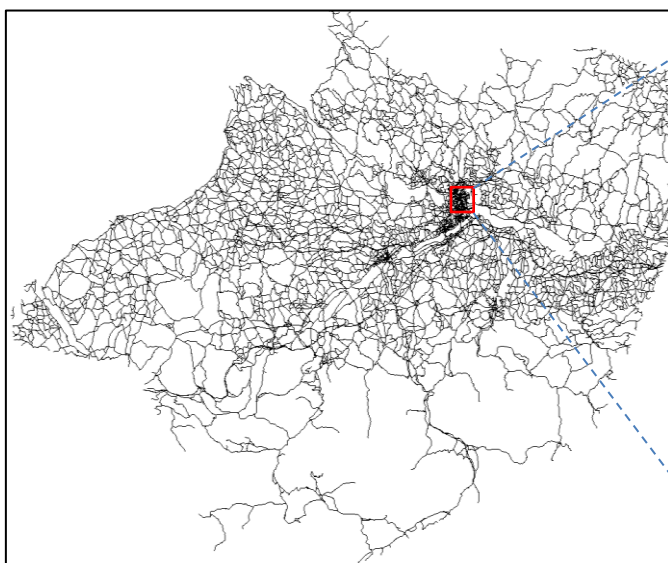


Figure 13-2: The whole road network



Figure 13-3: Part of an urban area (Linz).

Additional lists are used to refine and improve the road network, with traffic light signal systems (VLSA) and manual changes or corrections.

13.4.2 The Routes File

The currently available demand model was generated in 2008 by the department „Gesamtverkehrsplanung und öffentlicher Verkehr“ of the Upper Austrian government. It reflects all traffic in Upper Austria for a working day and is given as a source-destination matrix with the summary of trips, for which the source and destinations are districts. We generated a set of routes, distributed over a day, in a three-steps process:

1. We produce trips with clear origin-destination streets and start times.
2. For each origin-destination pair, we compute the best route. Routing algorithms (e.g. Dijkstra) can be employed, or tools already provided by traffic simulation software (SUMO makes a routing program available: DUAROUTER).
3. We try to balance the routes (see the next section for a detailed description), by
 - sending vehicles on alternative routes, to avoid congestions on critical road segments;
 - shifting vehicles temporally,

in order to reach a demand model with which the simulated traffic does not differ too much from the reality.

For the first step, time-variation curves are computed from historical sensor data, to be used as approximation basis for *distributing the trips during a day*. We differentiate the origin-destination relations, and thus the time-variation curves, by the locations of sensors. There are essentially three different relations: urban-to-urban, rural-to-rural, and urban-to-rural; their time-variation curves are quite dissimilar.

In the near future, a more precise trips file will be made available by the Upper Austrian government, in which the trips will already be given with their start time and origin-destination streets.

The routes are calculated in step 2 with a fast Dijkstra algorithm, parallelized so that, for each source, the shortest paths to all destinations are computed in a separate thread. The computation of all routes (around a million) on a 3.7 GHz computer with four cores with hyperthreading takes under two minutes.

13.5 Calibrating the Demand Model

The calibration process takes a road network and a routes file as input, both in SUMO format, and runs SUMO repeatedly with increasing end times, starting at 6:00AM and ending at 8:00PM. For each end time, a limited number of SUMO runs are allowed. If the number of vehicles arriving too late is relatively small (e.g. below 1000), the process starts working with the next end time (the end times advance is one hour).

In order to assess whether a vehicle arrives too late (or too early), we need estimated times of arrival (ETA). Currently we compute these times, for each vehicle, considering the average speed per edge and an estimated time for traversing crossroads. The alternative is to let SUMO compute these times for us: It should run with a simplified demand model, with exactly one trip for each individual route. These unique trips should be distributed temporally so that traffic jams or delays are improbable.

During the calibration process, the simulation output is obtained via the dumping mechanism implemented in SUMO. Since SUMO normally produces such a dump every second, we had to modify it so that the periodicity can be given as a parameter in the SUMO configuration file.

Every two minutes a SUMO dump is analyzed, and all vehicles currently running are checked against their expected arrival times.

- If a vehicle is found on the road after its ETA, it will be considered as delayed. If the delay is considerable (the minimum acceptable delay is parameterizable; currently we work with 5 minutes), the vehicle will need to be shifted (it will depart earlier) or sent on an alternative route. The decision is taken at random, with a ratio shift/reroute given as a parameter to the calibration process.
- A vehicle that has arrived too early will be shifted forward, so that in the calibrated routes file it will depart later.

The alternative routes are generated with a modified Dijkstra algorithm that tries to generate a new route between any give source and destination. The new route must be significantly different (for example, 50% of its length or 10 minutes drive time should differ from each of those already computed) and not too long (e.g., maximal 10 minutes longer than the optimal route). Also, the difference between the new route and any of the old routes must be a contiguous set of road segments.

After SUMO ends, a new routes file is generated, in which the vehicles that arrived too late or too early have either modified departure times or different alternative routes. This new routes file is given as input to SUMO at the next step.

The goal of the calibration process is to reach a stable offline traffic situation, where there are no catastrophic traffic jams and overly delayed vehicles. In our experience, SUMO runs with the initial routes file for more than five days (simulation time), in order to have all vehicles reach their destinations. Also, the rush hours are simulated by SUMO below real-time, and since a lot of road segments are filled with unmoving vehicles, huge queues of new vehicles wait to be inserted into the simulation.

The calibrated route files obtained so far produced very satisfactory results: The simulation was more fluid, and the traffic jams – inevitable during the rush hours – did not become persistent.

13.6 Integration of Real-Time Data

13.6.1 Vehicle detection loops

The *vehicle detection loops* (VDL) are automatically geocoded in the preprocessing step, so that their location (edge, position on edge, direction) is precisely identified in the network.

In the online system, the latest measurements from VDLs are aggregated and used to compute average traffic velocities on their corresponding road segments. These measurements also provide the number of passed vehicles, information which is used for the online calibration of the simulation: If the observed traffic is heavier (or lighter) than the simulated traffic, new vehicles are inserted into the simulation, with randomly chosen routes (or are removed from the simulation).

Figure 13-4 shows the positions of vehicle counters installed on Upper Austrian roads (white markers denote vehicle counters from which no data has been received in the last 15 minutes).

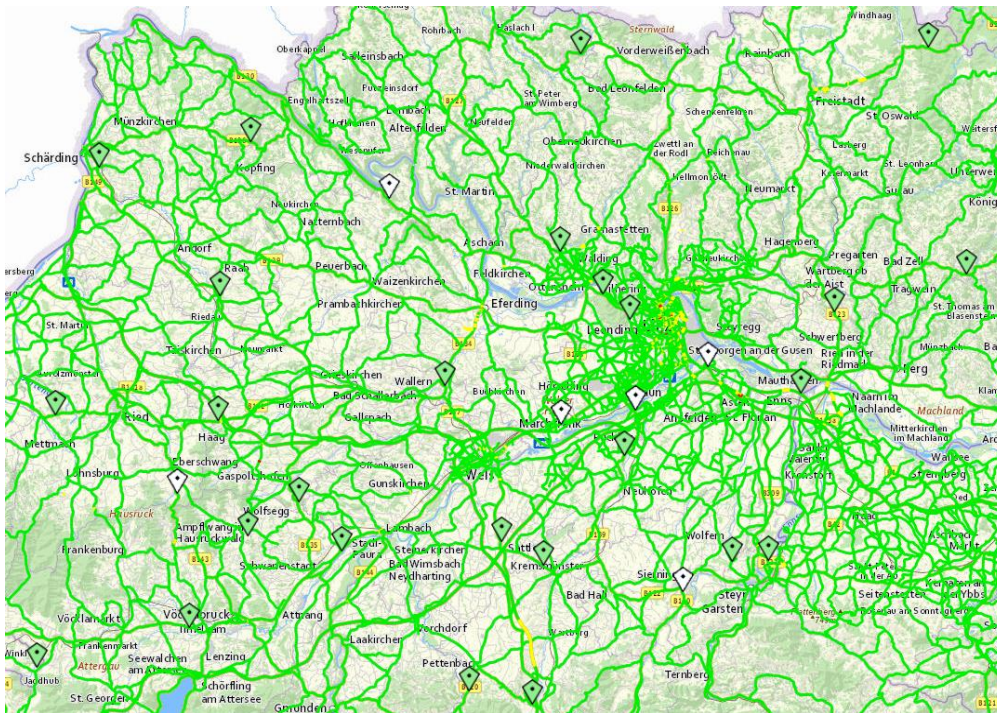


Figure 13-4: Locations of vehicle counters (Upper Austria)

13.6.2 Floating car data

Floating car data (FCD) come from various providers in intervals between 1 and 30 seconds. The measurements contain position information in geographic units, which is used to match the readings against the edges (road segments) of the road network.

In order to derive an accurate view of the traffic from FCD, it is not sufficient to determine the most probable road segments where readings were taken. Indeed, we received a lot of false positives, i.e., measurements with very slow speed values, which were not due to real problems in traffic but rather to waiting at a traffic light or slowing down and stopping for picking up a passenger (some of the sensors are installed on taxi cabs).

We designed a method for interpreting FCD where *vehicles' trajectories* are computed and analyzed with respect to time and velocity. This is not an easy task; we have to tackle the following problems:

- For a normally running car, measurements coming with a 30 seconds interval between them can be located on quite distant, non-adjacent road segments in the network. The missing road segments need to be guessed.
- The geographical coordinates can give a position that is far from any street (recall that the route network is an inherently incomplete sub-graph of the whole Upper Austrian road network). There may be more than one road segment equally distant from this point.

In order to validate the data of the last few minutes, a modified Dijkstra algorithm is employed. From a number of specific attributes (number of measurements, distance from the measurement to the road segment, direction, etc.) a value is computed, which is subtracted from the so-called *weight* of a road segment (usually its length, or drive time), a concept used by the Dijkstra algorithm: The shortest path is a sequence of road segments with minimal sum of weights. Consequently, the road segments containing (in the same direction, or closer to) measurements are preferentially chosen by the Dijkstra algorithm, so that the most probable route (i.e., trajectory) is eventually generated.

On this trajectory, a plausible matching of measurements to road segments can be performed. The travel time can be then used to update velocities on all road segments contained in the trajectory (including those not matched to any measurement).

13.6.3 Roadworks and roadblocks

The *roadworks information* is obtained via the Traffic Message Channel (TMC) published by VAO. This is a real-time data link, where different providers are bundled to a single connection.

13.7 TOMS

The main task of the Traffic Online Monitoring System TOMS is to generate periodically an online snapshot of the traffic situation of Upper Austria.

In order to achieve this, TOMS:

- Loads the static data and instantiates internal data structures which are fundamental for all processing steps;
- Starts a *simulation loop*, if a traffic simulation is running: TOMS collects its output and extracts traffic values for all roads with simulated vehicles; calibrates the simulation, inserting adjustments computed from the real-time data;
- Starts a *real-time data loop*: TOMS collects and processes real-time data; overrides the default or simulated velocities on monitored road segments; generates an LOS (Level of Service) output.

In Figure 13-5 we show a diagram of this process.

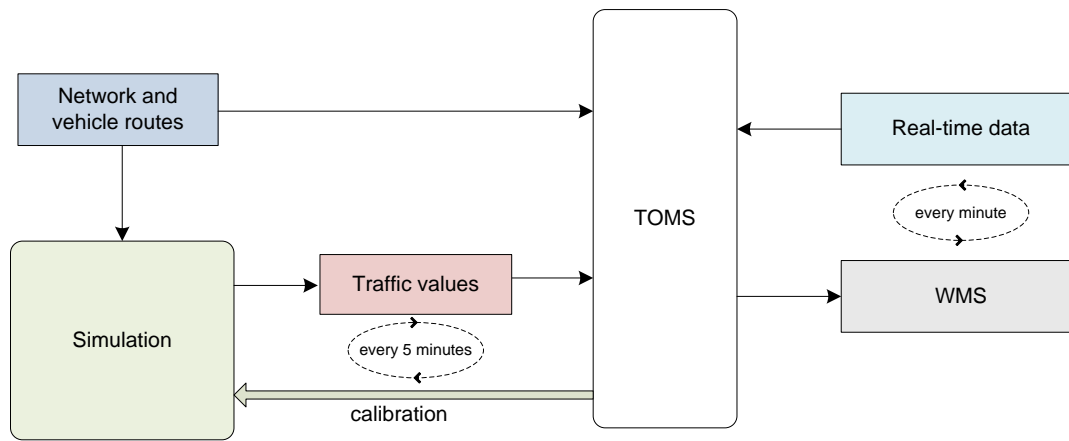


Figure 133-5 The two main loops in which TOMS collects traffic information.

The *scenario builder* is responsible for the dialog with the traffic simulation applications: It prepares scenarios – the input for simulations; it starts simulations, communicates with them and adjusts them as needed.

The main requirement that must be met by the simulation is that it runs faster than real-time. Our initial attempts at integrating SUMO showed that, with the uncalibrated demand model, the rush hours cannot be simulated faster than real-time. We described our efforts to parallelize SUMO in our paper [2].

Usually, a simulation runs in 5-minute steps. At the end of each step, the results are collected (e.g. as a SUMO dump) and sent to the scenario builder. These results, containing the status of every vehicle in the simulated network, are compared to the latest available real-time data, and adjustment decisions are taken (vehicles are added or removed, etc.). The adjustments are sent to the simulation, to be integrated for the next simulation step.

We use TraCI, the online control interface integrated in SUMO, to control the simulation and calibrate it according to the latest information received from sensors. The messages that TOMS currently sends to SUMO via TraCI are:

1. "Add New Route": A new route is defined, its ID and edges constitute the body of the message.
2. "Add Vehicle": A new vehicle is introduced into the simulation. The message contains the ID of the new vehicle and the ID of its route.
3. "Remove Vehicle": A vehicle is removed from the simulation. The vehicle ID is given in the message.
4. "Simulate To": SUMO is announced that it has to run until a given time (contained in the message) is reached.

A detailed description of our use of TraCI is contained in our paper [2].

We have to mention that it is possible to further calibrate the simulation with the average velocity on monitored road segments (and on roads with floating car data). For normal traffic situations, our experiments were unsatisfactory, leading to simulated traffic far slower than the real traffic. We reserve this calibration method for cases where the traffic is stopped due to events on the roads (e.g. accidents).

13.8 Conclusions and Future Work

In this paper, we gave an overview of the traffic online monitoring system developed in the project “ITS Austria West”. Based on soon-to-be public collections of road data, the system integrates real-time information received from various types of sensors to generate periodic snapshots of the traffic situation. Traffic simulation applications can be used to generate plausible traffic information on streets not covered by sensors. The output of our system is used by the Austrian traffic information system VAO, which provides services for multimodal routing.

The system is configured to monitor the traffic on Upper Austrian roads, but can be effortlessly set up for any other regional scenario.

The project is by no means finalized. Here are some directions in which we shall concentrate our efforts:

1. Currently only SUMO was used in our simulation scenarios. We intend to integrate MATSim as well, both in the offline calibration and in TOMS.
2. We are working on integrating the mesoscopic version of SUMO, provided by DLR for testing purposes. The calibration, which takes several weeks with the microscopic simulation, should be accomplished in reasonable time.
3. Our demand model is distributed along the day with historical data from vehicle detection loops. The time-variation curves obtained from these historical data do not accurately reflect the daily development of the traffic, and thus our demand model does not properly reflect the reality. We are looking forward to a new origin-destination matrix with hour-based distribution of trips, soon to be made available by the Upper Austrian government.
4. As a means to validating our traffic snapshots, streams from a set of video cameras are visually inspected and checked against the traffic situation exported by TOMS. However, most of the currently available video cameras are positioned in rather irrelevant locations, with little or no traffic, or where traffic jams are not probable.

We plan to contact other providers of visual traffic information, either with static or airborne cameras.
5. An important point on our agenda is replaying the traffic development in the last week, in addition to the online generation of traffic snapshots. It shall thus be possible to check our results against videos or images older than a day.

13.9 References

- [1] Behrisch, M. Bieker, L., Erdmann, J., Krajzewicz, D.: SUMO - Simulation of Urban MObility: An Overview, Barcelona, Spain, SIMUL 2011, 2011, 63-68
- [2] Kastner, K.-H., Keber, R., Pau, P., Samal, M. : Real-Time Traffic Conditions with SUMO for ITS Austria West, Proc. of the 1st SUMO Conference Berlin 2013, Springer Verlag, to appear.

14 A Railway Simulation Landscape Creation Tool Chain Considering OpenStreetMap Geo Data

Christian Rahmig and Andreas Richter;

*German Aerospace Center , Lilienthalplatz 7, 38108 Braunschweig, Germany
{Christian.Rahmig, Andreas.Richter}@DLR.de*

14.1 Abstract

This paper presents a tool chain with which various geo data from the free available geo data base OpenStreetMap (OSM) can be used as input for modelling a railway line to be used in a railway driver's cab simulation. Based on the Vires simulation software VirtualTestDrive and the so called SimWorld data base that has been developed by the Automotive department of the DLR Institute of Transportation Systems, this tool chain enables the fusion of heterogeneous data from different data sources. Further, it is analysed, which data are implicitly or explicitly described in the OSM data base. It is shown that elementary components of a railway simulation landscape including the railway track network as well as environment information can be extracted from these open data.

Keywords: OpenStreetMap, Railway, Simulation.

14.2 Motivation

The Railway Simulation Environment for Train Drivers and Operators (RailSET®) at the DLR Institute of Transportation Systems in Braunschweig is a laboratory for analysing the train driver's role in railway operation with a special focus on human factors (cf. [5]). Technically, the RailSET® is a realistic model of the train driver's cab, where the movement of the vehicle is simulated by the moving track and landscape in the front and side window view as can be seen in Figure 14-1.

This driver's cab simulation requires a realistic virtual 3-dimensional model of the railway line including the track alignment and the railway signalling components, e.g. signals, balises and magnets, which are relevant for operation on the track. Further, the surrounding landscape in terms of houses, trees, bridges, tunnels and stations plays an important role in making the driver's cab simulation as realistic as possible.

Currently, the simulation in the laboratory RailSET® is based on the simulation software tool Zusi (cp. [2]). Zusi contains integrated interfaces for modelling own tracks and scenarios. However, this process is very time-consuming, error-prone and it does not consider the combination with existing, real geo data, e.g. digital terrain models or other public available geo data bases and digital maps.



Figure 14-1: The train driver's cab simulation laboratory RailSET®.

The aim of the neo-geography project OpenStreetMaps (OSM) founded in 2004 is to create a free geo data base, from which digital maps for various topic-specific applications, e.g. also simulations, can be generated. Based on "volunteered geographic information" (cp. [3]) these digital maps may suffer from various drawbacks including an unknown position accuracy and data inconsistencies (cp. [4]). However, the OSM data base with currently more than 3.8 billion recorded positions stored in 2.3 billion nodes and 223.5 million ways also provides a big potential for creating new and enriching existing digital maps (cf. [12]).

This paper presents a tool chain with which these OSM geo data can be used as input for modelling a railway line to be used in the railway driver's cab simulation RailSET® to solve the problems of a manual content creation process. The following chapter describes the approach in detail: At first, the role of OSM geo data for generating digital railway maps is analysed. Secondly, the SimWorld tool chain for fusing heterogeneous spatial data from different data sources is presented. The adaption of the tool chain for the purpose described in this contribution closes the chapter. Section 14.4 gives an overview about the current state of implementation and the last section 14.5 provides a conclusion sketching future development.

14.3 Approach

14.3.1 Using OpenStreetMap geo data

In a previous paper we presented an approach how a railway network can be extracted from the OSM data by means of introduced topic-specific attributes for describing the railway infrastructure in terms of topology, geometry, topography and accuracy (cf. [13]). We show in [15] that by using simple geometry algorithms, the required railway track network topology and geometry information can be generated directly from the existing OSM data base.

The extracted railway track network description is then imported into the RailDrIVE® data base, which acts as central geo data storage for various railway applications, e.g. map-based positioning and track network visualisation. Additionally to the interface for importing from OSM, the data base contains also railway specific geo data, which have been gathered during measurement campaigns with the Railway Driving and Validation Environment (RailDrIVE®) laboratory vehicle. In contrast to other data sources including OSM, these recorded positions are very accurate. The data base itself is implemented as a file-based SQLite data base and the aforementioned railway applications access the data by specified accessor and mutator methods. The data model follows the OSM structure driven by the concept “the simplest thing that could possibly work” (cp. [18]) defining only three basic datatypes: nodes, ways and relations. It is further possible to export the functional railway track description from the data base into the standardized railway data exchange format railML®.

RailML® is a data exchange format based on the Extensible Markup Language (XML) focussing on railway applications (cp. [7]). At the same time, railML.org is an open source initiative working constantly at the development of this data exchange format for railway applications, which currently exists in version 2.2. The XML schema files and corresponding documentation can be downloaded for free from the organisation’s website [16]. From the three existing railML® sub-schemas, infrastructure, timetable and rollingstock (cf. [17]), only the first one is used for the geo data export from the RailDrIVE® data base.

The resulting tool chain for converting spatial data of the railway track network from OSM to railML® is depicted in Figure 14-2.

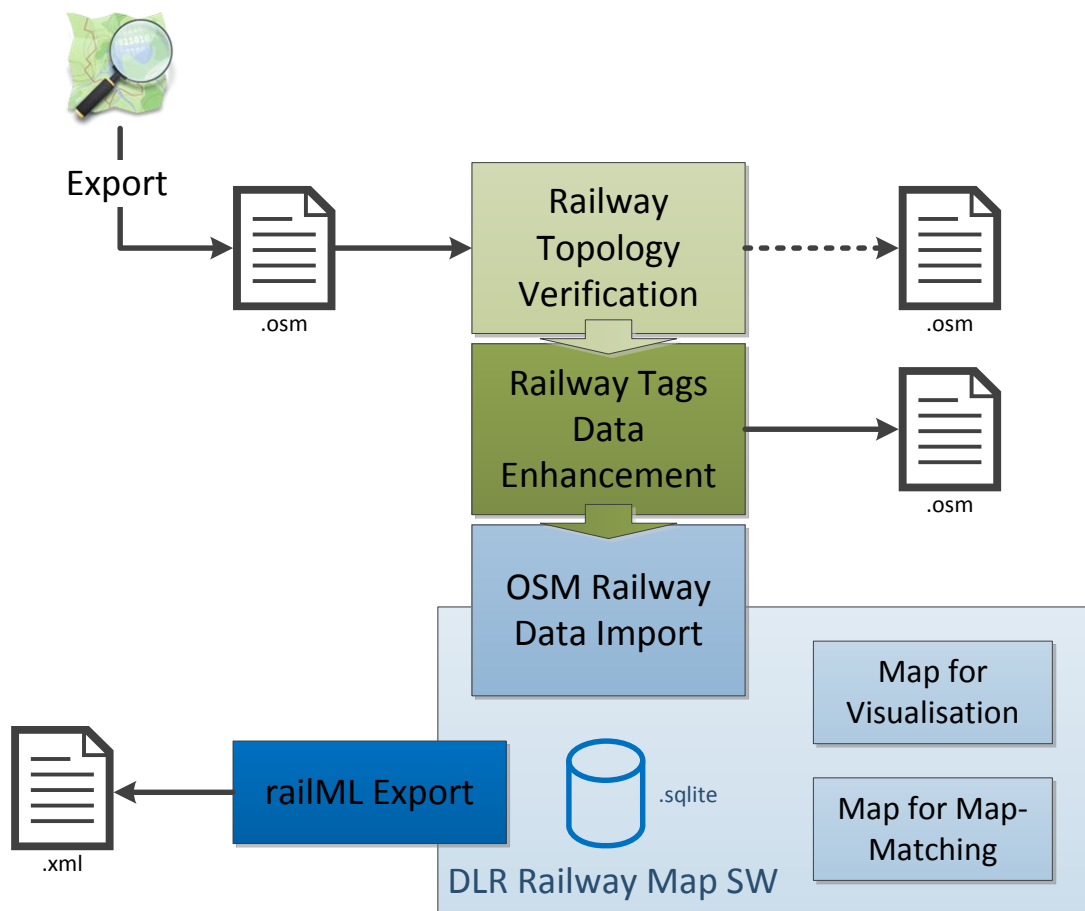


Figure 14-2: The tool chain for using spatial data of the railway track network as stored in the OSM data base for various railway applications.

The presented tool chain can be used to extract a topologic railway track network from the OSM data base and export it into the standardized data interface for railway applications railML®. However, this functional description of the railway infrastructure is not sufficient for a 3-dimensional simulation of a railway line. Required components like a digital terrain model or landscape model cannot be described using railML® and they are only partly and implicitly given in the OSM data. Therefore, it is analysed in the following how the spatial data available in the RailDriVE® data base can be used for the driver's cab simulation application in the RailSET® laboratory.

14.3.2 The SimWorld tool chain

In the Automotive department of the Institute of Transportation Systems driving simulators are established tools for the development work. The only difference between the RailSET® and the Automotive driving simulators is the type of the driven vehicle. The necessary software and hardware components are similar. Even the manual set up of virtual environments for Automotive driving simulators is time consuming and error-prone. That's why DLR came up with the idea to use remote sensing data for autonomous generation of these virtual environments (cp. [20]). The project SimWorld developed a first prototype of a tool chain to generate motoways and rural roads but the project also showed that only using remote sensing data isn't really working. In a second stage the project SimWorld^{URBAN} improved the tool chain by introducing established third party modules and also various data sources (cp. [19]) for being able to generate urban environments, too. E.g. for generating the 3-dimensional model of the roads Vires RoadDesigner is used, the city model is generated by Esri's CityEngine and the merging of all generated 3-dimensional models is done with the help of an improved TrianGraphic's Trian3D Builder.

The generation of the roads is an essential part for a driving simulator in Automotive domain and includes different sources from cadastral data though to surveying (cp. [1]). OSM data is in this case not very suitable because only logical information and only few layout data are available. For creating the surrounding at least the city model is important to re-create a city realistically. For SimWorld^{URBAN} a rule based approach is used (cf. [19]) to have the possibility for an easy update of the city model. Parameters for tuning the content creation are also gathered with the help of heterogeneous data. In a Bachelor thesis done in 2013 in the context of the SimWorld^{URBAN} project an approach is presented, how OSM data can be used to create a LOD1 (Level of Detail) model of the buildings (cf. [21]). In combination with the software tool CityEngine from Esri, it is possible to partly automatically generate a textured and modelled LOD2 city model. Despite all the available data within the OSM data base a lot of cleaning work has to be done and additional data like layouts are necessary. That's representative for many details that are essential for a realistic virtual environment in a driving simulator. The following figure shows an early version of a composed environment.



Figure 14-3: automatically generated and composed driving simulator environment

The tool chain approach extends the SimWorld data base of the previous project. The heart of this data base is the description of the road and the environment in a each standardized way. For the road description OpenDRIVE® is used. OpenDRIVE® is an open XML based format for describing the network logic and the road layout. railML® as described above can be seen as an analogue description format. The benefit of using open and established description formats is the possibility to use the stored data not only for driving simulators but also for large-scale traffic simulations. [6] describes a coupling between SUMO and a typical driving simulator like Vires VirtualTestDrive. Common basis is the usage of the very same road network defined in OpenDRIVE® format. Additional to that, SUMO can also import Shapefile data for improving the visualisation that is also stored in the SimWorld data based and generated by the tool chain before.

14.3.3 Adapting the tool chain for railway applications

By changing the RailSET® driver's cab simulation software environment from Zusi to Vires VirtualTestDrive, the basis for adapting the SimWorld tool chain enabling the usage of fused heterogeneous geo data from various sources for describing the transport network and the landscape is provided. In parallel to the Vires Road Designer the Vires Track Editor is a central component of the tool chain as it converts the functional description of the railway track network into a 3-dimensional model of the line, which can be visualized by the Vires simulation tools. As described above, the SimWorld data base is able to process heterogeneous geo data from various sources, e.g. digital terrain models, city models and track descriptions. By using the Trian3D Builder software tool the 3-dimensional track model can be fused with these data forming a complete 3-dimensional model of the landscape. The 3-dimensional model of the road is in a way replaced by the model of the railway line, all other generated parts remaining the same and the overall composing isn't touched. The resulting tool chain is depicted in the following figure.

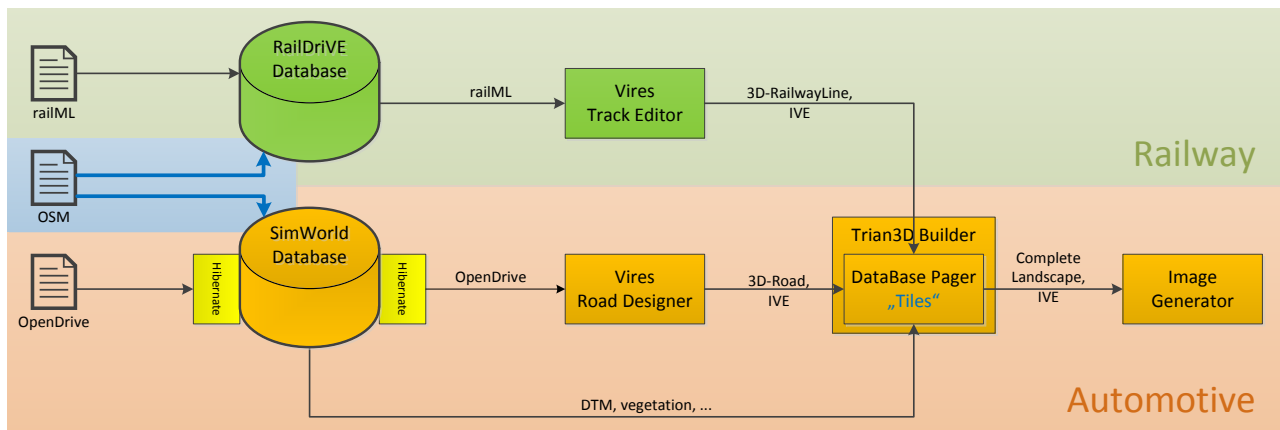


Figure 14-4: Structure of the tool chain for simulation-based landscape generation in the RailSET®

14.4 Implementation

The described tool chain connecting the RailDrIVE® geo data base with the driver's cab simulation RailSET® and in parallel connecting railway and automotive spatial data processing is currently being implemented. For both, the RailDrIVE® and the SimWorld data bases, the public available data base of OpenStreetMap can be seen as a huge data source providing various input for the road/rail network description and the model of the surrounding landscape.

The generation of a topological railway track network from the OSM data base has already been described in section 14.3.1. Figure 14-5 depicts an example of this processing applied for the railway line from Braunschweig to Gifhorn. The result in form of a topologically correct and with further tags enhanced railway infrastructure model is imported into the RailDrIVE® data base.

At the same time, OSM data can be also used for determining a digital terrain model: In general, the OSM nodes do not contain a fixed parameter describing their vertical position and only 2% of all the nodes have the tag "ele", which is defined as the elevation of a point in reference to the WGS84 coordinate system (cf. [10], [8]). However, in [9] and [11] a method for integrating global elevation data that has been collected during the Shuttle Radar Topography Mission (SRTM) with the OSM data base and thus gather information about the altitude of the landscape is described. The work presented in this paper uses the digital terrain model that has been obtained from the ordnance survey institution.

As already described in section 14.3.2 it is further possible to extract information about the height of buildings from the OSM data base. The data is used as input for the creation of the 3-dimensional LOD1 model of the buildings being a central part of the landscape in urban environments (cf. [21]).

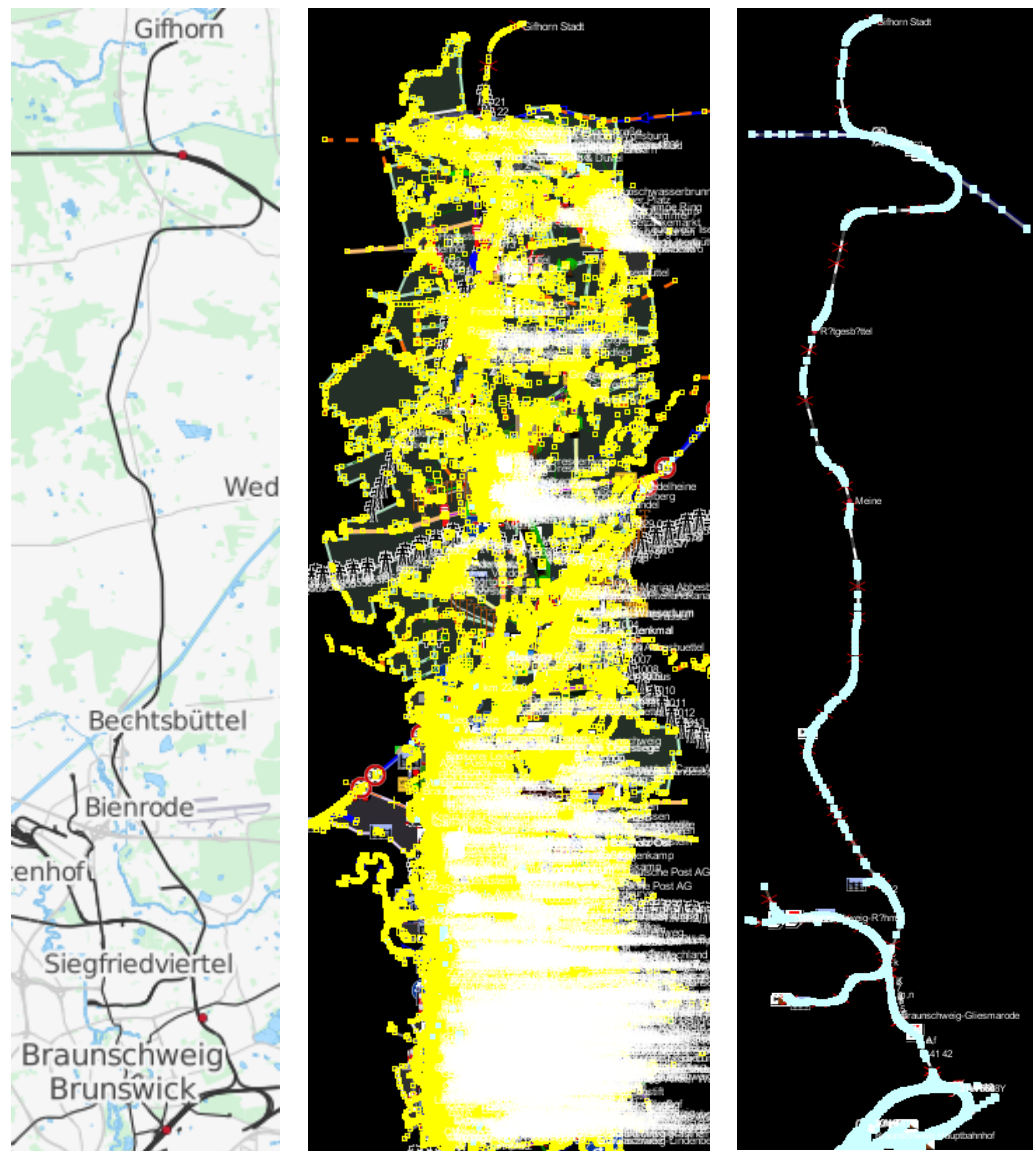


Figure 14-5: Generating railway track network from OSM: The first picture shows the rendered OSM geo data for the railway line between Braunschweig and Gifhorn (left). The geo data for this area are then extracted from the OSM data base and visualized in JOSM (middle). After processing the OSM-4-Railways tool chain, the corrected and enhanced railway line infrastructure remains (right) [14].

14.5 Conclusion

This paper described a tool chain with which various geo data from the free available geo data base OpenStreetMap can be used as input for modelling a railway line to be used in a railway driver's cab simulation. Further, this paper gave a short overview about the different options of using OSM data as input for the described railway simulation landscape creation tool chain focusing especially on the generation of the track network information. It has been sketched how, based on the Vires simulation software VirtualTestDrive and the so called SimWorld data base, the presented tool chain enables the fusion of heterogeneous data from different data sources.

In the next step, the performance of the railway simulation landscape creation tool chain shall be validated by comparing its result with a "conventionally" modelled railway line. Since the company Vires generated a simulation of the railway line from Braunschweig to Gifhorn crossing an area for which the DLR Institute of Transportation Systems already owns a lot of different spatial data sets, the validation will be applied for this track. It is expected that the

available geo data including OSM provides a basic set of information required to build a 3-dimensional simulation landscape. The results of this study will be presented in a follow-up publication.

14.6 References

- [1] Friedl, H.; Richter, A.: Fusion heterogener Geodaten zur Erstellung realer 3D-Welten am Beispiel einer Fahrsimulation. Shaker. GEOINFORMATIK 2012, Braunschweig, Germany, March 28-30, 2012.
- [2] Gantz, O.; Knollmann, V.; Jaschke, K. P.; Lemmer, K.: Visualisierung im Bahnlabor RailSiTe®. In: EURNEX - Zel 2006, 14th International Symposium EDIS, pp. 162-168 (2006).
- [3] Goodchild, M. F.: Citizens as sensors - the world of volunteered geography. In: GeoJournal, 69(4), pp. 211-221 (2007).
- [4] Haklay, M.: How good is volunteered geographic information? A comparative study of OpenStreetMap and Ordnance Survey datasets. In: Environment and Planning B: Planning and Design, volume 37, pp. 682-703 (2010).
- [5] Hammerl, M.; De Filippis, M.; Steinhäuser, I.; Torens, C.; Gantz, O.; Meyer zu Hörste, M.; Lemmer, K.: From a testing laboratory for railway technical components to a human factors simulation environment. In: 3rd International Rail Human Factors Conference, Lille, France, March 03-05, 2009.
- [6] Krajzewicz, D.; Richter, A.; Behrisch, M.; Erdmann, J.: Abbildung des Umgebungsverkehrs in einem Fahrsimulator. VDI Verlag. 4. Berliner Fachtagung Fahrermodellierung, Berlin, Germany, June 13-14, 2013.
- [7] Nash, A.; Huerlimann, D.; Schuette, J.; Krauss, V. P.: RailML - a standard data interface for railroad applications. In: Computers in Railways IX, pp. 233-240 (2004).
- [8] OpenStreetMap: taginfo keys – ele. <http://taginfo.openstreetmap.org/keys/?key=ele#overview>; last access: 31.12.2013.
- [9] OpenStreetMap: Contours. <http://wiki.openstreetmap.org/wiki/Contours>; last access: 31.12.2013.
- [10] OpenStreetMap: Key:ele. <http://wiki.openstreetmap.org/wiki/Key:ele>; last access: 31.12.2013.
- [11] OpenStreetMap: SRTM. <http://wiki.openstreetmap.org/wiki/SRTM>; last access: 02.01.2014.
- [12] OpenStreetMap: OpenStreetMap stats report run at 2014-03-24 00:00:14 +0000. http://www.openstreetmap.org/stats/data_stats.html; last access: 24.03.2014.
- [13] Rahmig, C.; Kluge, A.: Digital Maps for Railway Applications based on OpenStreetMap Data. In: Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems (ITSC 2013), pp. 1322-1327, The Hague, The Netherlands, 2013.
- [14] Rahmig, C.; Simon, A.: Using OpenStreetMap Geo Data for Map-Matching Applications in Railways. Accepted for: European Navigation Conference (ENC 2014), Rotterdam, The Netherlands, April 14-17, 2014.

- [15] Rahmig, C.; Simon, A.: Extracting Topology and Geometry Information from OpenStreetMap Data for Digital Maps for Railway Applications. Accepted for: 10th ITS European Congress, Helsinki, Finland, June 16-19, 2014.
- [16] railML.org: The XML-Interface for railway applications. <http://railml.org//index.php/home.html>; last access: 04.04.2014.
- [17] railML.org: The railML® subschemas. <http://railml.org//index.php/railml-schemas.html>; last access: 04.04.2014.
- [18] Ramm, F.; Topf, J.; Chilton, S.: OpenStreetMap. Using and Enhancing the Free Map of the World. UIT Cambridge, 2010.
- [19] Richter, A.; Friedl, H.; Guraj, V.; Ruppert, T.; Köster, F.: Developing a toolchain for providing automatically highly accurate 3D database. Verlag der Bauhaus-Universität Weimar; ConVR2011; Weimar, Germany, November 3-4, 2011.
- [20] Sparwasser, N.; Stöbe, M.; Friedl, H.; Krauß, T.; Meisner, R.: SimWorld – Automatic Generation of realistic Landscape models for Real Time Simulation Environments – a Remote Sensing and GIS-Data based Processing Chain. In: Road Safety and Simulation RSS 2007, Rom, Italy, November 7-9, 2007.
- [21] Weiter, J: Erstellen von 3D-Stadtmodellen aus nutzergenerierten, frei verfügbaren Geodaten in Kombination mit der Esri CityEngine. Bachelor thesis, Hochschule Karlsruhe – Technik und Wirtschaft (2013).

15 Advanced Traffic Light Information in OpenStreetMap for Traffic Simulations

David Rieck¹, Björn Schünemann², Ilja Radusch²

¹Fraunhofer FOKUS, Berlin, Automotive Services and Communication Technologies
david.rieck@fokus.fraunhofer.de

²Technische Universität Berlin, OKS/ Daimler Center for Automotive IT Innovations
{bjoern.schuenemann, ilja.radusch}@dcaiti.com

15.1 Abstract

In this paper, we show the development process of a new proposed feature for OpenStreetMap (OSM) traffic light tags. We introduce the needs for such kind of information in OSM and define requirements for our simulation needs. After comparing different traffic light tagging ideas and matching them to our requirements we come to the conclusion to extend the current classic way of tagging with OSM relations, which define turn restrictions and traffic light information. As a proof of concept a plugin for the popular OSM editor JOSM is shown as well as a conversion implementation of a complex intersection from OSM to SUMO is presented.

Keywords: OpenStreetMap, Traffic Signals, Traffic Simulation

15.2 Introduction

The use of OpenStreetMap (OSM) (1) data in traffic simulation environments is very common nowadays (2), (3) , (4). No other traffic network data sources offer such high quality data in urban areas for free without difficult licensing restrictions. Nevertheless, there are still some areas in OpenStreetMap, which could be improved to make traffic simulations out of OpenStreetMap data even better.

Traffic lights and lane information are OSM features which are still underrepresented even in areas, which already have been mapped in great detail. Reasons for this are mostly ease of use or need for this specialized information. Even simple information such as the number of lanes of a road are still used sparsely.

In this paper, we show how we extended the current OpenStreetMap traffic signal model with more detailed traffic signal data, how to convert this new information to a valid SUMO simulation scenario and how to use the traffic signal information in our Vehicle-2-X Simulation environment.

15.3 Extending the OSM Format

Traffic Lights in OpenStreetMap are usually modeled using only one node per intersection, regardless of the number of actual traffic lights, number of lanes at that intersection or intersection geometry (see Figure 15-, Figure 15-).

Today, there exists no concept in OpenStreetMap, which can be used to represent detailed traffic light information. There are proposed features that try to model more advanced signals information at intersections⁷ with focus on optimized information for navigation systems, but these cannot be used to include signal information nor are they optimized for simulation purposes.

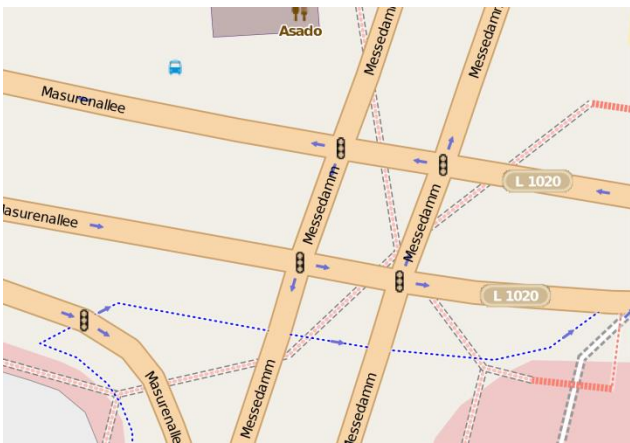


Figure 15-2 Complex Traffic Light Controlled Intersection

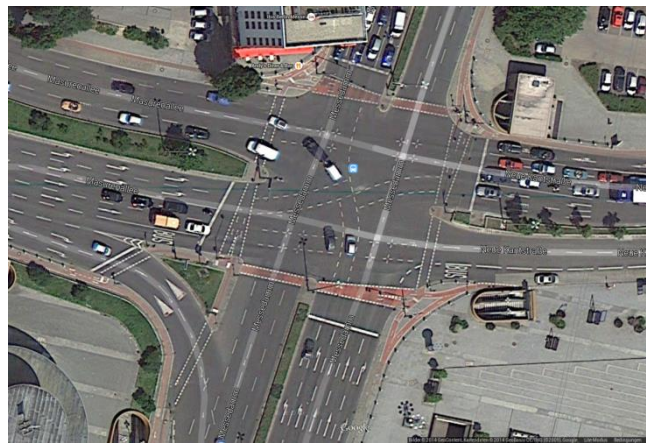


Figure 15-1 Satellite View of Complex Intersection

To enhance the traffic light model in OSM, we collected different requirements that a new solution might address and added a weighting from one to ten (ten being most important) to each requirement (in parentheses):

1. All possible (physical) assignments between lanes and traffic lights can be captured (10)
2. There are no adjustments needed for simple one-lane intersections (8)
3. Signal phases and timing information can be defined per traffic light head (5)
4. Map visualization is possible (2)
5. Mapping of intersections can be done efficient with existing tools (7)
6. Technical evaluation of intersections, lanes and traffic lights is possible (i.e. no undefined states, unique interpretation possibilities) (7)
7. Downwards compatibility, i.e. intersection geometry information remains untouched, existing tools should still work with the extended attributes (10)

⁷ https://wiki.openstreetmap.org/wiki/Proposed_features/Set_of_Traffic_Signals

To find a suitable solution, we analyzed different traffic light (TL) tagging ideas.

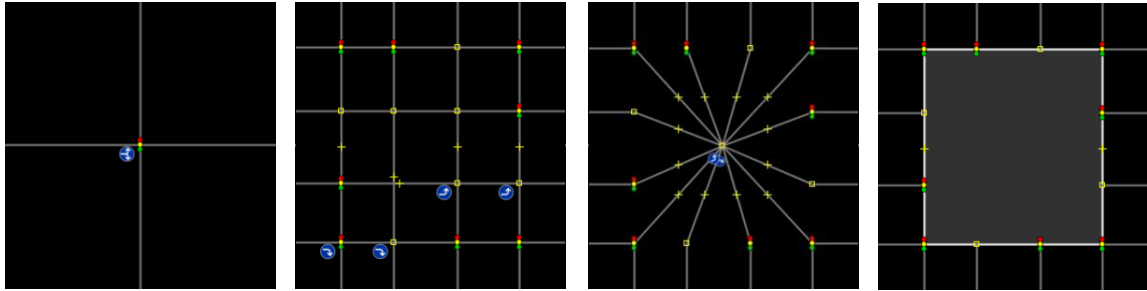


Figure 15-3 Classic, Lane, Star and Area Traffic Light Tagging

The *classic traffic light tagging* as it is used currently, contains one traffic light tag per intersection. Turn restrictions usually belong to whole ways.

The *lane traffic light tagging* uses a visual way of traffic light tagging. Each lane is modeled individually, traffic lights and turn restrictions are applied directly to the lanes.

Star traffic light tagging is an abstract way to model lanes logical. The intersection node as used in the classic traffic light tagging stays the same, but roads are splitted into individual lanes.

The *area traffic light tagging* models the whole intersection as one OSM-area, where each lane connects to the intersection.

Table 15-1 Comparison of different Traffic Light Tagging Methods

Tagging Type	Pros	Cons
Classic TL Tagging	<ul style="list-style-type: none"> • Simple geometry 	<ul style="list-style-type: none"> • Phases and timing not possible • No lane-specific TL-Tagging
Lane TL Tagging	<ul style="list-style-type: none"> • Traffic Light tags per lane • Turn restrictions per lane 	<ul style="list-style-type: none"> • Complex geometry • High mapping costs • Uses huge amount of data
Star TL Tagging	<ul style="list-style-type: none"> • Logic modeling of lanes • TL and Turn restrictions per lane • Downwards compatible • Medium mapping costs • Individual lanes 	<ul style="list-style-type: none"> • Visual representation != logic representation
Area TL Tagging	<ul style="list-style-type: none"> • Individual lanes 	<ul style="list-style-type: none"> • Incompatible with current routing engines • Improper usage of areas to model intersection-connections

By comparing each requirement with the various traffic light modeling methods, we came to the following requirements matrix (see Table 15-):

Table 15-3 Fields of the new traffic_signal relation

Attribute	Description
type	type description of the relation (traffic_signals)
ref:lanes:from	mapping of input lanes
ref:lanes:via	mapping of via lanes
ref:lanes:to	mapping of outgoing lanes
phases	description of the traffic signal phases
timing	description of traffic signal timings

15.4 Conversion of OSM Files

To convert standard OSM data to our simulator specific formats, we already use a tool called *VSimRTI scenario-convert* to import OSM data and export to different formats, e.g. SUMO *.nod.xml, *.edg.xml and *.tll.xml files. We extended this tool to make use of the additional traffic light information and export the relevant files to a SUMO and VSimRTI compatible format.

In this paper, we show the conversion process from the raw OSM file to the SUMO traffic network. Some OSM features can be translated direct to the corresponding parts in the SUMO files (see Figure 15-), while in other parts a more complex transition is needed (e.g. intersection lane modeling)

```

<relation id='-36'>
  <member type='way' ref='-16' role='to' />
  <member type='node' ref='-4' role='signal' />
  <member type='way' ref='-14' role='from' />

  <tag k='ref:lanes:from' v='1|2' />
  <tag k='ref:lanes:to' v='3|4' />
  <tag k='phases' v='g|y|r|r' />
  <tag k='timing' v='31|4|31|4' />
  <tag k='type' v='traffic_signals' />
</relation>

<tlLogic id="0" type="static">
  <phase duration="31" state="gr" />
  <phase duration="4.0" state="yr" />
  <phase duration="31" state="rg" />
  <phase duration="4.0" state="ry" />
</tlLogic>

<connection from="7_-8" to="6_-4"
  fromLane="0" toLane="0"
  t1="0" linkIndex="0" />

<connection from="7_-8" to="6_-4"
  fromLane="1" toLane="1"
  t1="0" linkIndex="0" />

```

Figure 15-5 OSM Traffic Signal Extension and Corresponding Tags and Values in SUMO

A screenshot of the conversion can be seen in Figure 15-. In this example, the intersection in Figure 15-3 was extended with the advanced traffic signal information, which was gathered by measuring the traffic signal phases. Then, the OSM file was parsed using *VSimRTI scenario-convert* and *SUMO netconvert* created the SUMO files.

The SUMO tool *netconvert* offers also an osm conversion feature to import OpenStreetMap files directly and write SUMO compatible files, including traffic light guessing and intersection joining. Unfortunately, this method didn't work due to the intersection complexity. With further effort on modeling the intersections, better results are expected.

15.5 Simulation

We use the generated traffic network and traffic light programs in our V2X simulations using the Vehicle-2-X Simulation Runtime Infrastructure (VSimRTI) (5).

VSimRTI is developed by Fraunhofer FOKUS is a framework for simulation of Vehicle-2-X scenarios by coupling different simulators (e.g., traffic simulator, network simulator, application simulator...). The framework is based on the High Level Architecture (6) which offers mechanisms to connect and synchronize different simulators using a common runtime infrastructure.

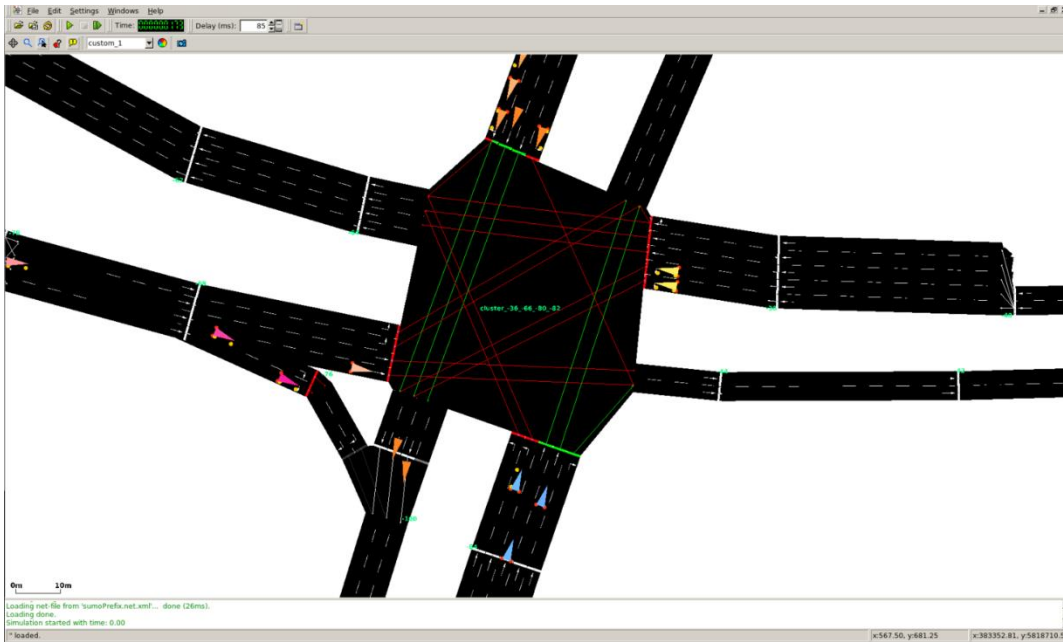


Figure 15-6 SUMO Traffic Simulation with junction connections shown

SUMO is mainly used as traffic simulator in VSimRTI, whereas communication and applications for vehicles are simulated on other simulators. Information about traffic lights is simulated in SUMO, but can be altered from an application running on a vehicle.

15.6 Conclusion

The presented extension of the classic traffic light definition in OpenStreetMap offers a lot of advantages for traffic simulations. By adding relations with information of phases, timing and turn restrictions, even complex intersections can be modeled comparatively easy using only OpenStreetMap. Furthermore, this information can also be used easily in traffic simulation tools such as SUMO. Lane based turn restrictions or even lane numbering alone being one crucial information for routing engines, this feature can also help to spread OpenStreetMap data even more.

15.7 Outlook

Although the presented methods allow for tagging complex intersections and advanced traffic light definitions, the proposed features currently only include static traffic light information. Further traffic signaling mechanisms, e.g. induction loops or camera controlled

traffic lights, public transportation prioritized traffic lights or green wave settings or daily/weekday setups are not handled by the presented feature. Adding these or offer possibilities to include some kind of online requests for the current status might add some valuable information to next generation routing applications.

15.8 Acknowledgements

We'd like to thank Tristan Wagner, Andreas Mentz, André Beyer and Rujun Wang for their valuable contribution to this paper.

15.9 References

- [1] *Openstreetmap: User-generated street maps*. Haklay, Mordechai und Weber, Patrick. 4, s.l. : IEEE, 2008, Pervasive Computing, IEEE, Bd. 7, S. 12-18.
- [2] *Sumo-simulation of urban mobility-an overview*. Behrisch, Michael, et al., et al. 2011. SIMUL 2011, The Third International Conference on Advances in System Simulation. S. 55-60.
- [3] *OpenStreetMap for traffic simulation*. Zilske, Michael, Neumann, Andreas und Nagel, Kai. 2011. Proceedings of the 1st European State of the Map--OpenStreetMap conference. S. 126-134.
- [4] *Efficient traffic simulator coupling in a distributed V2X simulation environment*. Rieck, David, et al., et al. 2010. Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques. S. 72.
- [5] *V2X simulation runtime infrastructure VSimRTI: An assessment tool to design smart traffic management systems*. Schünemann, Björn. New York, NY, USA : Elsevier North-Holland, Inc., October 2011, Computer Networks, Bd. 55, S. 3189-3198. ISSN: 1389-1286 DOI: <http://dx.doi.org/10.1016/j.comnet.2011.05.005>.
- [6] of Electrical, Institute und Engineers, Electronics. *IEEE standard for modeling and simulation (M&S) High Level Architecture (HLA)--Framework and Rules. IEEE Standard 1516.1*. New York, NY, USA : IEEE, 2000. S. 475. ISBN: 0-7381-2622-5.
- [7] *Performance study of a Green Light Optimized Speed Advisory (GLOSA) application using an integrated cooperative ITS simulation platform*. Katsaros, Konstantinos, et al., et al. 2011. Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International. S. 918-923.

16 Towards a Mobile-based ADAS Simulation Framework

João S. V. Gonçalves¹, João Jacob², Rosaldo J. F. Rossetti¹, António Coelho² and Rui Rodrigues²

Departamento de Engenharia Informática, Faculdade de Engenharia da Universidade do Porto

¹Laboratório de Inteligência Artificial e Ciência de Computadores

²INESC TEC

R. Dr. Roberto Frias s/n, 4200-465 Porto, Portugal

{joao.sa.vinhas, joajac, rossetti, acoelho, ruirodrig}@fe.up.pt

16.1 Abstract

The increasing number of vehicles and mobile users has led to a huge increase in the development of Advanced Driver Assistance Systems (ADAS). In this paper we propose a multi-agent-based driving simulator which integrates a test-bed that allows ADAS developers to compress testing time and carry out tests in a controlled environment while using a low-cost setup. We use the SUMO microscopic simulator and a serious-game-based driving simulator which has geodata provided from standard open sources. This simulator connects to an Android device and sends data such as the current GPS coordinates and transportation network data. One important feature of this application is that it allows ADAS validation without the need of field testing. Also important is the suitability of our architecture to serve as an appropriate means to conduct behaviour elicitation through peer-designed agents, as well as to collect performance measures related to drivers' interaction with ADAS solutions.

Keywords: Mobile ADAS, Driving Simulators, Serious Games, SUMO.

16.2 Introduction

The technological advances on both the mobile and transportation industries are remarkable. This has made the development of ADAS an interesting topic [1]. However, even though most high-end cars nowadays ship with built-in embedded systems, most of the older cars do not have such devices. This brings about an interesting research opportunity, which is to develop and test ADAS that run on low-cost devices, such as an Android tablet or smartphone.

The main goal of this paper is to describe the methodology of our MultiAgent System (MAS) based driving simulator, integrating SUMO microscopic simulator with driving simulators. We also intend to describe our implementation of a test-bed to easily develop ADAS using the system, simulating their use in a low-cost and controlled environment.

There are several benefits to testing an ADAS in a simulated environment rather than on a real scenario. As the tests are not conducted in a real physical location, they are not subjected to travel times, traffic or other adverse conditions which could render them mute. This, as well as being able to deploy the simulator in low-cost computers, and therefore reaching more test subjects, leads to time compression of the tests. Noticeably, cost reduction is

another significant benefit as the electrical cost of running a simulator is dismissive when compared to fuel costs of real world testing. Besides preventing the safety risks inherent to driving, simulation allows us to control the test environment and manipulate it according to the specificities of the ADAS being tested.

The objective of our work was to develop the MAS and include a test-bed that was easy to implement and replicate in low-cost environments. We aimed to combine SUMO microscopic simulator with IC-DEEP, which is a driving simulator developed at LIACC [2], with the GeoStream framework developed at SI&CG, as well as to enhance them with logs of simulated GPS positions in a mobile device. To achieve so, a mobile application/service was developed in order to receive this communication from the simulator and override the default GPS sensor of the device. We wanted to make it easy to extend the communication between the simulator and a mobile device, providing the latter with more information such as the current speed limit, semaphoric information, or other data from the network.

This work aims to contribute with a novel multi-faceted methodology to simulate and research multiple human factors in Intelligent Transportation Systems (ITS) and, particularly, with a novel approach to test ADAS that will enable developers to validate and test their applications more easily and efficiently while reducing costs.

In the following sections we describe the development and results of our implementation. In section 3 we introduce some related state-of-the-art works on the subject of ITS, focusing on simulation, on the integration of different scope simulators and also on the topic of serious games. We then describe our approach, architecture and development details. Finally we present our preliminary verification as well as their analysis in section 5. We finish this paper with a set of conclusions and interesting future work.

16.3 Background & Related Work

The Artificial Transportation Systems (ATS) [3], [4] concept has been one of the main research topics in the IEEE ITS Society [5]. A typical approach to ATS modeling and development is the MAS metaphor. Another potentially concomitant approach is the HLA concept. The concept is based on the idea of distributed simulation, so as to meet the requirements of all usages and users rather than of a single simulation model and analysis perspective [6].

In [7], authors propose to integrate a driving simulator and a traffic microsimulator, in an attempt to tackle the mutual-dependence between the driver's behavior and traffic conditions.

Combining SUMO microscopic traffic simulation [8], using MAS capabilities, with other simulators has also been researched [9]. Authors in [6] have studied a HLA-based approach to simulate electric vehicles in Simulink and SUMO. Driver-centric simulation has been researched by authors in [10], where they have developed a simulation tool that provides feedback back to the network based on the driver's behaviour.

Driving simulators are no doubt an important tool when researching ATS, specially so when studying the influence of human factors in driving faults [2]. These faults often occur in direct consequence of performing secondary tasks while driving [11]. In [12] authors introduce a game-engine-based for modeling and computing platform for ATS. They describe the artificial population both in their macroscopic and microscopic aspects.

Regarding driving simulators with ADAS testing capabilities, authors in [13] propose a reconfigurable driving simulator with several components to accommodate different ADAS testing or training. A framework for ADAS assessment and benchmarking has been developed by [14], with configurable scenarios and 3D scenes and multiple sensors input.

Authors in [15] developed a Full Speed Range Adaptive Cruise Control with their platform for ADAS prototyping and evaluation, SiVIC. The platform is capable of reproducing vehicle and sensor behaviors in a realistic fashion, according to the configured environment in the simulator. The developed platform also simulates noised and imperfect data.

A system comprising a large scale driving simulator, built in a 360 deg full dome with 3D scenes from real city area has been developed by [16]. The system contains a multitude of features such as real-time hardware-in-the-loop, wireless communication devices and bio signal analysis and is used to develop and test ADAS as well as Advanced Safety Vehicle, ITS infrastructure and others.

16.4 Methodological Approach

The proposed system architecture is as described in Figure 16-1. The main module of the system is the SUMO simulator, which is responsible for the network's multi-agent microscopic simulation, and has multiple driving agents. This module provides an overview of the whole MAS and can be manipulated directly.

The SUMO module also acts as a "central server", providing all the essential information for both IC-DEEP and the High Fidelity Simulator. This information consists of the network infrastructure and the agents in the system, whereas terrain morphology and road or building geometry are provided by the GeoStream framework.

Both of the driving simulators have a local representation of the whole MAS and are capable of controlling any driving agent. The simulators are also able to connect to an Android device and pass along all the information deemed necessary, such as the GPS coordinates of the current driving agent being controlled. The Android device is running a service that receives the incoming connections from the simulator and also the ADAS being tested. The dotted area in Figure 16-1 corresponds to the developed components as of the writing of this paper.

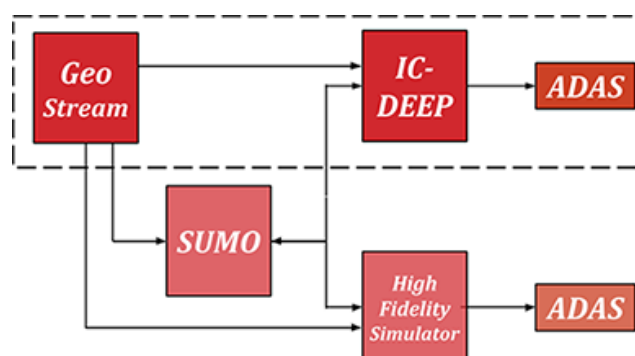


Figure 16-1: Overview of the system's architecture

16.4.1 Simulators & GeoStream Framework

The proposed system architecture has two simulators; however, as of the writing of this paper, only the IC-DEEP simulator has been enhanced and integrated into the framework. The latter is implemented in Unity3D and has the GeoStream framework embedded directly. The GeoStream framework connects with OpenStreetMaps, Google Geolocation API, Google Altitude API and other data providers in order to fetch the required geographical information of a given location and remodel it in a fashion which can be interpreted by all the simulators in a coherent and consistent way. This is specially important to the SUMO microscopic simulator as the raw network data imported from OpenStreetMaps, typically, generates unrealistic ways and intersections. The information generated by GeoStream framework is then parsed into both simulators to generate a 3D scene that is representative of the chosen test location.

16.4.2 Mobile Device

The Android service sets the current GPS location using *MockLocation* API to override the default location provider. All applications running on the mobile device that use or perceive the current location will also be affected by the running service.

The new GPS coordinates are sent from the simulator every second; however, this value is a parameter of the simulator and so can be adjusted to the specific needs of each scenario. The developed service can be run as a standalone application, and thus testing the ADAS mobile applications independently, as shown in the left side of Figure 16-2. There is also the option to use the service as a library in any Android application, as long as it matches API level 19, as shown in the right side of Figure 16-2.

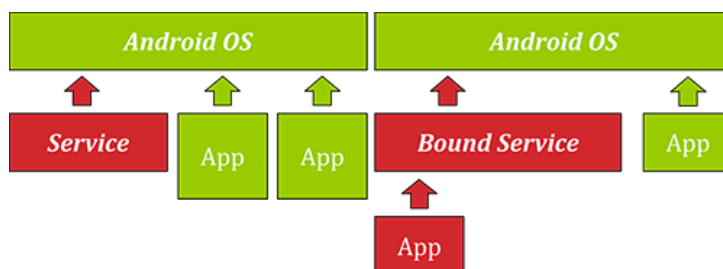


Figure 16-2: Standalone mobile application (left) and mobile application with included library (right)

16.4.3 Interaction of Driving Simulators and Android

A typical interaction between the simulators and the mobile devices is shown in Figure 166-3. The modules are connected via TCP-IP sockets due to implementation simplicity. The communication messages are formatted in JSON and therefore the message contents can be easily changed to add different kind of data.

The basic message template contains two compulsory fields, which are *latitude* and *longitude*. Other optional fields are the current *speed*, the GPS *accuracy*, the message *timestamp* or even the *speed limit* from the current location. A specific instantiation of this interaction is discussed in the next section.

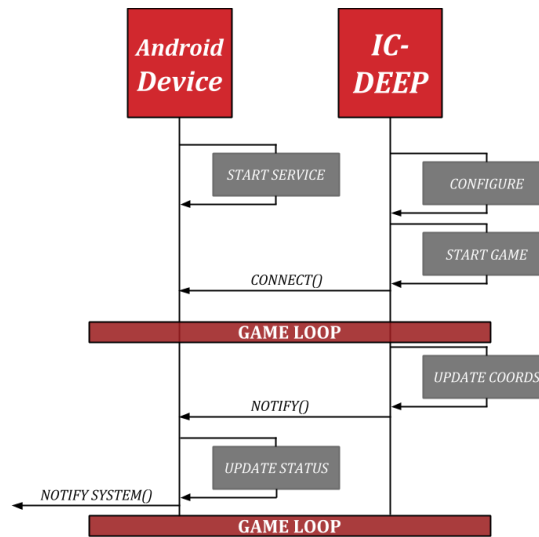


Figure 166-3: Typical interaction between IC-DEEP simulator and an ADAS

The coupling of SUMO microscopic simulator with the driving simulators, namely with IC-Deep, uses the same methodology as implemented and described elsewhere [17]. However, this raises some issues regarding the communication channel, as the SUMO TraCI protocol uses sockets and currently does not support more than one active socket. This is obviously a bottleneck when controlling multiple driving agents and a possible SUMO extension to support parallelism in terms of communication is in study.

16.5 Preliminary Verification

The preliminary verification to assess the proof of concept and also the efficiency of the developed architecture focused on the modules in the dotted area of Figure 16-1, the remainder of the system will be developed later on, as mentioned in the next chapter. We have divided the verification into two independent tests. Both of the tests were performed in the same geographical location, which was Porto's downtown, on *Avenida dos Aliados*, as seen in Figure 16-4.

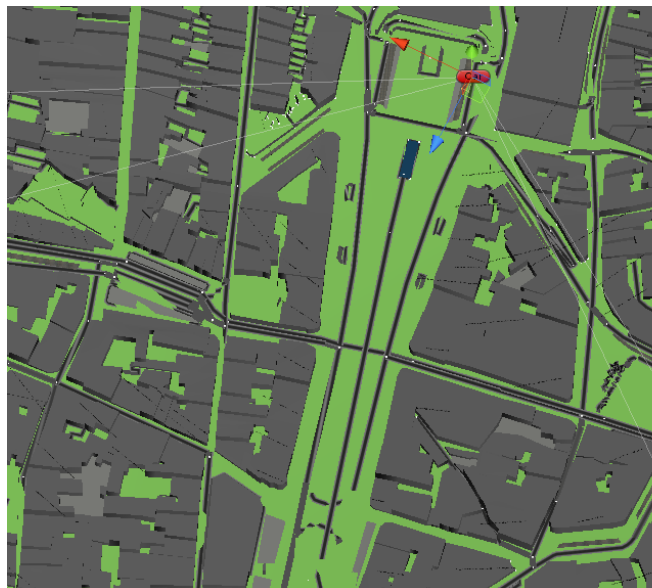


Figure 16-4: Generated 3D Scene orthographic view

In the first experiment we test the simulator accuracy to represent real-world scenarios using the GeoStream framework. The other test aims to emulate the GPS signal on the mobile device.

16.5.1 Simulator Accuracy

To test the simulator accuracy we have collected multiple GPS trace logs while driving a real car in the selected geographical location. We have then overlaid a visual representation of the obtained traces on the simulator and on Google Earth both, the results can be seen in Figure 166-5. The results show that the generated 3D scene is highly representative of the real world location. In one of the trace logs we have noticed an error and highlighted it in Figure 166-5, this error happens due to the data being collected as raw, untreated GPS, where the *road matching* algorithm [18] has not been applied. This is also an interesting result, as the error can be seen in both the simulator and Google Earth alike.



Figure 166-5: GPS traces on the simulator (left) and Google Earth (right)

Apart from testing the fidelity of the simulation with GPS trace logs, it is also noticeable that the ortographic view of the generated 3D scene very much resembles the satellite image of the same location, as it can be seen in Figure 16-6.



Figure 16-6: Satellite image of Av. dos Aliados

16.5.2 Mobile Device ADAS

To test the communication and emulation in the mobile device the setup consisted of a basic usage scenario, using a simple *ADAS* that shows the user his current and average speed, the total kilometers traveled, and, most importantly, warns him when he exceeds the speed limit of the current location as shown in Figure 166-7.

The interaction between the simulator and the mobile application followed that of Figure 166-3. To run the simulation the mobile application must be started and the device's IP address, which is shown in the application initial screen, must be entered in the simulator configuration screen. After this the simulation can start and the simulator internally updates the geographical coordinates as the driver traverses the network. These coordinates are passed on to the mobile device as described above, every second and via a JSON formatted message over a TCP-IP socket.

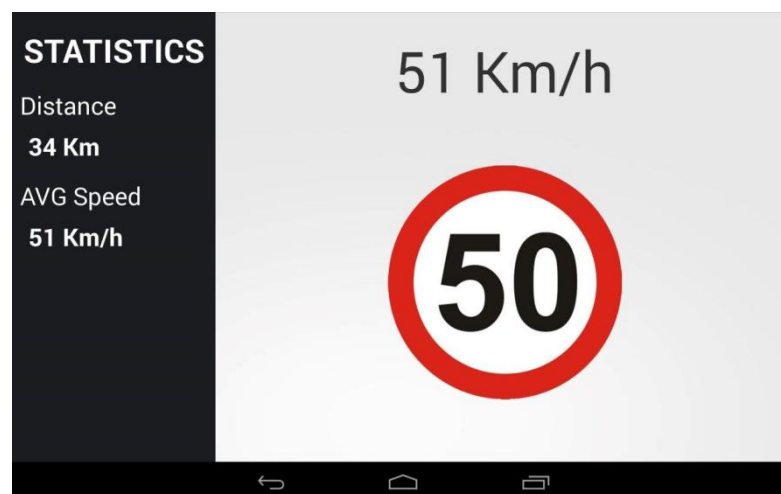


Figure 166-7: Developed ADAS

In this particular simulation the information sent to the mobile device consists of the current GPS coordinates and the speed limit of the current location. The heading of the vehicle is calculated internally by the mobile application using a simple algorithm that computes the

bearing with the last two known locations and thus, even though this can be done easily, there is no need to pass on an orientation variable from the simulator.

The main goal of this experimentation was to understand whether or not the mobile device simulated GPS position and calculated speed matched those of the simulator. We have used the driving simulator and Google Maps application to compare the marked position while driving. To compare the driving speed we have used the developed ADAS.

Even though the results from the simulator and the mobile device were not recorded, any inaccuracies were not noticed when testing. The only possible minor differences would be due to the fact that the Google Location API on the Android device automatically adjusts the current position to the nearest road, which is, as mentioned above, a technique called *road matching*.

16.6 Conclusion

In this paper we presented a multi-faceted MAS-based driving simulator methodology. The presented framework can be used to simulate and test multiple aspects in human factors in ITS, generally. Among others we identify some that we consider more expressive of the system's spread, such as supporting a Serious Game [19] to test driving behaviours and ergonomics, simulating driver's idiosyncrasies effects on the ATS with peer-designed agents, and also prototyping and validating Advanced Driver Assistance Systems.

The preliminary verification has illustrated the system efficiency and usability, as well as its ability to accurately represent real-world scenarios without the need of extensive 3D modeling or expensive hardware setups. This ability allows researchers to conduct studies regarding singularities of the different geographical locations.

As a great advantage over other systems we point out the fact that our system is always up to date in terms of real-world mapping, and also that there is no need to waste any time creating a scene when the sole purpose is to test an ADAS or do any other kind of simulation.

In addition to implementing the remaining components of the proposed methodology, there is an ambitious workload of further developments. We would like to point out some that we consider priority and more challenging. We believe it would be interesting to support batch simulations, in order to collect significant data and extract more elaborate conclusions. There are also improvements specific to driving simulators that we envisage, such as more detailed scenarios and improved physics.

It would also be interesting to develop cache servers that could store the responses from external services, and thus improve loading times. Another interesting enhancement would be to allow multiple agents to connect to multiple ADAS, simulating distributed ADAS applications while extending SUMO capabilities. There are also refinements to be done in the GeoStream framework, specially regarding road generation and also importing models and textures for different buildings.

16.7 References

- [1] M. Baumann, A. Keinath, J. F. Krems, and K. Bengler, "Evaluation of in-vehicle HMI using occlusion techniques: experimental results and practical implications.," *Appl. Ergon.*, vol. 35, no. 3, pp. 197–205, May 2004.
- [2] J. Goncalves, R. J. F. Rossetti, and C. Olaverri-Monreal, "IC-DEEP: A serious games based application to assess the ergonomics of in-vehicle information systems," in

- Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on, 2012*, pp. 1809–1814.
- [3] F.-Y. Wang, "Integrated intelligent control and management for urban traffic systems," in *Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE, 2003*, vol. 2, pp. 1313–1317 vol.2.
- [4] F.-Y. Wang and S. Tang, "A framework for artificial transportation systems: from computer simulations to computational experiments," in *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE, 2005*, pp. 1130–1134.
- [5] R. J. F. Rossetti, R. Liu, and S. Tang, "Guest Editorial Special Issue on Artificial Transportation Systems and Simulation," *Intell. Transp. Syst. IEEE Trans.*, vol. 12, no. 2, pp. 309–312, 2011.
- [6] J. Macedo, Z. Kokkinogenis, G. Soares, D. Perrotta, and R. J. F. Rossetti, "A HLA-based multi-resolution approach to simulating electric vehicles in simulink and SUMO," in *Intelligent Transportation Systems - (ITSC), 2013 16th International IEEE Conference on, 2013*, pp. 2367–2372.
- [7] V. Punzo and B. Ciuffo, "Integration of Driving and Traffic Simulation: Issues and First Solutions," *Intell. Transp. Syst. IEEE Trans.*, vol. 12, no. 2, pp. 354–363, 2011.
- [8] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo-simulation of urban mobility-an overview," in *SIMUL 2011, The Third International Conference on Advances in System Simulation, 2011*, pp. 55–60.
- [9] R. Maia, M. Silva, R. Araujo, and U. Nunes, "Electric vehicle simulator for energy consumption studies in electric mobility systems," in *Integrated and Sustainable Transportation System (FISTS), 2011 IEEE Forum on, 2011*, pp. 227–232.
- [10] P. Gomes, C. Olaverri-Monreal, M. Ferreira, and L. Damas, "Driver-centric VANET simulation," in *Communication Technologies for Vehicles*, Springer, 2011, pp. 143–154.
- [11] D. Kern, M. Müller, S. Schneegaß, L. Wolejko-Wolejszo, and A. Schmidt, "CARS-Configurable Automotive Research Simulator.," in *Mensch & Computer Workshopband, 2008*, pp. 256–260.
- [12] Q. Miao, F. Zhu, Y. Lv, C. Cheng, C. Chen, and X. Qiu, "A Game-Engine-Based Platform for Modeling and Computing Artificial Transportation Systems," *Intell. Transp. Syst. IEEE Trans.*, vol. 12, no. 2, pp. 343–353, 2011.
- [13] B. Hassan, J. Berssenbrugge, I. Al Qaisi, and J. Stocklein, "Reconfigurable driving simulator for testing and training of advanced driver assistance systems," in *Assembly and Manufacturing (ISAM), 2013 IEEE International Symposium on, 2013*, pp. 337–339.
- [14] S. Noth, J. Edelbrunner, and I. Iossifidis, "An integrated architecture for the development and assessment of ADAS," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on, 2012*, pp. 347–354.

- [15] D. Gruyer, S. Pechberti, and S. Glaser, "Development of Full Speed Range ACC with SiVIC, a virtual platform for ADAS Prototyping, Test and Evaluation," in *Intelligent Vehicles Symposium Workshops (IV Workshops), 2013 IEEE*, 2013, pp. 93–98.
- [16] S. Yu, S.-Y. Lee, M.-S. Kim, and D.-G. Lee, "Development and Evaluation of ITS Devices Using KAAS(KATECH Advanced Automotive Simulator) System," in *SICE-ICASE, 2006. International Joint Conference*, 2006, pp. 2116–2120.
- [17] J. L. F. Pereira and R. J. F. Rossetti, "An integrated architecture for autonomous vehicles simulation," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, 2012, pp. 286–292.
- [18] M. El Najjar and P. Bonnifait, "A Road-Matching Method for Precise Vehicle Localization Using Belief Theory and Kalman Filtering," *Auton. Robots*, vol. 19, no. 2, pp. 173–191, 2005.
- [19] R. J. F. Rossetti, J. E. Almeida, Z. Kokkinogenis, and J. Goncalves, "Playing Transportation Seriously: Applications of Serious Games to Artificial Transportation Systems," *Intell. Syst. IEEE*, vol. 28, no. 4, pp. 107–112, 2013.

17 Map matching and cycling infrastructure analyses with SUMO and python

Joerg Schweizer, Federico Rupi;

University of Bologna, DICAM, Viale Risorgimento, 2, 40136 Bologna, Italy

{Joerg.Schweizer,Federico.Rupi}@unibo.it

17.1 Abstract

The open tools SUMO and Python and the open database OpenstreetMap (OSM) are used to analyze urban cycling infrastructure. Endomondo is a well known database to upload GPC traces for particular purposes, as for example cycling for work/study purposes. SUMO has been used to convert OSM to a road network while different Python modules helped to match Endomondo GPS traces to network edges and to reconstruct the routes. This means that the attribute-rich OpenstreetMap database can now be combined with routes generated from GPS traces. One possible application is to calibrate trip time model or route choice models for cyclists. This work describes a road matching procedure, and shows road matching results from traces created during an Endomondo contest in Bologna. Successively the identified routes are used to calibrate different trip-time models.

Keywords: SUMO, Python, OpenStreetMap, Endomondo, GPS, road-matching, SUMOPy, bikeways, trip time models, route choice models.

17.2 Introduction

The use of open software and open data for transport modeling in general and bikeway planning in particular has attracted considerable research interests in recent years. There has been a constant evolution in the use and analyses of geo-referenced data: in 1995 demand models for cycling have been calibrated with geo-referenced topological characteristics and stated preference surveys [1, 2] whereas a similar work has been performed by combining data from Geographic Information Systems (GIS) with web-based questionnaires [3]. But only the recent diffusion of smart phones has brought the data collection to an unprecedented quality level: on the one hand, citizens started to record their commuter trips with the GPS functionality of their cell phones and to upload the traces on central databases like Endomondo. On the other hand, the community helped to expand and refine the geo-referenced maps on OpenStreetMap. The combination of transport supply (the road network) and transport demand (the bike trips) are very valuable information for transport planners and for the traffic management because it allows analyzing how the network is used and why. This means there is a real necessity to reconstruct the routes recorded by smart phones on geo-referenced maps and to conduct analyses using specific road attributes.

So called map-matching algorithms have been developed to identify the road segments based on imprecise GPS data. In an early attempt, Dalumpines and Scott have proposed an algorithm where GPS data is only used to identify the location of start and end location of the

trip, while a shortest path algorithm applied to the road network has been used to connect origin and destination with a feasible route [4]. This map matcher has been used to extract basic statistics on cycling behavior in Austin, Texas, such as location (heat maps), the type of used roads as well as average speeds [5]. A road matching algorithm developed by Marchal et. al.[6] is based on maximizing the likelihood that a route is identical to the real route from where the GPS trace has been sampled. The latter approach takes into account all GPS points measured from the start to the end of the trip. Such map-matched GPS traces have been used to calibrate route choice models of cyclists [7, 8]. In the present work, the geo-referenced world-wide road network of OpenStreetMap [9] has been used to identify the routes of volunteer cyclists who registered their GPS traces by means of cell phones on Endomondo [10] (or any other system that provides GPS traces). Once the single road edges of the route are recognized, it is possible to analyze the road attributes associated with the edges of the road or bikeway chosen by the cyclists. Such route-attributes can be retrieved again through OpenStreetMap data or other resources. In the present work, the OpenStreetMap bikeway attributes within the concerned study area have been enriched by cycling specific characteristics. By retrieving the edge characteristics from the matched routes, one can extract myriads of information on the infrastructure usage and the cyclist's behavior. It is not only possible to assess how often and where cyclists use bikeways or roads; the GPS traces provide also timing information with each transmitted geo-referenced point. This allows estimating the duration of the cyclists on each edge of the route and comparing it against the attributes of the road or bikeway. Based on this information, a travel time model has been calibrated which contains various bikeway attributes. Such analyses can help the city to verify whether the present bikeway network is effectively used and where improvements are most needed.

The software framework used for the present analyses is centered around various tools of SUMO [11,13] and Python [13]. In brief, SUMO has been used to generate a transport network from the open database OpenStreetMap and to identify the routes that fit best the GSM traces. For this map matching process also the python modules pyproj and shapely [16] have been involved. The numerical analyses and model calibration has made extensive use of the Scipy [15] and Numpy [14] package. The main focus of this work is on the map matching algorithm as proposed in Sec. 17.3. In Sec 17.4. the map-matching has been applied to Endomondo traces collected during a competition in Bologna. Thereafter the routes have been used to calibrate trip time models in Sec. 17.5. Section 17.6 is a critical review of the result obtained and makes suggestions for future improvements.

17.3 The road matching

Each GPS trace is a sequence of GPS points, recorded by the cell phone of a cyclist during her tour. Each point consists of a tuple with longitude, latitude and time information. The task of map matching is to identify the sequence of edges of a given road network, that corresponds most likely to the road links chosen by the cyclist who recorded the trace. The precision of a cell-phone is approximately +/- 20m with one GPS point every 5-10s. This is sufficient to associate an edge on pure proximity bases. However, the severe problems occur in urban areas with a dense street grid (e.g junction distances less than 20m) and a decreased GPS accuracy due to shadowing. This means, proximity alone is no longer an option, instead one

must guess the most likely sequence of joined edges that explain the cyclist's route. The algorithm used for the present work, can be divided into two steps:

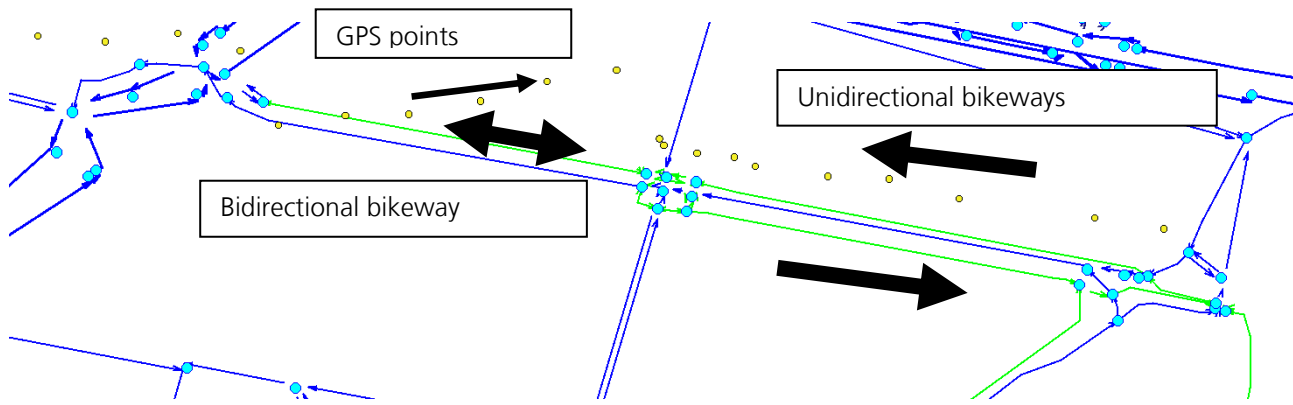
1.) assignment of weights to *all* edges of the network: around each edge, a buffer of ca. 30m is created (polygon approximation). Then the number of GPS points within each edge buffer is detected, which is the computationally most expensive part. If one GPS point j can be found in m_j edge buffers then its weight contribution to each of those buffer is $1/m_j$. Finally the weight w_a of edge a is given by

$$w_a = - \sum_{j \in P_a} \frac{1}{m_j} + cL_a \quad (1)$$

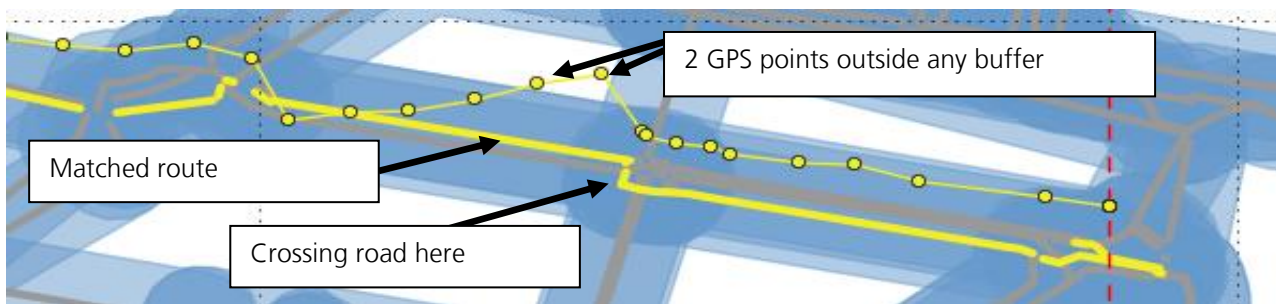
where L_a is the length of edge a and the point set P_a contains all GPS points that are within edge buffer a . The small constant $c=0.01$ for ordinary road edges and $c=0.005$ for bikeways. The idea behind is that the minimum total route costs will be the routes whose buffers contain the maximum of GPS point. In the absents of a GPS points, the route should follow the shortest path. For this reason the length of the edge contributes as a small penalty, biased towards bikeways.

2.) Shortest path routing: all edge buffers that contain the first GPS point constitute the set of source edges; and all edge buffers that contain the last GPS point constitute the set of sink points. Next, the shortest path (or the path that minimizes the edge weights) is calculated for each pair of source/sink edges. The matched route is finally the one that scores the lowest total edge weight of all possible routes.

Figure 17-1 shows an example of a matched route on a road network in a particularly challenging situation with uni-directional and bi-directional bikeways. Note that the road matching algorithm identified correctly the route on the bikeway, despite the significant drift of the GPS points with respect to the road coordinates. In Figure 17-1, two GPS points are even outside the 30m edge buffer.



(a)



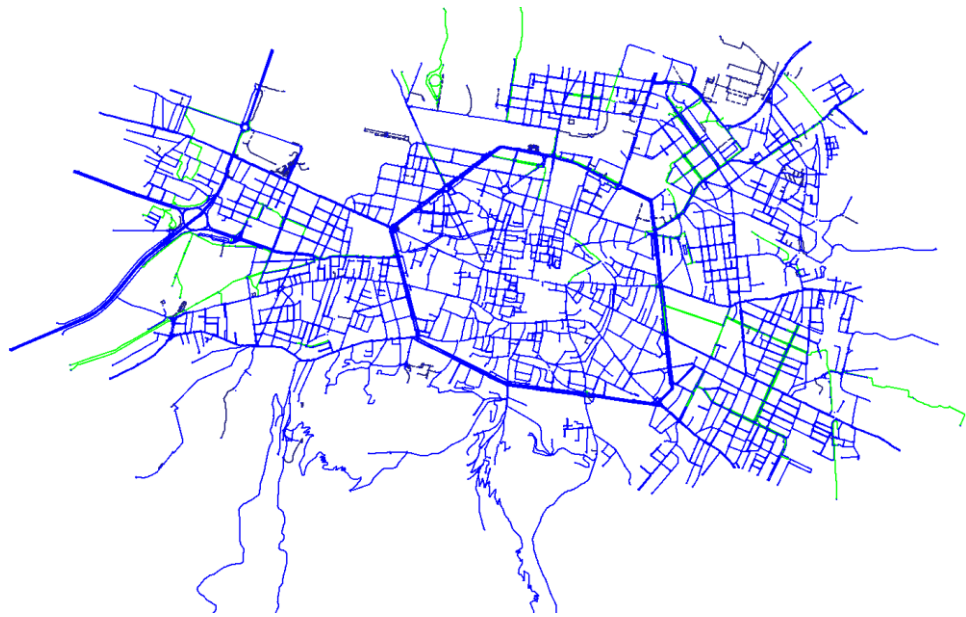
(b)

Figure 17-1: Exemplification of the map matching algorithm. (a) Road network in blue and bikeways in green (on the left part bidirectional, on the right part split to two unidirectional). The GPS points are the yellow dots, representing a bicycle movement from left to right. (b) Same street network, but with 30m wide buffers around edge in blue – buffers are partially overlapping (darker blue). The yellow line with circles represents again the recorded GPS points, whereas the bold yellow line indicates the identified road and bikeway edges. Note that the matched route crosses the road in the middle as to use the bikeway in the correct direction.

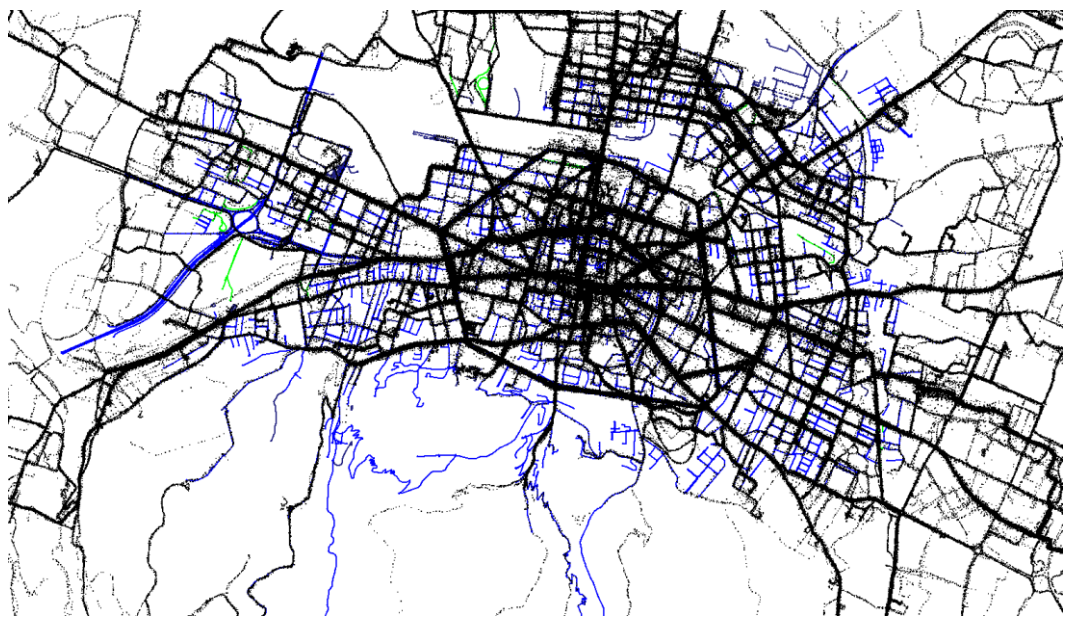
Once the route edges are identified, it is possible to estimate the time interval t_a spent on each edge -- but only for the edges which contain GPS points in their buffer. The adopted strategy is the following: the time instance when a new edge is entered corresponds to the time of the first GPS point outside the previous buffer. The exit time of the edge is the time of the last GPS point in the same edge's buffer.

17.4 Road matching of Endomondo traces in Bologna

GPS traces collected from bike-commuters during several month in the metropolitan area of Bologna have been matched on the SUMO network imported from OSM. The network and the recorded GPS points are shown in Figure 17-2. These are 9478 GPS traces, covering the entire road network.



(a)



(b)

Figure 17-2: (a) Road network (blue) and bikeway network (green) of central Bologna. (b) All GPS points (each is a black dot) of Endomondo mapped on the above road network. Note that the GPS points concentrate on the roads and appear as plain black.

The technical details of the road matching procedure are the following:

- 1.) Conversion of one-way roads into bidirectional roads in OSM format. This has been necessary to track down cyclists who went in the opposite direction of one way streets. In some cases a two way street has been modeled as one way road, due to errors in the Openstreet database.

- 2.) The SUMO network file has been generated by netconvert. One important option has been to merge node within a radius of 15m to simplify many complex junctions.
- 3.) Creation of edge buffer polygons: The SUMOPy tool parsed the network for edges. The edge buffers have been created with function of module shapely.
- 4.) Splitting of GPS traces: The road-matching algorithms can fail if the cyclists make circular tours. For this reason it is necessary to split traces into possibly straight pieces in a pre-processing step. This simple algorithm has been used to avoid circular traces: if the sequence of GPS points contains a point that is half the maximum distance from the initial point, then the trace is split in two traces at the point of the maximum distance of the original trace.
- 5.) Trace by trace matching: The GPS points of each trace have been projected with functions of the python module pyproj. The association of GPS points with edge buffers has been again performed by module functions of shapely. For each trace all, network edge weights have been calculated by applying the procedure as outlined above. The routing has been performed using a modified function from the Dijkstra python module from the SUMO toolbox.

The entire map matching is now available as a python plug-in of SUMOPy. The statistics of the matched results are shown in Table 17-1:

Table 17-1: Basic statistics of the matching process with the Endomondo trace set.

Total unidirectional road network length	660.61 km
Unidirectional bike network length	72.67 km
Number of routes detected	5979
Number of routes containing at least one bikeway edge	3556
Number of identified edges	88135
Number of identified bikeway edges	17301
Total distance travelled on all identified routes	15009.08 km
Length of all edges with identified time intervals	10064.11 km
Average speed on edges with identified time intervals	13.74 km/h
Distance travelled on roads without bikeway	7185.64 km
Average speed on roads without bikeway	13.5 km/h
Distance travelled on roads with bikeway	2878.47 km
Average speed on bikeways	14.35 km/h

Some remarks are in place: the proposed map matching algorithm will always find a route as long as the start and end edges are. Approximately 63% of all the provided traces have been matched to road edges. Most of the unmatched traces were simply outside the considered road network.

If during the trip some of the edges have no GPS points in their buffer, then the Dijkstra algorithms will bridge the missing pieces with the shortest path, no matter how many edges

are not covered by GPS points. However, this can also become a source of misinterpretation. In fact, biggest routing errors occur when edges which are connected in reality, but represented disconnected in the OSM database. If a GPS trace runs over such disconnected edges, then the router tries to find the shortest connected alternative route. This false alternative is often much longer than the real route and contains edges that are not part of the trip. The total length of all routes is 15000km, but only 2/3 of the distance is made of edges with GPS points in the respective buffer (which allows also identifying time intervals). This means 1/3 of the distance has been bridged by shortest path routing. However, it has not been attempted to quantify the occasions when false routing took place due to a disconnected network.

Regarding the statistics, the average velocities are realistic city-velocities for experienced bike commuters. The speed difference between bikeway and normal road are not significant. Approximately 30% of the cycled distance has been on bike ways. On the other hand only just above 10% of all roads are bikeways.

17.5 Link time model calibrations

With link time models, one tries to estimate the time necessary to cross a network link (or edge) based on link attributes such as for example the length L_a or the type of successive intersection. In this work, different models have been calibrated with the aim to investigate the calibration quality, and to find out which link attributes can be included in the model. The various model parameters have been estimated by minimizing the quadratic error between the estimated model time \hat{t}_a and the measured time t_a , which has been determined from the above described road matching process.

As a first approach the simplest model is estimated, which is nothing but the average velocity, hence

$$\hat{t}_a = \alpha L_a \quad (2)$$

where L_a has been the link length and the single parameter $\alpha = 1/\bar{v}$ is the inverse average speed.

The previously identified routes and intervals of the Endomondo competition have been used to calculate this average velocity. It is interesting to have a closer look how good the average velocity estimates the link times. For this purpose, a further data cleaning has been necessary to prevent unrealistic values to distort results. As the GPS points have a limited time- and space resolution, all edges below 20m and time intervals below 20s have been excluded from the calibrations. Furthermore, edges with edge speeds over 40km/h and below 1m/s have also been excluded.

The resulting average speed is $\bar{v} = 15.1 \text{ km/h}$, which is slightly faster than the overall average (see previous section), mainly because short links have been eliminated. Short links are often at intersection with particularly long waiting times. In addition, acceleration/deceleration maneuvers are dominant and lower the average speed. Figure 17-3 shows the estimated times \hat{t}_a versus the measured times t_a for each link. Easily recognizable are some horizontal point-patterns. Each horizontal pattern corresponds to an edge with a certain estimated time

\hat{t}_a (which is constant because proportional to its length). But for the same edge, there is a large range of measured times t_a , presumably produced by different cyclists crossing the specific link. These large horizontal time intervals reveal the problem that absolute travel times depend strongly on the physical condition and driving style of the cyclist.

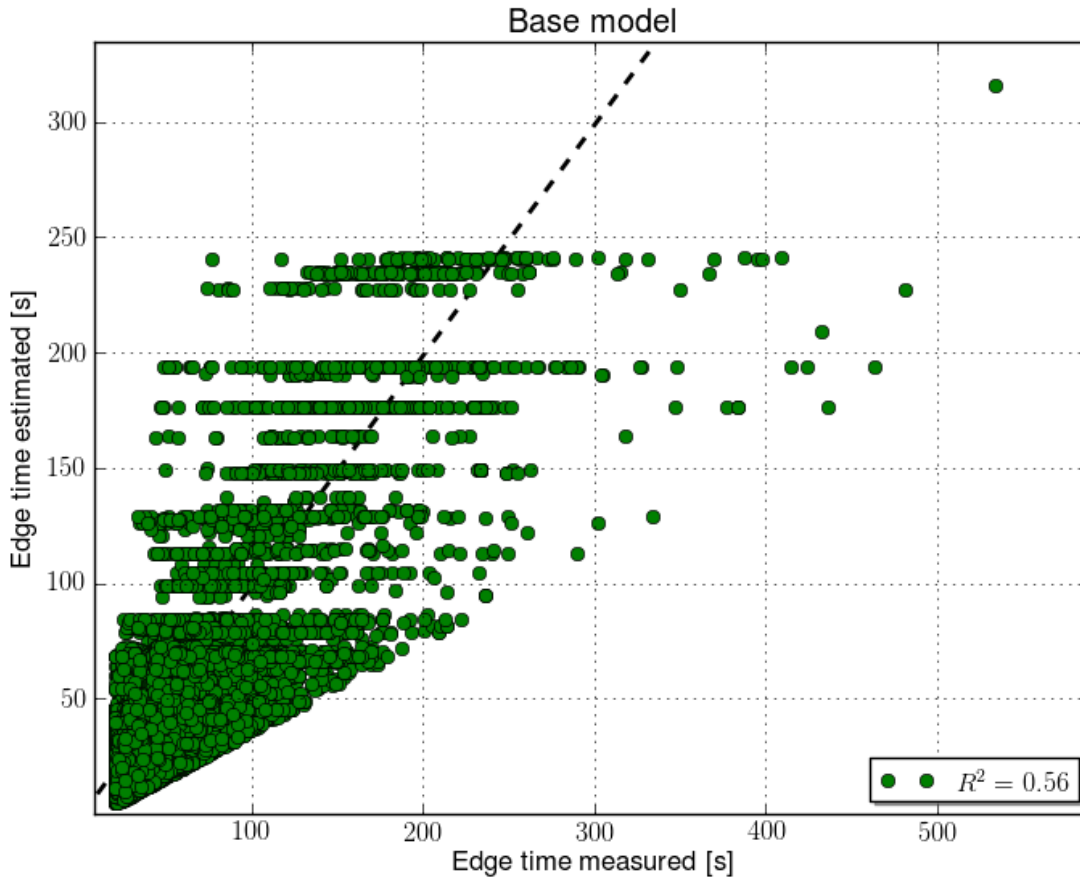


Figure 17-3: Estimated times \hat{t}_a versus measured times t_a using the average velocity, see from Eq. (2).

In order to account for the cyclist's driving style, her free-flow speed has been used to normalize the model as follows:

$$\hat{t}_a = \beta \frac{L_a}{V_F} \quad (3)$$

The free-flow speed V_F has been determined by taking the maximum encountered speed in each GPS trace, assuming that the trace has been recorded by the same person. Surely this is not always the maximum speed of a cyclist, but the best possible indicator retrievable from the available GPS traces.

The calibration resulted in an optimal parameter $\beta = 1.38$. In other words, the average bicycle speed equals 0.72 times the free flow speed. Figure 17-4 shows the results with the speed normalized model. The clear cut upper bound represents the edge-times for which free-flow conditions have been detected. Note that the calibration quality improved, with a coefficient of determination rising from $R^2=0.56$ (basic model) to $R^2=0.66$.

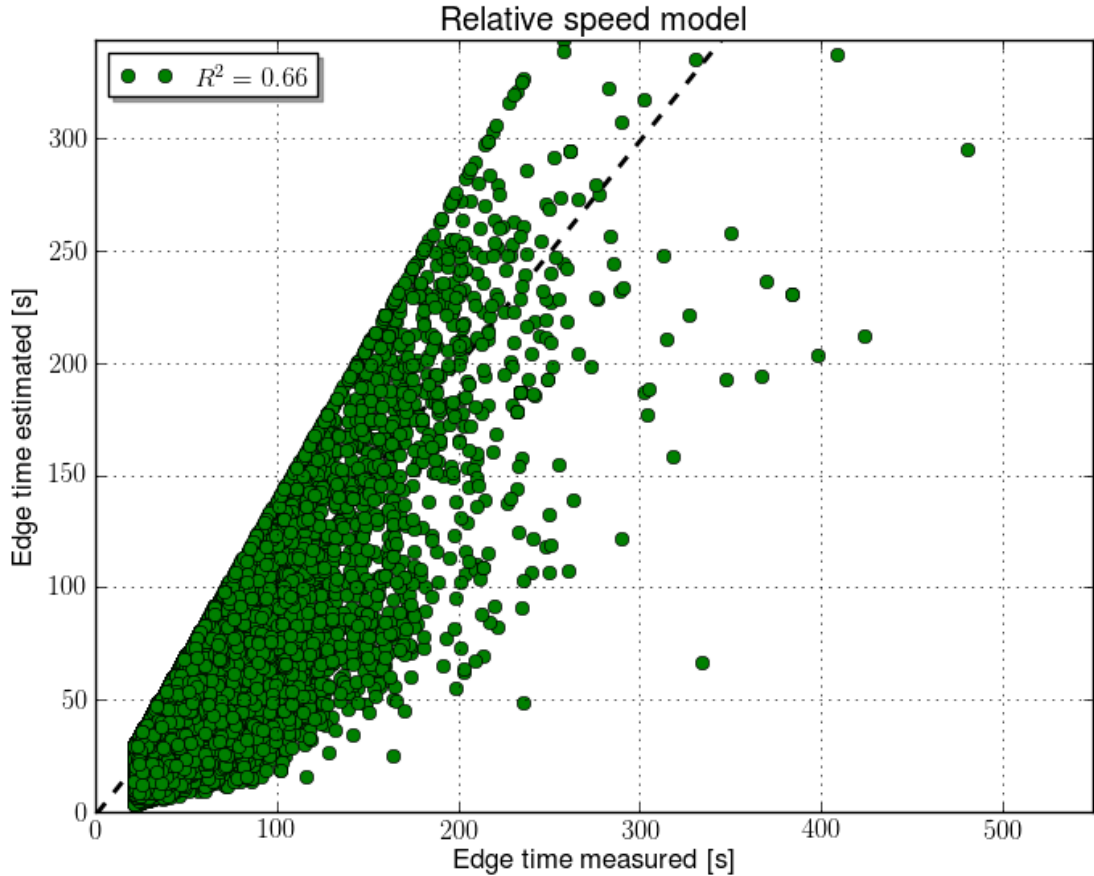


Figure 17-4: Estimated times \hat{t}_a versus measured times t_a using the normalized speed model from Eq. (3).

In an attempt to further improve the time estimation, different more sophisticated models have been calibrated, taking into account the type of intersection that follows the edge. As an example, the following three parameter model has been considered:

$$\hat{t}_a = \beta \frac{L_a}{V_F} + \gamma_1 N_a + \gamma_2 M_a \quad (4)$$

where N_a is the number of conflicting road legs of the successive junction and M_a equals one if the junction is controlled by a traffic light, otherwise zero; either N_a or M_a is zero.

However, for this model the coefficient of determination could not be improved with respect to the previous model from Eq.(2). Also other link attributes, such as lane width or physical obstacles (which have been available for a limited zone) did not further improve the calibration quality.

17.6 Conclusions

A map matching algorithm has been developed, capable of a detailed analyses of the road network usage as it allows combining the road attributes provided by OpenStreetMap with the geo-referenced traces generated from GPS data. Data and all tools such as Sumo and Python are open.

The map matching algorithm has been tested with GPS traces collected during an Endomondo cycle-to-work competition in Bologna and approximately 15000km km of routes could be identified. Furthermore the timing information in the GPS traces have been used to calibrate link time models for cyclists.

Concerning the road-matching process, various error sources have been encountered. The by far biggest error source is due to false routing at network nodes where the edges in the OpenStreetMap network are not connected, even though they are connected in reality. The consequence is that the route matching algorithm must create a route around the disconnected edge. Another problem is that cyclists often drive through pedestrian zones or parks which are either not modeled in Openstreet or have not been imported by netconvert (net gets too heavy when importing all footpaths)- Also in this case the route matching will try to find the shortest route around the un-modeled area. Approximately 30% of the identified routes are routed and not covered by GPS point. But the amount of false routing could not be quantified.

A by-product of the route matching algorithm are the entry- and exit times of route-edges. These times have been used in an attempt to calibrate trip time models for cyclists. Also in the determination of the entry- and exit times are uncertainties due to the special errors of the GPS points and the low temporal resolution. Another source of error is the cyclists herself, not being aware of participating in an experiment, she may have slowed down or stopped during the trip without stopping the GPS recordings. By eliminating points with unrealistic speeds, some of such events may have been eliminated but certainly not all. In addition, the physical capabilities of cyclists differ significantly. In order to compensate the effect of driving styles, the free flow speed has been used to normalize the travel time estimate in the model. But the estimation of the free-flow speed as the maximum speed in a trace is yet another source of errors.

It has been estimated that the average speed of the cyclist participating in the Endomondo competition is 0.72 times her free flow speed. The attempt to calibrate models including more edge attributes failed, probably due to the sum of all previously mentioned error sources.

Therefore, it is not recommended to use the timing information of the Endomondo data for calibrating more sophisticated models. The calibration of path choice models using directly edge attributes is more feasible with the Endomondo data.

Dedicated tests with trace recordings on a well selected set of routes are necessary to obtain less noisy data. There are also GPS devices available with higher precision and better time resolution than normal smart phones. Another prerequisite for good results is to verify, and if necessary edit, the OpenstreetMap network within the study area.

Another concern is the speed of the road matching algorithm for large networks, as all edge weights must be determined for all edges and for each trace. But it should be possible to pre-eliminate insignificant edges. Or significant edges for a trace could be preselected by a k-shortest path algorithm between first and last trace point.

17.7 References

- [1] H. Yuanlin, Y. Gordon Selecting Bicycle Commuting Routes Using GIS. Berkeley Planning Journal, 10(1) (1995).
<http://www.escholarship.org/uc/item/9pf2j1pd>
- [2] Lisa Aultman-Hall, Fred Hall, Brian Baetz (1997), Analysis of Bicycle Commuter Routes Using Geographic Information Systems: Implications for Bicycle Planning Transportation Research Record: Journal of the Transportation Research Board, Vol. 1578, No. -1. pp. 102-110, (1997),.
- [3] I. Sener, N. Eluru and C. Bhat (2009), An analysis of bicycle route choice preferences in Texas, US, Transportation, vol. 36, issue 5, pages 511-539 (2009).
- [4] R. Dalumpines, D. M. Scott. GIS-Based Map-Matching: Development and Demonstration of a Post-Processing Map-Matching Algorithm for Transportation Research. Transportation Research Board (2011).
- [5] Joan G. Hudson, Jennifer C. Duthie, Yatinkumar K., Rathod, Katie A. Larsen, Joel L. Meyer, Using Smartphones to Collect Bicycle Travel Data in Texas, DOT Grant No. DTRT06-G-0044, Project UTCM Project #11-35-69 , Final report. Texas Transportation Institute (2012).
- [6] Marchal, F., J.K. Hackney und K.W. Axhausen, Efficient map-matching of large GPS data sets - Tests on a speed monitoring experiment in Zurich, Transportation Research Record, 1935, 93-100 (2006).
- [7] G Menghini, N Carrasco, N Schüssler, KW Axhausen, Route choice of cyclists in Zurich: GPS-based discrete choice models, Arbeitsberichte Verkehrs- und Raumplanung 544, ETH, Zurich (2009),.
- [8] J. Hood, E. Sall, B. Charlton (2011), A GPS-based bicycle route choice model for San Francisco, California. Transportation Letters: The International Journal of Transportation Research (2011) 3: (63-75).
- [9] <http://www.openstreetmap.org>
- [10] <http://www.endomondo.com/>
- [11] Michael Behrisch, Laura Bieker, Jakob Erdmann and Daniel Krajzewicz. SUMO - Simulation of Urban MObility: An Overview In: SIMUL 2011, The Third International Conference on Advances in System Simulation (2011). <http://sumo-sim.org/>
- [12] D. Krajzewicz, M. Hartinger, G. Hertkorn, P. Mieth, C. Rössel, P. Wagner, J. Ringel. The "Simulation of Urban MObility" package: An open source traffic simulation. In 2003 European Simulation and Modeling Conference (2003)
- [13] <http://www.python.org/>
- [14] <http://www.numpy.org/>
- [15] <http://www.scipy.org/>
- [16] <http://toblerity.org/shapely/>

18 Authors Index

Acosta Gil,A.....	138	Ma,X.....	109
Baines,V.....	10	Mai,S.....	121
Behrisch,M.....	95, 108	McDonagh,P.....	53
Bieker,L.....	27	Michelacci,C.....	27
Blokpoel,R.....	36, 43	Morra,A.....	27
Bragard,Q.....	53	Murphy,L.....	53
Cartolano,F.....	27	Nguyen,T.....	121
Coelho,A.....	174	Padget,J.....	10
Dangel,U.....	53	Pau,P.....	149
Düring,M.....	62	Radusch,I.....	167
Erdmann,J.....	83	Rahmig,C.....	158
Espinosa,Jairo.....	138	Richter,A.....	158
Espinosa,Jorge.....	138	Rieck,D.....	167
Fullerton,M.....	121	Rodrigues,R.....	174
Gonçalves,J.....	174	Rossetti,R.....	174
Gonter,M.....	62	Rupi,F.....	184
Gurczik,G.....	108	Schünemann,B.....	167
Hausberger,S.....	95	Schweizer,J.....	184
Jacob,J.....	174	Thiel,F.....	62
Jin,J.....	109	Ventresque,A.....	53
Kastner,K.-H.....	149	Wagner,P.....	95
Krajzewicz,D.....	27, 95, 108, 121	Weinert,F.....	62
Krumnow,M.....	95		

ISSN 1866-721X