Lecture Notes in Mobility

Michael Behrisch
Melanie Weber *Editors*

Modeling Mobility with Open Data

2nd SUMO Conference 2014 Berlin, Germany, May 15–16, 2014

DLR

Springer

Michael Behrisch · Melanie Weber
Editors

Modeling Mobility with Open Data

2nd SUMO Conference 2014
Berlin, Germany, May 15–16, 2014

Springer

Editors

Michael Behrisch
Institute of Transportation Systems
German Aerospace Center (DLR)
Berlin
Germany

Melanie Weber
Institute of Transportation Systems
German Aerospace Center (DLR)
Berlin
Germany

ISSN 2196-5544 ISSN 2196-5552 (electronic)
Lecture Notes in Mobility
ISBN 978-3-319-15023-9 ISBN 978-3-319-15024-6 (eBook)
DOI 10.1007/978-3-319-15024-6

Library of Congress Control Number: 2014960040

Springer Cham Heidelberg New York Dordrecht London

Preface

The advance of new data sources for traffic networks, especially freely available mapping sources such as Open Street Map, provides major opportunities to the scientific and the applied traffic modelling community. Together with readily available tools, such as the open source package Simulation of Urban Mobility (SUMO), building a working prototype of a simulation scenario in virtually no time becomes feasible. Adding demand data, which is usually not openly available yet, enables the detailed estimation of the effects of engineering measurements as well as emerging new technologies through the means of individual (microscopic) traffic simulation. This simulation of every single actor allows the integration of behavioral data which can interface with the existing models to gather new insights into the social dynamics of traffic as well.

This volume contains the proceedings of the second SUMO User Conference (SUMO2014), which was held from 15 to 16 May 2014 in Berlin-Adlershof, Germany. SUMO is a well-established microscopic traffic simulation suite which has been available since 2001 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 the incorporation of mapping data and traffic signals 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 workflow improvements to govern microscopic traffic simulation results.

Several articles give outlines of detailed aspects of network preparation and demand modeling when setting up a simulation with SUMO as well as an overview of the application of the tool in large-scale scenarios or for emission modeling and for the evaluation of the results. Further contributions include the simulation of emergency vehicles as well as the extension for the implementation of new behavioral models or remote control of the simulation using various programming environments. The conference series' aim is bringing together the large international user community and exchanging experience in using SUMO, while presenting results or solutions obtained using the software. This collection should inspire you to try your next project with the SUMO suite as well or to find new applications in your existing environment.

Berlin, November 2014

Michael Behrisch
Melanie Weber

SUMO2014 Organization

SUMO2014 was organized by the Institute of Transportations Systems, German Aerospace Center, Berlin.

International Scientific Committee

Michael Behrisch (German Aerospace Center, Germany)
Laura Bieker (German Aerospace Center, Germany)
Robbin Blokpoel (Imtech Traffic & Infra, Netherlands)
David Eckhoff (University of Erlangen, Germany)
Jakob Erdmann (German Aerospace Center, Germany)
Jérôme Härri (Institute EURECOM, France)
Daniel Krajzewicz (German Aerospace Center, Germany)
Mario Krumnow (University of Technology Dresden, Germany)
Andreas Schadschneider (University of Cologne, Germany)
Christoph Sommer (University of Innsbruck, Austria)
Peter Wagner (German Aerospace Center, Germany)

Organization Committee

Michael Behrisch (German Aerospace Center, Germany)
Melanie Weber (German Aerospace Center, Germany)

Contents

Part I
Data Acquisition and Integration

DFROUTER—Estimation of Vehicle Routes from Cross-Section Measurements

TeRon V. Nguyen, Daniel Krajzewicz, Matthew Fullerton and Eric Nicolay

Abstract This contribution evaluates and improves the open-source “DFROUTER” tool that is contained in the SUMO traffic simulation suite. DFROUTER uses vehicle counts (e.g. from inductive loops) to calculate routes of vehicles through road networks. This approach is designed for highway corridors that are covered with measurement facilities at all entry and exit points. The study analyzes DFROUTER’s current functionality and compares it with other approaches that have a similar purpose. Tests performed using different networks and sensor coverage amounts are presented. Additionally, an extension to the software is presented that completes missing flows, increasing the correctness of the tool’s results.

1 Introduction

Transport planners and traffic engineers worldwide challenge with the increase in traffic amount. A wide range of measures is implemented to tackle this problem, ranging from large-scale traffic management strategies to in-vehicle Intelligent Transport Systems (ITS). Accordingly, the deployed methods range from traffic access regulations, such as calming areas or speed limits over route guidance, to in-vehicle solutions that advice a speed to use to pass the next traffic light at green or that help in changing lanes. All these solutions target the improvement of traffic in means of safety or efficiency and a more optimal use of natural resources.

One first step to take within the development of such solutions is to model the situation on roads. Besides the representation of the road networks, a proper

T.V. Nguyen · M. Fullerton
Institute of Transportation, Technische Universität München, Arcisstraße 21, 80333 Munich, Germany

D. Krajzewicz (✉) · E. Nicolay
German Aerospace Center, Institute of Transportation Systems, Rutherfordstraße 2, 12489 Berlin, Germany
e-mail: daniel.krajzewicz@dlr.de

M. Behrisch and M. Weber (eds.), *Modeling Mobility with Open Data*, Lecture Notes in Mobility, DOI 10.1007/978-3-319-15024-6_1

representation of the traffic demand is one of the major inputs for transportation system operation, design, analysis, and planning [1]. Origin-destination (O-D) matrices are one of such representations. They contain information about the spatial and temporal distribution of activities in different traffic zones in an area. Various methods for generating origin-destination matrices have been proposed, like household surveys, roadside interviews, license plate recognition and returnable-post card interviews [2]. However they are all expensive and obtaining the data is cumbersome [3].

Meanwhile, many data about vehicles flows (numbers of vehicles) is being collected for other traffic management purposes. Often, inductive loop detectors are used that are installed under the road surface. Inductive loops usually collect information such as vehicle type and speed, traffic volume, and detector occupancy. Being continuously retrieved for other purposes, such data is usually available at a lower price than the employment of previously mentioned methods. But while induction loops are a good source of information about the number of vehicles on a street, they fail to provide information about the vehicles' further routes. Hence, methods for estimating traffic flows between origin/destination pairs have been developed. Usually, they efficiently combine traffic count based data with other available information [4].

Within this report, the application "DFROUTER" is analyzed. DFROUTER uses detector values to calculate routes for simulated vehicles through a given motorway/ corridor simulation network. It is included in the microscopic road traffic simulation package SUMO [5]. Besides routes, this tool also generates the according demand for the traffic simulation, consisting of single (microscopic) vehicle insertion definitions the traffic simulation can read. This paper provides a detailed 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 of the tool where reproduction of the flows' probabilities is the decisive indicator for this evaluation.

This report starts with an introduction into the problem of O-D matrix estimation. Then, DFROUTER and the algorithms it uses are discussed. Afterwards, evaluations of the DFROUTER and comparisons to other approaches are given. Then, an extension to the DFROUTER is presented that allows using it on highway networks that are not completely covered with detectors. This report ends with a conclusion.

2 Theoretical Background

O-D matrices describe traffic demand by dividing a given area into so-called "traffic assignment zones" (TAZs). For every TAZ at which traffic participants start (the demand origins), the number of participants that approach a destination TAZ is given. O-D matrix estimation methods based on traffic counts have been developed over the last 30 years. There are two major types of O-D matrix estimation: the static method assumes O-D flows are constant over time for determining an average O-D demand for long-time transport planning and design purposes; whereas the dynamic method considers O-D flows with time variation for short-term strategic traffic control and management [6]. One could state that dynamic methods are an extension of static methods considering the time varying dimension. Furthermore, due to congestion effects, the O-D matrix estimation can use proportional assignment (uncongested) or equilibrium assignment (congested), resulting in four basic cases of O-D estimation [1].

Contrary to urban road networks where more than one route between one origin-destination pair exists, an O-D pair in a highway corridor consisting of an on- and an off-ramp has only one possible route. This less complicated characteristic facilitates the estimation of vehicle routes and traffic demand based on detector data. Specifically for highways, the problem of determining an O-D matrix from traffic counts can be formulated as follows:

$$\sum_{i} b_{ij} O_i = D_j \tag{1}$$

$$\sum_{j} b_{ij} = 1 \tag{2}$$

where

b_{ij}	proportion of trip from i to j;
O_i	on-ramp counts (origin flows);
D_j	off-ramp counts (destination flows).

Considering an example highway section that illustrates the O-D matrix estimation problem, the one shown in Fig. 1 could be used.

As shown in Table 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.

Many O-D matrix estimation techniques exist, where Information Minimization (IM) and Entropy Maximization (EM) [2], Maximum Likelihood (ML) [4], Generalized Least Squared (GLS) [3], or Bayesian Inference approach [7] could be named as the most popular static ones. Regarding dynamic methods, one can find

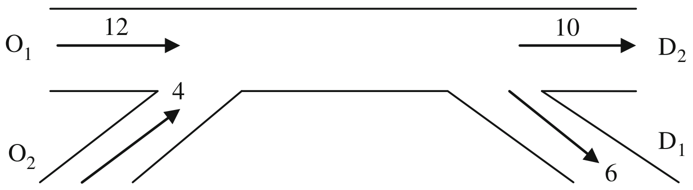


Fig. 1 Sample highway segment

Table 1 Some examples for O-D matrices that correctly represent the flows of the example from Fig. 1

	D_1	D_2	Sum		D_1	D_2	Sum		D_1	D_2	Sum
O_1	8	4	12		10	2	12		6	6	12
O_2	2	2	4		0	4	4		4	0	4
Sum	10	6	16		10	6	16		10	6	16

approaches such as Cross-correlation matrices, Constrained optimization, Recursive estimation, Kalman filtering [8], Recursive least square [9], Artificial Neural Networks [10], and Combined estimators [11].

3 DFROUTER

3.1 Development Context

DFROUTER builds upon experience gained during the set-up of a large-scale traffic observation and prediction project which results were applied during the Pope's visit in Germany in the year 2005 [12]. About 1 Mio persons were expected to participate in this event that took place on a green field near to the city of Cologne. The project's scope was to support the police and the local traffic management with on-line information about the state on the roads. The deployed system consisted of an airborne camera-based traffic surveillance system mounted under a zeppelin that sent information about recognized vehicles to a traffic management center. Together with measurements from inductive loops, this data was used to calibrate a mesoscopic traffic simulation. This simulation had the task to extrapolate the traffic counted at measurement points over the road network as well as to predict the traffic situation half an hour into the future. The so obtained states of the road network were visualized at the traffic management center of the city of Cologne (Fig. 2).

To achieve the goal of predicting traffic, an initial demand was needed, that could be calibrated using on-line measurements after deployment. The available input data included two commercial O-D matrices. The first one described a usual

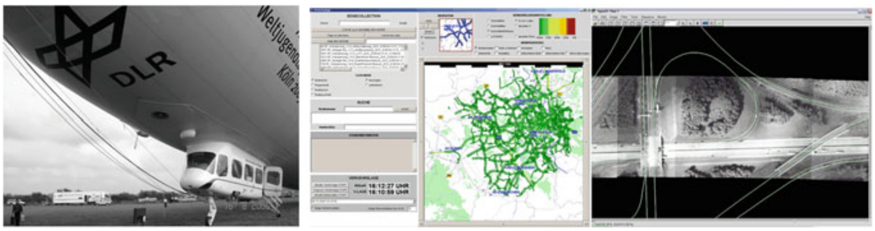


Fig. 2 *Left* the zeppelin that carried the airborne traffic surveillance system; *Right* the visualization

working day. The second one was a prediction of the road traffic during the Pope's visit, but covered only one half of the area that was defined to be simulated. Both matrices were static, containing the demand of a complete day. Additionally available was a microscopic demand from the TAPAS project [13], which was based on a synthetic population. On-line data were supported by the highway administration of North Rhine-Westphalia and by the traffic management center of the City of Cologne.

Besides the amounts of passing vehicles, on-line calibration requires information about the routes which newly inserted vehicles shall use for continuing their journey. For the reasons outlined in the following, it was decided to use the inductive loop measurements not only for adapting the simulated traffic flow volumes to the measured vehicle numbers, but also as the ground truth for computing routes across the highway part of the simulated area.

The first reason to name is the uncertainty whether the available demand descriptions were applicable. To obtain routes running over the measurement points, a traffic assignment [14] would have to be performed, first. But this process is very sensitive to both, the road network representation as well as to the used demand. As both were not completely revalidated at this decision step, it was assumed that the resulting routes distribution would be erroneous. In addition, the given matrices resembled different traffic conditions (usual day vs. visitors' traffic) and had different granularities (microscopic from a synthetic population demand vs. static O-D matrices). Attempts to combine these matrices were dismissed.

Moreover, first system runs have shown that using the given O-D matrices as source of routes distributions yields in a too high memory consumption: at each measurement point, a distribution of routes to use has to be given. The simulated network was very large and routes are defined as a list of all road network edges the simulated vehicle shall pass within the used simulation SUMO. To decrease both, their number as well as the sizes of the routes stored for each measurement point, the demand was split into highway and non-highway-parts. Every vehicle that entered the highway was given a new route. When leaving the highway, the vehicle obtained a new route again. This kept the routes relatively small. This was only possible, because the area around the city of Cologne is well-covered with sensors, including all highway entries and exits.

This kind of modelling breaks all previously existent O-D relationships of single vehicles as their routes are constructed from different route distributions. This was acceptable, because the project's target was to resemble the flows on the simulated road network, not the mobility of single participants. What was realized as a tool for the Pope's visit was transformed into DFROUTER in subsequent projects.

3.2 Algorithm

DFROUTER performs several steps to obtain routes and the vehicle insertion definitions, being mainly:

1. Reading the road network to route on, the detector positions, and their measurements,
2. Detector Classification,
3. Routes computation,
4. Flows generation,
5. Writing the results.

The major algorithms and overall features of the DFROUTER are explained more detailed in the following subsections.

3.2.1 Detector Classification

The needed functionality included a classification of the detectors into the following types:

- "pure sources": starting points of routes—vehicles that enter the highway get a new route assigned;
- "in between": the simulated vehicle numbers are adapted to the measurements at these positions; only vehicles that are added obtain a new route;
- "pure sinks": ending point of routes—vehicles get a new route assigned that is based on given data from the available demand descriptions.

A detector is classified as a "pure source" if the following constraints are valid:

- there is no other detector on the same street in front of it,
- there is no detector on any foregoing street.

Analogous, a detector is classified as a "pure sink" if

- there is no other detector on the same street behind,
- there is no detector on any following street.

3.2.2 Routes Computation

The main steps of the algorithm that computes the route usage probabilities are as following. Please note that usually measurements are given per-lane and need to be summarized for each cross section.

- Step 1: Determine downstream detectors (taking into account downstream road junctions) for all source and in-between detectors.
- Step 2: Calculate the proportion of flow for each junction using detector data; junction directions not equipped with detectors get a probability of 1.0 as default (what is a fallback to work with real-life networks).
- Step 3: Calculate destination distributions for all source detectors by multiplying all flow probabilities on all edges constructing that route.

Simply spoken, the algorithm computes routes by taking the destination proportion as route probability at every junction. If all sink detectors are supplied, the flows should be replicated correctly. But a single solution to the O-D guessing problem can be only obtained if there is only one origin and the network is fully covered by detectors. This is rather not the case for real-world networks.

This simple algorithm fails in the case of missing detectors, especially detector data on split edges (in-between or sink detectors) as it is not able to guess the missing data and thereby cannot compute the probability to choose one of the subsequent roads. As a default, the probability to use the non-observed road is set to 100 %, overestimating it. This default is rather arbitrary chosen—any other used value would be incorrect as well.

3.2.3 Output Generation

Vehicles are inserted at source detector positions. For every detector recognized as being a "pure source", DFROUTER generates the routes distributions and a list of vehicles that shall be inserted into the simulation network at this position. A route distribution is defined as a route and a probability to choose it, where a route is defined as a list of edges to pass.

As during computation, a routes distribution was obtained for every cross section, DFROUTER can write inputs to in-simulation flow "calibrators" for each in-between detector. These calibrators may be loaded into a simulation scenario. There, they adapt the number of passing vehicles to the read values by adding/removing vehicles into/from the simulation.

A further output consists of "variable speed sign" definitions for sink detectors, as well readable by the simulation. These simulation instances read a time line of speeds and apply them to a defined lane. This feature is mainly used in jam formation analysis to model boundary conditions properly. The speed is read from the detector measurements while reading the flow amounts. Besides "variable speed signs", so-called "rerouters" may be additionally written for sink detectors. Equipped with—externally generated—route distributions, these in-simulation instances assign a new route to passing vehicles. Within the Pope's visit, they were used to assign new routes to vehicles that leave the highway.

4 Evaluations

In the following, different evaluations of DFROUTER are presented. At first, synthetic scenarios are given to DFROUTER to determine how well it can reproduce an originally completely known flow. In a second step, a single scenario is used to compare DFROUTER to some selected O-D estimation algorithms.

4.1 Replication of Synthetic Scenarios

In order to analyze the algorithm, several abstract highway networks and demands, ranging from simple to complex, were applied. The four factors to be considered are the network type, the number of detectors, vehicle flows, and routes. These elements were altered to test the generated results. It was expected that the algorithm works well in simple cases but may fail when being confronted to more complex ones.

Beginning with two on- and off-ramps, the initial network was incrementally extended to more complicated scenarios with extra ramps, lanes, entrances, and exits (origins and destinations). Basically, there is one main highway line connected to several on- and off-ramps equipped with detectors.

The evaluation is performed by generating virtual detector data using the simulation SUMO. The resulting measurements from simulated inductive loops are then given to DFROUTER for generating routes and demand definitions. Routes and vehicle flows are the main indicators for this evaluation. In general, the flows/routes/detectors generated by DFROUTER should be identical to the initial input for SUMO simulation. The general work flow of this analysis is shown in Fig. 3.

The used synthetic scenarios are shown in the following figures (Figs. 4, 5 and 6).

A comparison of the output generated by DFROUTER against the initial input for the three cases shows that:

- The algorithm works well if the network is fully covered with detectors and generates routes comprising all O-D 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.

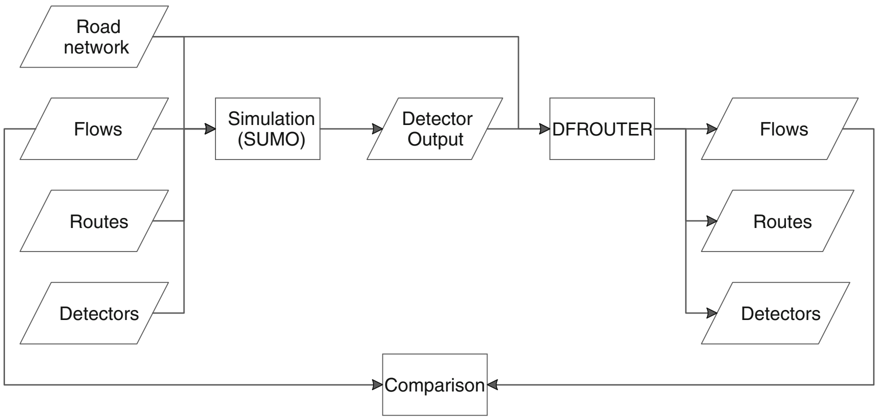


Fig. 3 The work flow of the evaluation process

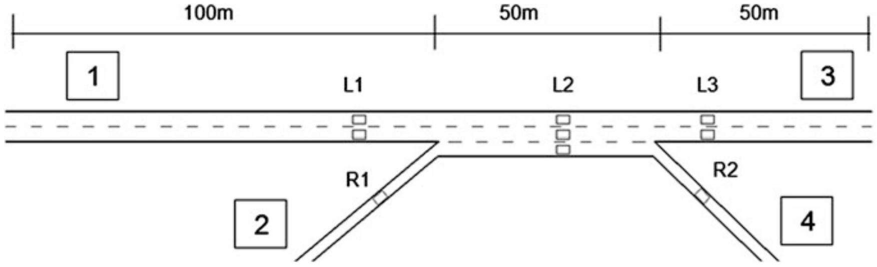


Fig. 4 CASE 1—2 origins, 2 destinations

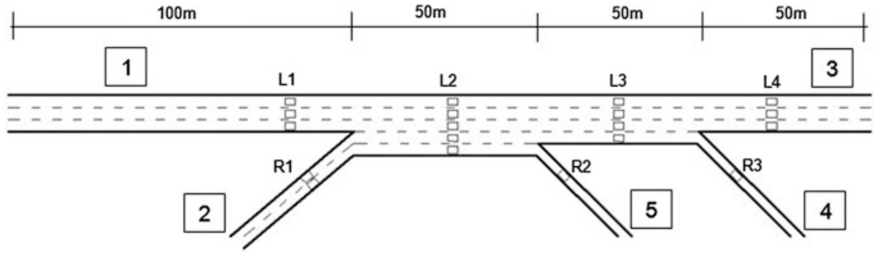


Fig. 5 CASE 2—2 origins, 3 destinations

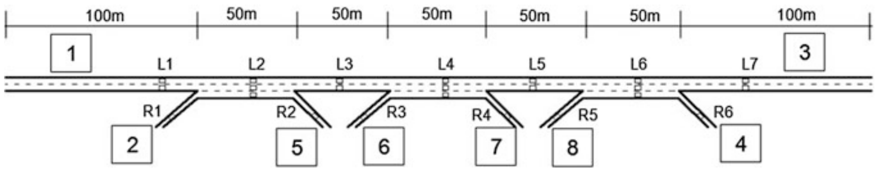


Fig. 6 CASE 3—3 origins, 3 destinations

- Missing in-between detectors (in three cases) do 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.
- This simple algorithm does 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.

4.2 Comparison with Other Approaches

In the following, some O-D matrix estimation approaches are described and compared with DFROUTER's algorithm. Some of these approached do not necessarily place restrictions to the network type. For a fair comparison, the same scenario is given to the compared algorithms. The main used performance indicator is the route probability.

DFROUTER generates route/demand data based merely on proportions of flows on split edges. The destination distribution is an average result of different calculations performed on time slices with a duration of 60 s, as default. Congestion effects and the travel time between the origin and the destination are not considered. This method is most likely to work for the static O-D 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 min). 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 no optimization function (e.g. minimization differences between estimated and observed link flows).

The used scenario, summarized in Fig. 7, comprises detector data as shown in Table 2 and the highway network as used for the initially described CASE 2 (Fig. 5).

The DFROUTER algorithm calculates flow probabilities for each of the split edges by examining the outflows of each junction considering off-ramp counts and

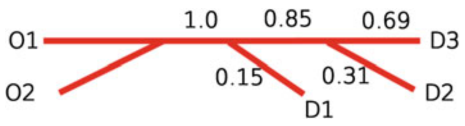


Fig. 7 Comparison test case configuration

Table 2 Network settings and detector data used for comparison

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

Table 3 The DFROUTER O-D matrix

O\D	D_1	D_2	D_3
O_1	= 1.0 * 0.15	= 1.0 * 0.85 * 0.31	= 1.0 * 0.85 * 0.69
	= 0.15	= 0.26	= 0.59
O_2	= 1.0 * 0.15	= 1.0 * 0.85 * 0.31	= 1.0 * 0.85 * 0.69
	= 0.15	= 0.26	= 0.59

Table 4 The equally split O-D matrix

	D_1	D_2	D_3
O_1	1/3	1/3	1/3
O_2	1/3	1/3	1/3

mainline counts, e.g. 70/460, 390/460, 120/390, 270/390 (equal to 0.15, 0.85, 0.31 and 0.69 respectively).

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

This O-D 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 O-D matrix, the proportional O-D matrix, iterative methods, the gravity model, and turning percentages.

4.2.1 The Equally Split O-D 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 (Fig. 7) with three destinations, the method concludes that D_1, D_2 and D_3 are equally likely for trips from origin O_1 and O_2, so the proportion will be 1/3 (33.3 %) (Table 4).

4.2.2 Proportional O-D Matrix

This is one of the most common and oldest methods to estimate an O-D matrix [15]. 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 yields in a higher flow proportion. The origin flow will hence be distributed according to destination flows.

Considering the test case (Fig. 7) where destination flows collected at D_1, D_2, D_3 are 70, 120, 270 vehicles respectively, the proportional O-D matrix can be computed manually as follows, which is identical to DFROUTER's calculation (Table 5).

Table 5 The proportional O-D matrix

	D_1	D_2	D_3
O_1	= 70/(270 + 120 + 70)	= 120/(270 + 120 + 70)	= 270/(270 + 120 + 70)
	= 0.15	= 0.26	= 0.59
O_2	= 70/(270 + 120 + 70)	= 120/(270 + 120 + 70)	= 270/(270 + 120 + 70)
	= 0.15	= 0.26	= 0.59

4.2.3 Iterative Method

This is considered as a hybrid proportional assignment technique that balances both inflows and outflows [15], adopted from Wills and May (1981) based on an iterative fitting algorithm. The algorithm computes each O-D cell iteratively until a convergence is reached. The algorithm steps are given below.

Step 0

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

Step 1

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

where O'_i is the observed volume at point *i* adjusted for all known demands from i.
Step 2

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

where D'_j is the observed exit volume at point *j* adjusted for all known trips that end at j.
Step 3

$$\text{if } T_{ij}^{(2k+2)} - T_{ij}^{(2k)} < \delta \quad \text{for all i, j then STOP}$$
$$\text{else set } k = k + 1 \text{ and go to Step 1}$$

Using this algorithm to compute the O-D matrix for the test case (Fig. 7) yields in the results shown in Table 6. The algorithm produced a converged output after two iterations.

The final iterative O-D matrix estimation, however, contains the same values as that of the proportional O-D matrix estimation. This may be because the method computes O-D elements iteratively but does not consider any constraint such as

Table 6 The iterative O-D matrix estimate

	D_1	D_2	D_3
O_1	0.15	0.26	0.59
O_2	0.15	0.26	0.59

distance or travel time as inputs to a deterrence function. The following gravity model will take these parameters into account.

4.2.4 The Gravity Model

The gravity model is one of the oldest trip distribution methods and is widely used in macroscopic modelling. An extension proposed by Nancy Nihan [16] uses the impedance function to estimate the trip proportion between ramps. 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:

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

where

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

The cell entries in the O-D matrix are defined as:

$$T_{ij} = \frac{b_j F_{ij}}{\sum_j b_j F_{ij}} O_i \tag{6}$$

where

$\mathrm{T_{ij}}$	trip interchange between pair (i, j);
$\mathrm{b_j}$	balance factor from iterations;
$\mathrm{O_i}$	production at *i*;
$\mathrm{D_j}$	attraction at *j*.

In addition, the following constraint has to be fulfilled:

$$\sum_i T_{ij} = D_j \tag{7}$$

In the implementation of the algorithm, the balancing factor was ignored for the first iteration. The average trip length from the geometry is:

$$(100 * 280 + 50 * 460 + 50 * 390 + 50 * 270)/(280 + 180) = 183.$$

Table 7 The Gravity model O-D matrix

	D_1	D_2	D_3
O_1	0.4312	0.3371	0.2317
O_2	0.2222	0.3909	0.3869

Table 8 Turning percentage O-D matrix

	D_1	D_2	D_3
O_1	0.15	= 0.31*(1 − 0.15)	= 1 − 0.15 − 0.26
		= 0.26	= 0.59
O_2	0.15	= 0.31*(1 − 0.15)	= 1 − 0.15 − 0.26
		= 0.26	= 0.59

Therefore parameter $\beta = 3/183 = 0.016$. Using these parameters and the distance matrix, the O-D matrix results are calculated accordingly (Table 7).

4.2.5 The Turning Percentage

This is the most intuitive method of estimating an O-D matrix for a freeway section. Similar to the equally split and proportional O-D estimate methods, it assumes that turning percentages at any given off-ramp are independent of the trip origin [15]. Therefore the O-D matrix is derived by tracking the turning percentages in each section. Using the test case (Fig. 7), there are four 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 O-D matrix is shown in Table 8.

4.2.6 Discussion

The equally split O-D matrix method did not generate a plausible result. Due to the missing value for the balance factor b_j, the gravity model has not been examined thoroughly and therefore produced rather incomplete output in the first calculation iteration. Similar O-D matrices were achieved from various approaches: DFROUTER, the proportional O-D matrix, the used iterative method, and the 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. In contrary to the iterative method, it does not take distance, time, or any deterrence parameter into account, but performs its computations based on the number of origin and destination counts only. The results therefore are proportional to these counts.

Furthermore, DFROUTER and the proportional O-D matrix also have similar working mechanisms. Considering a tree graph as follows including one origin and seven destinations where a, b, c, d, e, f are the respective detector data on edges (Fig. 8).

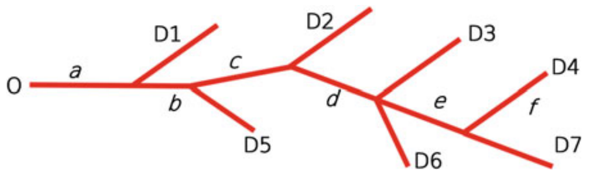


Fig. 8 A tree graph

Then the flow probability at each destination, e.g. D_4, is computed as:

$$\text{DFROUTER:}\ \ pro = \frac{b}{a} \times \frac{c}{b} \times \frac{d}{c} \times \frac{e}{d} \times \frac{f}{e} = \frac{f}{a}$$

$$\text{Proportional O-D matrix:}\ \ pro = \frac{f}{\sum_{j=1}^{n} D_j} = \frac{f}{\sum_{j=1}^{n} O_i} = \frac{f}{a}$$

From above, it could be said that for the case of one origin, DFROUTER and the proportional O-D matrix use a basically same approach. The proportional O-D matrix works more simple than DFROUTER as it does not take into account in-between detectors or split edges; only the data at sink detectors are used for calculation. A different approach named SYNOD has been developed to synthesize the required O-D matrix based on proportional O-D matrix approach. This simple proportionality scheme is on the other hand 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 described in [16].

Due to the drawbacks of these methods, they are often used to generate a starting solution (seed or target, a priori matrix) for the O-D estimation problem to solve the minimization function of difference between estimated and observed link flows or O-D matrix [15].

5 Extension for Completing Missing Measurements

From the analysis of DFROUTER and other, already known issues, several improvements to the algorithm 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.
- Regarding the travel times when computing route usage probabilities; currently the probabilities are only computed regarding the same time slice of detector measurements.
- Computing route probabilities individually for passenger and heavy duty vehicles; albeit both types are usually explicitly counted and given in according

measurements, DFROUTER computes only route probabilities for the overall vehicle amount.
- 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 two (or several) split edges. By doing this, DFROUTER could perform well even in case of not all "pure sinks" being covered with detectors. The overestimation problem of the current DFROUTER that assigns probability = 1.0 as default for missing detector data could be eliminated. The following subsections describe this extension. At first, the algorithm to compute the missing data is given, followed by an evaluation of its function in an abstract road network. This section closes with a report on the application of the improved DFROUTER for a complex, real-world network.

5.1 Calculating Missing Data

The initial algorithm takes only those split edges that have a detector on them into account and omits those without a detector. This problem could be solved by the algorithm proposed in the following:

- Step 1: Calculate the flow value on each edge of the highway network using backward or forward recursion.
- Step 2: Determine split edges after a junction for all routes starting from source or between detectors to sink detectors.
- Step 3: Calculating flow proportion of split edges based on computed flow so that each split edge contains a different probability.
- Step 4: Calculate destination distribution by multiplying all flow probabilities on all edges constructing that route for routes starting from source detectors only.

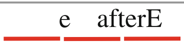
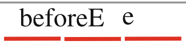
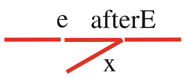
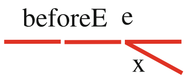
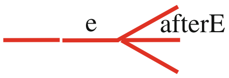
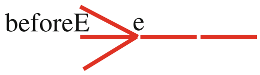
Of all steps above, the first is the most challenging one as there are many dependencies to consider. Consider an edge e for which detector values are missing. Forward recursion will be performed when there is no detector before *e* and backward recursion works in the opposite way.

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

5.2 Application in an Abstract Network

A hypothetical highway network was developed to test the improved DFROUTER. It was designed to contain all cases listed in Table 9. There are only seven detectors at the location of L1, L9, R1, R2, R5, R66, R7, the remaining on- and off-ramps are

Table 9 Cases to consider in the recursion algorithm

	Recursion forward	Recursion backward
1	 e = afterE	 e = beforeE
2	 e = afterE - x	 e = beforeE - x
3	 e = ∑afterE	 e = ∑beforeE

missing, including some in-between and sink detectors. The input probabilities will be compared with DFROUTER's output (Fig. 9).

In order to calculate flow at a certain edge, the recursion function will be used, whatever it is forward or backward. For instance, R3 will be computed as follows (Fig. 10):

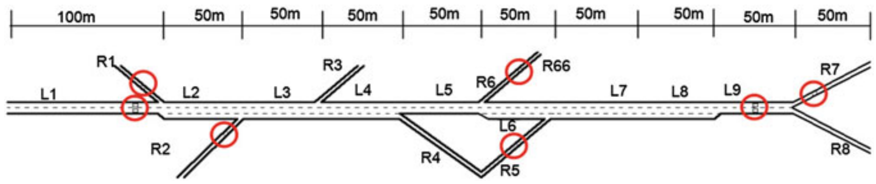


Fig. 9 Abstract network with missing detectors

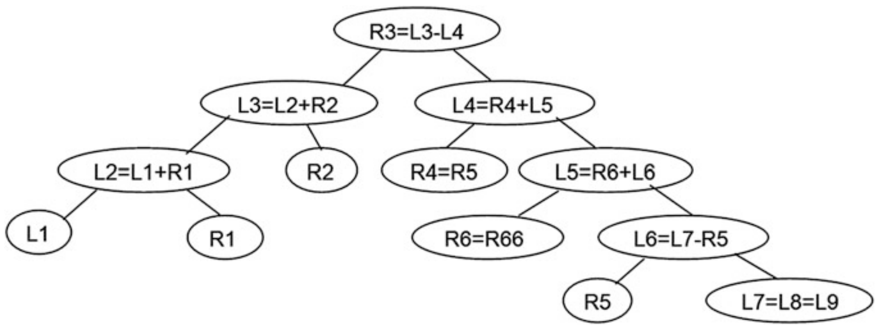


Fig. 10 Example of calculating R3

Table 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	1,100	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

Calculated results and their comparisons are shown below regarding both probabilities at destinations 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 are more accurate compared to the original DFROUTER (Table 10).

5.3 Application in a Larger Network

To evaluate the algorithm on a more complex, real scenario, a larger network containing three main interchanges in Nuremberg was converted from OpenStreetMap data. Each interchange is equipped with different numbers of detectors (see Fig. 11).

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

The flow probabilities produced by the improved DFROUTER are different from those computed by the original DFROUTER as shown in the Table 11. As expected, the results from the improved DFROUTER are more accurate.

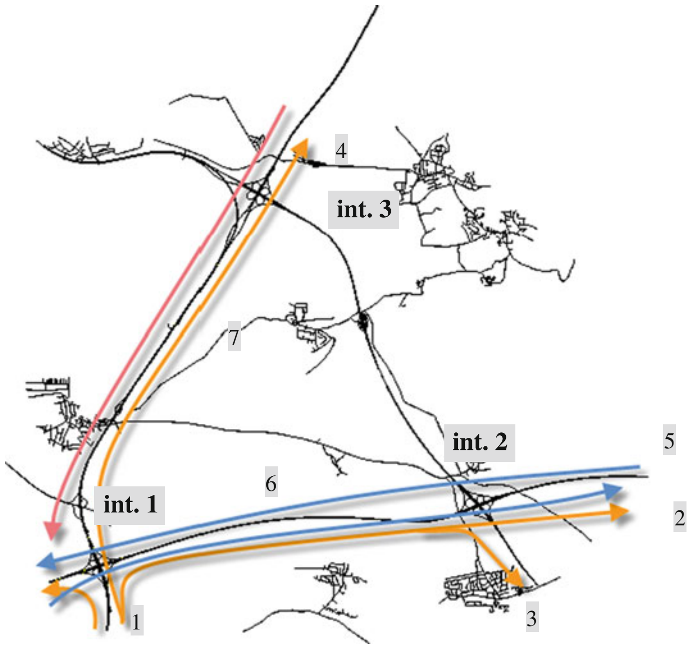


Fig. 11 Nuremberg highway network

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

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

6 Conclusion

The study has been conducted to analyze SUMO's DFROUTER tool. It sought to answer the following 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?

DFROUTER's results for several typical highway corridors were examined, first. Additionally, the algorithm has been compared with some O-D matrix estimation approaches based on the same abstract highway corridor. The literature review has indicated two main groups of O-D estimation: static and dynamic, which have been developed over the last 30 years. DFROUTER's approach of dividing incoming flow proportionally to off-ramp counts makes it simple and fast in calculating respective flows. In parallel, it computes results that are similar to those obtained from other algorithms.

An algorithm improvement has been proposed and applied successfully to a large highway network. It produced reliable results using recursion to guess missing data, assuring that each edge after a junction will contain a certain traffic count and relative probability. The method of multiplying individual probabilities is left unchanged. The sometimes present problem of missing detectors at destinations is thereby partially solved. The extension will be included in SUMO's standard release. The improved algorithm, however, is applicable to highway corridors (one way street) only. Future research and extension possibilities have been outlined and may be performed in the future.

References

1. Bert E (2009) Dynamic urban origin-destination matrix estimation methodology. EPFL
2. Van Zuylen HJ, Willumsen LG (1980) The most likely trip matrix estimated from traffic counts. Transp Res Part B: Methodol 14(3):281–293
3. Cascetta E (1984) Estimation of trip matrices from traffic counts and survey data: a generalized least squares estimator. Transp Res Part B: Methodol 18(4–5):289–299
4. Cascetta E, Nguyen S (1988) A unified framework for estimating or updating origin/destination matrices from traffic counts. Transp Res Part B: Methodol 22(6):437–455
5. Krajzewicz D et al (2012) Recent development and applications of SUMO—Simulation of Urban MObility. Int J Adv Syst Meas 5(3 and 4):128–138
6. Kattan L, Abdulhai B (2011) Traffic origin-destination estimation in handbook of transportation engineering, Volume II: applications and technologies, 2nd edn. McGraw Hill Professional, Access Engineering, New York
7. Maher MJ (1983) Inferences on trip matrices from observations on link volumes: a Bayesian statistical approach. Transp Res Part B: Methodol 17(6):435–447
8. Cremer M, Keller H (1987) A new class of dynamic methods for the identification of origin-destination flows. Transp Res Part B: Methodol 21(2):117–132

9. Nihan NL, Davis GA (1987) Recursive estimation of origin-destination matrices from input/output counts. Transp Res Part B: Methodol 21(2):149–163
10. Yang H et al (1992) Estimation of origin-destination matrices from link traffic counts on congested networks. Transp Res Part B: Methodol 26(6):417–434
11. Bell MGH (1991) The real time estimation of origin-destination flows in the presence of platoon dispersion. Transp Res Part B: Methodol 25(2–3):115–125
12. Niebel W et al (2008) Traffic surveillance and forecast for large-scale events, monitoring and simulating the world youth day 2005 and the soccer world cup 2006. PROM: list studenata Fakulteta prometnih znanosti (21):64–66. ISSN:1332-2613
13. Justen A, Cyganski R (2008) Decision-making by microscopic demand modeling: a case study. In: Transportation decision making: issues, tools, models and case studies. ISBN:9-78-88-96049-06-8
14. Dafermos SC, Sparrow FT (1969) The traffic assignment problem for a general network. J Res Natl Bur Stan 73B:91–118
15. Muthuswamy S et al (2005) Improving the estimation of travel demand for traffic simulation: Part I, in final report 2005, Department of Civil Engineering, University of Minnesota
16. Nihan NL (1979) Use of volume data to reproduce ramp-to-ramp freeway trip patterns—a pilot study. Technical report standard, Washington State Department of Transportation, WSDOT-35.1

Advanced Traffic Light Information in OpenStreetMap for Traffic Simulations

David Rieck, Björn Schünemann and Ilja Radusch

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.

1 Introduction

The use of OpenStreetMap (OSM) [2] data in traffic simulation environments is very common nowadays [1, 4, 6]. 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

D. Rieck (✉)
Fraunhofer Institute for Open Communication Systems (FOKUS), Automotive Services and Communication Technologies, Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany
e-mail: david.rieck@fokus.fraunhofer.de

B. Schünemann · I. Radusch
Daimler Center for Automotive Information Technology Innovations, Technische Universität Berlin, Sekr. DCAITI Ernst-Reuter-Platz 7, 10587 Berlin, Germany
e-mail: bjoern.schuenemann@dcaiti.com

I. Radusch
e-mail: ilja.radusch@dcaiti.com

M. Behrisch and M. Weber (eds.), *Modeling Mobility with Open Data*,
Lecture Notes in Mobility, DOI 10.1007/978-3-319-15024-6_2

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.

2 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 Figs. 1, 2).

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 intersections with focus on optimized information for navigation systems, but these cannot be used to include signal information nor are they optimized for simulation purposes.

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):

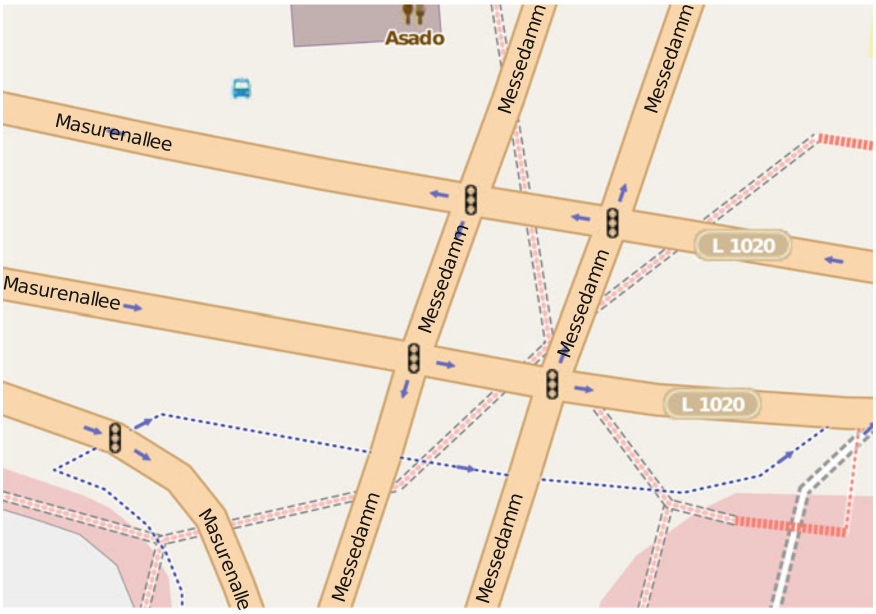


Fig. 1 Complex traffic light controlled intersection

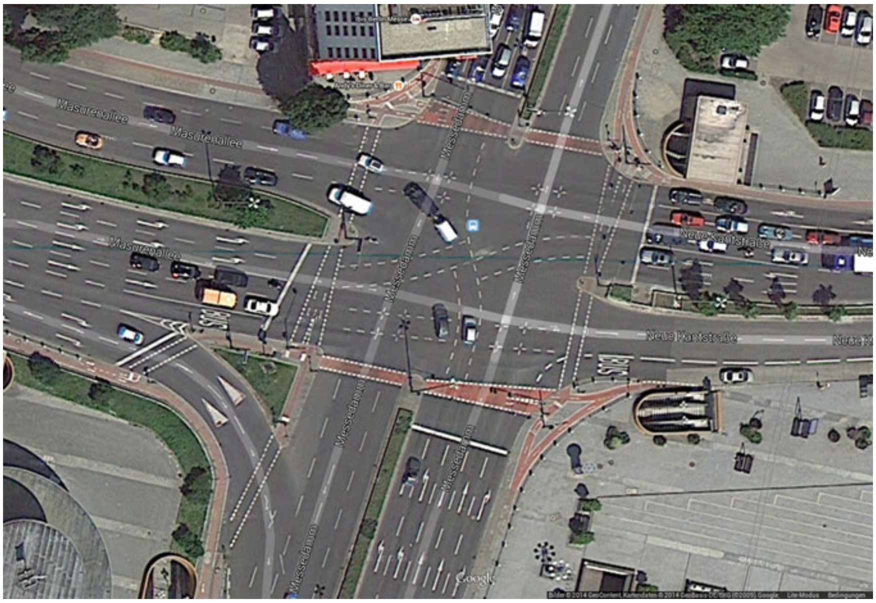


Fig. 2 Satellite view of complex intersection

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)

To find a suitable solution, we analyzed different traffic light (TL) tagging ideas (Table 1).

By comparing each requirement with the various traffic light modeling methods, we came to the following requirements matrix (see Table 2).

One can see that the alternative tagging methods do not do better than the classic tagging method at least with regard to the defined requirements. Therefore we came to the conclusion to introduce an extension of the current relation-model by adding a *traffic signal* relation, which enables lane precise traffic signal modeling (right, left, straight or combinations of all directions). This offers a high flexibility but also keeps the classic tagging system.

By using the common concept of referencing already existing attributes (lanes) and extending them with new options (from, via, to) existing information can easily

Table 1 Comparison of traffic light tagging ideas

Tagging type	Pros	Cons
Classic TL tagging		
	• Simple geometry	• Phases and timing not possible
		• No lane-specific TL-Tagging
Lane TL tagging		
	• Traffic light tags per lane	• Complex geometry
	• Turn restrictions per lane	• High mapping costs
		• Uses huge amount of data
Star TL tagging		
	• Logic modeling of lanes	• Visual representation != logic representation
	• TL and turn restrictions per lane	
	• Downwards compatible	
	• Medium mapping costs	
Area TL tagging		
	• Individual lanes	• Incompatible with current routing engines
		• Improper usage of areas to model intersection-connections

Table 2 Requirements table

	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Load	10	8	5	2	7	7	10
Classic	0	10	0	7	10	10	10
Line	10	10	0	10	0	5	10
Star	10	10	0	3	5	5	10
Area	0	10	0	3	5	0	0

Table 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

be reused. Phase and timing information is described in a similar way to SUMOs way of representing traffic signal information (|-separated values for phases and timings). Table 3 shows the extensions made to the relation.

We also show the usage of our easy-to-use plugin for the popular Java OpenStreetMap editor (JOSM), which supports the user in creating new or updating existing traffic light information in his/her area (see Fig. 8).

3 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 Fig. 9), while in other parts a more complex transition is needed (e.g. intersection lane modeling)

A screenshot of the conversion can be seen in Fig. 7. In this example, the intersection in Fig. 2 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 (Figs. 3, 4, 5, 6).

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 did not work due to the intersection complexity. With further effort on modeling the intersections, better results are expected.

4 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 [3] which offers mechanisms to connect and synchronize different simulators using a common runtime infrastructure.

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.

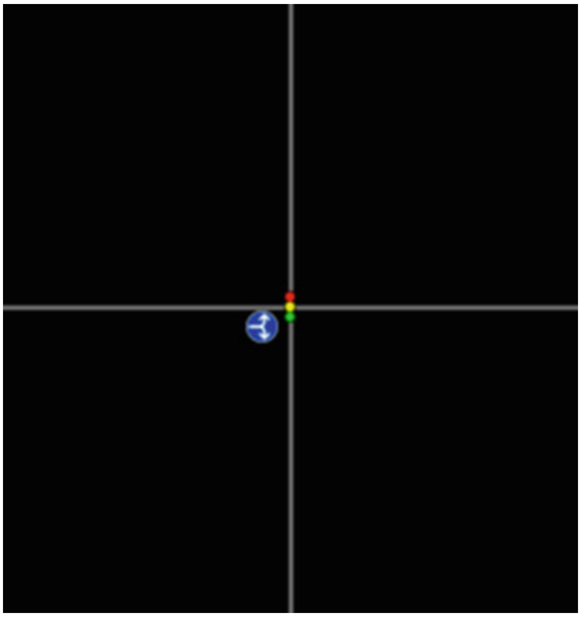


Fig. 3 The classic traffic light tagging as it is used currently, contains one traffic light tag per intersection. Turn restrictions usually belong to whole ways

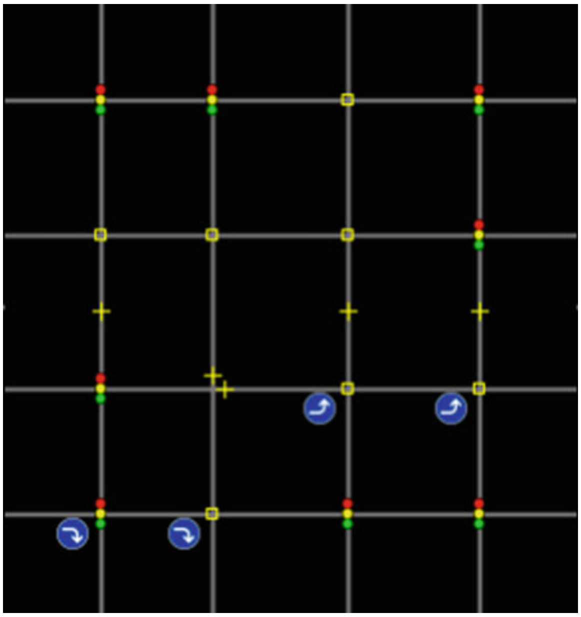


Fig. 4 The lane traffic light tagging uses a visual way of traffic light tagging. Each lane is modeled individually, traffic lights and turn restrictions are applied directly to the lanes

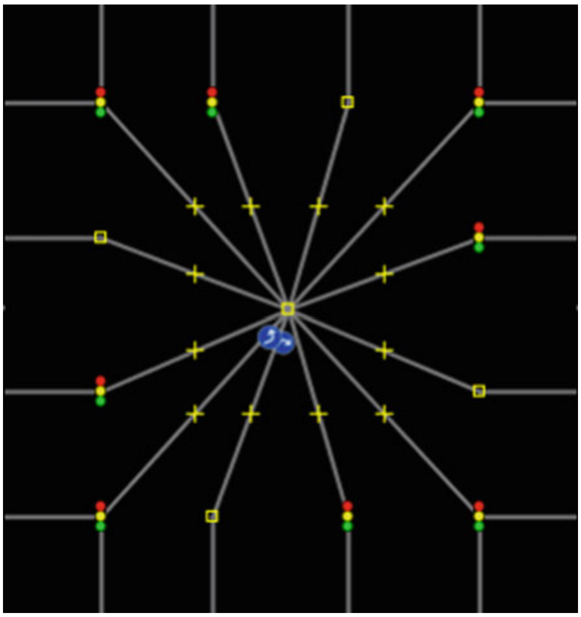


Fig. 5 Star traffic light tagging is an abstract way to model lanes logical. The intersection node as used in the classic traffic light tagging stays the same, but roads are splitted into individual lanes

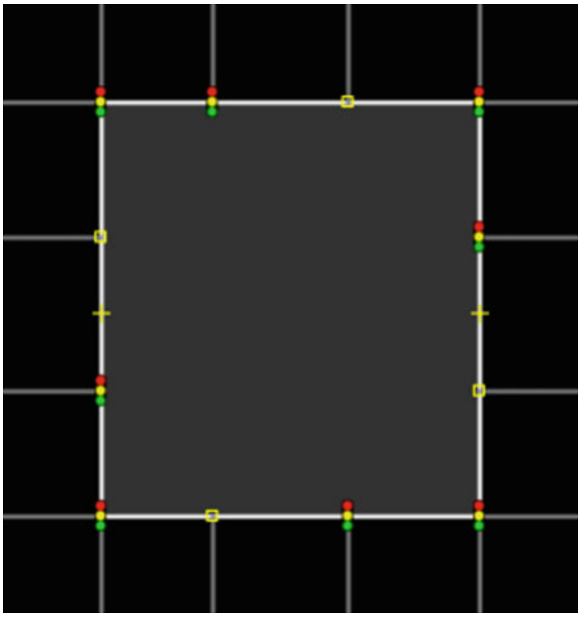
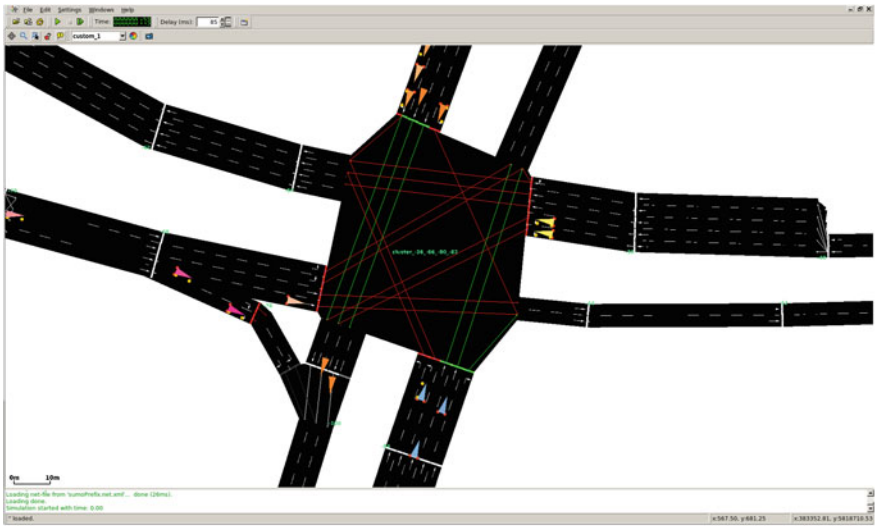


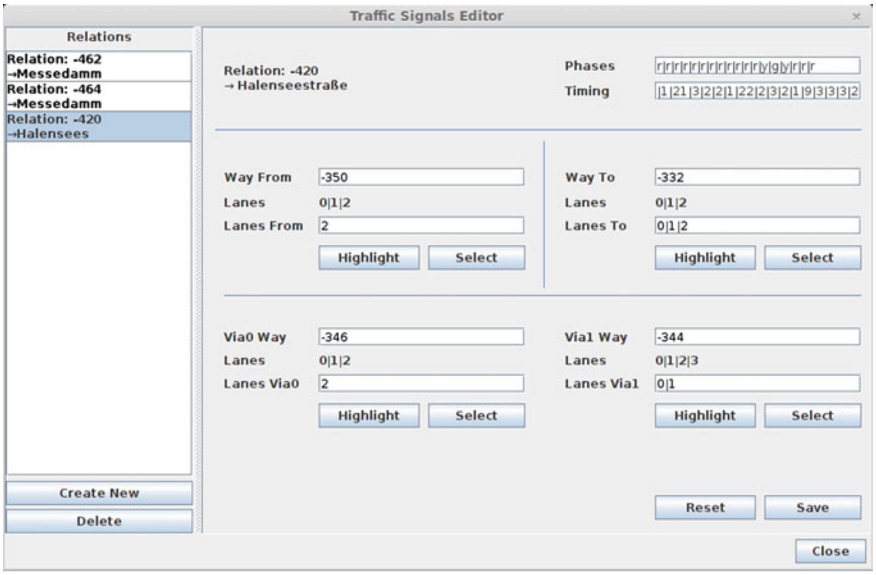
Fig. 6 The area traffic light tagging models the whole intersection as one OSM-area, where each lane connects to the intersection



```
Loading net-file from 'sumoPrefix.net.xml'...  done (26ms).
Loading done.
Simulation started with time: 0.00
```

```
* loaded.    x:567.50, y:681.25    x:383352.81, y:5818710.53
```

Fig. 7 SUMO traffic simulation with junction connections shown



Relation: -462
→Messedamm
Relation: -464
→Messedamm
Relation: -420
→Halensees

Relation: -420
→ Halenseestraße

Phases: r|r|r|r|r|r|r|r|r|r|r|r|y|g|y|r|r|r
Timing: |1|21|3|2|2|1|22|2|3|2|1|9|3|3|3|2

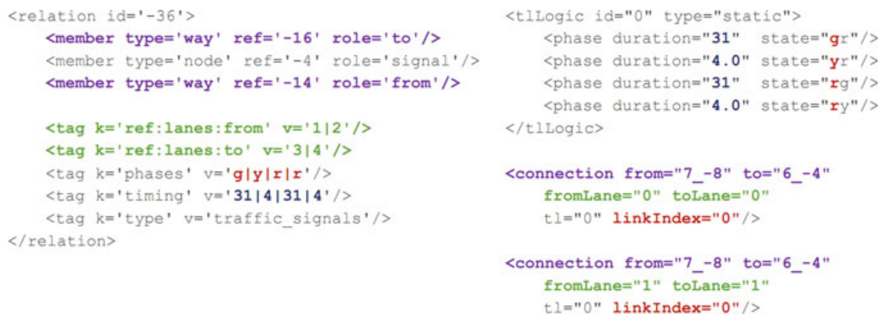
Way From: -350
Lanes: 0|1|2
Lanes From: 2

Way To: -332
Lanes: 0|1|2
Lanes To: 0|1|2

Via0 Way: -346
Lanes: 0|1|2
Lanes Via0: 2

Via1 Way: -344
Lanes: 0|1|2|3
Lanes Via1: 0|1

Fig. 8 JOSM traffic signal editor plugin



```
<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"
    tl="0" linkIndex="0"/>

<connection from="7_-8" to="6_-4"
    fromLane="1" toLane="1"
    tl="0" linkIndex="0"/>
```

Fig. 9 OSM traffic signal extension and corresponding tags and values in SUMO

5 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.

6 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.

Acknowledgements We would like to thank Tristan Wagner, Andreas Mentz, Andre Beyer and Rujun Wang for their valuable contribution to this paper.

References

1. Behrisch M, Bieker L, Erdmann J, Krajzewicz D (2011) Sumo-simulation of urban mobility-an overview. In: SIMUL 2011, The Third international conference on advances in system simulation, pp 55–60
2. Mordechai H, Patrick W (2008) Openstreetmap: user-generated street maps. Pervasive Computing, IEEE 7(4):12–18

3. Institute of Electrical and Electronics Engineers (2000) IEEE standard for modeling and simulation (M&S) high level architecture (HLA)–framework and rules. IEEE Standard 1516.1. IEEE, New York
4. Rieck D, Schünemann B, Radusch I, Meinel C (2010) Efficient traffic simulator coupling in a distributed v2x simulation environment. In: Proceedings of the 3rd international ICST conference on simulation tools and techniques, p 72
5. Schünemann B (2011) V2x simulation runtime infrastructure vsimrti: An assessment tool to design smart traffic management systems. Comput Netw 55:3189–3198
6. Zilske M, Neumann A, Nagel K (2011) Openstreetmap for traffic simulation. In: Proceedings of the 1st European State of the Map–OpenStreetMap conference, number 11-10, pp 126–134

Online Micro Modelling Using Proprietary Controllers and SUMO

Robbin Blokpoel and Jaap Vreeswijk

Abstract Over the past years the open source traffic simulator SUMO has been significantly improved and extended. One of the most important elements of urban traffic simulation is the proper handling of traffic light control. Currently available are elementary control methods like embedded fixed time and actuated control, but also controllers external to SUMO that use SUMO's extensive TraCI interface that enables reading and changing of many simulation parameters. This interface, however, has as yet not been used to link to proprietary controllers, which would enable the use of SUMO for accurate studies in a multivendor environment. Moreover, the TraCI interface accepts the injection of vehicles from external sources during the simulation. This opens up possibilities for using real-world sensor data directly in the simulation environment. This paper describes how state-of-the-art Imtech controllers are linked to SUMO. The paper covers topics like architecture, vehicle detection, signal group control, simulation speed optimization and contains a comparison of the SUMO simulation to the commercial Vissim simulator for an identical scenario. The last section of this paper introduces embedded real-time micro simulation as part of the control environment, which was able to approach.

1 Introduction

Over the past years the open source traffic simulator SUMO has been improved and extended significantly with at the time of writing a 19th version available. With a large community involved and a history of more than 10 years, the simulator can be

R. Blokpoel (✉) · J. Vreeswijk
Imtech Traffic & Infra, 2542, 3800GB Amersfoort, The Netherlands
e-mail: robbin.blokpoel@imtech.com

J. Vreeswijk
e-mail: jaap.vreeswijk@imtech.com

M. Behrisch and M. Weber (eds.), *Modeling Mobility with Open Data*,
Lecture Notes in Mobility, DOI 10.1007/978-3-319-15024-6_3

considered a serious alternative to commercially available solutions. The open source nature and easy access to almost all parameters during runtime make the simulator particularly 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 [4], in which both vehicle-to-infrastructure (V2I) and vehicle-to-vehicle (V2V) communications are modelled using the ns-3 [7] 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 control are fully supported by SUMO. For other types of control and variations on the embedded vehicle actuated method, external controllers can take over SUMO's control through TraCI (Traffic Control Interface). Currently, these external controllers, like the example Python program that comes with SUMO, are stand-alone applications specifically made for connection to SUMO. However, for a good comparison between traffic systems currently running on the street and results from 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 to what is currently available on the market. This comparison would be even stronger when real time or historical traffic demands over a long period of time can be fed to the system. That would prove the system could work over an extended period of time and not just in one scenario.

Fixed time and vehicle actuated controllers can be realistically approximated by either SUMOs 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 favour using the real world controllers. This holds even stronger for traffic adaptive control, like Imflow [9], which is too complicated to be approximated by the control available in Sumo. Furthermore, the differences between competing products are too large to simulate their behavior with a reference application.

For these reasons it was decided to create an interface between Imtech's real world controllers and SUMO. This paper describes 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. Network conversion between Imflow controlled networks and SUMO for increased ease of use is described in [2].

Additionally, this interface enables to use the simulation environment as accurate online model that only requires traffic demand data. Unlike many other models [3] no data about travel times is required. Exceptional changes in demand, which are often a problem for traffic models based on travel time measurements and neural networks [7], are handled better. This is because the real traffic light controllers are part of the model and behave the same way as the controllers on the street would behave. The paper shows a test case with a week of data from a network in Helmond in the Netherlands.

2 Architecture

The architecture of the interface and all involved components is described in the picture (Fig. 1).

The TLC (Traffic Light Controller) blocks in the diagram use the same software as is running inside real-world traffic light controller. The TLC blocks can accept just as easily real sensor input as data generated by the simulation environment. The equivalent situation holds for the actuators. The TLC interfaces to the simulation environment by means of the SimInterface. The SimInterface is a C++ dynamic link library (dll) that can maintain connections to multiple TLCs in parallel. 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 two main flows, detection information going from SUMO to the TLCs and signal group status from the TLCs to SUMO.

After initialization of the interfaces the main interface process checks detector status and signal group status every 100 ms. Changes to signal group status are written to SUMO, while detector status is sent directly to the Siminterface dll. Finally, SUMO is ordered to execute another simulation step through TraCI.

3 Detection

Detection—the acquisition of traffic sensor data—is a key element for any adaptive controller, as without detection only fixed time control is possible. Therefore, having proper detection functionality is vital for accurate simulation. SUMO supports three 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

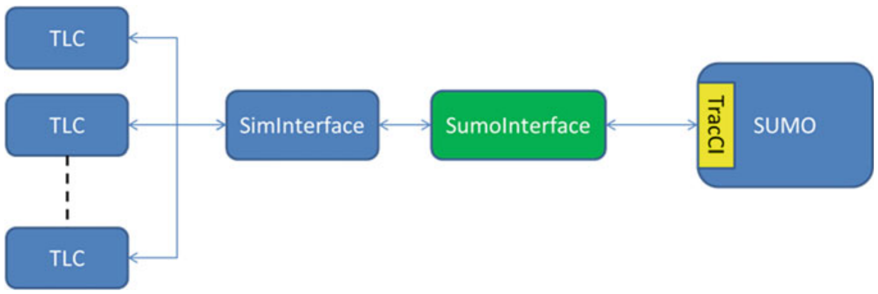


Fig. 1 Interface architecture

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 induction loop area detectors that can cover up to 30 m of the approach to an intersection. By means of these long detectors gaps in the approaching traffic can be detected, which can be used to determine the best moment to cut off the green phase.

Figure 2 shows a short loop close to the stopline and a 20 m loopat some 15 m in the upstream direction. When a vehicle leaves the long loop, the front of the vehicle is at a distance of at most 12 m from the stopline. Turning the light to amber at this moment will not make the vehicle stop, since the distance is less than 1 s. This enables the controller to utilize part of the amber time by letting the last vehicle of a 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 m 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 m from the stopline, but its efficacy would depend on a presumed fixed vehicle speed, which is not realistic close to an intersection. Moreover, this work focusses on simulating real world controllers and these expect long area loops. Using different loop configurations 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 the command "get LaneAreaDetector Variable" and the variable to acquire 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

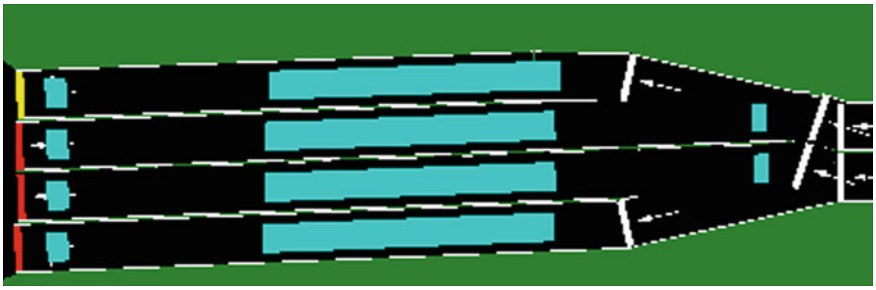


Fig. 2 Typical detection field for vehicle actuated control

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 uses the correct numbering scheme.

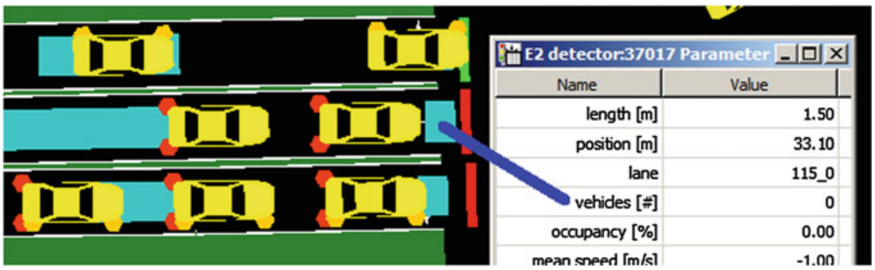
As described in the architecture section, the update time for detectors is 100 ms. This is done in order to never miss any detection event. Motorcycles can be as short as 2 m and on the highway, their speed can be over 30 m per second. This means they occupy a detector for only 100 ms. When vehicles are shorter and drive faster, a shorter update time is required. In urban environments the simulation may be speed up by checking the detectors less frequently if vehicle speeds are lower. For 4 m vehicles at 15 m/s, a 300 ms cycle suffices.

Another important aspect to consider is the stopping distance in front of a red light. This is shown in the figure (Fig. 3).

The loop indicated by the blue line is not occupied when vehicles stop at 2.5 m before the signal head, as they did in older SUMO versions. 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 1.0 m.

4 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 comprise 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, the through direction, a left turn and a 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 numbers to SUMO identification numbers. An example of a translation file is shown below:


E2 detector:37017 Parameter

Name	Value
length [m]	1.50
position [m]	33.10
lane	115_0
vehicles [#]	0
occupancy [%]	0.00
mean speed [m/s]	-1.00

Fig. 3 Problem with detector location and stopping distance

```
<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: rGGGrrrr.

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 100 ms the software checks whether the status has changed and if so sends a "Change Traffic Lights State" command with a new state tuple 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 or predefine them in a SUMO configuration file and selecting the right phase index during operation would be a lot of work.

5 Simulation Speed

It was noticed that the network used for testing the SUMO interface between TraCI and the SimInterface was running much slower after the detectors were connected. Although the number of detectors is high, with 168 divided over 5 intersections, the delay was much larger than expected. An implementation that sends separate TraCI commands for each detector requires up to 30 ms per intersection per simulation step of 100 ms. 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 real-time. A hypothesis that the large number of Traci calls caused this led to combining all detector requests of one intersection in one call. This led to an increase in simulation speed to almost 2x real-time 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 number 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.

6 Comparison Between Vissim and SUMO

For the scenario of Assen-Noord, a small network in the north of the Dutch city Assen, a comparison was made between VISSIM and SUMO. 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 join (Fig. 4)

When watching the simulations in both Vissim and SUMO, no clear differences could be noted, except that SUMO has uniform vehicle injection and the same acceleration at the stopline. SUMO was used in a standard way creating the routes with a trip file that injects 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 correctly. 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 s higher for SUMO than for Vissim. On the other hand, for normal vehicles this delay was 1.3 s lower for SUMO.

These results were obtained using standard settings as much 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 the Vissim simulation results.

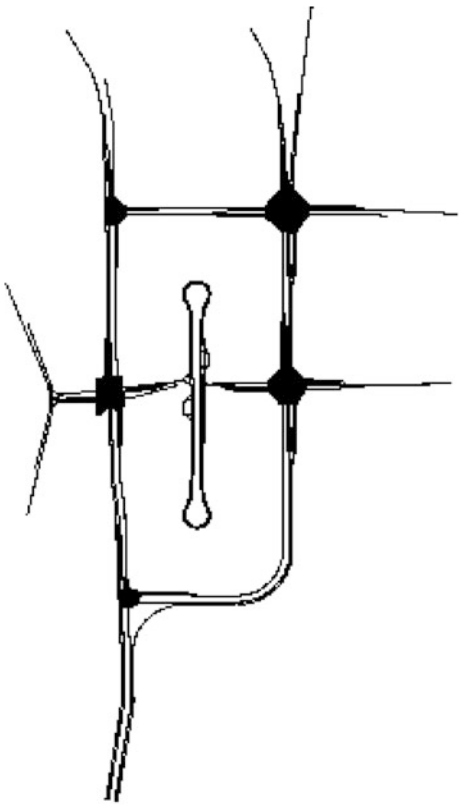


Fig. 4 Simulation network of Assen-Noord

7 Online Micromodelling

For accurate online micromodelling there are three main components that should be considered: traffic behaviour, traffic demand and traffic light control. The latter is covered by connecting the real traffic light controllers to the simulation. Traffic behaviour was found to be close to Vissim, which is generally considered an accurate model [10]. Therefore, the only remaining component is the traffic demand.

Generally, traffic counts are easily available, since traffic light controllers require counting sensors, like inductive loops to function properly. A lot of research has already been carried out for estimating OD matrices from loop counts [6, 7]. The method used in this paper only uses traffic counts, but no vehicle class specific counts. A method with class specific counts is presented in [8]. For this research the formula taken from [6] will be used:

$$\sum_{w \in W} p_w^a t_w = v_a$$

t_w is the number of trips of O-D pair w,w∈W
p_w^a is the proportion of trips O-D pair w∈W traversing link a ∈ A
v_a is the expected link flow for the link a ∈ A

There are, however, multiple valid solutions for this equation and therefore some assumptions are needed. The flow per origin can be measured directly since there are inductive loops at each entry of the simulation network under investigation, but for destinations there are multiple possibilities.

This is best understood when considering Fig. 5; a vehicle entering the network at intersection 102 going in the eastern direction can leave the network at 104 in three different directions. The same holds for a vehicle that entered at 101. Since these vehicles mix with each other inside platoons, they cannot be distinguished anymore at the loops of intersection 104. This also demonstrates that this knowledge is not necessary, because of the same mixing effects. Only in extreme cases the effects of different destination ratios from different origins will be noticable. For example when all vehicles entering the network at 102 turn left at 104, while none of 101 and 103 do that, an unexpected gap in a platoon may occur because those vehicles will still be relatively close together inside the platoon. Similarly, at the first intersection after entering the network the vehicles of a certain specific origin will also arrive at a specific time. In this case it is important to have a more specific indication in which direction they proceed, since this is important to model the effects of coordination between intersections correctly.

The solution for determining the destination in the online model is to use the ratios from a detailed static OD model for the first intersection a vehicle encounters and afterwards group traffic to follow general turning percentages at the following intersections. This detailed model was made to accurately represent the traffic demand of the network in agreement with the road operator. The resulting flows per OD pair are used as the basis of a Poisson arrival process, which injects vehicles into SUMO through the TraCI interface.

The data of the traffic counts also contains total red phase durations, which should approximate the total red duration of the online model environment. This was tested for 2 weeks of data in March 2014. As a comparison the actual total red time duration of each signal group was compared between the actual street data and

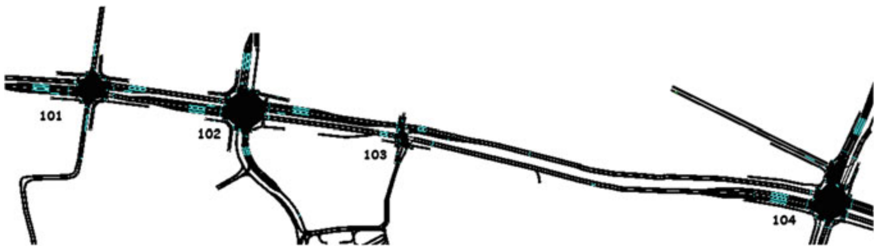


Fig. 5 Simulation network of Helmond for online modelling

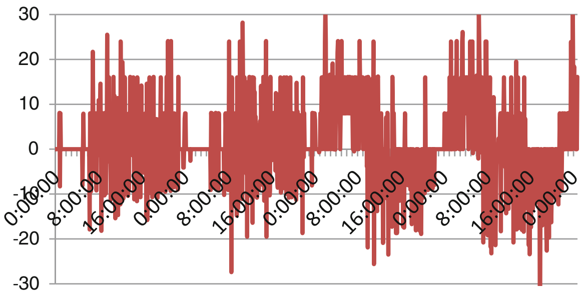


Fig. 6 Difference between simulation and actual street data from 7–10th of March 2014

the data from the simulation. The resulting difference in total red duration per 5 min for signal group 3 (left turn from the main direction) at intersection 104 is shown in the figure (Fig. 6).

As can be seen from the figure, the difference between the actual data and the simulated data is very irregular on the first 2 days. This can be explained by sampling effects, a green phase that occurred at the border of an interval can be either in the first or second interval. Moreover, the Poisson arrival process introduced some randomness in the demand as well, which could easily lead a difference of one green phase per sampling period. When data is aggregated over 30 min, the deviation does not vary between −15 and +15, with exceptions up to 30 anymore, but only increases to −30 to +30 with exceptions up to 60. This effectively doubles the variation, while the interval length is increased by a factor of 6. However, the last 2 days have a clear pattern on the difference. From midnight to midday the simulation has less red time, while at PM hours it is the other way around. A deeper investigation into the log files revealed that the controller switched to another plan at the 9th of March at 0:50AM.

When looking at the average deviation for signal group 3 of 1.5 s, it can be concluded that the simulation was quite accurate. The average absolute error was 4.9 s, but this includes the noise introduced by the Poisson arrival process and the sampling effects. For all intersections the average error and average absolute error over all signal groups were calculated and are shown in Table 1. The average absolute error is partially due to deviations from the Poisson arrival process. When putting these errors of 0.0 and 8.9 s in perspective with the data aggregation period of 300 s, the average total error is <0.1 % and the average absolute error is 3.0 %.

Table 1 Average errors per intersection

Intersection	Average error (s)	Average absolute error (s)
101	−0.7	6.2[a]
102	0.0	9.2
104	−0.6	11.4
Total	0.0	8.9

[a] When a correction is not applied for the neighboring intersection switching to a different control mode the average absolute error for intersection 101 goes up to 8.1 s

Note that 103 is just a pedestrian crossing with only public transport being allowed to use the street coming from the south. Therefore, it has no complete logging of the signal phases and only detection counts could be acquired. Apart from the missing data of intersection 103, some corrections had to be applied for certain external factors. Intersection 102 went to vehicle actuated mode at the 9th of March. Therefore, all data after the 9th was discarded. This also had a significant effect on the neighbor intersection 101, and the same correction was applied to this data set. Another issue was found with the default stance when there is no traffic on intersection 101. On the street it stayed in a stage with only signal group 8 and a long maximum green time, while in simulation it went to the stage with signal group 2, 8 and all parallel pedestrians and bicycles with a shorter maximum green time. Therefore, those signal groups were discarded from the data set.

Other inaccuracies of the model for which the data set was not corrected can be considered directions for future improvement:

- Pedestrians and bicycles were counted by the amount of times the push button was used. However, this is an inaccurate measure as a pedestrian may get impatient and push multiple times and a second pedestrian arriving may not push at all. The time between the start of red and the first time the button gets pushed would be a better indication for the arrival rate.
- Public transport priority and the respective bus schedules were not included in the simulation. These priority calls, even though not frequent can have a significant impact on the intersection.
- An extension loop at intersection 104 would sometimes freeze in the occupied state. Therefore, the corresponding signal group got a lot more green time than in the simulation, which had no broken loop. Conflicting signal groups on the other hand had fewer green. Correction for this would be difficult, but the system could create a warning that a detection inconsistency was detected.
- No special train functionality at intersection 104 was implemented in the simulation. When the railroad crossing to the south of 104 closes, special priority is given to cars coming from the south to ensure the railroad crossing to be free of cars. Additionally, after the railroad opens again, there is again a special priority for these vehicles as they have waited for a long time at the railroad already.

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, the challenges of 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.

An extension to combine this work with dynamic traffic demand patterns demonstrated the possibilities of creating an online micro-model. The results show that it is possible to model with a mean absolute error of 3.0 % for the resulting total red time per signal group in 5 min intervals. With this accurate online model, different traffic management strategies can be tested online before actually effectuating them. It also allows for testing multiple potential strategies in parallel to assist in the decision of which strategy to select.

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.

References

1. Bera S, Krishna Rao KV (2011) Estimation of origin-destination matrix from traffic counts: the state of the art. Europ Transp 49:3–23
2. Blokpoel RJ (2014) Network conversion for SUMO integration. In: 2nd SUMO conference, Berlin, Germany
3. Hoogendoorn SP, Bovy PHL (2001) State-of-the-art of vehicular traffic flow modelling. J Syst Control Eng 215:283–303 (1 June 2001)
4. Koenders E, in't Velt R (2011) Cooperative technology deployed. ITS Europe, Lyon, France
5. Krajzewicz D et al (2013) COLOMBO: investigating the potential of V2X for traffic management purposes assuming low penetration rates. In: ITS Europe congress, Dublin, Ireland, 4 June 2013
6. Liu H et al (2009) A neurak network model for travel time prediction. In: IEEE conference on intelligent computing and intelligent system
7. ns-3 (2014). ns-3 project web-pages.http://www.nsnam.org/. Accessed 11 Feb 2014
8. Seungkiri B, Hyunmyung K, Yongteak L, Gangwon L (2001) Multi-vehicle OD trip matrix estimation from traffic counts. J East Asia Soc Transp Stud 4(2)
9. Van Vliet K, Turksma S (2013) ImFlow: policy-based adaptive urban traffic control first field experience. ITS Europe, Dublin, Ireland
10. Xiao H et al (2005) Methodology for selecting microscopic simulators: comparative evaluation of AIMSUN and VISSIM. Research report, University of Minnesota

Traffic Simulation for All: A Real World Traffic Scenario from the City of Bologna

Laura Bieker, Daniel Krajzewicz, AntonioPio Morra, Carlo Michelacci and Fabio Cartolano

Abstract A large effort is needed to gather, convert and adapt all the data needed to replicate a part of a real road network. To allow performing real-world evaluations using SUMO out-of-the-box, three "real-world" traffic simulation scenarios that represent parts 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. This article describes, evaluates, and discusses the scenarios.

Keywords Real world traffic scenario · Open data · Validation · Microscopic simulation

1 Traffic Simulation and Open Data

For modelling real world scenarios a traffic simulation needs data that describe the infrastructure as well as data about the real-world traffic conditions. Only having both in an adequate quality allows obtaining valid simulation results. A traffic simulation requires representations of the following:

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

L. Bieker (✉) · D. Krajzewicz
German Aerospace Center, Rutherfordstraße 2, 12489 Berlin, Germany
e-mail: laura.bieker@dlr.de

A. Morra · C. Michelacci
Comune di Bologna, Piazza Maggiore 6, 40124 Bologna, Italy

F. Cartolano
Consorzio ib Innovation, Via Altabella 15, 40126 Bologna, Italy

M. Behrisch and M. Weber (eds.), *Modeling Mobility with Open Data*,
Lecture Notes in Mobility, DOI 10.1007/978-3-319-15024-6_4

Without these representations, a realistic traffic simulation is hardly possible. But collecting, processing and validating given input data may be time consuming. Sometimes, it is already difficult to gather data that describes the real world situation in the regarded area. Especially traffic light signal plans are rarely open to the public and are often not available in a digital format. Available road network representations 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, the preparation of a real-world scenario is usually time-consuming. Thus, real world scenarios from Bologna were prepared and are described in this work. They have been made publicly available within the SUMO package since version 0.21 [1, 2]. By making the scenarios available to the public, the scenarios can be used for further research with little effort.

2 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") [3]. 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 [4]. A large part of the project was dedicated to determining the state of the traffic in the city of Bologna (Italy, see Fig. 1) as well as modelling it. Major contribution to this task was performed by the municipality of Bologna who was a project partner in iTETRIS. Besides describing the situation and the traffic 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 [5]. Each of the scenarios modelled the peak hour in Bologna (8:00 am–9:00 am). Additional data sets supported by the municipality of Bologna included positions of traffic lights, traffic light plans, positions of inductive loop and their measures and many others. A further scenario, "joined", was implemented within iTETRIS by merging both previously named Vissim scenarios.

In the following, the three areas of Bologna which are used for the traffic simulation scenario are described.

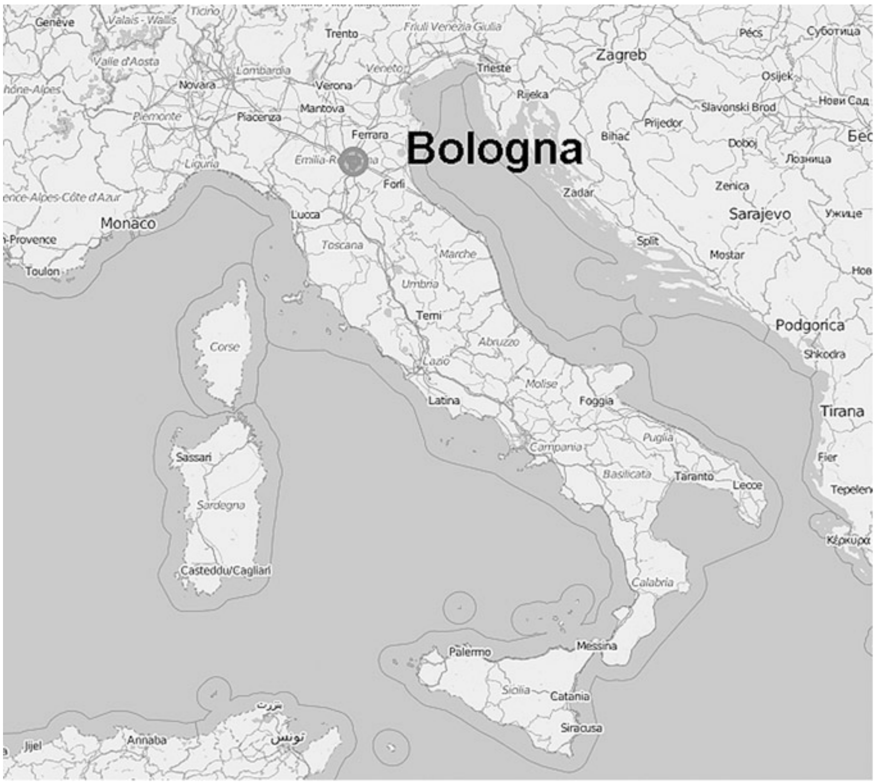


Fig. 1 Location of Bologna in Italy [12]

2.1 Andrea Costa Scenario

The simulation scenario including the roads around the street named "Andrea Costa" is described in the following as "Andrea Costa scenario". The Andrea Costa scenario is located nearby a football stadium and was set up to simulate the mobility of big events such as football matches or concerts (see Fig. 2).

In Table 1 a description of the simulation scenario for Andrea Costa is given. These numbers should help users to identify the appropriate scenario for their needs.

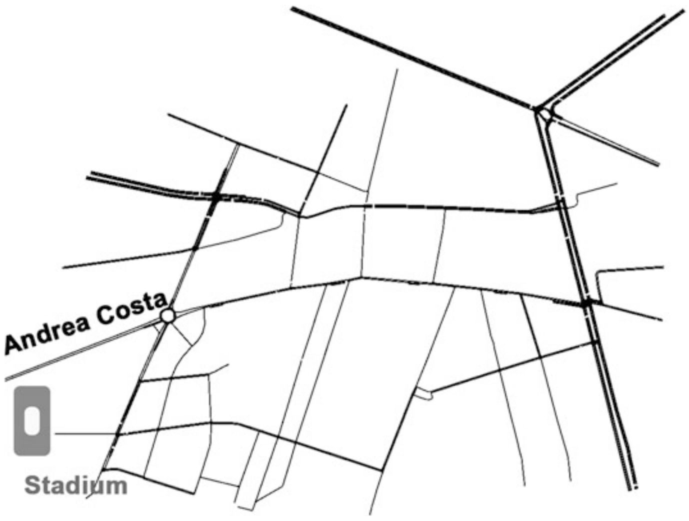


Fig. 2 Andrea Costa traffic network in SUMO

Table 1 Scenario description for Andrea Costa

Area size of network	2.45 km^2
Total number of edges	179
Number of edges with 1 lane	119 (~66 %)
Number of edges with 2 lanes	32 (~18 %)
Number of edges with 3 lanes	28 (~16 %)
Number of edges with 4 or more lanes	0
Total number of nodes	112
Traffic lights	7
Induction loops	60

2.2 Pasubio Scenario

The simulation scenario around the street "Pasubio" is named in the following "Pasubio scenario". The Pasubio scenario extends the scenarios by the area around the hospital and includes also common routes to the football stadium (see Fig. 3).

The pasubio scenario has almost the same size as the Andrea Costa scenario (see Table 2). Also the number of traffic lights and induction loops is almost equal. The major difference is the existence of roads with more than three lanes within the Pasubio scenario. If a traffic scenario with a street which have more than 3 lanes is needed the Pasubio scenario should be used.

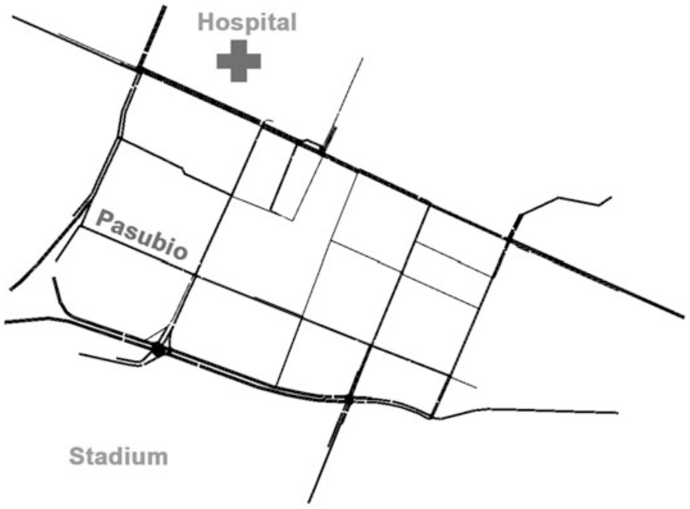


Fig. 3 Pasubio traffic network in SUMO

Table 2 Scenario description for Pasubio

Area size of network	2.45 km^2
Total number of edges	111
Number of edges with 1 lane	56 ($\sim$50 %)
Number of edges with 2 lanes	36 ($\sim$32 %)
Number of edges with 3 lanes	18 ($\sim$16 %)
Number of edges with 4 or more lanes	1 ($\sim$1 %)
Total number of nodes	65
Traffic lights	8
Induction loops	64

2.3 Andrea Costa and Pasubio Joined Scenario

Both scenarios, Andrea Costa and Pasubio, cover only a relatively small area. Therefore, it was decided to generate one scenario composed of both. This scenario was obtained by joining both given scenarios, as the areas they cover overlap (Fig. 4).

The joined scenario with its larger size (see Table 3) provides more alternatives for traffic simulations, for example routes directly from the stadium to the hospital can be simulated. The network has more route choices and longer routes can be evaluated.

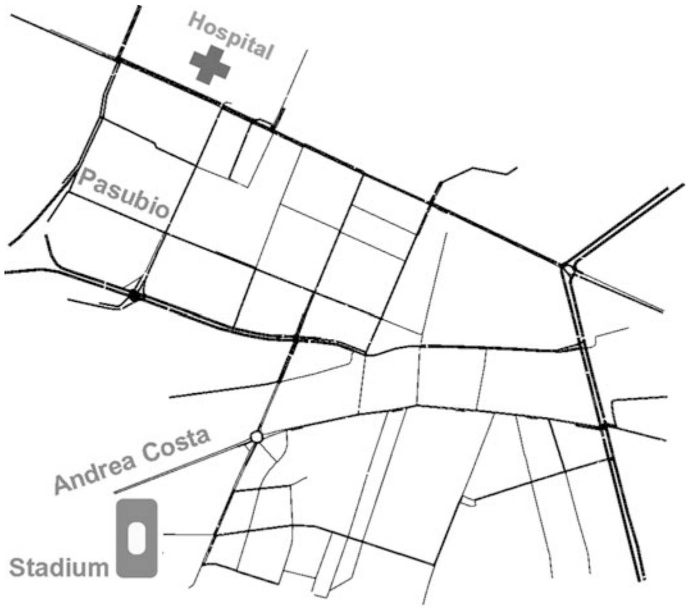


Fig. 4 Andrea Costa and Pasubio traffic network in SUMO

Table 3 Scenario description for Andrea Costa-Pasubio joined scenario

Area size of network	4.15 km^2
Total number of edges	268
Number of edges with 1 lane	165 (~62 %)
Number of edges with 2 lanes	63 (~24 %)
Number of edges with 3 lanes	39 (~15 %)
Number of edges with 4 or more lanes	1 (<1 %)
Total number of nodes	159
Traffic lights	13
Induction loops	106

3 Development of the Scenarios

The conversion of the originally given scenarios (to make them be readable by SUMO) included different working stages. First it was needed to import the traffic road network into a SUMO readable format. Next, the traffic lights have been integrated into the simulation network. Finally, the traffic demand was generated. The development steps are described in the following.

3.1 Traffic Road Network

The Andrea Costa and Pasubio scenarios were supported as VISSIM scenarios. They describe a slightly pruned road network that does not contain some smaller streets and/or pathways. Traffic lights are defined, including their positions and signal plans.

Although VISSIM is a microscopic simulation just as SUMO, it follows a completely different concept of modelling the road network. The main difference is that VISSIM is not using a graph concept, consisting of nodes (intersections) and edges (streets/roads) as SUMO does, but only of roads and connections between them. This difference makes importing VISSIM networks cumbersome and the results must often be edited by hand after an initial conversion. Figure 5 shows the differences between the network representation in Vissim and SUMO.

Because SUMO only supports the import of VISSIM networks stored in German language (at that time, the VISSIM format used a man-machine language for network descriptions) the supported networks had to be translated from English into German, first. The complete workflow of importing networks from VISSIM was as shown in Fig. 6. The manual validation step shown in this figure was done by comparing the network with images from Google Earth [6] and Google Maps [7], 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.

After the road network was successfully imported into SUMO, bus lanes were integrated into the traffic network. Bus lanes (lanes or even roads which must not be used by passenger vehicles) were not modelled within the original VISSIM network explicitly. After the networks have been prepared initially, this information was manually inserted into them. Rather surprising was the fact that within the given VISSIM scenarios, some passenger vehicles were using lanes dedicated to busses only. This was clarified by the municipality of Bologna: the given scenarios represent the reality—including passenger car drivers which ignore the prohibitions.

While the work on the Andrea Costa and the Pasubio scenario was finished at this stage, the generation of the "joined" scenario required further steps. For being

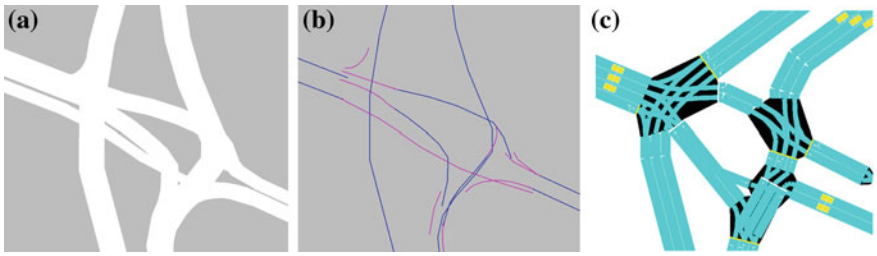


Fig. 5 Representation of an intersection in Vissim and SUMO; **a** as traffic area in Vissim, **b** as edges/connections graph in Vissim, **c** as nodes/edges graph in SUMO

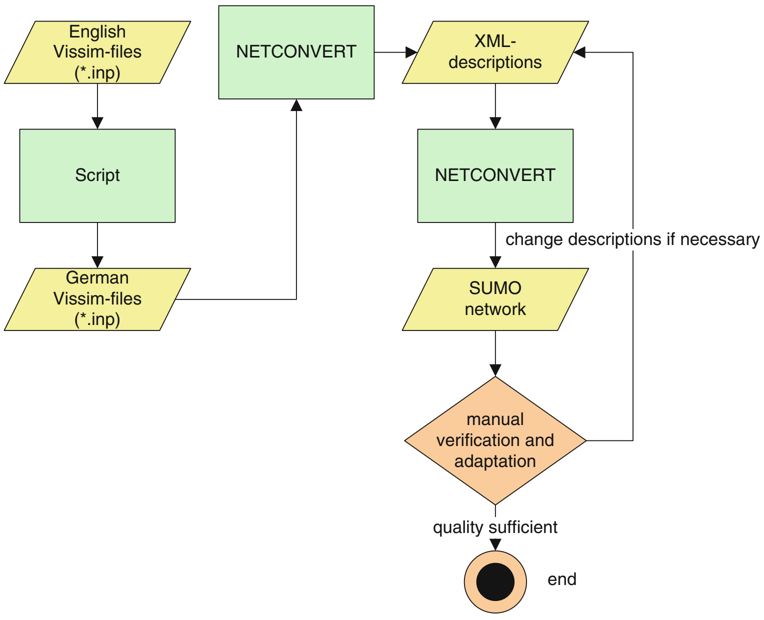

English Vissim-files (*.inp)
Script
German Vissim-files (*.inp)
NETCONVERT
XML-descriptions
NETCONVERT
change descriptions if necessary
SUMO network
manual verification and adaptation
quality sufficient
end

Fig. 6 Workflow of VISSIM network import

joined, both networks must use distinct ids for naming nodes (intersections) and edges (roads). The most straight forward approach was to use a prefix for these structures. All edges' and nodes' IDs of both networks were changed by adding an "a" and a "b", respectively. For the used networks, their overlapping parts—roads and intersections included in both—were determined. For each combination, it was decided which shall be kept and the according one from the other network was not included in the final, joined network. The unused edges were removed from the according edges file.

3.2 *Traffic Lights*

For including traffic lights into an existing road network the positions of the traffic lights as well as the definition of the signal time plans are needed. The municipally of Bologna provided in addition to the road network also definitions of traffic lights given as telemetry files and signal time plans. To import them, a script has been written that reads information about lane-based linkage (obtained from the telemetry files) and time plans (obtained from the signal time plans) and build traffic light descriptions SUMO can read.

It should be said that the city of Bologna has installed the UTOPIA system for traffic light control. This traffic light control system adapts timing and in some cases even the order of traffic light phases to the current demand on the controlled roads. The supported data set included 55 telemetry files and 39 signal time plans in the whole area of Bologna.

3.3 Traffic Demand

Likewise the traffic network and the traffic lights also the traffic demand for the scenarios were given by the municipal of Bologna. Passenger vehicles are described in an aggregated manner in both VISSIM simulations: 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.

The demand for the joined network was built using a script written especially for this task. The script reads both route files obtained from the import of the original VISSIM files. These files contain vehicles with their departure times and routes. At first, all vehicles from both files which are starting at roads that are not at the boundaries of the joined network are kept in memory for later use. Then, the script processes all vehicles which start at the network's boundaries. Each vehicle's route is examined by going through its edges. If an edge occurs in the route which was included in both networks, routes starting at this edge are retrieved. The route's continuation is then chosen randomly from these routes.

In addition to the passenger vehicles, all three scenarios include a description of the public bus transport in the regarded area. Both, positions of the bus stops as well as bus routes and schedules are given and were imported into the SUMO scenarios.

3.4 Additional Traffic Infrastructure Data

The resulting networks replicate the scenario descriptions from the VISSIM files. In addition, induction loops were added to the SUMO scenarios. Their positions were retrieved from the GIS database.

4 Demand Evaluation

For the demand generation and evaluation the municipality of Bologna supplied two datasets of detector measures. 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 compared to other sites where the number of errors in the detector data is known to be higher.

The second data set contains the measures for the same days, aggregated into intervals of 5 min. The quality corresponds to the data set aggregated into 1800s. Unfortunately, the detectors are measuring only the amount of vehicles which are passing the detectors within 5 min 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 Fig. 7. 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.). The traffic flow decreases afterwards a little bit and remains at a certain level. In the evening the traffic flow is rising to another peak.

For the validation of the simulation the real world measurements were compared to the results of the simulation. As mentioned, only the flows 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 are presented. Table 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 p.m. against the ones obtained from the simulation using the demands imported from VISSIM. The optimal results would be a bisectrix.

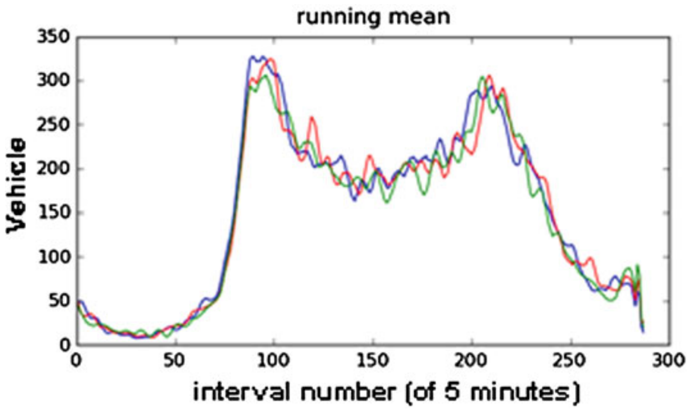
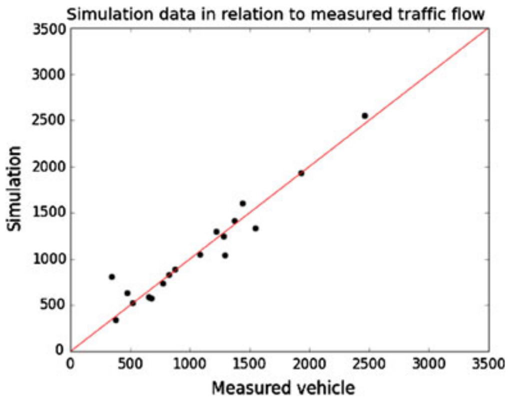


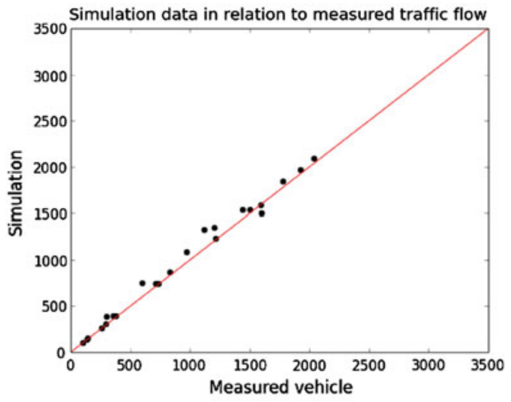
Fig. 7 Example traffic flow of 3 days (11.11.2008, 12.11.2008, 13.11.2008)

Table 4 Comparison of the simulation results and the measured values from induction loops

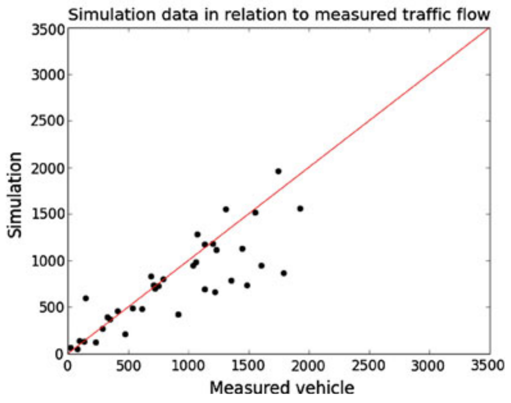
Andrea Costa: one hour simulated



Pasubio: one hour simulated



Andrea Costa Pasubio joined: one hour



As a result the following can be seen: The overall numbers of vehicles which are simulated in SUMO are relatively well, but it can be seen that the simulation has problems to simulate the traffic demand in the joined scenario within 1 h. 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 a stable traffic state is reached. This simulation “warm-up” is a known need within traffic simulations. Usually, double of the maximum travel time through the network is used [8].

5 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 to 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. First, the demand is given for only 1 h. Considering a warming up phase of the simulation, there are only approximately 30 min of simulation which can be used. Furthermore, the network sizes of the scenarios are relatively small so the route choices in the scenarios are very limited. Therefore rerouting algorithms are not recommended to be simulated with the provided scenarios.

6 Examples of Usage

The scenarios have been used to study different research questions. Some of them are briefly described here to draw a picture of the possible usage of the simulation scenarios.

6.1 Bus Lane Management

In the iTETRIS project an application for traffic management was evaluated to open bus lanes for heavy occupied vehicles in case of a big event. 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 1 week in prior. Additional routes were generated from the analyzed data set [9] (see Fig. 8). If a higher traffic demand was detected in the simulation, bus lanes were opened also to passenger vehicles.

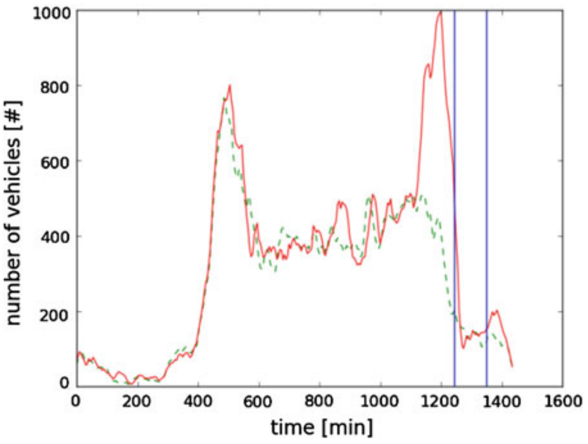


Fig. 8 Comparison between a normal Wednesday (17.10.2010; *dashed line*) and one at which a football match took place (24.10.2010; *continues line*) as measured by one chosen induction loop. The *vertical lines* denote the begin and the end of the football match

6.2 Emergency Vehicle Evaluations

The Bologna scenarios were used for evaluation of the emergency vehicle prioritization. Emergency vehicles are sending their position and their desired route via V2I communication to the road side units. The traffic lights of the route of the emergency vehicle is set to green when the emergency vehicle is approaching and to red for all other traffic participants. After the emergency vehicle has passed the traffic light it continues its normal operation [10].

6.3 Pedestrian Modelling

In the COLOMBO project SUMO was extended by a pedestrian model [11]. The Bologna scenarios were used to test the new model with real world intersections and traffic lights.

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.

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.

References

1. Krajzewicz D, Erdmann J, Behrisch M, Bieker L (2012) Recent development and applications of SUMO—simulation of urban mobility. Int J Adv Syst Meas 5(3 and 4):128–138
2. Simulation of Urban MObility (2014). www.sumo.dlr.de. Accessed 6 Jan 2015
3. iTETRIS (2014). http://www.ict-itetris.eu/10-10-10-community/. Accessed 13 Oct 2014
4. iTETRIS Consortium (publisher): iTETRIS Deliverable 3.2—Traffic Modelling: ITS Algorithms, April 2010
5. PTV AG (2014) PTV AG web site. http://www.ptv.de/. Accessed 11 Sept 2014
6. Google (2010) Google Earth web site. http://earth.google.com.
7. Google (2010) Google Maps. http://maps.google.com/.
8. Antoniou C, Barcelo J, Brackstone M, Celikoglu HB, Ciuffo B, Punzo V, Sykes P, Toledo T, Vortisch P, Wagner P (2014) Traffic simulation: case for guidelines. http://publications.jrc.ec.europa.eu/repository/bitstream/111111111/30680/1/2014_multitude_guidelines_on-line.pdf.
9. Bieker L, Krajzewicz D (2011) Evaluation of opening bus lanes for private traffic triggered via V2X communication. In: Proceedings of the first forum on integrated and sustainable transportation systems (FISTS), 29 June–1 July 2011, Vienna, Austria
10. Bieker L (2011) Emergency vehicle prioritization using vehicle-to-vehicle communication. Young researchers seminar, 8–10 June 2011, Copenhagen, Denmark
11. COLOMBO consortium (publisher): COLOMBO project's Deliverable D2.1: "Policy Definition and dynamic Policy Selection Algorithms", November 2013
12. OpenStreetMap (2014) http://www.openstreetmap.org. Accessed 30 Sept 2014

Can Road Traffic Volume Information Improve Partitioning for Distributed SUMO?

Ulrich Dangel, Quentin Bragard, Patrick McDonagh, Anthony Ventresque and Liam Murphy

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 Quadtrees and Quadtrees, 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

U. Dangel · Q. Bragard (✉) · A. Ventresque · L. Murphy
Lero@UCD, School of Computer Science and Informatics,
University College Dublin, Dublin, Ireland
e-mail: quentin.bragard@ucdconnect.ie

U. Dangel
e-mail: ulrich.dangel@ucdconnect.ie

A. Ventresque
e-mail: anthony.ventresque@ucd.ie

L. Murphy
e-mail: liam.murphy@ucd.ie

P. McDonagh
Lero@DCU, School of Electronic Engineering, Dublin City University, Dublin, Ireland
e-mail: patrick.mcdonagh@dcu.ie

1 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].

2 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.

3 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

IRIS.[1] 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.

3.1 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.

$$N_w = \sum w_t \frac{c_{tn}}{c_t}. \tag{1}$$

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).

$$w_t = \frac{c_{tn}}{c_{max}}. \tag{2}$$

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

3.2 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 Sect. 3.1 above, 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.

[1] http://iris.dot.state.mn.us/.

3.3 Modification of Smart Quadtrees

Smart Quadtrees, also referred as grid based partitioning, are an extension to Quadtrees 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 Quadtrees as Quadtree divides a map into 4 similar regions, while Smart Quadtrees divides a map into small grids and merges them until all grids are merged (Fig. 1).

The unmodified version of the Smart Quadtree implementation uses the street size (street lengths * number of lanes) as a heuristic. We modify Smart Quadtrees by changing the heuristic to use the sum of the volume data, as described in Sect. 3.1, for each grid. The difference between the two implementations is shown in Fig. 2.

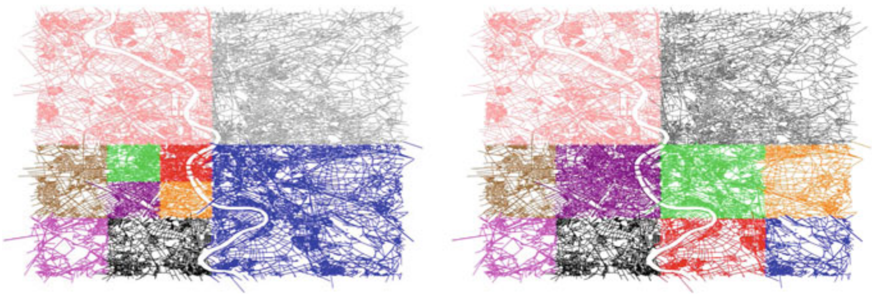


Fig. 1 Output of the modified quadtree algorithm (*left*) and unmodified quadtree (*right*) with ten partitions or three divisions

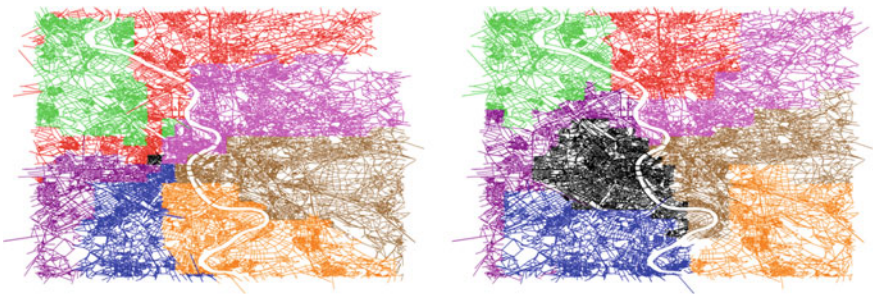


Fig. 2 Output of the modified smart quadtree algorithm (*left*) and unmodified smart quadtree (*right*) with eight partitions

3.4 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 Sect. 3.1) instead of using the nodes with the highest degree. Figure 3 shows the partitioning result of both the unmodified and modified version of SParTSim.

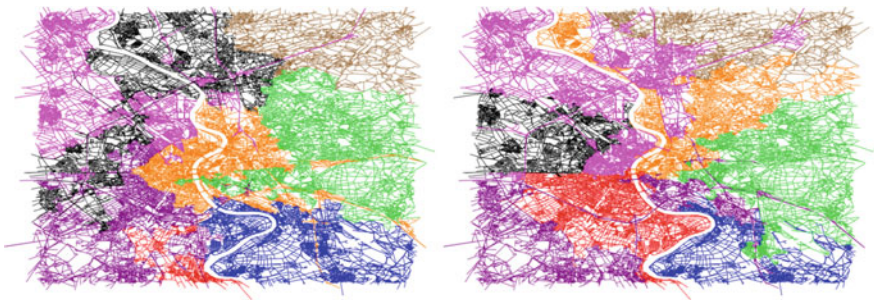


Fig. 3 Output of the modified SParTSim algorithm (*left*) and unmodified SParTSim (*right*) with eight partitions

3.5 Simulation of a Distributed Simulation

In this paper we are only interested in the impact of urban data on the quality of the partitioning, not on the distributed simulation itself. For more details on the latter, we refer the reader to our previous work on a distributed version of SUMO [5]. We have then decided to focus on the partitioning and only "simulate" the distributed simulation: TAPAS Cologne gives us the position of every vehicle at any given time, so that we are able to know precisely when a vehicle crosses the border of a partition for every partitioning schemes; this allows us to run SUMO for each partition and to manage the passing of vehicles in an ad hoc manner, without using the communication and synchronisation mechanisms of a real distributed simulation. It is important to tell again that while we avoid here all the characteristics of a distributed simulation (e.g., communication time and synchronisation mechanism), this does not have any impact on the focus of our study, i.e., an evaluation of the improvement of using urban traffic data for the partitioning in a distributed simulation.

4 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 Fig. 1, with the outputs for Smart Quadtrees with eight partitions is shown in Fig. 2 and those for SParTSim are shown in Fig. 3.

4.1 Metrics

In the first part of 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.

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

In the second part of our evaluation we simulate the use of a distributed simulation to measure the time required to process each simulation and the real time factor. In a conservative distributed simulation, the slowest node determines the time required to run the whole simulation. The more the simulation is load-balanced, the smaller the difference will be between the slowest and the fastest node. The second metric used is the average time required by a mock distributed simulation to be executed. The third metric is the number of Vehicle Per Second (VPS) which is measured by adding the number of vehicles processed at each step divided by the runtime. The last metric is the real time factor correspond to how much faster the simulation is compared to the real time. For instance, a real time factor of 10 means that the simulation execute 10 s of simulated time in 1 s of real time.

5 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 7,200 steps, with one step representing one second in real-time. TAPAS Cologne contains more than 250,000 vehicles traces for the 2 h period.

Figures 4 and 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.

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 (Fig. 1) for further partitioning, resulting in additional communication overhead. For the Smart Quadtree algorithm, our modified version created some small partitions (Fig. 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

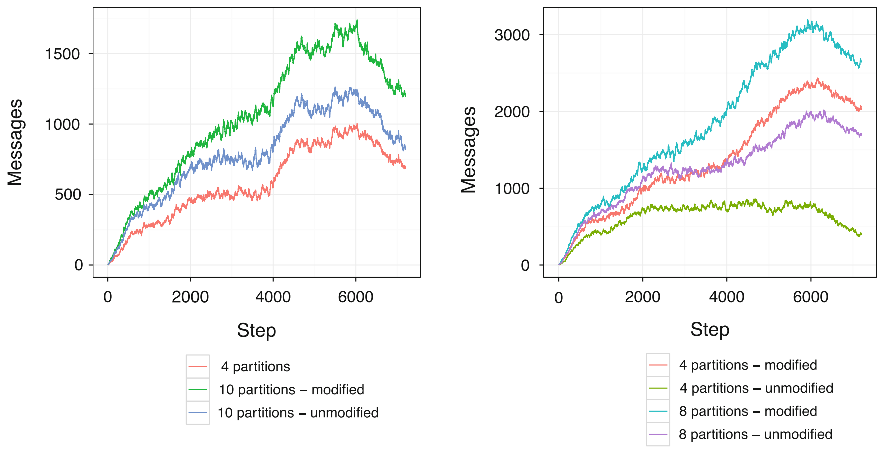
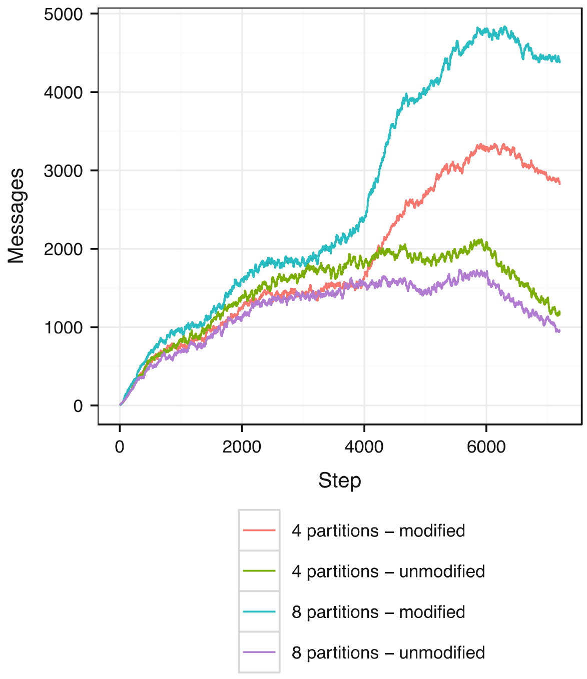


Fig. 4 Number of messages sent per simulation step for quadtrees (*left*) and smart quadtrees (*right*). Modified and unmodified versions are both shown

Fig. 5 Number of messages sent per simulation step for SParTSim, both modified and unmodified for 4 and 8 partitions



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 Fig. 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 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.

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 1 h 13 min to compute 10 partitions, Smart Quadtree took 1 h 22 min to compute eight partitions while SParTSim took 5 h 14 min for eight partitions on a 4 Core I 7-2,600 with 16 GB 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

Table 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

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.

While Table 1 shows the theoretical benefits of using urban data for the partitioning in terms of load balancing (i.e., Simpson Index), Table 2 presents some experimental results from the simulation of the distributed simulation. The results for Quadtree show that while the average time and the VPS stay stable for both configurations, the runtime is divided by more than 2 for the modified methods, using urban data, and the Real time factor is proportionally twice higher. Regarding Smart QuadTree, the modified versions for 4 and 8 partitions get a runtime improvement of respectively 13 and 22 %. While the average time and the VPS just slightly vary, Real time factor is improved in the same way than runtime. As we previously observed for the load balancing with the Simpson index, using urban data to optimise SParTSim does not seem to improve the simulation time.

When we increase the number of partitions to 16, the results seem to show a limited improvement. Despite increasing the number of partitions, QuadTree presents no improvement between 10 and 16 partitions for the modified version. Again the modified SParTSim shows results a little bit worse than the original SParTSim but the improvement due to the increase of the number of partitions goes from 6 to 15 %. On the other side, while the original Smart QuadTree achieves good results, the modified version shows a slower runtime. This decrease in performance comes from a bad choice of seeds. It appears that some seeds generated regions inside

Table 2 Execution time for every SUMO instances in the simulated version of a distributed SUMO

Name		Time(s)	Avg time(s)	VPS	Real time factor
Quadtree	4 partitions	247.46	107.75	78830.22	30.31
	10 partitions—modified	104.78	47.388	71834.39	71.58
	10—unmodified	223.52	44.12	75072.70	33.55
	16 partitions—modified	103.1	30.08	35056.79	72.75
	16—unmodified	198.11	27.86	83255.98	37.86
Smart quadtree	4 partitions—modified	115.78	97.13	112691.76	64.78
	4 partitions—unmodified	133.33	92.14	108292.90	56.26
	8 partitions—modified	93.62	51.44	59541.65	80.12
	8 partitions—unmodified	120.61	45.34	72850.18	34.80
	16 partitions—modified	197.79	27.87	83389.63	37.92
	16—unmodified	87.08	23.79	107091.92	86.13
SParTSim	4 partitions—modified	179.76	84.60	115305.59	41.72
	4 partitions—unmodified	176.34	82.42	110353.29	42.53
	8 partitions—modified	100.34	44.43	92839.74	74.75
	8 partitions—unmodified	88.29	42.71	97958.16	84.95
	16 partitions—modified	94.09	22.44	86193.98	79.72
	16—unmodified	75.45	23.84	86593.48	99.41

others and could not fully grow to form proper regions. As Smart QuadTree does not have any trading phase, it creates in this case tiny regions inside others, processing only few vehicles and spending most of their time communicating vehicles with other partitions.

These results lead to two observations. Firstly, we can see that the runtime and the Simpson Index match in most cases. The higher the Simpson Index is, the smaller the simulation time will be. This observation reinforce us in thinking that the Simpson Index is a good indicator of the load balanced state of a distributed simulation. Moreover, as expected, the average time and the number of vehicles per second prove to give no indication on the load balancing. Secondly, it looks like there is a direct correlation between the smartness of an algorithm and the improvement obtained with urban data: the smarter the algorithm, the smaller the impact of the optimisation. Optimised Quadtree, Optimised Smart Quadtree and SParTSim provide close results for 8–10 partitions.

6 Conclusion

In this paper we propose the of use 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. On the other side, we also show that the optimization using traffic volume has its limitation and cannot make a simple algorithm such as Smart QuadTree as reliable as a urban traffic dedicated partitioning algorithm such as SParTSim.

Acknowledgment 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)

References

1. Abbott J (2013) State of the world's cities: prosperity of cities, Australian Planner, pp 1–2
2. Karypis G, Kumar V (1998) A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J Sci Comput 20:359–392
3. Soares G, Macedo J, Kokkinogenis Z, Rossetti RJ (2013) An integrated framework for multi-agent traffic simulation using SUMO and JADE. In: SUMO2013, The first SUMO user conference, 15–17 May 2013—Berlin-Adlershof, Germany, pp 125–131
4. Bellifemine F, Bergenti F, Caire G, Poggi A (eds) (2005) JADE—a java agent development framework. Multi-agent programming, Springer, pp 125–147
5. Bragard Q, Ventresque A, Murphy L (2013) dSUMO: towards a distributed SUMO. In: SUMO2013, The first SUMO user conference, 15–17 May 2013—Berlin-Adlershof, Germany
6. Ventresque A, Bragard Q, Liu ES, Nowak D, Murphy L, Theodoropoulos G et al (2012) 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, pp 202–209
7. Finkel RA, Bentley JL (1974) Quad trees a data structure for retrieval on composite keys. Acta Informatica 4:1–9
8. Wang Y, Lees M, Cai W (2012) Grid-based partitioning for large-scale distributed agent-based crowd simulation. In: Proceedings of the winter simulation conference, p 241
9. Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. Commun ACM 51:107–113
10. Amanatides J Woo A (1987) A fast voxel traversal algorithm for ray tracing. In: proceedings of EUROGRAPHICS, pp 3–10
11. Radha H, Vetterli M, Leonardi R (1996) Image compression using binary space partitioning trees. Image Process IEEE Trans 5:1610–1624
12. Torres E (1990) Optimization of the binary space partition algorithm (BSP) for the visualization of dynamic scenes. In: Eurographics, pp 507–518
13. Steed A, Abou-Haidar R (2003) Partitioning crowded virtual environments. In: Proceedings of the ACM symposium on virtual reality software and technology, pp 7–14
14. Freisleben B, Hartmann D, Kielmann T (1997) Parallel raytracing: a case study on partitioning and scheduling on workstation clusters. In: Proceedings of the thirtieth hawaii international conference on system sciences, 1997, pp 596–605
15. Dhillon IS (2001) 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, pp 269–274
16. Pothen A, Simon HD, Liou K-P (1990) Partitioning sparse matrices with eigenvectors of graphs. SIAM J Matrix Anal Appl 11:430–452
17. Hendrickson B, Leland R (1995) An improved spectral graph partitioning algorithm for mapping parallel computations. SIAM J Sci Comput 16:452–469
18. Kernighan BW, Lin S (1970) An efficient heuristic procedure for partitioning graphs. Bell Syst Tech J 49:291–307
19. Fjällström PO (1998) Algorithms for graph partitioning: a survey. linköping electron art comput inf sci 3(10):1–37
20. Hendrickson B, Leland RW (1995) A multi-level algorithm for partitioning graphs. SC 95:28
21. Andreev K, Racke H (2006) Balanced graph partitioning. Theor Comput Syst 39:929–939
22. Karypis G, Aggarwal R, Kumar V, Shekhar S (1999) Multilevel hypergraph partitioning: applications in VLSI domain. IEEE Trans Very Large Scale Integr VLSI Syst 7:69–79
23. Lowrie P (1990) Scats, sydney co-ordinated adaptive traffic system: a traffic responsive method of controlling urban traffic
24. SUMO (2014) TAPAS-Cologne dataset. http://sourceforge.net/apps/mediawiki/sumo/index.php?title=Data/Scenarios/TAPASCologne
25. Simpson EH (1949) Measurement of diversity. Nature 163:688

26. Varschen C, Wagner P (2006) Mikroskopische modellierung der personenverkehrsnachfrage auf basis von zeitverwendungstagebüchern. Stadt Reg Land 81:63–69
27. Boukerche A, Das SK (1997) Dynamic load balancing strategies for conservative parallel simulations. In: Proceedings of 11th workshop on parallel and distributed simulation, 1997, pp 20–28
28. Devine KD, Boman EG, Heaphy RT, Hendrickson BA, Teresco JD, Faik J et al (2005) New challenges in dynamic load balancing. Appl Numer Math 52:133–152

Part II
Modelling and Processing

A Situational Awareness Approach to Intelligent Vehicle Agents

Vincent Baines and Julian Padget

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 goals 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.

1 Introduction

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 [1] and more recently Nissan's [2]) as well as collectively in convoys [3], 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?

V. Baines (✉) · J. Padget
University of Bath, Bath, UK
e-mail: v.f.baines@bath.ac.uk

J. Padget
e-mail: j.a.padget@bath.ac.uk

M. Behrisch and M. Weber (eds.), *Modeling Mobility with Open Data*,
Lecture Notes in Mobility, DOI 10.1007/978-3-319-15024-6_6

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) [4] 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 [5] 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[1] [6] platform), a rich simulation environment (SUMO [7]), 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 "InstAL"[2] [8], 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 [9]), where vehicle trips for a section of the M25 motorway over a 15 min 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

[1] Jason agent platform, http://jason.sourceforge.net, accessed 19th July 2014.

[2] InstAL institution framework, http://www.cs.bath.ac.uk/instal/, accessed 8th September 2014.

real-world vehicle data to reproduce actual initial conditions, and (iv) capturing conventions and regulations in institutional models to provide guidance to vehicle agents.

2 Research Background

As discussed in Sect.1, this work attempts to adopt themes from Endsley's [5] 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) [10] 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)[3] [11] which forms the simulation backbone.

Earlier work [12] considered Hourizi's [13] 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 Sect. 3 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 [3] demonstrating the ability of vehicles to function as vehicle platoons. Continued announcements of increasingly sophisticated autonomous vehicles such as Google's car [1], the Volkswagen based 'MadeInGermany' [14] vehicle, Nissan's self-drive [2], etc., demonstrate the increasing maturity of real world vehicles suitable for this work. Whilst SUMO [7] 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 [15] claiming that controlling vehicle speed based on

[3] Bath Sensor Framework, https://code.google.com/p/bsf/, accessed 30th August 2014.

upcoming traffic lights could reduce CO_2 emissions by fifteen percent. Similarly, experimentation in vehicle to traffic light communication [16] 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. whether a flash of headlights is 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 [17], and is explored in a scenario considering the use of variable speed limits to alleviate congestion, while the need for multiple institutions [18] 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.

3 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 inappropriate for agent-based modelling because of the clearly higher computational requirements. Our approach here is a mixed strategy, in which we 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. We establish these scenarios by using data obtained from the UK Highways Agency to generate SUMO vehicle populations that reflect real-world traffic patterns observed on the M25 (a circular motorway around London, UK). This section provides a technical overview of the framework, as well as performance findings.

3.1 Technical Overview

These informal requirements have lead to the creation of a distributed environment called the Bath Sensor Framework (BSF) [11], whose primary features are: (i) a bus-like communications system based on the eXtended Messaging and Presence Protocol (XMPP) [19], 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 [20, 21]. 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.

A sketch of the framework appears in Fig. 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 [22]
- The **normative framework** component introduces the element of institutions [8] into the simulation, allowing obligations to be issued to simulation

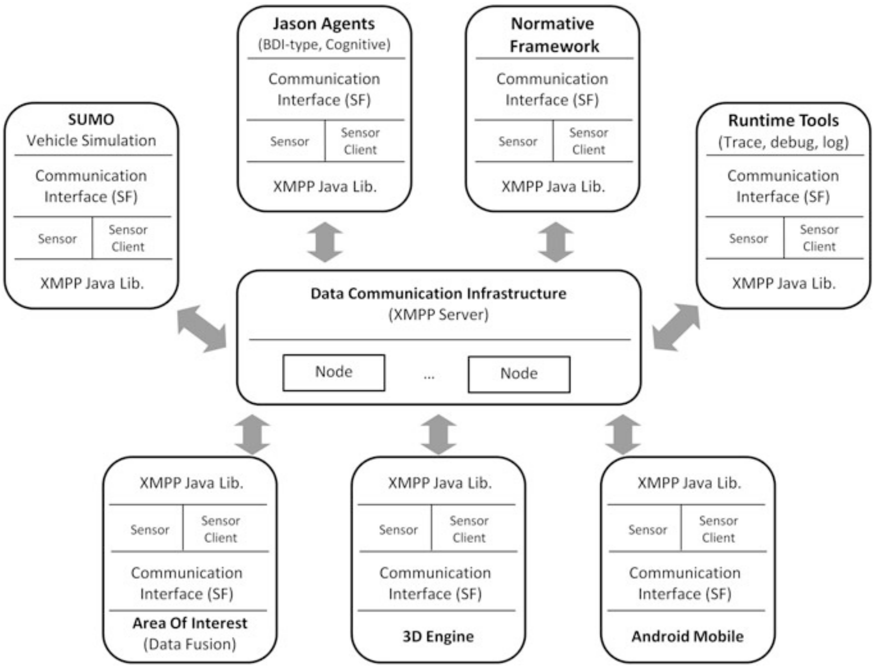


Fig. 1 Illustration of available BSF system components

participants (e.g. for a vehicle to slow down, or move out of the way), following the principles set out in [23] and developed for Jason in [24].

- 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 information from other systems (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 a 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 [25]. Some modifications 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 object data definition (e.g. metres per seconds, 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 [26], this would require SUMO to have a simulation step of 0.03 s 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 [12], 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.

3.2 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 reliable message exchange has led to a number of network tests being included as part of the build process (and reported back to a Jenkins[4] build server, accessed 27th August 2014) as otherwise simulations may appear to work but have totally unreliable results.

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 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

[4] Jenkins Continuous Integration Server, http://jenkins-ci.org, accessed August 30th 2014.

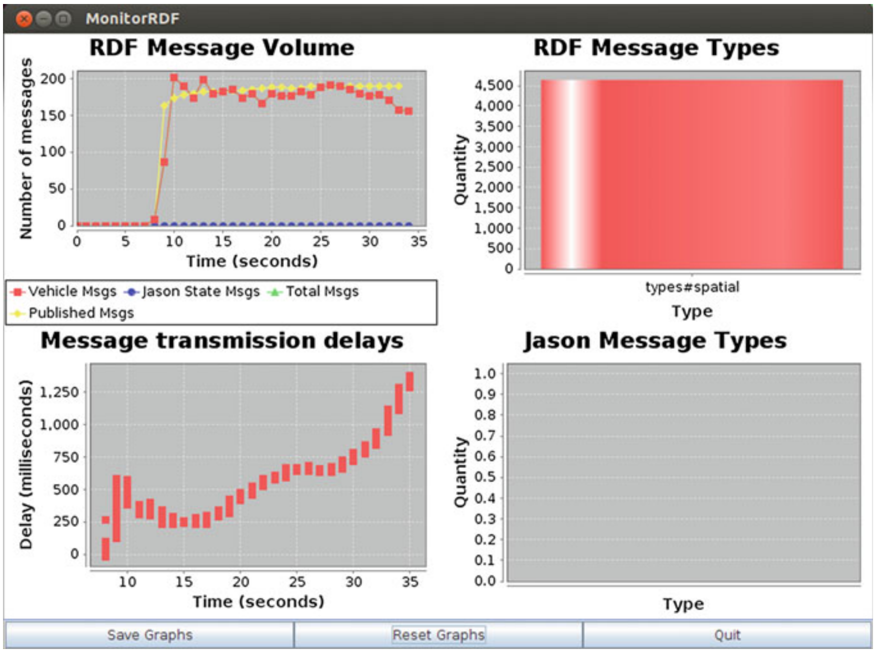


Fig. 2 The BSF rdfMonitor tool

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 at 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 the 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 effect the performance of the system, but as the BSF configurations used in these experiments so far have involved a low number of subscriptions, this has not been specifically investigated.

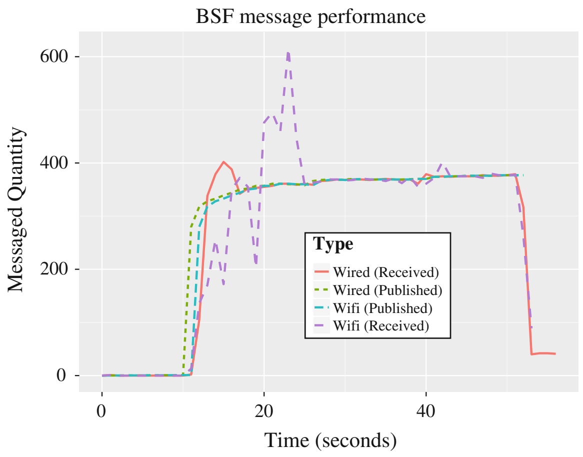


Fig. 3 Message delivery performance comparison between wired and wifi networks

The use of these tools helps establish an operating envelope for the intended configuration. In Fig. 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 [27] 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 real-time, as without timely and reliable message exchange, unexpected behaviour occurs. Previous work [12] 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.

4 Experimental Scenarios

This paper presents three 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 combinations[5] as well as re-use. Furthermore, the use of institutions offers a means for the coordination of multiple vehicles (e.g. to ease congestion), where individual drivers may be able to perceive the problem, but are unable to bring about a solution by individual action alone. The details of these three scenarios are now presented in more depth.

4.1 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).

[5] 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.

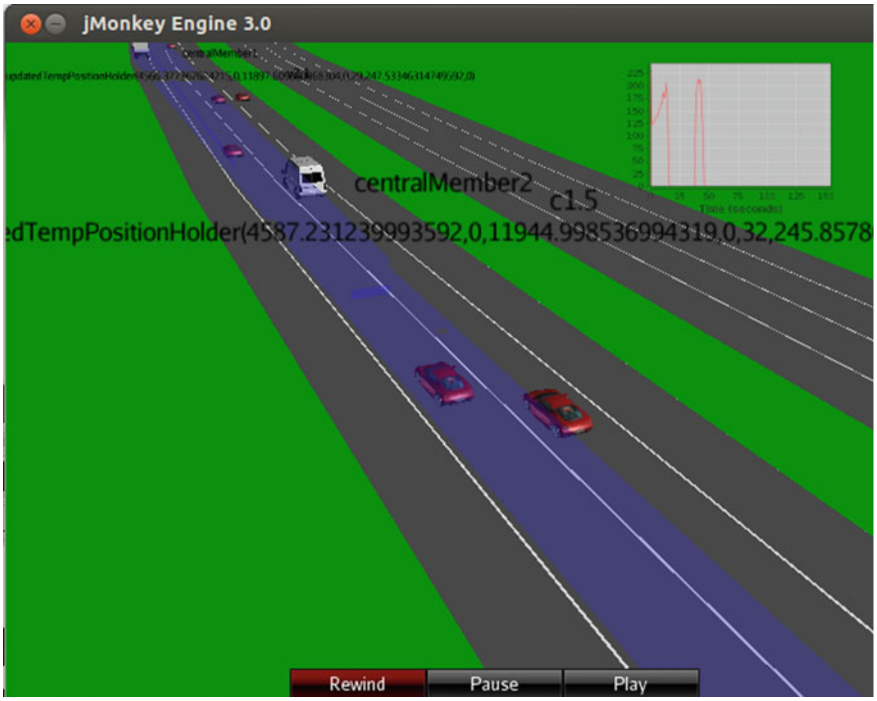


Fig. 4 M25 Scenario in 3D viewer

Figure 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 s into SUMO from Jason with speed of 29 m/s.
2. Vehicle V2 injected at 11 s into SUMO from Jason with speed of 30 m/s.
3. After 7 s, V2 increases speed to 32 m/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 `detectionInCol-lisionZone(Name, Distance)`.
6. If Distance is less than 45 m then agent plan `brakeHard` is triggered, if Distance greater than 45 m and less than 65 m then plan `flashLights` is triggered.

4.1.1 Lights Flash with No Institution

In this case, when approaching the vehicle, V2 flashes its lights at V1 but there is no response. V2 continues to gain on V1 until the distance is below 45 m. The `brakeHard` plan causes the vehicle to slow to 10 m/s until V1 is outside of the collision zone, after which it resumes the previous speed. This behaviour then repeats, as V2 begins to gain on V1 again, and will show the same `brakeHard` behaviour as a result.

4.1.2 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.

4.2 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 light interactions which are explored in this scenario, investigating the role institutions could play in managing vehicles’ speed in order to coordinate with traffic light states.

The context of this scenario is shown in Fig. 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.

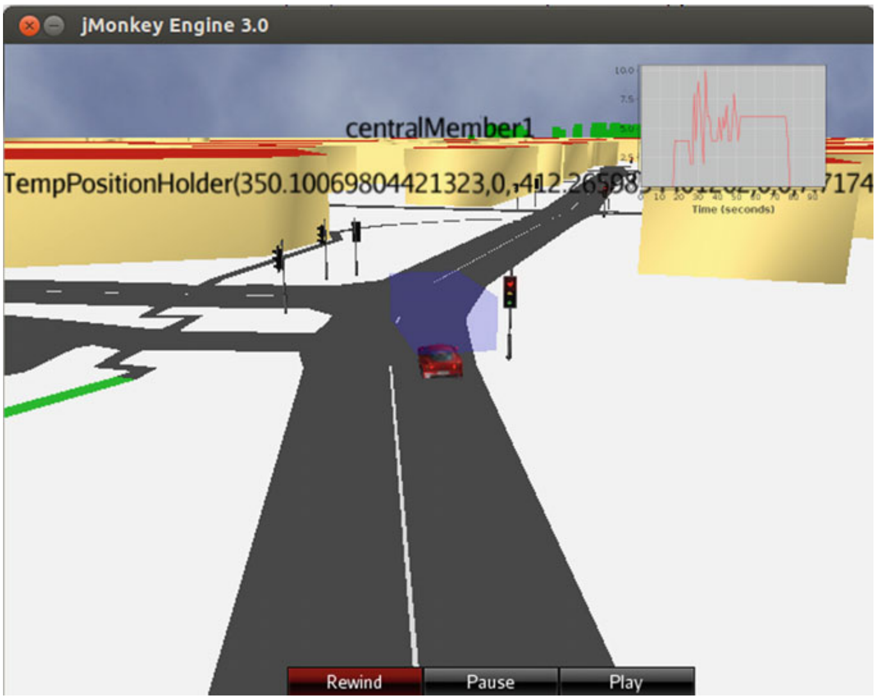


Fig. 5 City scenario in 3D viewer

4.2.1 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 s.
2. Vehicle progresses along route, stopping at red traffic lights.

4.2.2 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 s.
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 7 m/s for 35 s, after which speed control returns to SUMO.
7. Vehicle arrives at traffic lights after they have turned back to green and proceeds along route.

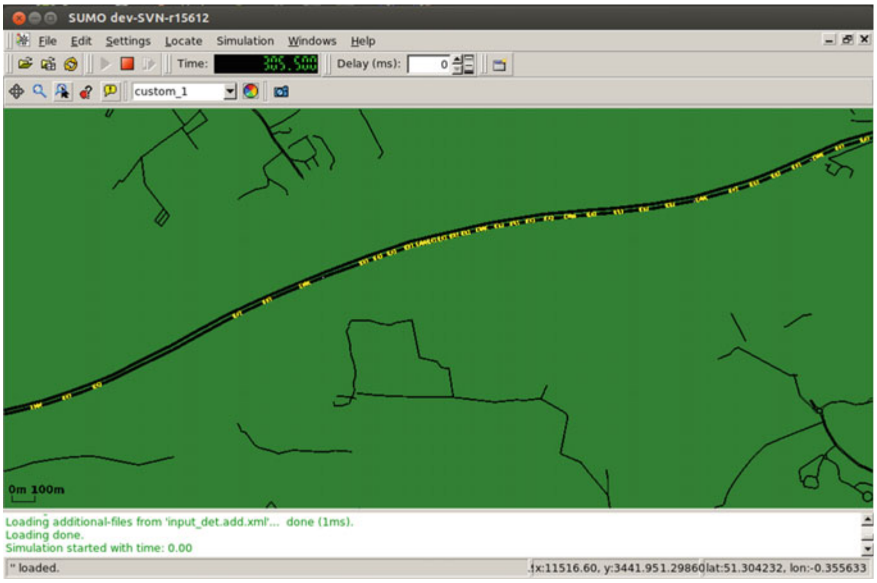
4.3 Scenario 3: Variable Speed Limits

Having demonstrated the application of institutions to resolving the ambiguity of a flash of headlights, and in managing vehicle speeds such that they arrive at traffic lights when they are in a green state, a third scenario is now presented, which focusses on the use of institutions targeted at the enforcement of variable speed limits. The motivation for deploying a Variable Speed Limit (VSL) is that there are a variety of situations where traffic flow can be improved by reducing the speed allowed, such as following an accident or lane closures due to maintenance. Speed has been put forward [28] as a direct link to the severity of a crash, and as a weaker link to the probability of a crash occurring. Three justifications are put forward [28] for imposing speed restrictions on an individual: (i) their misjudgement of impact on other individuals (e.g. cost, risk), (ii) that a driver may have insufficient information to judge an appropriate speed, and (iii) that a driver may be unable to judge the impact of their speed on the likelihood and severity of a crash. In [29] the use of VSL is attributed to reducing accidents by 25–50 %. From this, we conclude that imposing a VSL removes the element of the driver travelling at potentially unsafe speeds, and combined with the physical implications of driving slower (i.e. shorter braking distances, greater time to react) may not only reduce the likelihood of an accident, but improve the possibility of managing congestion as the incidence and impact of undesirable behaviour (excessive braking, harsh acceleration, late reactions) may be reduced. There are a number of road sections in the UK where such VSL areas have been established, with studies considering their effectiveness on UK motorways [30, 31], which observe some benefits (speed homogenization, and reduced variance in journey times) but also highlight a challenging problem, namely that of suppressing the shock waves induced by braking events. In this case, shock waves are the phenomenon of traffic slowing down for no apparent reason, but which is typically triggered by some event (such as a driver slowing down), that establishes a wave-like pattern of vehicles arriving at the congestion, slowing down, until they are able to accelerate back to their previous speed. The wave is maintained by new vehicles arriving at the back of the wave and slowing, while vehicles ahead of them travel through the congestion and eventually depart from the front of the wave.

In this scenario, the problem of shock waves is considered as a case where a disturbance occurs and triggers the formation of a wave-like pattern, that can be detected by vehicles being closely grouped together in terms of distance or speed. Variable Speed Limits are a mechanism that can potentially dampen this wave, in order to ease congestion and improve the flow rate of traffic. Details of a specific motorway control scheme are provided in earlier referenced work [31], where measured speed and flow levels are used to determine the congestion level, at which point signs can be activated along the route to display a new speed limit. We consider this as a *global* speed limit, as the speed limit applies to all vehicles travelling in that lane, up until a sign stating that the national speed limit applies is encountered. In this work [31] goes on to suggest that such an approach to the implementation of VSL may struggle to achieve the goal of suppressing such waves. In response, we put forward a number of observations. Firstly, there is the time lag between the wave triggering action (e.g. a vehicle suddenly braking hard) and action being taken to issue the VSL. Secondly, there is the problem that drivers are likely to wish to continue to travel as fast as possible, and so may travel above this (temporary, variable) speed limit, with some vehicles significantly in excess of the limit.

Thus, in addition to the question of the effectiveness of VSL, existing approaches are also quite coarse, as the speed limit is imposed over relatively long road sections, with the aim of reducing *all* vehicle speeds in each lane. In contrast, we propose to investigate the question of whether shock wave suppression can be achieved through tailored speed modifications for a smaller number of vehicles. Assuming the availability of V2V communication (as discussed in Sect. 1), the braking vehicle could communicate its action immediately, at which time vehicles likely to be affected could be advised to implement appropriate changes to their speed. To achieve this, we draw on the institution approach to issue appropriate obligations to vehicles that are potentially affected, with the intention of dampening the behaviour that triggers wave formation, before the wave becomes established, thus addressing the time-lag issue identified above.

The context of the scenario is shown in Fig. 6, where in a flow of motorway traffic, a braking vehicle causes disruption to the vehicles behind, which have to adjust their speed to avoid a collision, establishing a standing wave. Three variations of the scenario are considered: in the first no VSL is used and a shockwave should form, in the second a global VSL is set with the aim of dampening the shockwave, and in the third an institution issues obligations to specific vehicles to reduce their speed, with the same aim. The no-VSL scenario is used to test the assertion that a hard braking vehicle can cause a shockwave to form, and to support the development of metrics aimed at capturing what this event looks like. The global-VSL approach aims to replicate the existing approach to speed management in order to measure the impact this has on the shockwave itself but also on the overall vehicle population. The institutional-VSL approach seeks to assess the feasibility of targeting specific vehicles rather than the global population. If the no-VSL and global-VSL variations are show an impact on vehicle behaviour and congestion, then they provide a benchmark to compare the institution approach against.



```
Loading additional-files from 'input_det.add.xml'... done (1ms).
Loading done.
Simulation started with time: 0.00
" loaded.
x:11516.60, y:3441.951.29860lat:51.304232, lon:-0.355633
```

Fig. 6 Traffic congestion shockwave example

In all scenario variations, five Jason controlled vehicles are inserted: `jasonCar1` is the lead vehicle which performs the hard brake, `jasonCar2` is the vehicle immediately behind `jasonCar1`, and so on up to `jasonCar5`. With this experimental context established, we now present the specific events of each scenario.

4.3.1 No VSL

This case provides the baseline, where we seek to establish an undesirable congestion pattern of a shock-wave forming, which can be measured using the existing BSF monitoring tools described in Sect. 3. In this scenario, the following events occur:

1. The lead `jasonCar1` vehicle is inserted into the SUMO simulation at 35.5 s.
2. Following Jason vehicles (`jasonCar2` to `jasonCar5`) are inserted into the same lane at 40, 44.5, 49 and 53.5 s.
3. At 110 s, the `jasonCar1` vehicle brakes for a period of 6.5 s.
4. The following vehicles are forced to reduce their speed in order to avoid colliding with the vehicle ahead.
5. The scenario runs for 240 s, to allow observation of emerging shockwave pattern.

The hypothesis of this scenario variation is that after selecting appropriate parameters for the SUMO vehicles to represent bad driving behaviour (e.g. late reactions, over braking), that `jasonCar2` will over compensate for the hard brake of `jasonCar1`, with `jasonCar3` reacting in similar fashion, and so on, causing the wave to form. This will provide a comparison for findings from the other two scenarios, and support refinement of metrics to ensure this congestion pattern can be captured.

4.3.2 Global VSL

For this case, a similar initial set of events occurs, up to the point of the lead vehicle breaking, however following this event, a global VSL is established and all vehicles reduce speed. Variation in the timing of the imposition of the VSL is a matter for further exploration, but in this case, it is assumed that the braking behaviour can be detected and a VSL introduced immediately, with the following set of events occurring:

1. The lead `jasonCar1` vehicle is inserted into the SUMO simulation at 35.5 s.
2. Following Jason vehicles (`jasonCar2` to `jasonCar5`) are inserted into the same lane at 40, 44.5, 49 and 53.5 s.
3. At 110 s, the `jasonCar1` vehicle brakes for a period of 6.5 s.
4. Also at 110 s, a global VSL of 22 m/s is imposed on all lanes.
5. At this point all vehicles reduce their speed to the new limit.
6. The Jason vehicles (`jasonCar2` to `jasonCar5`) have to reduce their speed further so as to avoid colliding with the braking vehicle.
7. The scenario runs for 240 s, to allow observation of any shockwave pattern.

The hypothesis with this approach is that the global impact on all vehicle speeds should be observable in the captured metrics, though any impact on the shockwave is difficult to predict. However, if this approach is found to be capable of suppressing or removing the shockwave, then it provides a competitor to the institution VSL approach.

4.3.3 Institution VSL

In this version of the scenario, the braking vehicle broadcasts the fact it is performing an `emergencyBrake`, and the institution receives this message. Along with this message, the braking vehicle transmits details of the vehicles behind it (this could be calculated by the Area of Interest module described earlier in the technical overview, but is fixed in this experiment). Based on this information, the institution issues obligations to these vehicles, that if met result in appropriate speed reductions. In this scenario, the duration of the speed reduction has been set to 5 s, as no reasoning has been implemented as to when the vehicles should return to their previous speed. The aim of this short speed reduction is to have reduced the speed of these vehicles before they (over)react to a braking vehicle ahead. The set of events occurring in this scenario are as follows:

1. The lead `jasonCar1` vehicle is inserted into the SUMO simulation at 35.5 s.
2. Following Jason vehicles (`jasonCar2` to `jasonCar5`) are inserted into the same lane at 40, 44.5, 49 and 53.5 s.
3. At 110 s, the `jasonCar1` vehicle brakes for a period of 6.5 s.
4. At the same time, the lead vehicle transmits `emergencyBrake`, along with vehicle positions `vehiclePosition(jasonCar2, p1)`, `vehiclePosition(jasonCar3, p2)`, `vehiclePosition(jasonCar4, p3)`.
5. The institution issues obligations `slowDown(jasonCar2, slow)`, `slowDown(jasonCar3, mediumSlow)`, `slowDown(jasonCar4, medium)`.
6. Jason agents resolve these obligations according to their beliefs about `speedModifier(medium, 20)`, `speedModifier(mediumSlow, 12)`, `speedModifier(slow, 5)` and reduce their speed accordingly.
7. After 5 s at reduced speed, Jason vehicles `jasonCar2` to `jasonCar5` consider the `slowDown` obligation fulfilled and return to their previous speed.

The corresponding institution state change is shown in Fig. 7, with the initial state S_0, containing the values for `vehiclePosition`, and the `emergencyBrake(jasonCar1)` bringing about the new state S_1 where the `slowDown` obligations arise.

The hypothesis of this scenario is that the effect on an individual vehicle of the VSL should be clearly observable, but that any impact on the wider vehicle population may be harder to identify. Ideally, by managing the behaviour of the vehicles initially affected by the excessive braking behaviour, the shockwave may be prevented from occurring. However, this is optimistic, and instead if the mechanism is shown to have an effect, then future work could refine and tune the parameters (i.e. how many vehicles should slow down, to what speed, and for how long) of the VSL and vehicles it is applied to.

Having described the main features of and differences between the scenarios, we now present the experiment results.

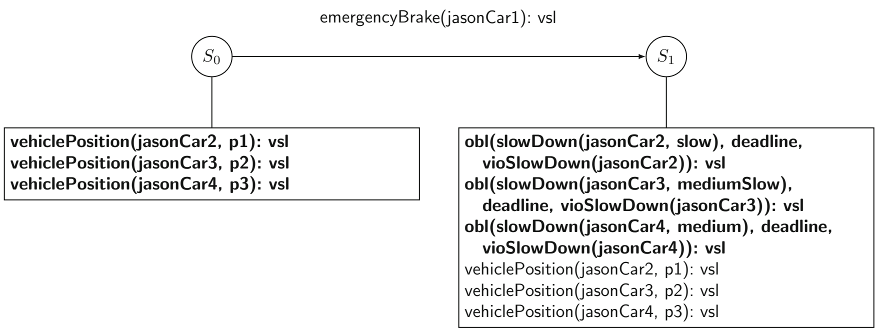

emergencyBrake(jasonCar1): vsl
S_0
S_1
vehiclePosition(jasonCar2, p1): vsl
vehiclePosition(jasonCar3, p2): vsl
vehiclePosition(jasonCar4, p3): vsl
obl(slowDown(jasonCar2, slow), deadline, vioSlowDown(jasonCar2)): vsl
obl(slowDown(jasonCar3, mediumSlow), deadline, vioSlowDown(jasonCar3)): vsl
obl(slowDown(jasonCar4, medium), deadline, vioSlowDown(jasonCar4)): vsl
vehiclePosition(jasonCar2, p1): vsl
vehiclePosition(jasonCar3, p2): vsl
vehiclePosition(jasonCar4, p3): vsl

Fig. 7 Institution state change during VSL scenario

5 Results

In this section, the simulation results for the motorway and traffic lights scenarios (as described in the previous Sect. 4) are presented.

5.1 Scenario 1: Motorway Change Lane Request

In this scenario, the objective is to measure the impact of the institution on maintaining average traffic speed and ameliorating congestion. To do so, we 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-kilometre 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.

The results of this scenario can be seen in Fig. 8, 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 500 m along the route, where with the institution active speeds remain at a reasonably constant 70 mph. With the institution not active, as outlined in Sect. 4 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 been seen that either side of this disruption (i.e. ahead by

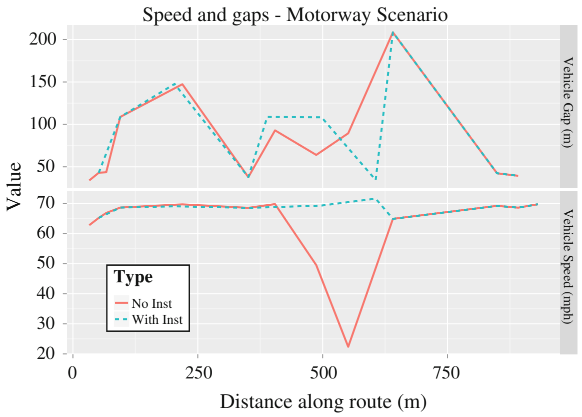


Fig. 8 Vehicle speed and gaps for change lane scenario

625 m or behind by 375 m) 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. The management of this of this kind of 'ripple effect' is precisely the kind of behaviour considered by the variable speed limit scenario presented in this work.

5.2 Scenario 2: City Traffic Lights

In this scenario, the objective is to measure the effect of the institution in managing vehicle arrival times at traffic lights. The key metric being used is the measurement of fuel consumption, based on the premise 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.

In Fig. 9 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, an obligation is issued 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

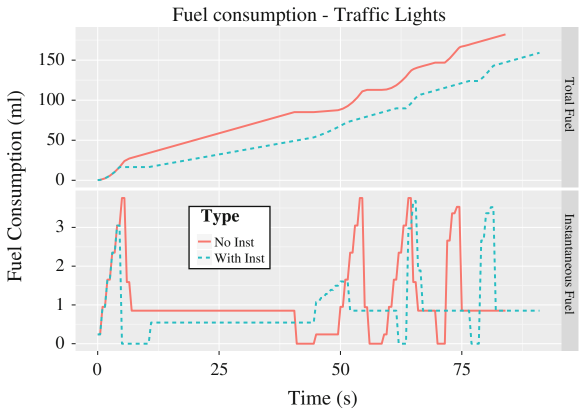


Fig. 9 Impact on fuel consumption

be an increase in journey time, but in fact the vehicle only loses approximately 10 s 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.

5.3 Scenario 3: Variable Speed Limits

In this scenario, the objective is to explore congestion shockwaves within SUMO, specifically how different approaches to establishing a Variable Speed Limit (VSL) can reduce such a wave. Whilst a human observer of the simulation GUI is able to spot when such congestion occurs (e.g. as shown earlier in Fig. 6), developing a metric to quantify such an occurrence proves challenging. The mean speed per lane or number of vehicles per lane were considered, but as lanes can be arbitrarily long and congestion may be localised to a small region, alternative approaches were sought, leading to the speed and gap analysis shown in Fig. 8. Analysis based on the gap-ahead technique did not show particularly useful results for this scenario, and so instead we focus on vehicle speeds. There is a drawback to this approach in that vehicle speeds can only be shown for a given point in the simulation time, and so any temporary state occurring just at that time (i.e. a vehicle braking because another vehicle changed lane) could appear to be more significant, despite it not being present in the next simulation time step. For this reason, we capture a set of vehicle speeds at 240 s into the simulation (the end of the simulation run) as well a set of speeds as at 200 s into the simulation. This allows for comparison of vehicle states at two different time points, in an effort to de-emphasize temporary effects and observe trends instead.

The outcome of this analysis is shown in Fig. 10 for three data series: with no VSL, with a global VSL and with an institution VSL, captured at 200 and 240 s during the scenario. These data series are presented from two different times, as it is difficult to show the vehicle speed of the entire traffic population changing over time. Comparing the results at these two time samples, there are some consistencies which can be identified. It can be seen in both cases that with no VSL there is a significant cluster of slow vehicles near the 3,750 m route distance mark. It can also be seen that before and after this distance, both the no VSL and institution VSL show vehicle speeds in the region of 80 mph, and that the global VSL shows speeds reduced to the region of 55 mph.

There are a number of interesting differences that may be observed in Fig. 10, concerning the behaviour of global and institutional VSL approaches. At the 200 s

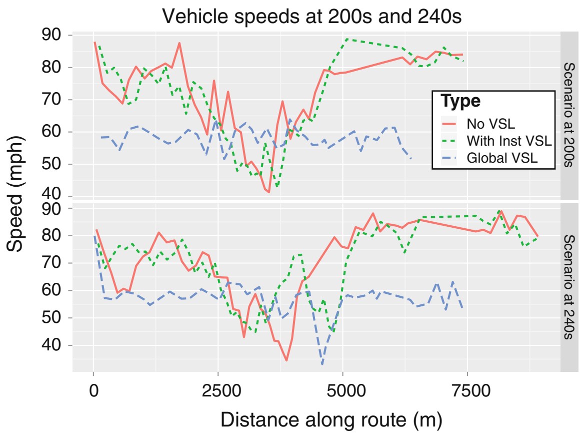


Fig. 10 Speed comparison at 200 versus 240 s

point, the institution VSL seems to have only slightly improved the situation, but 40 s later at the 240 s sample there is a significant improvement. The global VSL approach shows the opposite behaviour, where at 200 s all vehicles are travelling at approximately 60 mph, but at 240 s a cluster of slow moving vehicles is visible. This suggests an area where further work would be beneficial, and we would propose two areas specific to the results seen in Fig. 10. Firstly, that a more advanced metric, capable of analysing all vehicle speed changes over time is developed. The intention is this would show whether changes occur over a few time steps, or are persistent over a longer time period and indicate a more substantial congestion behaviour occurring. This could be implemented as a derivative of the existing speed metric, taking a number of samples over time and reporting the measured rate of change in speed. Secondly, that using such a metric, it should be possible to tune the scenario parameters (e.g. how long to apply a speed limit for, how many vehicles to apply it to, and so on).

Shifting focus from the entire vehicle population to the vehicles controlled by Jason agents, in Fig. 11 the speed behaviour over time is shown for five Jason vehicles. With a smaller set of vehicles, it is possible to view how their behaviour changes over time, and the distinct impact of the VSL approaches can be seen. With the no VSL approach, we can see the lead vehicle C1 brakes suddenly at 110 s, and vehicles C2 to C5 slowly reduce their speed as they gain on the vehicle in front. In the institution VSL variation, all vehicles can be seen to brake hard at 110 s, but after this vehicles C2 to C5 increase their speed again, before having to reduce it as they gain on the vehicle ahead. This suggests that the institution `slowDown` obligation was not of a sufficient duration. Finally, in the global VSL variation it can be seen that all vehicles decrease their speed at 110 s, with vehicles C2 to C5 travelling at the new maximum speed for 10–20 s, before a gradual slow down as they gain on the vehicle ahead.

There is also some variation visible in Fig. 11 during the 150–200 s period, where both the global VSL and no VSL variants of the scenario settle to the

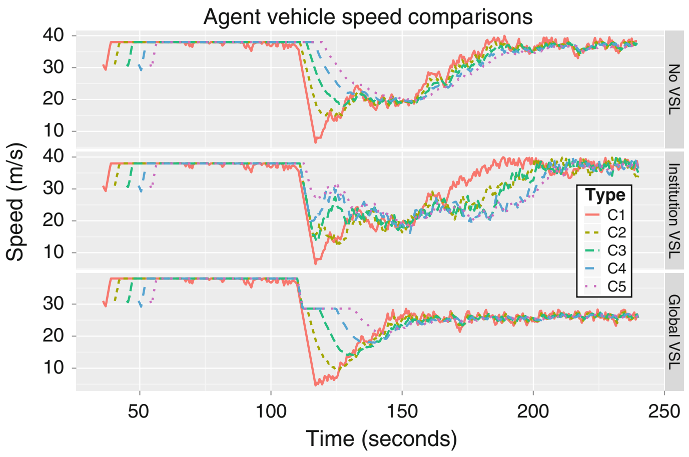


Fig. 11 Speed comparison of intelligent agent vehicles

permitted speed, while in the institution variation vehicles C2–C5 remain slower than vehicle C1 for approximately 50 s. This is an area requiring further investigation, as the institution obligation has expired after the 5 s period specified in the scenario description (see Sect. 4.3), yet vehicle behaviour is still disrupted, and shows different behaviour to the no VSL and global VSL variants of the scenario. One potential development would be to augment the 3D-viewer tool, following on from Sect. 3 where it was described that the 3D representation of vehicles could be augmented with additional vehicle state information, to extract data from SUMO in order to supplement this view with additional explanatory messages (e.g. reason why braking, distance to vehicle ahead). It may also be that the vehicle parameters regarding driver behaviour (i.e. reaction time, acceleration and deceleration, minimum gap to vehicle head) need improvement, as the vehicles may be over-sensitive to road conditions, which although not manifesting itself in the no VSL and global VSL variations, appears in the institution VSL approach. It could be that the institution approach of controlling five vehicles (to prevent the shockwave) causes the gaps between these vehicles to be small during the institution obligation (permissible as they are travelling at slower speeds) but when the obligation expires and they resume normal speed that they struggle to balance increasing speed with the requirements of maintaining a safe gap to the vehicle ahead. Whilst this would need further analysis to establish the necessary evidence, it offers a further role of the institution, that perhaps these vehicles could ignore their braking distance while under control of an institution, because the coordination is being performed by that external body, and that the institution should govern not just their `slowDown` behaviour, but also their return-to-normal behaviour.

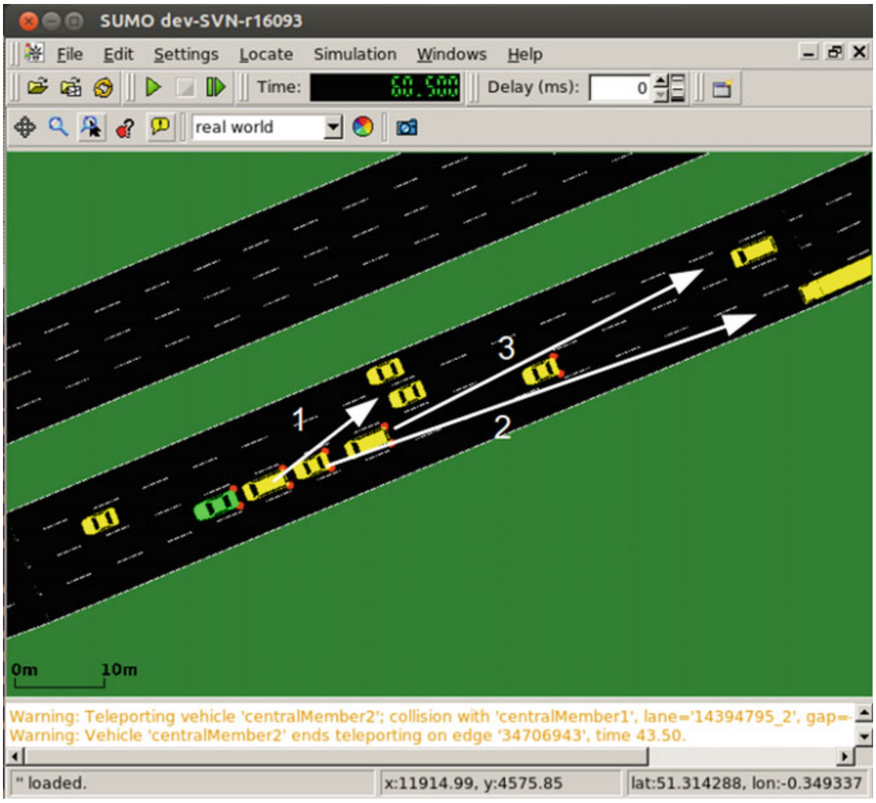
6 Discussion and Future Work

The work presented here demonstrates the use of an open source solution combined with realistic traffic data, real world metrics and a sophisticated architecture, in modelling a number of vehicle scenarios. Three scenarios are presented, in the first a city scenario is used to investigate the effect of an institution model in regulating arrival times at traffic lights in order to improve fuel consumption. In the second, a motorway based scenario is used to explore the application of an institution that issues obligations to move out of the way in order to prevent excessive braking and acceleration of following vehicles. In the third scenario, a motorway congestion scenario is presented, where three approaches to variable speed limits are assessed: no speed limit, a global speed limit, and an institutionally determined speed limit targeted at specific 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 metric 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) and the effect of speed limits across the entire motorway section. Alternative metrics are also desirable, to provide better observability of changes over time.

There are a number of areas of future work for the variable speed limit scenario to be developed. In the version presented, parameters such as how quickly any VSL should be established, how long it should last for, and how much the speed should be decreased to, have all been chosen somewhat arbitrarily. However, as the experiment is simulation based, it would be feasible to repeat over multiple iterations and tune these parameters, in order to develop improved handling of the shockwave congestion. However, further work should also take place in validating vehicle parameters concerning driver behaviour and physical vehicle performance, in order to improve the credibility of any findings. Finally, enhanced metrics would provide a suitable feedback signal for any iterative tuning of the scenario, and also improve the ability to analyse the outcome beyond vehicle gaps and speeds.

Having demonstrated potential benefits in the combination of the SA approach and institutional governance in the scenarios presented, future work is planned to develop a larger, multiple institution scenario. One such example under consideration is shown in Fig. 12, 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 potential 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.



```
Warning: Teleporting vehicle 'centralMember2'; collision with 'centralMember1', lane='14394795_2', gap=-
Warning: Vehicle 'centralMember2' ends teleporting on edge '34706943', time 43.50.
```

" loaded. x:11914.99, y:4575.85 lat:51.314288, lon:-0.349337

Fig. 12 Incident management example

The work we have presented demonstrates the application of institutional models as a 'late-binding' regulation mechanism, providing directed control for the achievement of social objectives, in the domain of intelligent vehicles. Information exchange between such vehicles has been considered in terms of Situational Awareness, to allow intelligent vehicles to communicate a rich set of semantically-annotated RDF messages. All experimentation has been based on open source data and software, so this approach to socio-cognitive intelligent vehicles can readily be explored in further scenarios and traffic contexts.

References

1. Markoff J (2010) Google cars drive themselves, in traffic. http://www.nytimes.com/2010/10/10/science/10google.html. Accessed 8 Oct 2011
2. Nikki Gordon-Bloomfield (2013) Nissan takes Japanese PM on autonomous LEAF test drive. http://transportevolved.com/2013/11/11/nissan-takes-japanese-pm-on-autonomous-leaf-test-drive/. Accessed 19 Jan 2014

3. Bergenhem C, Huang Q, Benmimoun A, Robinson T (2010) Challenges of platooning on public motorways. In: 17th world congress on intelligent transport systems
4. Naylor N (2014) 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. Accessed Feb 2014
5. Endsley MR (1995) Toward a theory of situation awareness in dynamic systems. Hum Factors: J Hum Factors Ergon Soc 37(1):32–64
6. Bordini RH, Hübner JF, Wooldridge M (2007) Programming multi-agent systems in AgentSpeak using Jason. Wiley, Hoboken
7. Krajzewicz D, Erdmann J, Behrisch M, Bieker L (2012) Recent development and applications of SUMO—Simulation of Urban MObility. Int J Adv Syst Meas 5(3 and 4):128–138
8. Cliffe O, De Vos M, Padget J (2006) Answer set programming for representing and reasoning about virtual institutions. In: Inoue K, Satoh K, Toni F (eds) CLIMA VII. Lecture notes in computer science, vol 4371, pp 60–79. Springer
9. UK Highways Agency (2014) Traffic flow database system. https://trads.hatris.co.uk. Accessed 26 Jan 2014
10. Bratman ME, Israel DJ, Pollack ME (1988) Plans and resource-bounded practical reasoning. Comput Intell 4:349–355
11. Lee JH, Baines V, Padget J (2012) Decoupling cognitive agents and virtual environments. In: Dignum F, Brom C, Hindriks KV, Beer MD, Richards D (eds) CAVE. Lecture notes in computer science, vol 7764, pp 17–36. Springer
12. Baines V, Padget J (2012) Communication and metrics in agent convoy organization. In: 7th international workshop on agents in traffic and transportation (ATT 2012 at AAMAS 2012), pp 69–77, June 2012
13. Hourizi R (1999) Awareness beyond mode error. Ph.D. thesis, University of Bath
14. Freie Universitat Berlin (2011) Autonomous car navigates the streets of Berlin. http://autonomos.inf.fu-berlin.de/news/press-release-92011. Accessed 8 Oct 2011
15. Volkswagen (2013) 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. Accessed 19 Jan 2014
16. Kim KT (2012) STVC: secure traffic-light to vehicle communication. In: ICUMT, pp 96–104. IEEE
17. Balke T (2011) Towards the governance of open distributed systems: a case study in wireless mobile grids. Ph.D. thesis, University of Bayreuth, September 2011
18. Cliffe O, De Vos M, Padget JA (2007) Specifying and reasoning about multiple institutions. In: Coordination, organizations, institutions, and norms in agent systems. Lecture notes in artificial intelligence, Springer
19. XMPP Standards Foundation (2014) The XMPP standards foundation homepage. http://www.xmpp.org, 20130129, no date
20. Stout L, Murphy MA, Goasguen S (2009) 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, pp 11:1–11:6. New York, NY, USA. ACM
21. Wagener J, Spjuth O, Willighagen E, Wikberg J (2009) XMPP for cloud computing in bioinformatics supporting discovery and invocation of asynchronous web services. BMC Bioinform 10(1):279
22. Lee JH, Li T, Padget J (2013) Towards polite virtual agents using social reasoning techniques. Comput Anim Virtual Worlds 24(3–4):335–343
23. Alechina N, Dastani M, Logan B (2012) Programming norm-aware agents. In: Proceedings of the 11th international conference on autonomous agents and multiagent systems, AAMAS'12, vol 2, pp 1057–1064. Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems

24. Lee JH, Padget J, Logan B, Alechina N, Dybalova D (2014) Run-time norm compliance in BDI agents. In: International conference on autonomous agents and multi-agent systems, AAMAS'14, Paris, France, May 2014. IFAAMAS (to appear)
25. Tobias Knerr (2013) Merging elevation raster data and OpenStreetMap vectors for 3D rendering. Master's thesis, University of Passau, May 2013
26. Claypool KT, Claypool M (2007) On frame rate and player performance in first person shooter games. Multimedia Syst 13(1):3–17
27. Smith E (2013) JVM-Serializers. https://github.com/eishay/jvm-serializers/wiki. Accessed 2 Feb 2014
28. National Research Council (U.S.) (1998) Transportation Research Board. Committee for Guidance on Setting and Enforcing Speed Limits. Managing speed: review of current practice for setting and enforcing speed limits. Number no. 254 in managing speed. Transportation Research Board, National Research Council, National Academy Press
29. Coleman JA, Paniati JF, Cotton RD, Parker MR Jr, Covey R, Pena HE Jr, Graham D, Robinson ML, MaClauley J, Taylor WC et al (1996) Fhwa study tour for speed management and enforcement technology. US Department of Transportation, Washington DC
30. Papageorgiou M, Kosmatopoulos E, Papamichail I (2008) Effects of variable speed limits on motorway traffic flow. Transp Res Rec 2047(1):37–48
31. Tafti MF (2008) An investigation on the approaches and methods used for variable speed limit control. In: 15th world congress on intelligent transport systems and ITS America's 2008 annual meeting

SUMO's Lane-Changing Model

Jakob Erdmann

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

1 Introduction

The SUMO application suite [1, 2] provides tools for the ***S**imulation **O**f **U**rban **MO**bility*. 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

J. Erdmann (✉)
German Aerospace Center, Berlin, Germany
e-mail: jakob.erdmann@dlr.de

M. Behrisch and M. Weber (eds.), *Modeling Mobility with Open Data*,
Lecture Notes in Mobility, DOI 10.1007/978-3-319-15024-6_7

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 Sect. 10.

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:

(1) Strategic change
(2) Cooperative change
(3) Tactical change
(4) Obligatory change

After discussing the general architecture of lane-changing within the simulation in Sect. 3, the handling of these four motivations will be discussed in detail in Sects. 4–6. 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]. In Sect. 7 external control of the lane-changing model via the TraCI interface [1] is discussed. Section 8 gives a detailed account of the way conflicting lane-change motivations are resolved. Section 9 documents simulation results of the lane changing model in comparison to older models and Sect.10 gives an outlook on further developments.

This paper is an extension of [4]. It gives a more detailed account of the decision trees for effecting speed adjustments (Sect. 3.3) and covers changes which were made since that publication. One major change is the handling of obligatory lane changes discussed in Sect. 7. Furthermore, it contains additional evaluation results.

2 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.

SUMO simulates the movement of vehicles along the aforementioned lanes. In the context of this work the term **vehicle** refers to the model of a real-world vehicle and its driver (sometimes called vehicle-driver unit). 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 Sect. 3.1. 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 to 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. By default, lane changes are instant.[1] A vehicle is situated completely on the original lane in one simulation step and in the next simulation step it is situated completely on the target lane.

[1] There exists the simulation option—**lanechange.duration** which enables continuous lane change maneuvers. The vehicle occupies both lanes for a part of the given duration depending on its width. This functionality is less mature than the default.

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 Sect. 9 we explain how conflicts between these motivations are resolved.

3 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.

3.1 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**).

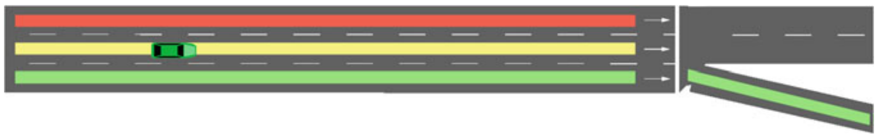


Fig. 1 The ego vehicle (*green*) needs to move to the bottom lane (with index 0) in order continue on its desired route (*green*). This lane has a bestLaneOffset of 0. The *yellow lane* (with index 1) has a bestLaneOffset of—1 indicating a necessary lane change to the right. The *red lane* (with index 2) has a bestLaneOffset of—2 and is strategically unadvisable

(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 without changing lanes and minimizing the number of necessary lane changes). Nevertheless multiple bugs were resolved in this part of the code base during the work on improving the lane-changing model in SUMO. Figure 1 illustrates the bestLaneOffset at a motorway off-ramp.

3.2 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 bestLaneOffset < 0 and changing right if bestLaneOffset > 0) correlates with the following factors:

(a) remaining distance to the dead-end (negative correlation)
(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 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:

$$d - o < lookAheadSpeed * abs(bestLaneOffset) * f$$

where the 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. The expected number of seconds until reaching the end of the dead lane is divided by the number of necessary lane changes (bestLaneOffset) to obtain the available time for the current lane change (**remainingSeconds**). This value 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 Sect. 10.

3.3 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 be 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 the **plannedSpeed** of the ego vehicle, blocker speed, gaps and remainingSeconds:

(1) **Leader is blocking**

   a. **able to overtake leader:** request leader to refrain from speeding up, prevent ego dawdling, (prevent overtaking on the right where forbidden by law)
   b. **unable to overtake leader**

      i. slow down to stay behind the leader
      ii. keep speed since the leader is faster anyway

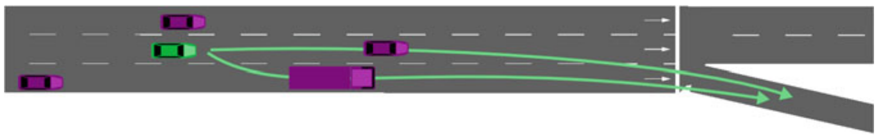


Fig. 2 The ego vehicle (*green*) needs to change to the *right* to continue with its route. It could either change behind the truck or it could overtake the truck and change in front of it

(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 decision whether a blocking leader may be overtaken (1a vs. 1b) is illustrated in Fig. 2. The choice is made by checking a list of necessary conditions for overtaking:

a. The ego vehicle is faster than the blocker (*dv = plannedSpeed-blocker speed > 0*). The plannedSpeed incorporates speed requests by surrounding vehicles.
b. The blocking vehicle is to the left of ego or overtaking on the right is allowed (in urban situations, on congested motorways or if the simulation option—*lane-change.overtake-right* is set)
c. The remaining space to the end of the dead lane is sufficient for overtaking
d. The time remainingSeconds is sufficient to overtake the leader at the current speed difference *dv*

The above decision tree results in an updated value of **plannedSpeed** with regard to a blocking leader. Another decision tree is used to compute the behavior in regard to a blocking follower. This decision tree considers plannedSpeed, blocker speed, gaps and remaining seconds.

(4) **Follower is blocking**

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

  c. **follower cannot overtake on the right:** request follower to slow down if above a minimum speed threshold

(5) **Follower is not blocking:** request follower to maintain speed so as to remain non-blocking
(6) **There is no follower:** drive with the maximum safe speed

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 27 m/s).

3.4 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 (refered to as **counterLane-Change**) 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. Figure 3 illustrates situations which may lead to a deadlock.

To prevent 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

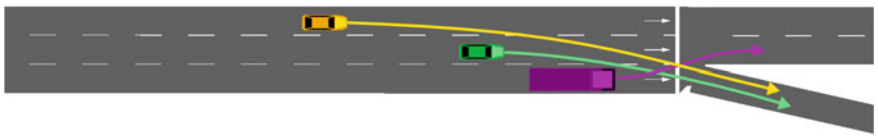


Fig. 3 A deadlock may arise between the *purple vehicle* which needs to change *left* and the *green vehicle* which needs to change *right* to continue with its route. Another deadlock may arise between the *purple vehicle* and the *yellow vehicle* which needs to change to the *right* twice

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.

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. Currently, additional space of 20 m is reserved for vehicles which need to change to the right by more than one lane and 40 m for vehicles which need to change to the left by more than one lane. 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.

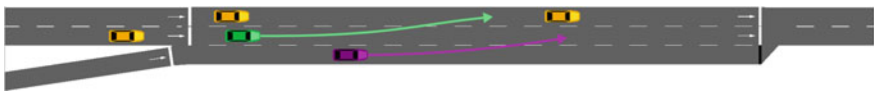
An important aspect of preventing stopping at a dead lane (and thus deadlocks) is avoiding detrimental lane changes. Generally speaking, the fewer lane change maneuvers vehicles have to perform, the less chance they have to become stuck. One change that was found to be quite beneficial was the avoidance 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. 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.

4 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 may have a beneficial impact on traffic flow 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. 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).



5 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. Figure 4 show a situation in which tactical lane changing may take place.

This part of the model is left unchanged from the old model [6]. Each vehicle maintains a signed variable **speedGainProbability** which by its sign indicates the beneficial change direction (-1 for right, 1 for left) and by its magnitude the expected benefit. If the magnitude exceeds a threshold value, a tactical lane change is attempted. If the lane change succeeds the value is reset to 0. The accumulation over multiple time steps prevents oscillations. During each simulation step and for each considered change direction d, the potential gain $g = (v - u)/v$ is computed. If g is positive, the variable speedGainProbability is incremented by $d * g$. If g is nonpositive and speedGainProbability has the same sign as d, it is halved instead.

Under some circumstances, overtaking another vehicle on its right side is forbidden. To overtake in this case, the ego vehicle needs to change lanes to the left

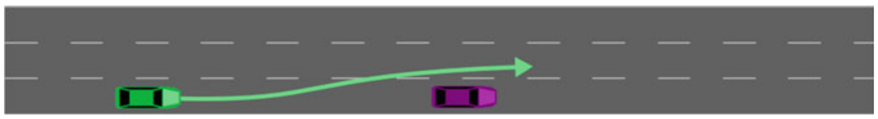


Fig. 4 The ego vehicle (*green*) is faster than its leader vehicle (*purple*). To prevent slow down, the ego vehicle may change to the *left* in order to overtake its leader

and wait for the slower leader to move to the right itself (effectively swapping lanes). If there are more than two lanes available, to ego vehicle may also change to the left twice in order to overtake the leader on its left side. As one of the additions of the new car following model, this behavior is now triggered if all of the following conditions are met:

- The option—**lanechange.overtake-right** is not set
- The ego vehicle is driving at a speed of 60 km/h or above
- There is a leader vehicle on the adjacent lane to the left
- The leader is slower than the ego vehicle
- The ego vehicle would need to slow down if it were to follow the leader vehicle (on the same lane)

In this case the ego vehicle will slow to the safe following speed and receive an impulse to change to the right by incrementing speedGainProbability by a fixed amount (enough to exceed the threshold if the conditions hold for three simulation steps). It should be noted that the 60 km/h check in the above conditions stands as a rough approximation for a far more complex set of rules which are dependent on legal rulings in the applicable country to be modelled [7]. For Germany, the value of 60 km/h corresponds not to a text of law but to a legal precedent which concretizes the wording of "slow speed".

6 Obligatory Lane Changing

In jurisdictions with right-handed driving, the left lane(s) are designated as overtaking lanes. Drivers are under the obligation to clear that lane whenever they do not use it for an overtaking maneuver. The obligation 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 Sect. 5 which is optional, the behavior described in this section is mandated by traffic laws [8]. Figure 5 illustrates a situation which calls for obligatory lane changing.

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 is exceeded (negative values are used in allusion to the variable **speedGainProbability**). The formula for computing the new value

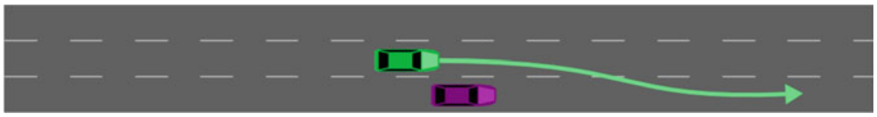


Fig. 5 The ego vehicle (*green*) is about to overtake. After finishing the maneuver, it is required to move back onto the *rightmost* unobstructed lane

of keepRightProbability q' from the old value q is designed for clearing the overtaking lane as soon as possible while at the same time avoiding oscillations from repeated cycles of overtaking on the left and changing back to the right lane.

$$q' = q - T\frac{t \times m}{d \times v}$$

Here, t denotes the expected time which for which the ego vehicle will be able to drive on the right lane with full speed (based on the distance to the leader vehicle and its speed). The value of m denotes the speed limit on the target lane while d denotes the desired speed of ego and v its current speed. T is a constant for tuning the urgency of obligatory changing.

7 Remote Controlled Lane Changing (TraCI)

Running SUMO simulations can be controlled by external programs using an interface called TraCI (**Tra**ffic **C**ontrol **I**nterface). Among the things that can be controlled is the choice of lane among the available lanes for each vehicle. External requests to change to a target lane or keep the current lane must be integrated with the "internal" 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.

For each of four change motivations discussed above the following options can be independently configured:

(a) Ignore internal request
(b) Ignore internal request when in conflict by an external request
(c) Always follow internal request regardless of external request

Furthermore, the following options for configuring the urgency of external requests:

(a) Following request regardless of surrounding vehicles, perform urgent speed adaptions
(b) Follow request unless it would cause an immediate collision but ignore safety gaps to surrounding vehicles, perform urgent speed adaptions
(c) Only change if all safety constraints are met, perform urgent speed adaptions
(d) Only change if all safety constraints are met, perform no speed adaptions

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.

8 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 the resulting change in lane index.

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

9 Improvements Over the Earlier Model

In the following we present measurements which document the effect of model changes on traffic flow and lane changing efficiency. Section 9.2 presents additional measurements which were undertaken since [4] when working on the model for obligatory lane changing.

9.1 Efficiency of Lane Changing

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 0.1 m/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 of Braunschweig (Brunswick) and the surrounding area with sections of motorway. The scenario spans one day and contains 650,000 vehicle movements. The threshold time for teleporting was set to 120 s.

The scenario *A92* consists of a motorway section in southern Germany with a length of 20 km. It contains 63,000 vehicle movements over the course of 1 day. The threshold time for teleporting was set to 300 s.

The scenario *ACOSTA* is comprised of a section of the Italian city of Bologna and contains 9,000 vehicles over the course of 1 h. It is notable for containing a 2-lane roundabout. The threshold time for teleporting was set to 300 s.

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

Compared to the old lane-changing model described in [6], 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

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

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

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 (in contrast to simulation with the old model). Also, the *Braunschweig* scenario exhibited deadlocks and jamming with a frequency that was utterly implausible for the demand model of a normal working day when simulation with the old model. 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.

9.2 Lane Usage

In the old model, obligatory changing was conflated with tactical lane changing by giving an additional decrement to speedGainProbability whenever the right lane allowed a high enough speed (for the next simulation step). This way, the decision for performing an obligatory lane change was strongly based on accumulated knowledge about the past rather than anticipated behavior in the future. The noticeable disadvantage of this design was the high delay between passing a slower vehicle and pulling back into the rightmost lane which had a detrimental effect on average road speed. The new model is designed to change to the right shortly after overtaking whenever the desired speed can expectedly be maintained on the right lane for a sufficient time.

To investigate lane usage properties a simulation scenario based on measurements from the German motorway A3 was set up. The input data consists of single vehicle detections from a detector cross-section with 3 lanes measured over a whole day.[2] The motorway experiences an average flow of 1,800 vehicles per hour with a peak flow of 3,500 vehicles per hour. The data is described in detail in [9]. Each data point contains a high-resolution time stamp, as well as the speed and the length of the vehicle. This data was fed into a simulated 3 lane motorway with a length of 30 km. The simulated vehicles were inserted with normally distributed values for *tau* and *sigma* by generating an individual *vType* element for each vehicle. The maximum speed, departure time and vehicle length were taken from the real-world measurements.

Simulated detectors were placed with a spacing of 1 km to accumulate the absolute number of vehicles passing on each lane as well as the average speed.

[2] 1995, 11th of March, a Saturday.

The investigation assumes that vehicles at the measurement location are in an equilibrium state in regard to their speeds and lane usage. Under that assumption, the absolute count of vehicles on each lane should remain nearly constant over the length of the motorway. Likewise, the average speed on each lane should remain constant over the length of the motorway.

Figure 6 shows the evolution of lane usage for different versions of the lane-changing model. It can be seen that older versions of the lane changing model deviate much stronger from the equilibrium assumption whereas newer versions deviate less. Specifically, the old model DK2008 placed excessive weight on

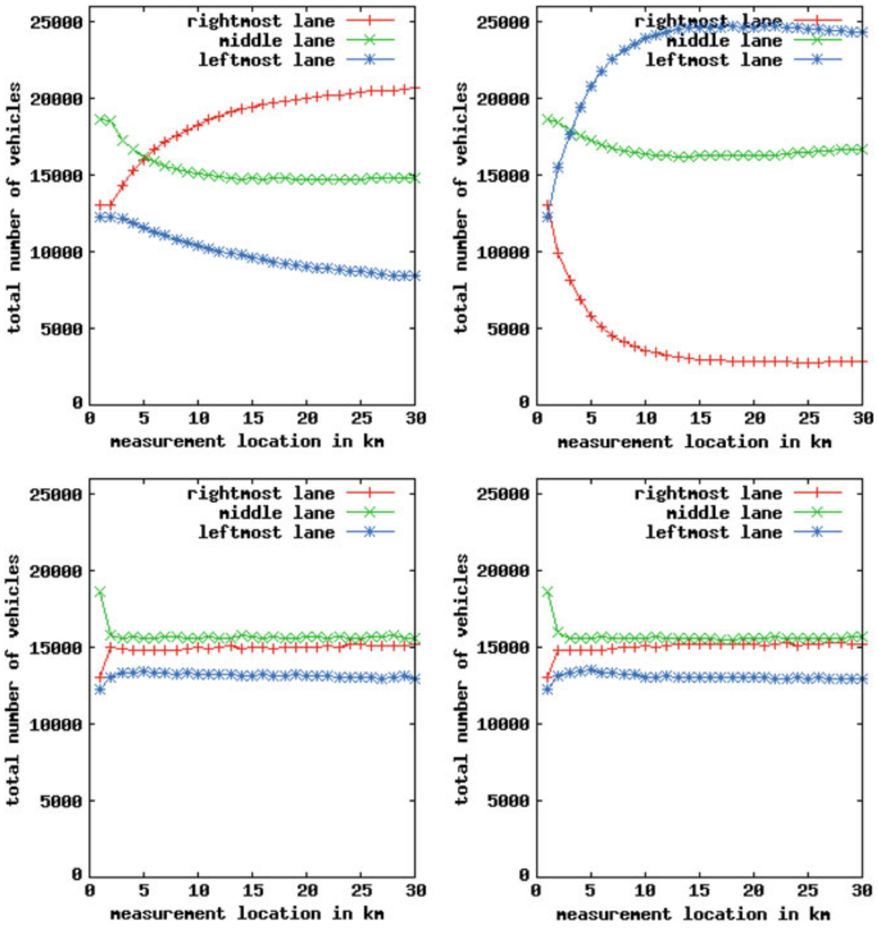


Fig. 6 Lane usage measurements for different versions of the lane changing model. *Top left* DK2008 sumo 0.18.0, *top right* JE2013 sumo 0.19.0, *bottom left* JE2013 sumo 0.20.0, *bottom right* JE2013 sumo 0.22.0. Version 0.21.0 is not shown due to being the same as 0.20.0 (The lane changing model slated for the upcoming release version 0.22.0 is already available as source code revision 17,102)

Table 2 Lane changing statistics for different model versions

Model	SUMO version	Lane changes per vehicle and km
DK2008	0.18.0	0.03
JE2013	0.19.0	0.04
JE2013	0.20.0	0.45
JE2013	0.21.0	0.45
JE2013	0.22.0[1]	0.26

[1] Upcoming release. Source code revision 17,102

obligatory lane changing whereas the intermediate version of the JE2013 model in revision 0.19.0 had no obligatory changing at all due to an implementation bug.

In addition to the lane usage and speeds, the average number of lane changes per vehicle and driven kilometer was measured for different lane change models. A survey by Lee et al. measured an average number of **0.26** lane changes per kilometer and vehicle while driving on American highways and interstates [10].[3] Table 2 gives the corresponding value for different model versions. A large jump can be seen from sumo version 0.19.0–0.20.0 when the new formula for effecting obligatory lane changes was introduced. However, due to a bug that was causing oscillations between tactical and obligatory lane changing[4] the number of lane changes was higher than intended. This bug is fixed in the upcoming version 0.22.0 resulting in a reduced number of lane changes. Overall an improvement in realism can be posited based on the measured values.

10 Outlook

The focus of the recent improvements of the lane-changing model was on deadlock-prevention and success rate of lane-changing as well as lane usage. The next goal is to perform calibration in order to reproduce lane-changing frequency.

To perform model calibration, several hard-coded model parameters shall be exposed to the end user. 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

[3] Lee et al. report 0.36 lane change maneuvers per mile or 0.22 per km with 16 % of the maneuvers consisting of more than one lane change.

[4] Vehicles were changing to the left in anticipation of overtaking a slower vehicle which was still very far away and shortly afterwards changing back to the right by obligation.

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

The type of deadlock discussed in Sect. 3.4 could be considered as a network modelling error. In reality, vehicles have the option of moving "diagonally" from their current lane to the target lane on a subsequent edge without requiring free space for their full length on the target lane. Typical traffic regulations would prohibit this kind of maneuver except for preventing deadlock. These connections are currently not modelled because SUMO currently cannot handle multiple connections from the same edge to the same target lane. It may be necessary to extend the network model to achieve more realistic traffic flow in deadlock-prone scenarios.

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 are used in reality. The fact that cooperative lane changes towards the blocked vehicle 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.

Another point that should be addressed in the future is the interaction between lane-changing and car-following. In SUMO vehicles always maintain sufficient gaps to allow safe stopping if their leader vehicle were to brake with maximum deceleration until stopped. Likewise, the ego vehicle only changes to a new lane when the follower vehicle has enough space to stop safely if the ego vehicle were to brake with maximum deceleration until stopped.

In reality, drivers may accept much lower front-headways during lane-changing which may be justified by any of the following arguments:

- Their leader vehicle will usually not start to brake hard (especially when there is no apparent blockade)
- They may change the lane again to avoid collision if necessary
- They have a lower reaction time by concentrating on a critical maneuver and thus are able to use smaller gaps safely

Likewise, drivers in reality may accept much lower rear-headways during lane-changing by using the following justifications:

- They will not suddenly start braking hard when there is no obstacle (which they could see in advance if it was there)
- They may plan to continue their lane change maneuver one lane further which only requires them to remain safely clear of the follower vehicle for a brief time window

Some of these justifications may be applicable to SUMO vehicles and could be used for altering the parameters of the car-following model when evaluating the safety of a lane change maneuver. This would go a long way towards increasing the realism in scenarios such as highway on- and off-ramps where urgent strategic changes are needed.

References

1. Behrisch M, Bieker L, Erdmann J, Krajzewicz D (2011) SUMO—simulation of urban mobility: an overview. In: SIMUL 2011, the third international conference on advances in system simulation
2. DLR and contributors: SUMO homepage (2013) http://sumo.sourceforge.net/
3. SUMO source code corresponding to this document. http://sumo-sim.org/trac.wsgi/browser/trunk/sumo/src/?rev=16164
4. Erdmann J (2014) Lane-changing model in SUMO. In: Proceedings of the SUMO2014 modeling mobility with open data, 24, Seiten 77–88. Deutsches Zentrum für Luft—und Raumfahrt e.V. SUMO2014, 16–16. Mai 2014, Berlin, Deutschland. ISSN 1866-721X
5. Krauß S, Wagner P, Gawron C (1997) Metastable states in a microscopic model of traffic flow. Phys Rev E Am Phys Soc 55:5597–5602
6. Krajzewicz D (2010) Traffic simulation with SUMO—simulation of urban mobility. In: Barceló J (ed) Fundamentals of traffic simulation, series: international series in operations research and management science, vol 145, Springer, ISBN 978-1-4419-6141-9
7. Section 7 StVO (German Straßenverkehrsordung)
8. Section 2 StVO (German Straßenverkehrsordung)
9. Knospe W, Santen L, Schadschneider A, Schreckenberg M (2002) Single-vehicle data of highway traffic: microscopic description of traffic phases. Phys Rev E Am Phys Soc 65:056133
10. Lee SE, Olsen ECB, Wierwille WW (2004). A comprehensive examination of naturalistic lane-changes. Report no. DOT HS 809 702. National Highway Traffic Safety Administration, Washington, DC

Development and Assessment of Cooperative V2X Applications for Emergency Vehicles in an Urban Environment Enabled by Behavioral Models

Florian Weinert and Michael Düring

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. Simulations may support the development of applications, as it is not possible to test them in a real traffic system. Simulation of Urban Mobility (SUMO) is one possible tool to conduct simulations of 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. Therefore, we adjust them with a video analysis and simulate them. Further, an assessment analyzes their usability as a reference for testing new applications. In order to identify supportive applications, we created and carried out a survey with 252 EV drivers. The deduced applications are a traffic light preemption via V2I and an automated formation of a rescue lane via V2V. We assess the models and applications by evaluating the travelling time, a speed profile of the EV, and speed profiles of the IVs. Additionally, we show the usefulness of the two applications for the EV as well as the IVs.

Keywords Simulation of an emergency vehicle · Simulation of EVs · Urban · Intersection · Real EV behavior · Rescue lane · Intelligent transportation system · Intelligent traffic light · V2X · Preemption · Automated formation of a rescue lane · Travelling time

F. Weinert (✉) · M. Düring
Volkswagen AG, Berliner Ring 2, 38440 Wolfsburg, Germany
e-mail: Florian.Weinert@Volkswagen.de

M. Düring
e-mail: Michael.Duering@Volkswagen.de

M. Behrisch and M. Weber (eds.), *Modeling Mobility with Open Data*,
Lecture Notes in Mobility, DOI 10.1007/978-3-319-15024-6_8

1 Introduction

Statistics about missions of rescue services in Germany indicate over 14 million missions a year [1]. This corresponds to several ten thousand missions of emergency vehicles (EVs) every 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 [2]. The travelling time of an EV may be influenced by any incident on the road. Especially in an urban environment, red traffic lights are a serious threat for reaching the destination in time [3–6]. 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 [7, 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 [8].

For supporting EVs in urban situations, research and development presented various systems [4–6, 9]. 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. In order 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. As this is a severe alteration of the traffic system, it is hardly imaginable that local authorities allow such a procedure. However, simulations are a suitable tool to perform the necessary potential analysis.

2 Survey of EV Drivers

Only few studies investigate accidents related to missions of EVs. Müller [8] did his studies based on traffic data from 1994. Since then, no reliable source stated data in comparable quality. One reason might be that accidents involving EVs are not monitored centrally. Potential sources to collect relevant data are emergency vehicle drivers who know about the problems and dangerous situation which occur on missions. Hence, we carry out a systematic survey with EV drivers, aiming to discover critical traffic situations and deduce possible solutions. In the following, we present the survey and an initial analysis. We intent to give a first outline about the answers on selected questions.

The online survey enfolds 252 drivers of police cars, ambulances, and fire trucks. Personnel driving fire trucks comprise drivers within the professional fire brigade and the voluntary fire brigade. The survey starts with sociodemographic questions, continues with statements about "Driving and Situational Awareness" and "Accidents", and closes with opinions about technical assistance systems.

Within the "Driving and Situational Awareness" section, the drivers have to answer questions concerning their medical condition before starting a mission and the assessment of different situations on their missions. 54 % of EV drivers indicate that pull-outs are routine for them.

However, 55 % admit that a mission means stress and 84 % say they are tensed during the pull-out. For 98 % of the interviewees, safety is more important than celerity. Regarding the different traffic situations, it becomes apparent that drivers categorize intersections as more critical than any other type of street. Moreover, they perceive driving through red traffic lights (96 %) and crossing intersections without traffic lights (80 %) as critical, while only 12 % define straight roads as critical. Blue light and siren gain enough attention to make others aware of the approaching EV according to 59 % of the EV drivers. The 41 % denying this statement mention that other drivers are distracted by media such as mobile phones or the radio. Additionally, the harmonic tone sequence might reduce the perceptibility of the siren.

The section "Accidents" reveals that 35 % of the interviewees already have had an accident. 93 % agree with the sentence "Accidents are caused by the individual traffic". Moreover, 91 % think that abrupt braking and wrong steering reaction cause accidents. These maneuvers are more critical than not reacting at all according to 79 % of the interviewees. Beyond, 56 % agree that "Accidents are caused by EV drivers".

The reasons of accidents may be divided into driving too fast near/in the intersection area (92 %) and driving through red lights (91 %). Moreover, they think that accidents mostly occur on the way to the place of assignment (91 %) and not on the way back to the department (13 %).

Concerning the technical assistance systems, we classified two different types: assistance systems for the EV and assistance systems for the IV. For the EV, we asked for the usefulness of a preemption system. For IVs, we wanted to know if additional information or even an automated reaction of IVs may help. The interviewees say that preemption systems can save time (92 %) and reduce the danger of driving through intersections (90 %). This also leads to less stress (75 %). Regarding the IV, we divided systems into three types: warning IV drivers about an approaching EV, advising the drivers on how to react, and an automatically reacting system that may override the drivers. Interviewees categorize that "detailed warning about the approaching EV" (91 %) is useful, 64 % think that "the advice on how to react" helps drivers. 30 % of the EV drivers find a system useful that conducts automated reactions on EVs. Concluding the comment areas of this last application, we can see that automated systems achieve only small acceptance because the drivers apprehend technical difficulties and the lack of robustness. Figure 1 shows the results in short.

Summarizing this initial analysis of the survey, we gather information on what would make missions of EVs safer. We deduce that assistance systems are useful

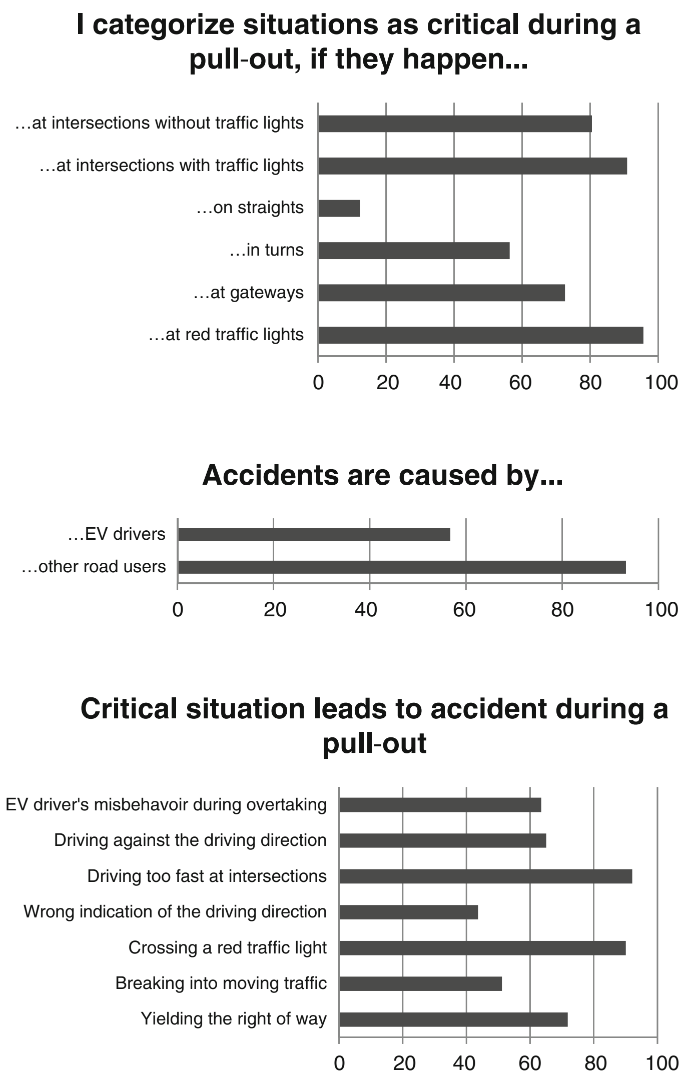


Fig. 1 Excerpt of answers to the EV survey (in percent)

for EV drivers. There is a demand for support at traffic light controlled and blind intersections. Moreover, the EV drivers state that the IV has a major influence and is responsible for hazardous situations. IV drivers could be supported by warning and advice systems. An automated vehicle reaction may also support the IV driver, but, according to the interviewees, such a system is beyond the technical possibilities and not realizable.

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 [10]. However, the simulation of special situations—e.g. situations comprising EVs—is not covered. EVs may override 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). The 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 [11]. Others argue that by neglecting these effects the potential of new applications may be overestimated [12]. Bieker [13] does not implement a driving through red lights because the EV coincidental arrives during the green phase. According to her, a model needs to be investigated to overcome the red light issue. 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 article, we want to present models enabling SUMO to simulate V2X applications improving traffic efficiency and safety involving EVs and conduct simulations of two applications, namely a preemption and an automated cooperative formation of a rescue lane by IVs. This article is organized as follows. Section 4 describes the simulative environment with all boundary conditions and input parameters. Section 5 explains the different implementations. Section 6 deals with the calibration of the proposed models. Section 7 describes two example applications as well as the simulation. Moreover, it contains the assessment of both the models and applications. Section 8 completes the article by giving a conclusion and outlook.

4 Simulative Environment

Material provided by OpenStreetMap is the basis for the traffic system used in this article. It is shown in Fig. 2 and includes three urban intersections in Braunschweig,[1] 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 3 shows the underlying data 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

[1] Intersections from west to east: Rebenring/Pockelsstraße, Rebenring/Hagenring, and Hans-Sommer-Straße/Langer Kamp.

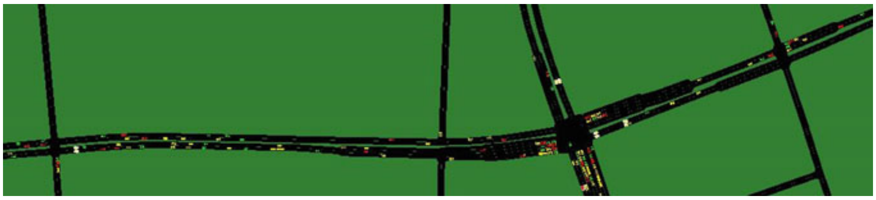


Fig. 2 The simulated traffic system

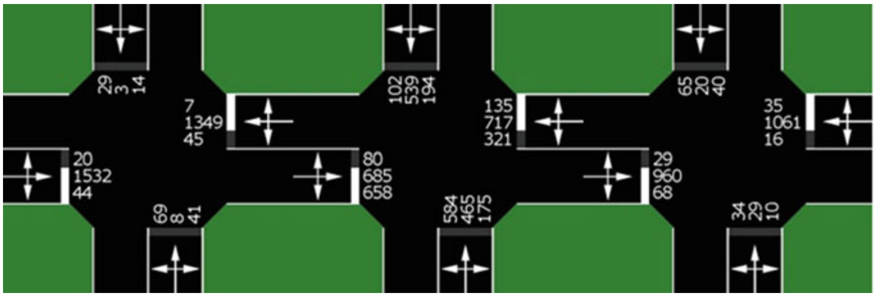


Fig. 3 Collected data of a traffic census at the relevant intersections during the peak hour

of trucks is 3 % with a distribution of semi-trailer trucks (Truck 1) and short trucks (Truck 2) 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 times of the drivers, and driver's attention. Values for type Car 1 are comparable to the vehicles of the A00 segment. Type Car 2 represents the A segment, type Car 3 equals the B segment, 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 1 shows vehicle related parameters and used driver models (minGap, Sigma and Impatience). The table also contains parameters used for the EV.

Table 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 (–)
Car 1	40	0.8	1.9	3.0	3.5	2.00	0.6	0.3
Car 2	50	0.95	2.6	3.5	4.2	1.20	0.8	0.5
Car 3	60	1.0	3.1	4.0	4.7	0.65	0.8	0.8
Truck 1	22	1.0	0.8	3.5	18.4	0.75	0.9	0.7
Truck 2	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

5 Models

5.1 EV Behavior

As stated before, EVs are allowed to override general traffic rules and SUMO is not able to model this necessary behavior with a predefined internal function. Our implementation concerning the EV's behavior considers speeding and the ability to drive through red lights. The usage of a siren and a light bar is not visualized within the simulation. However, their effect on the IV is described in Sect. 5.3.

5.1.1 Speeding

The EV may override speed restrictions by using the implemented speed factor. Table 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 may 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 can drive 16.56 m/s ($1.2 * 13.8$ m/s ≈ 60 km/h) within the traffic system, as the maximum speed of the EV (30 m/s) is not exceeded.

5.1.2 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. Normally, an EV approaching a red traffic light in the simulation would start to brake in order not to violate traffic rules. Even if no vehicle congests the intersection, the EV will wait until the traffic light switches to green. Figure 4 shows a flowchart of the implemented algorithm which allows an EV to drive through red lights. 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 Sect. 6). Second, it checks whether the EV is in front of 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 green signal is held or the red signal is switched to green.

If the EV is driving faster than the upper threshold, the signal is held or switched to red. This leads to an averaged approaching behavior of the EV which can be observed in real situations with EVs approaching intersections. The algorithm is executed every time step in the simulation.

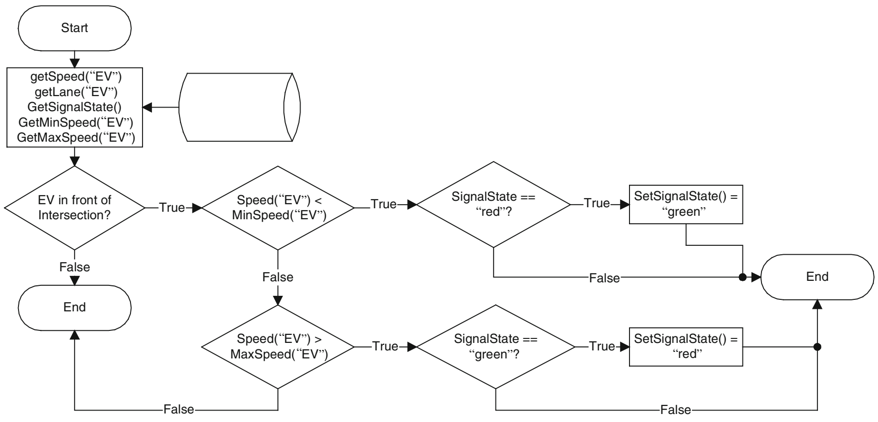


Fig. 4 Flowchart showing one EV model

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 only react to vehicles of the type EV. The distance between the loops and the intersection represents the V2X reception radius.

5.3 *IV Behavior*

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 [14]. One possibility to behave cooperatively is described in the Road Traffic Regulations [15] 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 hereinafter. An example situation clarifies the functional principle of the method.

Figure 5 (top) shows a oneway road with three lanes. Ten vehicles drive on that road as an EV approaches on the middle lane from behind. In this example, the method clears the middle lane by forcing the obstructing vehicles to change the lane. It induces a lane change maneuver by using the SUMO internal ChangeLane ()-function based on the SUMO vehicle dynamics. Figure 5 (middle) shows the turn signals indicating a lane change of the obstructing vehicles. The direction of the lane change may be parameterized according to the vehicles' destinations. Figure 5 (bottom) shows the final rescue lane. The flowchart in Fig. 6 shows the algorithm.

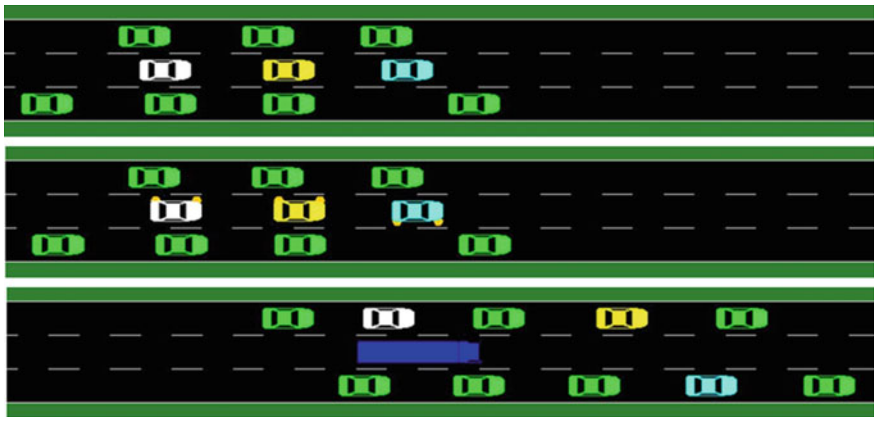


Fig. 5 Example situation for the IV behavior

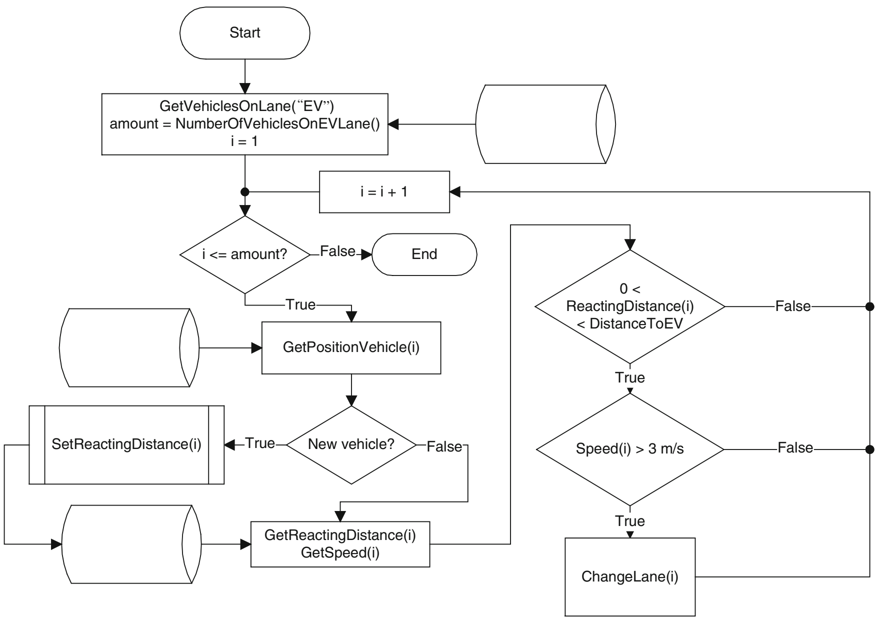


Fig. 6 Flowchart showing the model "IV behavior"

The algorithm determines the number of vehicles on the EV's lane (amount) and their identification number. After that, a procedure checks each vehicle. First, it determines the speed of the vehicle. 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 Sect. 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 a certain value. This procedure is repeated for each vehicle and each time step in the simulation. In the following, two different settings will be presented: IV behavior a and IV behavior b. Both IV behavior models have a threshold of 5 m/s for the vehicle's speed to induce a lane change.

5.3.1 Behavior a

The algorithm induces a lane change without taking the route of the vehicles into account. It may be possible that a vehicle is not able to reach its destination, because the algorithm forces it to change the lane to an undesired one. This behavior is the base for the investigations in [16].

5.3.2 Behavior b

This model takes the route of the vehicles into account. That may lead to longer travelling times caused by obstructing vehicles but has the advantage that every vehicle is able to reach its destination.

6 Calibration

The calibration of the 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 EVs 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.

Buchenscheit et al. [17] assessed videos to gain insights of interactions between EVs and IV. They mounted a camera on the dashboard of an EV and recorded 21 typical emergency response trips with a total length of 147 min. 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 min 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 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 min with 116 traffic light controlled intersections and a variety of numbers and composition of vehicles and environments. Concerning the traffic lights, the traffic light was red 56 times 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), crossing IVs almost leads to accidents in two instances. 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”, “50–20 m”, and “20 m and less”. Depending on the environment and the perception of the EV’s presence, approximately 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 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 crossing an intersection while having a red light 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 Fig. 4). 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 Fig. 6. The discovered distribution over the obtained reacting distances is modelled by a random, uniformly distributed float generator. Figure 7 shows the IV reaction model.

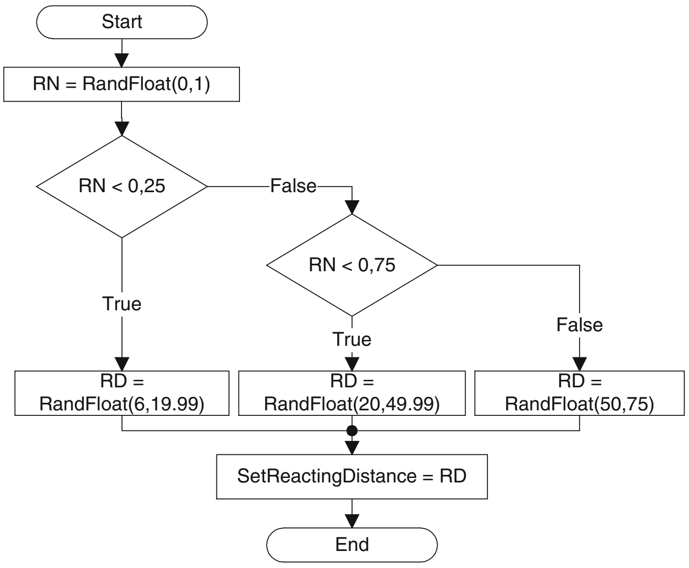


Fig. 7 Flowchart to calibrate the IV behavior model (SetReactingDistance)

7 Simulation of the Models and V2X-Applications

We present an intelligent traffic infrastructure and near realistic behavior of IVs and an 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 differences. Afterwards, we conduct simulations for two applications: a preemption system and an automated formation of a rescue lane. The next subsections describe the applications, the simulation procedure, and their assessment.

7.1 Preemption

A 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 principle is shown in Fig. 8. The algorithm determines if an EV preemption program is active. If not, it checks for a request at the starting induction loop, which represents the V2I reception distance. 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) [18] conform, predefined EV preemption program and starts a timer. If the EV preemption is active, the algorithm

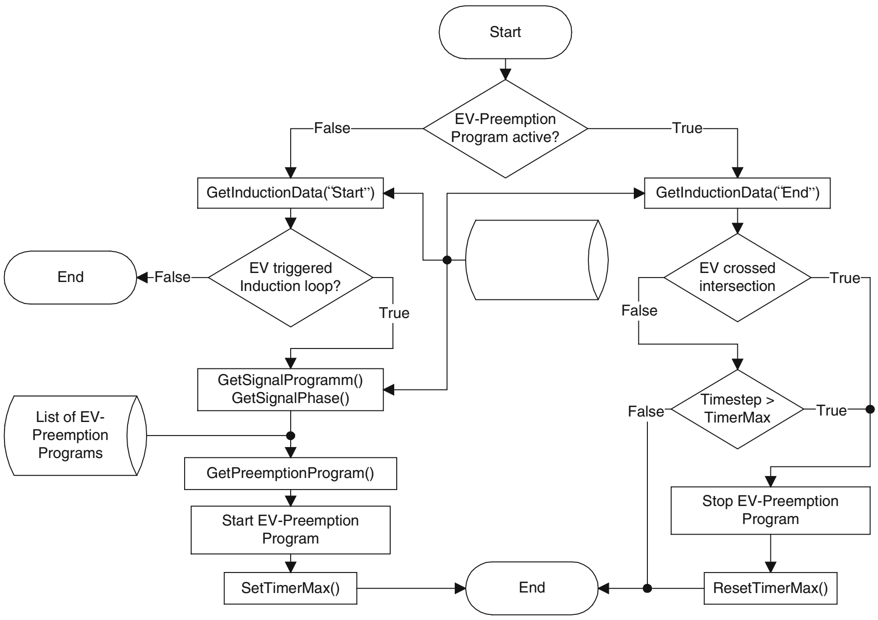


Fig. 8 Flowchart of the traffic light preemption

checks whether the EV triggered the ending induction loop or the maximal time-span has expired. There are two factors that influence the success of the preemption system.

First, the moment of registration at the infrastructure influences the possibilities to switch the signal phase according to the RiLSA.[2] 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 EV's traffic light shows green or yellow/red. When having green, the green light may be held as long as the other directions do not have a red light for more than 3 min. 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. In this article, the communication distance for the intersections is (west to east) 165 m, 450 m, and 165 m. The goal of a preemption is to reduce the travelling time of the EV in an urban environment. Moreover, the safety of the EV and IVs may be enhanced while minimizing the adverse impact on IVs.

[2] By observing the guidelines, our method does consider pedestrians implicitly.

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 [14]. Applying 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 proving additional information can be found in the literature [17]. Depending on the characteristic of the system, it gives additional information and thus assists the driver, gives direct advices, or induces 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. A cooperative coordination among IVs can be obtained by using different methods. In this application, a rule based approach is employed. Figure 6 shows the implemented algorithm to model IV behavior. The automated formation of a rescue lane is based on this flowchart, but the SetReactionDistance function of Fig. 7 is substituted by a constant value of 150 m. The threshold for inducing the ChangeLane() command is set to 10 m/s. The application obeys the route and the destination of the vehicles. It 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 run on the IVs which send ego information such as own lane and own position to the master. Additionally, it is assumed that the slaves execute the commands sent by the master without sending an acknowledgement.

7.3 Simulation Procedure

The simulation procedure describes the configuration of executed simulations. Table 2 shows employed models and applications for different runs of the simulation.

The simulation runs for 500 s without modification. After that, the EV enters the simulation and drives through the traffic system on a designated route. It always uses the most right lane possible to reach its destination. In contrast, the IVs starting on an edge with multiple lanes use a randomized departing lane.

The moment the EV enters the simulation is varied from the 500th s in steps of 1 s to the 585th s. The timespan of 85 s matches the timespan of one phase shift for all three intersections. Each test is performed 50 times to take randomized effects

Table 2 Setup of the different tests

No	Speeding	Driving through red lights	IV reaction	Preemption	Autom. rescue lane
M1	X				
M2 (b)	X	X	X		
A1	X	X	X	X	
A2	X			X	
A3	X	X	X		X
A4	X	X	X	X	X

into account and ends when the EV reaches a specific point at the end of the simulation. Models are implemented as discussed in Sect. 5. The two applications are employed as shown in Sects. 7.1 and 7.2. 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 proposed models to show the difference in travelling time.

The assessment of simulations enables to decide for a realistic reference. This reference serves to measure the effectiveness of the applications which 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. Test A3 simulates the application alone whereas Test A4 includes both applications. The figures in the following assessment 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.

7.4 Assessment of Models

Figure 9 shows the travelling time of the EV within Test M1. The EV behaves like a normal road user, as none of the models is activated. The travelling time is not constant over the introduction second. This variance is caused by a randomized starting lane and 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

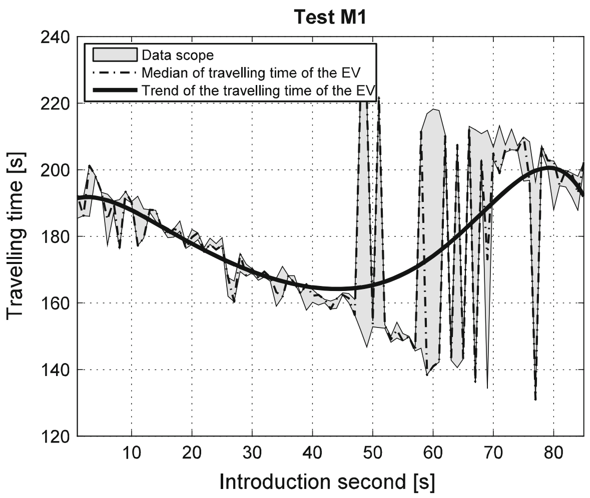


Fig. 9 Travelling time in Test M1

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 10 shows Test M2 with all three models activated. It shows that the travelling time depends on the introduction second. Moreover, there are different

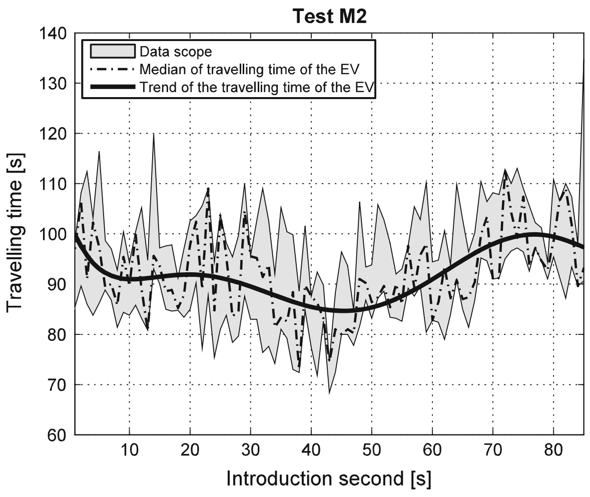


Fig. 10 Travelling time in Test M2

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 a 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 more 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 with 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 barely influenced by variations of IV behavior as the data scope is relatively small. 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 relatively high travelling times (about 100 s) for the EV. At least one traffic light is red with waiting and obstructing vehicles. This leads to an additional delay for the EV as a following traffic light also might change to red.

The comparison of the two simulations Test M1 and Test 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. That suggests using Test M2 as a reference estimating the potential of EV applications.

However, the following analysis compares the results of the two different IV behaviors, namely behavior a and behavior b (see Sect. 5.3). Figure 11 shows the results of IV behavior b. Remember the difference of IV behavior a and IV behavior b which is that within IV behavior a the vehicles clear the lane at any cost whereas IV behavior b takes the designated route into account. The results of Test M2b show that the course of the trendline fits the results of Test M2a with an offset of about 15 s. Moreover, the data scope is also greater which indicates a wider range of possible obstructions for the EV.

However, studying the different behavior models with the SUMO internal GUI yields arguments for IV behavior b: Vehicles that clear the lane at all costs can not follow their desired route and end up in a wrong lane obstruct succeeding traffic.

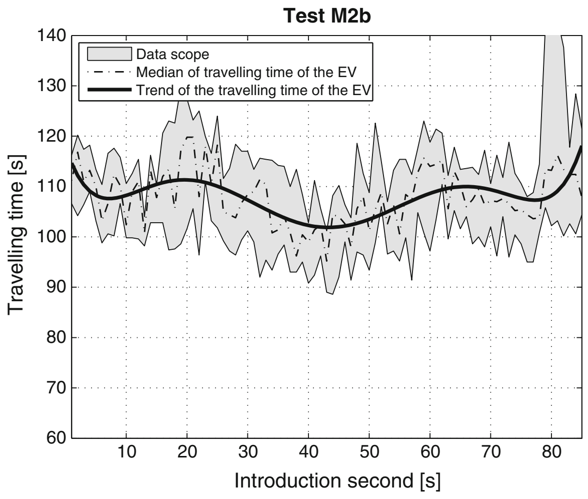


Fig. 11 Travelling time in Test M2b

This leads to situations in which vehicles end in a turning lane waiting to go straight which is not possible by settings. Finally, these vehicles are teleported after a period of 3 min. This situation does not occur in IV behavior b, because vehicles are only clearing the lane if they are still able to follow their route. This seems to be a more realistic behavior of the traffic systems. Consequently, the simulations use IV behavior b for further investigations and the results of Test M2b as a reference for estimating the potential of EV applications.

7.5 Assessment of Applications

The assessment is divided into two parts: Firstly, an analysis of the travelling time and the distribution of travelling times over the signal phase time takes place. Secondly, speed profiles of the EV give some indication about the usefulness of the two applications.

7.5.1 Travelling Time and Their Distribution

Figure 12 shows the travelling time over the introduction second for Test A1. The course of the trendline drops from a value of around 95 s at introduction second 1 to a minimum of 82 s around introduction second 35. Afterwards, it rises to a travelling time around 105 s 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 15–45, the preemption comes in time

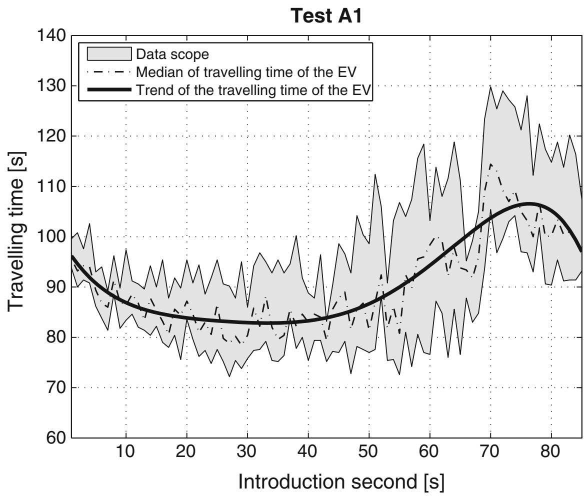


Fig. 12 Travelling time in Test A1

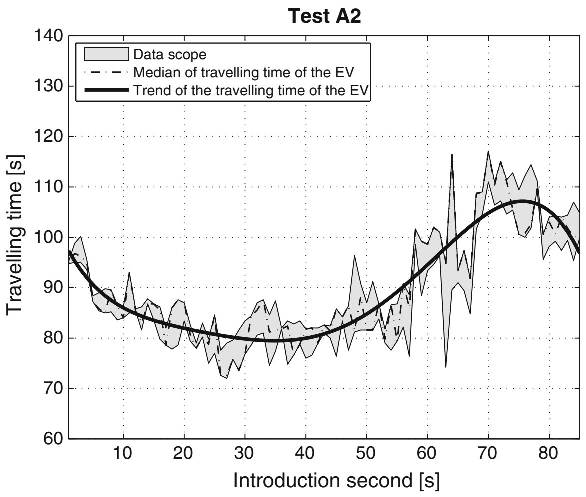
to either hold the green phase or change the red/yellow traffic light to green. The variances are caused by IVs randomly merging into the EVs lane. Taking median values into account, introduction seconds 26, 28, 43, 53, and 55 are advantageous moments for the preemption so that the travelling time is less than 83 s. 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 s.

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 M2b. The difference in travelling time is about 15 s for all introduction seconds except for a range of 15 s from introduction second 65 up to introduction second 80 in which the difference is smaller. During these introduction seconds, the preemption system for all three traffic lights needs to be triggered. The queue of vehicles obstructs the EV three times. This leads to the insight that a more detailed calibration of the traffic light preemption should take place. Analyzing the influences of parameters such as the distance between the induction loops and the intersections, congestion in front of the intersection, and the speed of participants will help to adjust a better system behavior. This study is out of scope of this contribution but will be addressed in further work.

The travelling time of Test A2 is represented by the graph in Fig. 13. The gray area is not very distinct as the IV behavior and the drive through red lights models are deactivated. From this point of view, it is comparable to Test M1. The course of the trendline starts at a travelling time of about 95 s, then declines to a minimum around 80 s at introduction second 36 and eventually rises to a maximum of 105 s around introduction second 75.

For small introduction seconds, the EV encounters delays by obstructing vehicles at the first intersection. However, for later introduction seconds, the vehicles

Fig. 13 Travelling time in Test A2



have more time to start. Moreover, the preemption shows its efficiency by holding the green phase at the first intersection. 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 or turn at an intersection. The maxima at introduction seconds 64 and 68–79 are caused by this effect at one and/or multiple intersections.

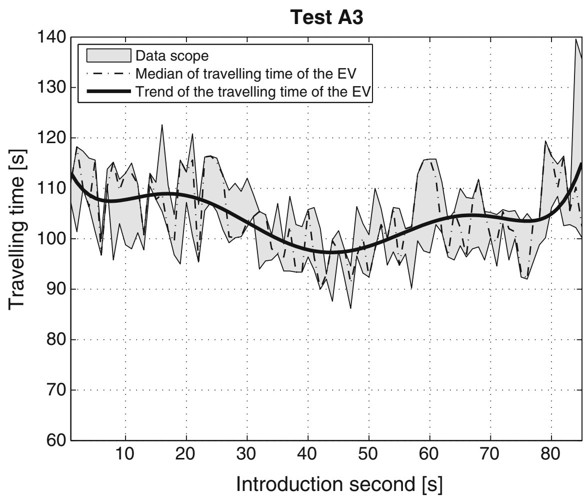
The results indicate that the travelling time is dramatically reduced compared to Test M1. The difference to Test M2b is much smaller. This indicates that studies comparing a preemption system to a reference situation without considering IV reaction and the EV behavior vastly overestimate the potential of a preemption system.

Comparing the results of Test A1 and Test A2 shows that the data scope of Test A2 is much smaller. The trendline of Test A2 drops below a travelling time of 80 s whereas the trendline of Test A1 has its minimum around a travelling time of 83 s. Except for this range around the minimum, the travelling times of Test A1 are slightly smaller. This shows that the IV behavior models have only little effect on the travelling time when the preemption system is activated.

Figure 14 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 second 8, then increases until introduction second 18, declines until second 43, rises until second 85 with a local minimum at second 75.

Starting from introduction second 1 to 8, the first traffic light switches to green, and vehicles have more time to start and clear the lane for the EV. On the EV's

Fig. 14 Travelling time in Test A3



route between the first and the second traffic light, the vehicles clear the lane for the EV if this corresponds with their route. Results of vehicles clearing the lane despite their routes can be found in [16].

The travelling time starts around 112 s and decreases to a minimum around introduction second 43 with a travelling time of 97 s. Afterwards it rises until a travelling time of 115 s for introduction second 85. The course of the graph is very alike compared to the results of Test M2b. However, the data scope is smaller in tests with the application running. This means that the 50 simulations for one introduction second lead to similar results.

The application may support the EV in reaching its destination in a more deterministic manner. The main issue, the waiting vehicles that obstruct the EV's route, cannot clear the lane because of the small speeds. This simple approach with fixed distances and fixed speed thresholds shows that the application does not decrease the performance. However, additional potential may be addressed by smart rules. 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 15 shows the travelling time of the EV in Test A4. The trendline declines from a travelling time of 95 s at the first introduction second to a local minimum of 85 s at introduction second 20. After that, the trendline rises to a maximum of 105 s around introduction second 78 and then declines to 95 s of travelling time at introduction second 85. At small introduction seconds, the two applications cannot clear the path for the EV effectively which leads to delays. With rising introduction seconds, the effectiveness of the applications rises, because the obstructing vehicles have enough time to start and afterwards clear the lane. After introduction second 13, the first traffic light is green and the IVs have enough time to start without

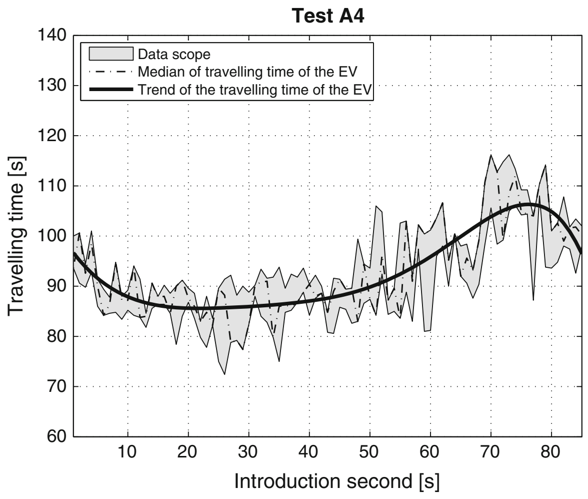


Fig. 15 Travelling time in Test A4

obstructing the EV. The introduction seconds between 15 and 40 are advantageous for the EV, because green phases may be held by the preemption system and the intersections are too short to make the vehicles disappear, especially at the first and the third intersection.

Comparing these results to the reference Test M2b shows that the two applications are advantageous for the travelling time of the EV. Around introduction second 78, the travelling times for both tests are very similar. The relatively high values for Test A4 originate from obstructing vehicles in front of the third intersection. The preemption system switches the traffic light of the second intersection in time with the third intersection still having red. The vehicles approach the third intersection, slow down, and eventually have to stop again. Thus, the EV needs to decelerate as well, because the triggering point for the preemption of the third intersection is located too near to the intersection. So, the vehicles do not have enough time to clear the path.

Comparing Test A1 and Test A4, it becomes apparent that the trend lines are more or less identical with a much smaller data scope for Test A4. This means that the formation of the rescue lane leads to a more predictable travelling time for an EV in this traffic system. A combination of a traffic light preemption system and an automated formation of a lane change integrates the advantages of both systems to support the EV reaching its destination as fast and predictable as possible.

7.5.2 EV's Speed Profiles

Speed profiles enable statements about the trip of the considered vehicle. Deviations in the velocity profiles may lead to losses in comfort and safety. Strong gradients in the velocity express emergency stops which may be used to describe safety critical states. Especially braking needs a reaction of the adjacent traffic—thereby every braking situation is a potential danger.

The following figures express the results of three different tests: Test M2b, Test A1, and Test A4. The plots underlie the starting parameter (introduction second 18 for the EV) in which the travelling time does neither reach a maximum nor a minimum. Also the data scope has an average variation for this introduction second. Figures 16, 17 and 18 show the diagrams of the three difference applications. Within these plots, the x-axis denotes the travelling time in seconds while the y-axis denotes the velocity in m/s. Each plot contains 25 speed profiles of the EV. If the speed profiles are equal, they cannot be distinguished in the plot.

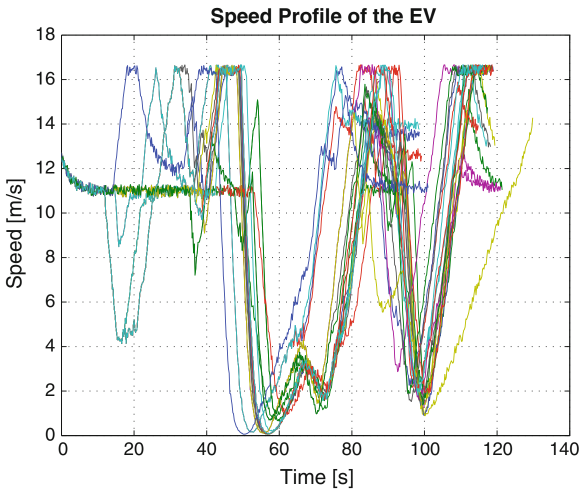


Fig. 16 EV in Test M2b

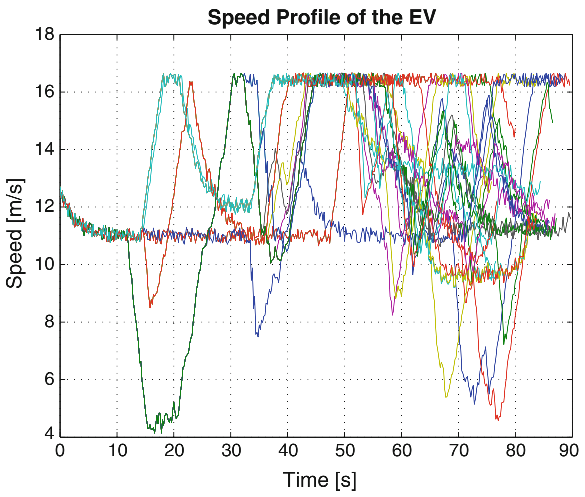


Fig. 17 EV in Test A1

Fig. 18 EV in Test A4

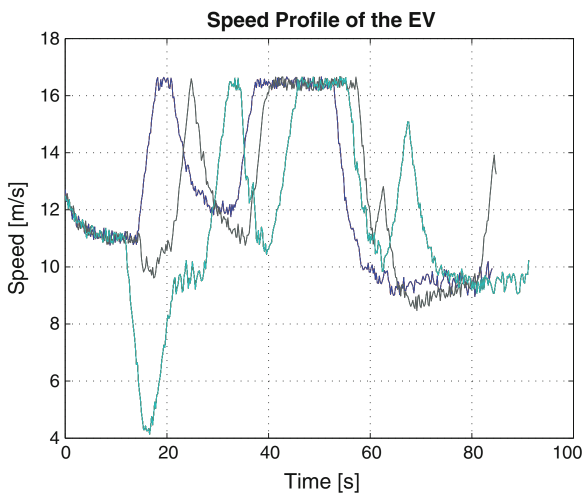


Figure 16 represents the reference behavior Test M2b. The speed profiles' trends are similar until second 12. During this time the EV is on its way to the first traffic light and not influenced by the IV. It only has to brake for slowly driving vehicles in front of it. After this period, distinctions become apparent caused by two stochastic effects: The first one makes vehicles use random lanes if they enter the simulation on a multi-lane road. As three different types of cars and two types of trucks exist, this leads to a variety in, e.g. the starting behavior and the maximum velocity. The dissolution to traffic jams is also influenced by this fact. The second effect deals with a randomized reacting distance model which allocates a reacting distance to each vehicle which is in the EV's way. The algorithm calculates the distance randomly in each run of the simulation. This leads to a different behavior of the IV which is still within the constraints and boundary conditions of the acquired IV model. As a consequence, the EV may reach the traffic light while having a green phase or a red phase. From second 50 to 80, the graphs show an approximated comparable trend. During that time span, the EV comes to a stop at the second traffic light and makes use of the "driving through red lights" model which leads to the W-shaped course of the graphs. After that, the courses of the graphs differ caused by the EV's arrival time at the traffic light. Except the short plateau in the beginning of the simulations, there is no continuous trend in the graphs. Rather, there is a temporal permanent braking, which is followed by acceleration to the maximum speed which cannot be kept for a longer time. This leads to the conclusion that the trends are neither comfortable nor time efficient in Test M2b.

Figure 17 shows the graph for Test A1. The graphs start around 12 m/s and drop until 11 m/s. Except for some outliers, the speed does not drop below 10 m/s for the entire simulation. In contrast, a plateau at a relatively high speed (16.56 m/s) emerges. Some graphs drop to low values. That is caused by a late preemption and thus a need for the driving through red light model at the third traffic light. Overall, there is less variance in the velocity profiles.

Comparing Fig. 17 with Fig. 16, it attracts attention that the second diagram has a plateau until second 40, which is visible in the first diagram, too. However, the second diagram has only few outliers. There is no distinctive minimum (W-shape) of the velocity between 60 and 100 s. Moreover, Fig. 17 shows a higher level of maximal speed than visible in Fig. 16. This leads to a reduction in travelling time by 20 to 30 s.

Figure 18 shows the speed profiles of Test A4. The speed profiles start around 12 m/s, drop until 11 m/s at 15 s. After that, three distinctive courses of the graphs can be observed. Two of those have in common that they drop and afterwards rise to the maximum of 16.56 m/s. The third graph rises to the maximum and decreases afterwards to rise to the same plateau with some delay. Together, they develop a plateau for about 20 s, and drop below 10 m/s. This speed is also the speed at the end of simulation after between 85 and 95 s.

Comparing these results with Figs. 17 and 16, the reduced variety of graphs attracts attention. The application formation of a rescue lane assigns a higher and consistent reacting distance to the IV. As planned by the algorithm the behavior gets more deterministic. The randomized entering lane and different vehicles types remain as variation parameter and lead to the derivations. The travelling time and speed profile gets more predictable by using both applications.

7.5.3 IVs' Speed Profiles

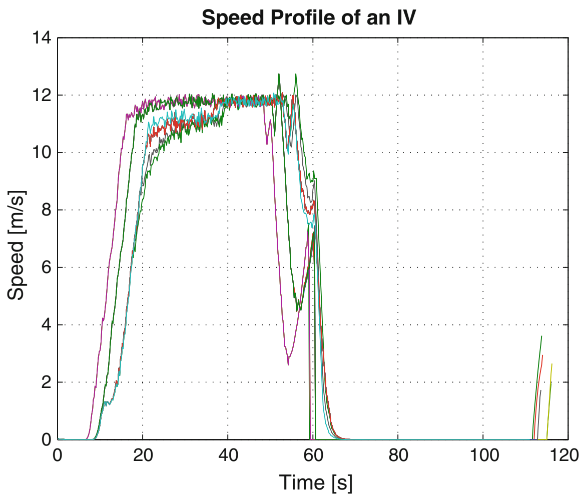
While the models and applications have an effect on the EV, there should also be an effect on the IVs. Two different crossing vehicles represent the IVs in conflict situations with the EV. The first vehicle profits whereas the second vehicle is impaired by the EV applications. The assessment is based on speed profiles, too. The measurement of the speed profile begins when the EV enters the simulation and ends when either the EV or the considered IV leaves the simulation.

The two plots in Fig. 19 describe the vehicle that profits from the applications and Fig. 20 shows a vehicle getting delayed by the applications. The two plots in both figures show speed profiles during Test M2b (top) and Test A4 (bottom).

Each diagram contains 25 graphs. Some graphs may be hidden behind a graph of an identical simulation result.

In Fig. 19 it becomes apparent, that until second 50 both diagrams have nearly the same course. Due to the EV's driving through red lights model, the IV has to stop and wait until second 110 in Test M2. In Test A4 the preemption speeds up the EV, so that it is not relevant for the specific IV. Hence, the IV is able to cross the intersection without obstruction. The IV drives faster and is not forced to stop by the applications. Thus, it reaches its destination after around 80 s.

The considered vehicle of Fig. 20 loses time with applications activated. The first 20 s of the diagrams are comparable. Afterwards the vehicle comes to a stop in



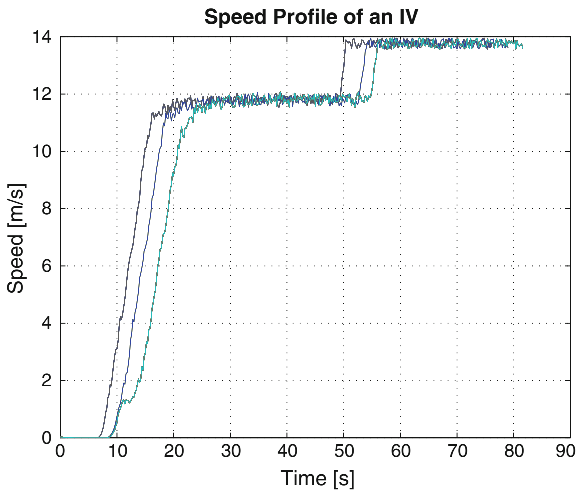


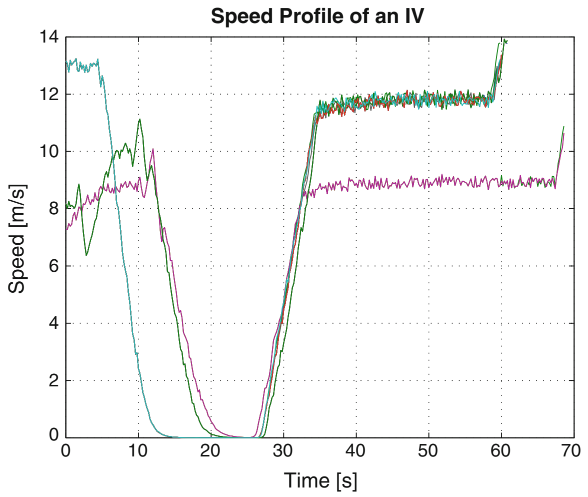
Fig. 19 IV profits by the applications

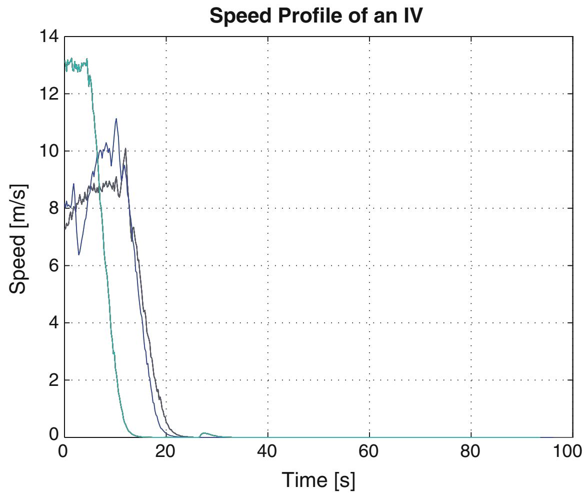
both cases. In the upper diagram the IV is not obstructed by the EV at all. It only has to wait for the normal red phase and continues its route after the traffic light switches to green again.

In the lower diagram, the EV is preempted just in the moment before the IVs traffic light switches to green. Thus, the IV's red phase is extended and the IV has to wait for a longer period of time. Before the IV's traffic light switches to green again, the simulation ends because the EV reaches its destination.

As a brief conclusion, we can say that there are vehicles and situation which profit from the applications. Vice versa, the application may increase waiting times of IVs. Further work analyzes the effects and uses the obtained insights to develop an advanced cooperative system behavior.

Fig. 20 IV needs to wait because of the applications





8 Conclusion and Outlook

Within this article, we showed that emergency vehicles (EVs) encounter a higher risk of getting involved in accidents during their missions. For supporting 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, the 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 traffic system was based on traffic census data during peak hour. We presented models regarding the road users EV and IV and the infrastructure. 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.

We created, conducted, and evaluated a survey with 252 EV drivers to deduce supportive applications. We simulated two applications: a traffic light preemption system via V2I and an automated formation of a rescue lane via V2X. By assessing the results of 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 time of the EV compared to a reference travelling time. The automated formation of a rescue lane does not necessarily reduce the travelling time as it does not accelerate the queuing vehicles. 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 and predictable.

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 [19]), 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 two 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, too. 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. We showed that even by enhancing the concept of static rules, a fairly good result can be obtained. However, instead of using static rules, other maneuver planning methods may calculate better behaviors of IVs by considering additional information.

References

1. Bundesanstalt für Strassenwesen (2011) Leistungen des Rettungsdienstes 2008/09. In: Berichte der Bundesanstalt für Straßenwesen—Mensch und Sicherheit 217
2. Niedersachs GVBl (1993) Verordnung über die Bemessung des Bedarfs an Einrichtungen des Rettungsdienstes (BedarfVO-RettD). p 1
3. Bycraft J (2000) Green light pre-emption of traffic signals for emergency vehicles. Technical report on City of Richmond traffic signal control system
4. Cetin M, Jordan CA (2012) Making way for emergency vehicles at oversaturated signals under vehicle-to-vehicle communications. In: IEEE international conference on vehicular electronics and safety (ICVES), pp 279–284
5. Traffic signal preemption for emergency vehicles. Technical report, U.S. Department of Transportation (2006)
6. Traffic signal priority control for emergency vehicle preemption. Technical report on Global Traffic Technologies, LLC, (2011)
7. Kahn C, Pirrallo R, Kuhn E (2001) Characteristics of fatal ambulance crashes in the United States: an 11-year retrospective analysis. In: Prehospital emergency care 5, pp 261–269
8. Müller D (2007) Typische Gefahren bei Einsatzfahrten des Rettungsdienstes. Technical report, Institut für Verkehrsrecht und Verkehrsverhalten Bautzen
9. Eltayeb AS, Almubarak HO, Attia TA (2013) A GPS based traffic light pre-emption control system for emergency vehicles. In: International conference on computing, electrical and electronics engineering (ICCEEE), pp 724–729
10. Krajzewicz D et al (2012) Recent development and applications of SUMO—simulation of urban mobility. Int J Adv Syst Meas 5(3, 4):128–138
11. McHale G, Collura J (2002) Improving emergency vehicle traffic signal priority system assessment methodologies. Technical report, Federal Highway Administration and Virginia Polytechnic Institute and State University
12. Zhang L et al (2010) Simulation modeling and application with emergency vehicle presence in CORSIM. Technical report, Mississippi State University and Turner-Fair Bank Highway Research Center
13. Bieker L (2011) Emergency vehicle prioritization using vehicle-to-infrastructure communication. Technical report, German Aerospace Center—Institute of Transportation Systems
14. During M, Pascheka P (2014) Cooperative decentralized decision making for conflict resolution among autonomous agents. In: IEEE international symposium on innovations in intelligent systems and applications (INISTA) proceedings, pp 154–161, 23–25 June 2014 10.1109/INISTA.2014.6873612
15. Bundesanzeiger Verlag. Verordnung zur Neufassung der Strassenverkehrs-Ordnung (StVO). Bundesminisiterium der Justiz, (2013)
16. Düring M, Gonter M, Thiel F, Weinert F (2014) Models enabling simulations of V2X applications regarding emergency vehicles in urban environment. In: SUMO2014—modeling mobility with open data, S. 55–75. ISSN:1866-721X
17. Buchenscheit A et al (2009) A VANET-based emergency vehicle warning system. In: IEEE vehicular networking conference (VNC), pp 1–8
18. Arbeitsgruppe Verkehrsmanagement, ed. Richtlinien für Lichtsignalanlagen. Forschungsgesellschaft für Straßen- und Verkehrswesen (FGSV) Verlag (2010)
19. Bosserhoff D (2010) Handbuch für Verkehrssicherheit und Verkehrstechnik der Hessischen Straßen- und Verkehrsverwaltung. In: Follmann J (ed) Forschungsgesellschaft für Straßen- und Verkehrswesen (FGSV), Chap. Knotenpunkte mit Lichtsignalanlagen, pp 4.5–1–4.5.131

TraCI4Matlab: Enabling the Integration of the SUMO Road Traffic Simulator and Matlab® Through a Software Re-engineering Process

Andrés F. Acosta, Jorge E. Espinosa and Jairo Espinosa

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 Traffic Control Interface (TraCI), SUMO offers a high level of flexibility, allowing a client to retrieve and modify the objects in the simulation. Two implementations of TraCI have been released to date for Python (TraCI-Python) and Java (TraCI4j). 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 platforms share strengths that benefit the development of control strategies for road traffic. The desire of combining both strengths motivated the interest to develop a TraCI implementation for Matlab. This chapter describes an adaptive software re-engineering process of TraCI-Python used to implement TraCI4Matlab (TraCI for Matlab).

1 Introduction

SUMO (Simulation of Urban Mobility) is an open-source software project that incorporates a set of tools to create and execute microscopic road traffic simulation scenarios [16]. These tools are grouped in three categories:

J.E. Espinosa
Politécnico Colombiano Jaime Isaza Cadavid, Cra 48 No 7-151, Medellin, Colombia
e-mail: jeespinosa@elpoli.edu.co

A.F. Acosta · J. Espinosa (✉)
Universidad Nacional de Colombia, Cra 80 No. 66-223, Medellin, Colombia
e-mail: jespinov@unal.edu.co

A.F. Acosta
e-mail: afacostag@unal.edu.co

- 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. This network is represented as a directed graph, where nodes define the intersections and edges define the streets.
- 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 the Traffic Control Interface (TraCI), which simplifies the retrieval and modification of the SUMO objects through an application protocol, allowing applications like vehicular communications, dynamic routing and traffic light control algorithms [8]. Furthermore, TraCI subscription and context subscription commands allow to retrieve several attributes of an object, or those of its surrounding objects, on every simulation step.

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) [6].

Depending on the application of interest, an implementation of TraCI can benefit from the programming language in which it is developed. In the case of applications involving control and optimization, Matlab® has proven to be an excellent alternative, featuring toolboxes for optimization, robust control and model predictive control, among others [9]. This motivated the development of an implementation of the TraCI protocol for the Matlab® programming language, namely TraCI4Matlab.

Particularly, TraCI4Matlab was proposed as a requirement in the MOYCOT project [11], where optimization-based traffic lights coordination strategies are being developed using Matlab®.

However, developing a new implementation of TraCI could be more expensive than doing it based on an existing one, especially taking into account the open-source nature of the later. In this regard, a re-engineering approach has many advantages over direct code translation either by hand or using semi-automated tools [7].

This chapter describes a re-engineering process of TraCI-Python used to implement TraCI4Matlab. This chapter is organized as follows: Sect. 2 describes the software re-engineering process related to software maintenance, reverse engineering, refactoring and forward engineering; Sect. 3 describes the reverse engineering sub-process of TraCI-Python and shows the extracted architectural and design component models obtained; Sect. 4 describes the refactoring tasks needed to adapt the obtained models to the constraints imposed by the Matlab® language and the subsequent forward engineering sub-process resulting in TraCI4Matlab; Sect. 5 shows results and discussion; finally, Sect. 6 shows conclusions and future work.

2 The Re-engineering Process

Chikofsky and Cross [3] define software re-engineering as:

> The examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form

They also state that the re-engineering process involves some form of reverse engineering, followed by some form of forward engineering and may include modifications related to new requirements.

It is important to note that software re-engineering is not only applied in legacy software, but also in cases where new requirements arise, such as [2] improving performance, exploiting new technologies and porting the subject software to a new platform.

In general, Chikofsky and Cross [3] related the re-engineering process with the software lifecycle and introduced formal definitions regarding the software transformations at different levels of abstraction. In the case of TraCI4Matlab, only the design level is considered, and the relationships with the software lifecycle take the form showed in Fig. 1, where the subject software corresponds to TraCI-Python and the new implementation, to TraCI4Matlab. Levels of abstraction refer to the representation of the software in the different phases of the cycle. For example, software is usually represented in terms of UML diagrams at the design phase, while in the implementation phase, the representation corresponds to the source code.

The taxonomy proposed by Chikofsky and Cross includes the following definitions, represented as sub-processes in Fig. 1:

- **Reverse-engineering**: In many cases, the subject software needs to be reverse-engineered because "usually, the system's maintainers were not its designers". Moreover, open-source software can be developed and extended in a distributed fashion by different developer teams. Therefore, reverse engineering enables

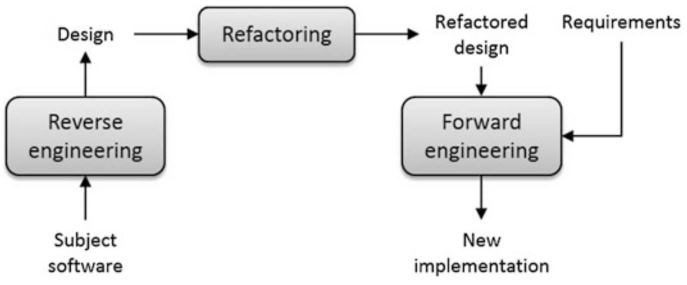


Fig. 1 Re-engineering process used in TraCI4Matlab

software comprehension by extracting artifacts from different levels of abstraction. In other words, reverse engineering allows to extract the knowledge of the software previous to its implementation, in a language that is easier to understand.

- **Restructuring**: It involves the "transformation from one representation to another at the same relative abstraction level, while preserving the subject system's external behavior". Thus, restructuring is related to the modification of the software without altering or extending its functionality, with the goal of improving its quality (structure, performance, etc.). In the case of Object-Oriented Programming (OOP), restructuring is known as refactoring [10].

Another important concept related to the software lifecycle is the use of software patterns, which have been defined by Vincke et al. as "The description of a general solution to a recurring problem" [19]. Hence, patterns allow to reuse the experience and best practices in the solution of common problems found in the development of a software product. Demeyer et al. proposed a set of Object Oriented Re-engineering Patterns [5], which they describe, as follows:

> Re-engineering patterns codify and record knowledge about modifying legacy software … We see re-engineering patterns as stable units of expertise which can be consulted in any re-engineering effort

Some of these re-engineering patterns were applied in the implementation of TraCI4Matlab, and will be explained in the subsequent sections that correspond to the sub-processes mentioned earlier.

Regarding the requirements considered for the implementation of TraCI4Matlab, there were two-fold:

- The migration to the Matlab® language: Naturally, the main requirement was to implement the TraCI API in Matlab®, taking into account its features and limitations.
- Preserving the TraCI-Python's end-user functions' structure: Since TraCI4Matlab was conceived to be open-source, it is important to simplify its use as much as possible. Therefore, the TraCI4Matlab's end-user functions' structure should be very similar to the TraCI-Python's.

It is important to note that there were not requirements related to performance, which favored the rapid release of TraCI4Matlab, this was done at the cost of a lower performance compared to TraCI-Python's, as it will be discussed later. Additionally, the implementation of TraCI4Matlab assumed that TraCI-Python is well structured, which means that the focus was not put on detecting code duplication or code smells [12].

The following sections explain the sub-processes involved in the implementation of TraCI4Matlab.

3 Reverse Engineering of the TraCI-Python Implementation

Re-engineering patterns used

Since the size of the subject software (TraCI-Python) is relatively small (around 4,300 lines of code) and the number of requirements was low, it was not necessary to apply many Object-Oriented Re-engineering Patterns. Particularly, the following patterns were used:

- **Chat with the maintainers**. According to Demeyer et al. [5] 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". However, in TraCI4Matlab it was not important to learn about the historical and political context of TraCI-Python because it is part of an active software project (SUMO) and the re-engineering effort is not related to quality improvements, but to the achievement of the requirements described previously, which are related to an extension of SUMO. Instead, in TraCI4Matlab, the pattern chat with the maintainers was applied to learn about the technical aspects of TraCI-Python that are not clear enough in the documentation or are part of exceptional cases. Therefore, in this case, the intent of this pattern could be stated more generally as: "Learn about the context and the technical aspects of the project through discussions with the people maintaining the system". It's important to note that this pattern couldn't be possible without the SUMO mailing list system.
- **Read all the code in one hour and skim the documentation**, whose intent is to "Assess the state of a software system by means of a brief, but intensive code review". These patterns were enough to approach to the recovery of the subject system's design (i.e. to identify the components of TraCI-Python, their responsibilities and how they collaborate) taking into account its small size.
- **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 [20], which is a graphical Python debugger, and StarUML [15], which is a program to draw UML diagrams. Thus, the TraCI4Traffic Lights tutorial, provided with the SUMO installation, was debugged with some modifications to understand all the components of the subject software.

These tasks helped to conclude 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. Those components take the form of namespaces, which, in Python, are accessed through the *dot* operator. Thus, TraCI-Python takes advantage of the fact that Python allows to associate variables, functions and classes to namespaces [1]. Figure 2 shows two UML

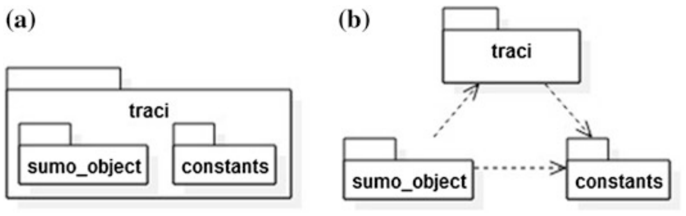


Fig. 2 TraCI-Python's components: **a** Deployment diagram, **b** Dependency diagram

package diagrams that represent the TraCI-Python's namespaces in terms of their deployment and their dependence relationships. Note that the *sumo_object* abstract package was defined to generalize the namespaces representing the SUMO objects, which have some variables and functions in common. Additionally, since in UML namespaces can be represented as packages, these terms will be used interchangeably in the remainder of this chapter.

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

3.1 The TraCI Package

This is the top-level package. It contains the namespaces corresponding to the SUMO objects (the modules variable) plus five public functions and others with, at most, package visibility. Through these functions, the responsibilities of the TraCI package could be extracted, being:

- Initialize and close the connection to the SUMO server through the functions `init()` and `close()`.
- Allow several SUMO instances to be controlled by the same client and switching among the corresponding connections, thanks to the port argument in the `init()` function and the `switch()` function.
- Perform a simulation step through the `simulationStep()` function, including a `step` argument, which allow to increase or decrease the simulation step in milliseconds.
- Populate the subscription results related to each SUMO object, using the `readSubscription()` function.
- Construct and send the outgoing messages according to the TraCI protocol, through a set of functions beginning with the word `send`, which have been grouped in an abstract function called *sender* for illustration purposes. The *sender* functions prepare the `message` variable according to the desired data type to send to the SUMO server.
- Read the responses from the SUMO server and check them for errors throwing the corresponding exceptions, using the `recvExact()` function.

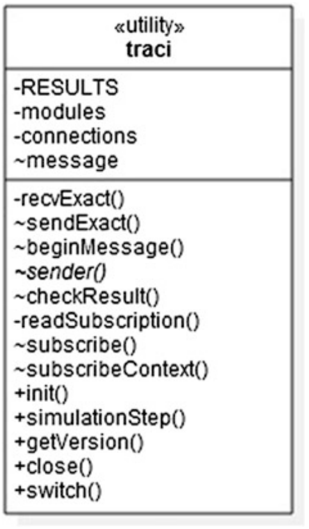


Fig. 3 Variables and functions in the TraCI package

Figure 3 shows an UML class diagram including the functions of the TraCI package. Note that the utility stereotype has been used, since those functions do not belong to a class but to a namespace.

3.2 Packages Corresponding to the SUMO Objects

These packages can be briefly summarized through the so called getters and setters which allow the end user to retrieve and modify the properties of the objects in the SUMO simulation. 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 functions are defined to represent the set of public functions 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 4 shows an UML sequence diagram, which is an example of the above process in the case of a `getter`. Note how the different components collaborate: the end user calls the `get_wrapper()` which calls the universal getter of the `sumo_object` component, 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 5 shows a class diagram corresponding to the abstract class `sumo_object`. It is worth to notice, that this abstract class was not

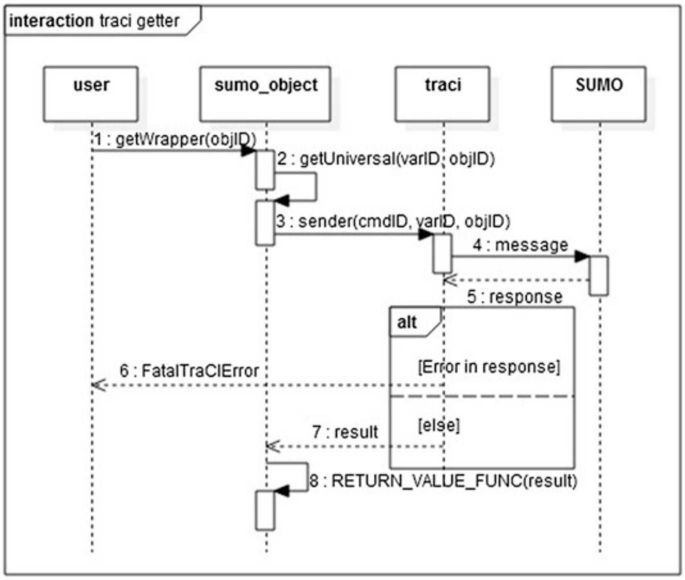


Fig. 4 UML sequence diagram for the get process in TraCI-Python

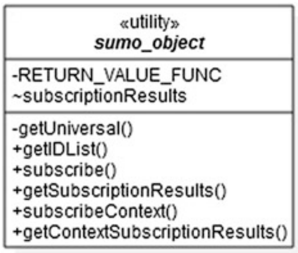


Fig. 5 Abstract package *sumo_object* representing the namespaces corresponding to SUMO objects

physically implemented, but serves as a way to explain the packages corresponding to the SUMO objects and their variables and functions in common.

3.3 TraCI Constants

This is a namespace containing the command, variable-type and data-type codes as constants from the TraCI protocol specification. TraCI-Python's components use these constants as parameters for their functions. For example, referring to Fig. 4, the parameters `varID` and `cmdID` are taken from the TraCI constants module.

4 Forward Engineering Sub-process

4.1 Re-engineering Patterns Used

As stated in the reverse engineering sub-process, because of the relatively small size of the subject system and the low number of requirements, some re-engineering patterns were not necessary. Moreover, since the official Matlab® unit testing framework is quite new [13] and taking into account that the majority of TraCI-Python's functions are not associated to a class, as explained in the previous section, tests were created in a single m-file that reproduces the TraCI4 Traffic Lights tutorial. Therefore, patterns related to *use a testing framework* were not applied. Additionally, recall that the focus of the re-engineering process is on the migration of TraCI-Python, not on the improvement of its quality. Hence, the patterns related to *write tests to understand* were not used. Taking into account these considerations, the following patterns were applied:

- **Write tests to enable evolution**: The test created in the implementation of TraCI4Matlab allowed to identify the limitations of the Matlab® programming language that prevented a direct implementation of the subject software's recovered design and, consequently, the necessary refactoring tasks to perform. Thus, according to Demeyer et al. [5], the risk of "failing to accommodate future change" was mitigated.
- **Grow your tests base incrementally**: Every time a TraCI4Matlab's component was implemented, the corresponding test was modified to incorporate the new functionalities, which enabled to *always have a running version*.
- **Conserve familiarity**: The requirement related to the preservation of the TraCI-Python's end-user functions' structure enabled to *conserve familiarity*.
- **Use profiler before optimizing**: During the implementation of TraCI4Matlab, it was noticed that its performance was lower than TraCI-Python's. Later, the stakeholders concluded that a new requirement was necessary to address this issue. In this regard, the Matlab® profiler helped to identify the bottleneck that caused the low performance. Consequently, the refactoring tasks needed to satisfy the related requirement could be identified, as will be explained in brief.

It was found in the implementation and testing phase of TraCI4Matlab that the Matlab® language specification has some limitations forcing 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. In contrast, Python allows to have namespace's 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 functions in other namespaces.

In order 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 [4]. Note that this option conflicts with property 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 [14]. 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 does not 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 functions that require them and can be accessed by any other function.

There were some special cases where there was no need to use global variables. For example, it was found that some variables were used only by one function. Therefore, those variables were defined inside the functions that use them. Another case is related to the `RETURN_VALUE_FUNC` dictionary of the *sumo_object* packages, which has constant values. In this case, a corresponding new class with only constant attributes was defined. Finally, it was found that the `modules` variable of the TraCI package only was used in two functions of the same package: `readSubscription()` and `simulationStep()`. The `modules` variable 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()` function, the `modules` variable 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()` function. On the other hand, the `modules` variable is used in the `simulationStep` module only to reset its values, i.e. the subscription results of each *sumo_object* namespace. Note that, in this case, it is not necessary to define a map. Therefore, a new array called `modules` was defined in the `readSubscription()` function.

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

Figure 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 variables associated to namespaces are defined in the functions that use them, e.g. the `RESULTS` and `modules` attributes of the TraCI package.

However, as it was explained before, TraCI4Matlab's performance resulted to be worse than TraCI-Python's. Figure 8 shows performance results of both implementations using the cProfile module in the case of TraCI-Python and the Matlab® profiler in the case of TraCI4Matlab. It can be seen that TraCI4Matlab spends much time in sending and receiving messages through the TCP-IP implementation, which is part of the instrument control toolbox. Particularly, in TraCI-Python the `_sendExact()` and `_recvExact()` functions sum 4.903 s while in TraCI4Matab, they sum 119.147 s.

Taking advantage of the high integration capacity of Matlab® and Java, the proposed solution was to develop a new TCP-IP implementation for TraCI4Matlab using Java sockets. The solution involved the creation of a `Socket` class in Matlab® that wraps a Java socket and uses a `DataReader` class [17] which enables to read the entire buffer of the input stream. Figure 9 shows the performance of TraCI4Matlab including the implementation of the `Socket` class. It can be seen that, using Java sockets, the TraCI4Matlab's performance improved substantially. In this case, the `_sendExact()` and `_recvExact()` functions sum 16.711 s, which represent a performance improvement of 85.97 %.

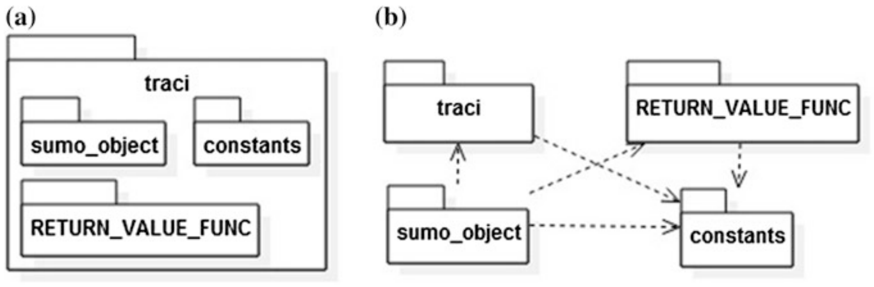


Fig. 6 TraCI4Matlab's components: **a** Deployment diagram, **b** Dependency diagram

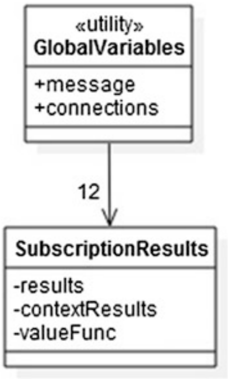
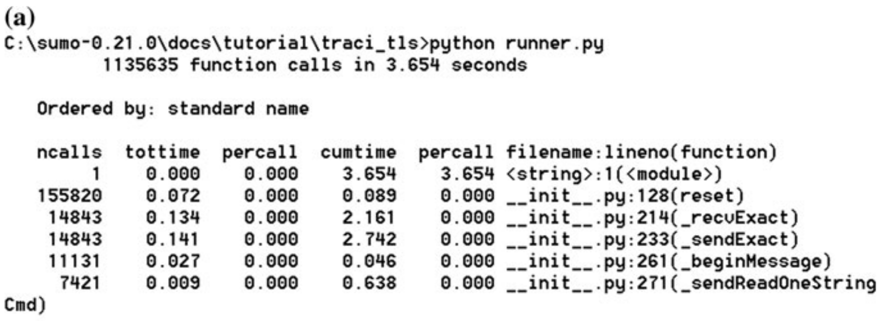


Fig. 7 Global variables defined in TraCI4Matlab

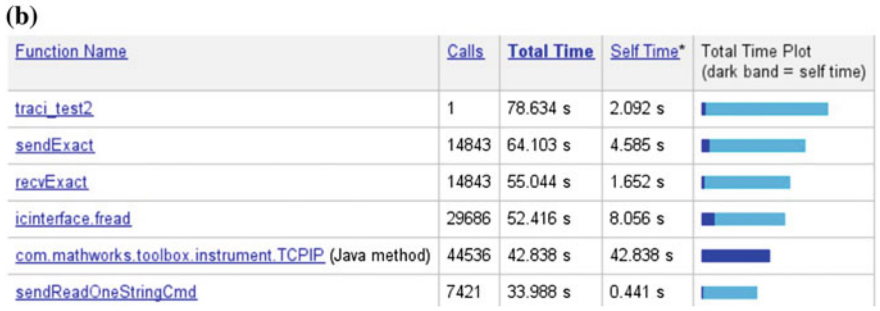


(a)

```
C:\sumo-0.21.0\docs\tutorial\traci_tls>python runner.py
         1135635 function calls in 3.654 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    3.654    3.654 <string>:1(<module>)
   155820    0.072    0.000    0.089    0.000 __init__.py:128(reset)
    14843    0.134    0.000    2.161    0.000 __init__.py:214(_recvExact)
    14843    0.141    0.000    2.742    0.000 __init__.py:233(_sendExact)
    11131    0.027    0.000    0.046    0.000 __init__.py:261(_beginMessage)
     7421    0.009    0.000    0.638    0.000 __init__.py:271(_sendReadOneString
Cmd)
```



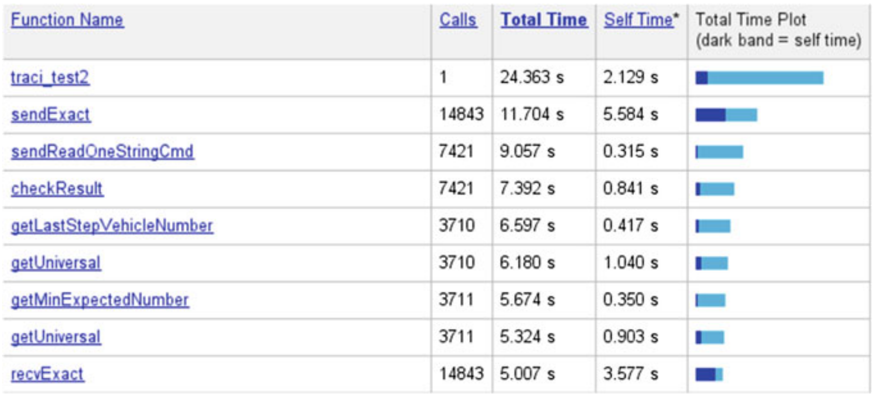
(b)

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
traci_test2	1	78.634 s	2.092 s	
sendExact	14843	64.103 s	4.585 s	
recvExact	14843	55.044 s	1.652 s	
icinterface.fread	29686	52.416 s	8.056 s	
com.mathworks.toolbox.instrument.TCPIP (Java method)	44536	42.838 s	42.838 s	
sendReadOneStringCmd	7421	33.988 s	0.441 s	

Fig. 8 Performance results of **a** TraCI-Python and **b** TraCI4Matlab

5 Results and Discussion

TraCI4Matlab was released on December 24, 2013 under the BSD license. It is free software and is available for the community at Matlab Central [18], or as part of the SUMO contributed tools since SUMO 0.20.0.



Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
traci_test2	1	24.363 s	2.129 s	
sendExact	14843	11.704 s	5.584 s	
sendReadOneStringCmd	7421	9.057 s	0.315 s	
checkResult	7421	7.392 s	0.841 s	
getLastStepVehicleNumber	3710	6.597 s	0.417 s	
getUniversal	3710	6.180 s	1.040 s	
getMinExpectedNumber	3711	5.674 s	0.350 s	
getUniversal	3711	5.324 s	0.903 s	
recvExact	14843	5.007 s	3.577 s	

Fig. 9 Performance results of TraCI4Matlab including a new TCP-IP implementation using Java sockets

Currently, TraCI4Matlab is being used in the project "Modelling and Control of Urban Traffic in the City of Medellin (MOYCOT)" [11]. 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 in the city of Medellin. 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 an isolated junction, showed in Fig. 10. 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 Fig. 11.

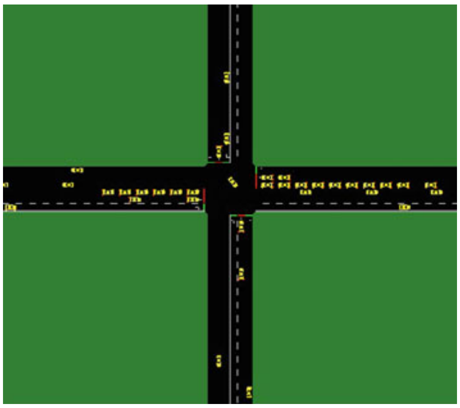


Fig. 10 The isolated junction scenario used in the MOYCOT [11] project to obtain parameters needed for a MPC traffic lights controller

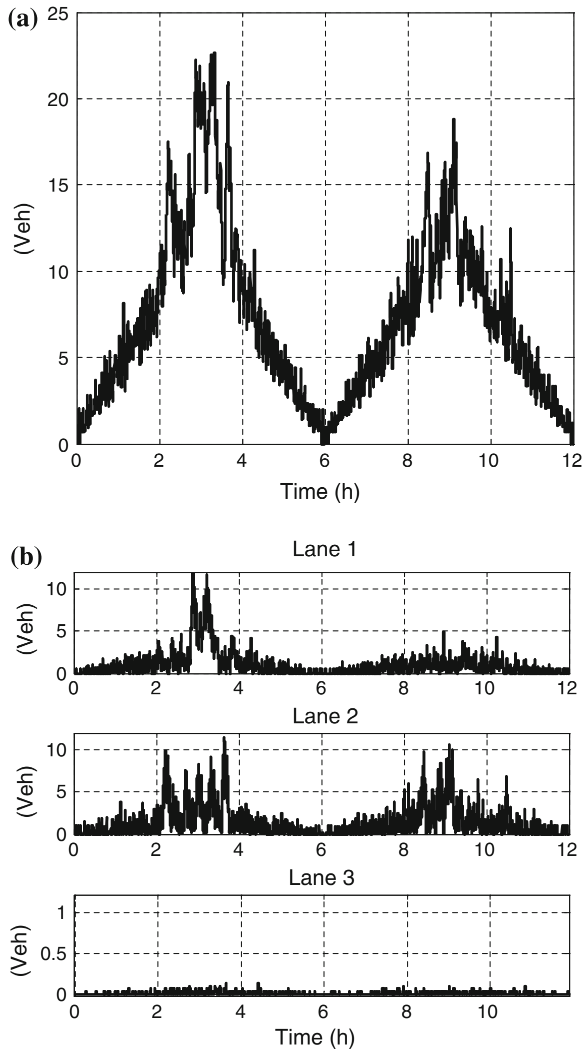


Fig. 11 Data obtained in the north-south edge using TraCI4Matlab: **a** Number of vehicles entering the edge, **b** Length of the queue on each lane in vehicles

6 Conclusions

In this chapter, the re-engineering process of the TraCI API's Python implementation (TraCI-Python) 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.

The re-engineering process was supported with *object-oriented re-engineering patterns*, which, in some cases, had to be adapted to the specific case of TraC-I4Matlab. These patterns provide useful guidelines for a re-engineering project, including small-sized projects like TraCI4Matlab.

One of the requirements formulated for TraCI4Matlab was to preserve the same structure of the TraCI-Python's end-user's functions. Although it could be accomplished through the approach described in the forward engineering process, performance implications were not considered. As a result, it was found that performance of TraCI4Matlab was much lower than TraCI-Python's. In order to overcome this issue, a TCP/IP implementation using Java sockets was proposed, which resulted in a substantial performance improvement.

However, the re-engineering process was focused on the migration of TraCI-Python to Matlab®, without taking into account the quality of the subject software in terms of its structure (namespaces as classes, code duplication and code smells). Future work should concentrate on this topic by using (semi) automated tools and possibly including a benchmark with TraCI4J. Further performance improvements should be also considered.

Finally, the design obtained through reverse engineering suggests some private functions and some others with package visibility. Although Matlab® allows to define private functions, it has not defined, to date, a similar approach for the case of functions with package visibility.

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.

References

1. Beazley DM (2009) Python essential reference, 4th edn. Addison-Wesley Professional, Upper Saddle River
2. CanforaHarman G, Di Penta M (2007) New frontiers of reverse engineering. In: Future of software engineering, FOSE '07. IEEE Computer Society, Washington, DC, USA, pp 326–341. doi:10.1109/FOSE.2007.15
3. Chikofsky EJ, Cross I JH (1990) Reverse engineering and design recovery: a taxonomy. IEEE Softw 7:13–17. doi:10.1109/52.43044
4. Comparing MATLAB with other oo languages—MATLAB and simulink, n.d. URL http://www.mathworks.com/help/matlab/matlab_oop/matlab-vs-other-oo-languages.html. Accessed 02 April 2014
5. Demeyer S, Ducasse S, Nierstrasz O (2002) Object-oriented reengineering patterns. Morgan Kaufmann, San Francisco
6. egueli/TraCI4J GitHub, n.d. URL https://github.com/egueli/TraCI4J. Accessed 01 April 2014
7. Ewer J, Knight B, Cowell D (1995) Case study: an incremental approach to re-engineering a legacy {FORTRAN} computational fluid dynamics code in C ++. Adv Eng Softw 22:153–168. doi:http://dx.doi.org/10.1016/0965-9978(95)00021-N
8. Krajzewicz D, Erdmann J, Behrisch M, Bieker L (2012) Recent development and applications of SUMO—Simulation of Urban mobility. Int J Adv Syst Meas 5:128–138

9. MATLAB—the language of technical Computing—B, n.d. URL http://www.mathworks.com/products/matlab/. Accessed 09 Sept 2014
10. Mens T, Tourwe T (2004) A survey of software refactoring. IEEE Trans Softw Eng 30:126–139. doi:10.1109/TSE.2004.1265817
11. MOYCOT | MOYCOT, n.d. URL http://www.moycot.org/. Accessed 09 Sept 2014
12. Olbrich S, Cruzes DS, Basili V, Zazworka N (2009) The evolution and impact of code smells: a case study of two open source systems. In: Proceedings of the 2009 3rd international symposium on empirical software engineering and measurement, ESEM '09. IEEE Computer Society, Washington, DC, USA, pp 390–400. doi:10.1109/ESEM.2009.5314231
13. Release notes for MATLAB—MATLAB and simulink, n.d. URL http://www.mathworks.com/help/matlab/release-notes.html. Accessed 09 Sept 2014
14. Share data between workspaces—MATLAB and simulink, n.d. URL http://www.mathworks.com/help/matlab/matlab_prog/share-data-between-workspaces.html. Accessed 02 April 2014
15. StarUML—The open source UML/MDA platform, n.d. URL http://staruml.sourceforge.net/en/. Accessed 30 Jan 2014
16. SUMO_User_Documentation—SUMO—simulation of urban mobility, n.d. URL http://sumo-sim.org/userdoc/. Accessed 30 Jan 2014
17. TCP/IP socket communications in MATLAB using java classes—file exchange—MATLAB central, n.d. URL http://www.mathworks.com/matlabcentral/fileexchange/file_infos/25249-tcp-ip-socket-communications-in-matlab-using-java-classes. Accessed 09 Sep 2014
18. TraCI4Matlab—file exchange—MATLAB central, n.d. URL http://www.mathworks.com/matlabcentral/fileexchange/file_infos/44805-traci4matlab. Accessed 01 April 2014
19. Vincke R, Van Landschoot S, Steegmans E, Boydens J (2012) Refactoring sequential embedded software for concurrent execution using design patterns. Annu J Electron 6:157–160
20. Winpdb—A platform independent python debugger, n.d. URL http://winpdb.org/. Accessed 30 Jan 2014

An Integrated Framework for Mobile-Based ADAS Simulation

João S.V. Gonçalves, João Jacob, Rosaldo J.F. Rossetti, António Coelho and Rui Rodrigues

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

J.S.V. Gonçalves · R.J.F. Rossetti (✉)
Departamento de Engenharia Informática, Faculdade de Engenharia da Universidade do Porto, Laboratório de Inteligência Artificial e Ciência de Computadores, R. Dr. Roberto Frias s/n, 4200-465 Porto, Portugal
e-mail: rossetti@fe.up.pt

J.S.V. Gonçalves
e-mail: joao.sa.vinhas@fe.up.pt

J. Jacob · A. Coelho · R. Rodrigues
Departamento de Engenharia Informática, Faculdade de Engenharia da Universidade do Porto, INESC TEC, R. Dr. Roberto Frias s/n, 4200-465 Porto, Portugal
e-mail: joao.jacob@fe.up.pt

A. Coelho
e-mail: acoelho@fe.up.pt

R. Rodrigues
e-mail: ruirodrig@fe.up.pt

1 Introduction

The technological advances in both the mobile and the 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 Multi Agent 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 ADAS in a simulated environment rather than in real-world scenarios. 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 and 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 Sect. 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 Sect. 5. We finish this paper with a set of conclusions and interesting future work.

2 Background and 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, which 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 an 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 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 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 in [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° full dome with 3D scenes from real city area has been developed in [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.

3 Methodological Approach

The proposed system architecture is as described in Fig. 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 Fig. 1 corresponds to the developed components as of the writing of this paper.

3.1 Simulators and 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 Open Street Maps, Google Geolocation API, Google Altitude API and other data providers in order to fetch the required geographical information of a given location and

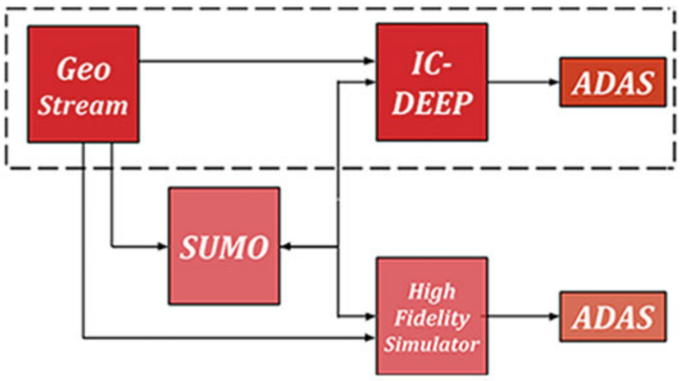


Fig. 1 Overview of the system's architecture

remodel it in a fashion which can be interpreted by all the simulators in a coherent and consistent way. This is especially important to the SUMO microscopic simulator as the raw network data imported from Open Street Maps, 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. The Geostream framework is explained in further detail in the following section.

3.2 The Geostream Framework

The Geostream Framework was originally designed to aggregate location-based data from multiple sources and recreate urban and rural environments for use in location-based games. One of the issues in location-based games is the need to accurately determine the user's current context. As such, the Geostream provides a foundation on top of which location-based games can be developed. However, since the information is of use to other areas (simulation, procedural modeling), the framework was further altered so that it can be used in non-game development.

The framework accesses the following web services, using them as providers for the respective context-sensitive data (Table 1):

The data retrieved by these sources is then combined as a means to recreate the current location context of the player (or in this case, the user). Certain sources may be optionally not used if their data adds little to the problem at hand. After loading data from the above sources, the scene graph looks similar to that of Fig. 2.

As Fig. 2 depicts, the typical scene graph of a Geostream based application consists of a World Map gameobject, and other several Geo-gameobjects (objects not based on the external sources of data, but that are placed in the game world, such as a player, enemies, cars, etc.). The World Map game object consists of

Table 1 Geostream currently used services and their data

Service	Data requested
Open street maps	Human structures information and meta-data (buildings, roads, bridges, traffic lights…) of a given area
Microsoft bing maps	Aerial photos and traffic information of given area
Google places API	Information about POIs (Points of Interest) in given area
Google elevation API	Altitude of given coordinates
Google geocoding API	Coordinates of a given address
Open weather	Detailed, current weather information of given location
Map quest	Aerial photos of a given area

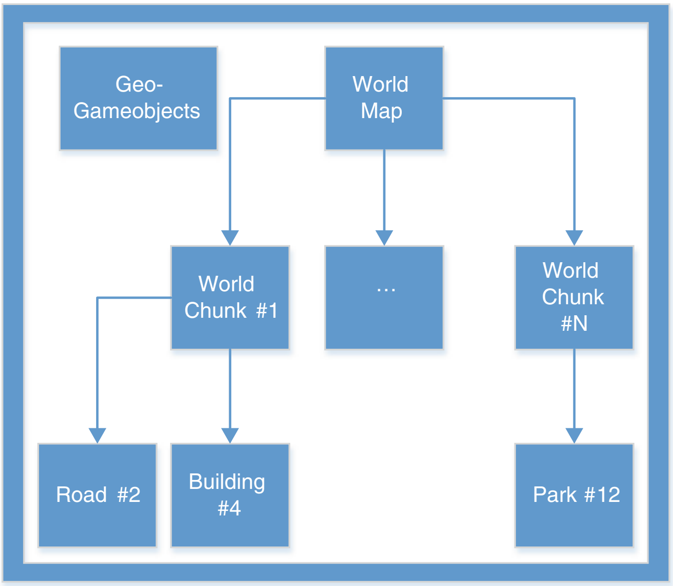


Fig. 2 Example of a Geostream scene graph

several Map chunks. Each Map chunk holds the information of a specific part of a location, such as the geometry of the terrain of that part (its elevation) and its aspect (the corresponding part of an aerial photo).

As Fig. 3 shows, a World Map is comprised of several World Chunks. In the above image, the grid mesh that is drawn with a blue color, represents the limits of a given World Chunk. In this case, the World Map consists of 16 World Chunks, each responsible for computing the information that belongs to itself (not only its geometry or texture, but those of the structures that belong to it: POIs, roads, buildings, natural structures, etc.). The behaviour of each World Chunk is summarized in the fluxogram of Fig. 4.

When a World Map is created, using a certain coordinate or address as a center reference, it will procede to create NxN World Chunks, each with a predefined width and height of fractions of degrees. Then, each World Chunk will proceed to look at the information it needs to reconstruct itself. Additionally, a World Map can have a Geo-Gameobject to "follow". This means that as the referenced Geo-Gameobject moves around, new World Chunks may be dynamically loaded, while others are unloaded to preserve memory.

New sources of information can be added to a World Chunk for it to make use of. Additionally, it is possible to change, in design time, how certain types of meta data are processed. For instance, in Unity, in the editor view, one can visualize what current tags of Open Street Maps or Google Places are being processed and how (as it can be seen in Fig. 5).

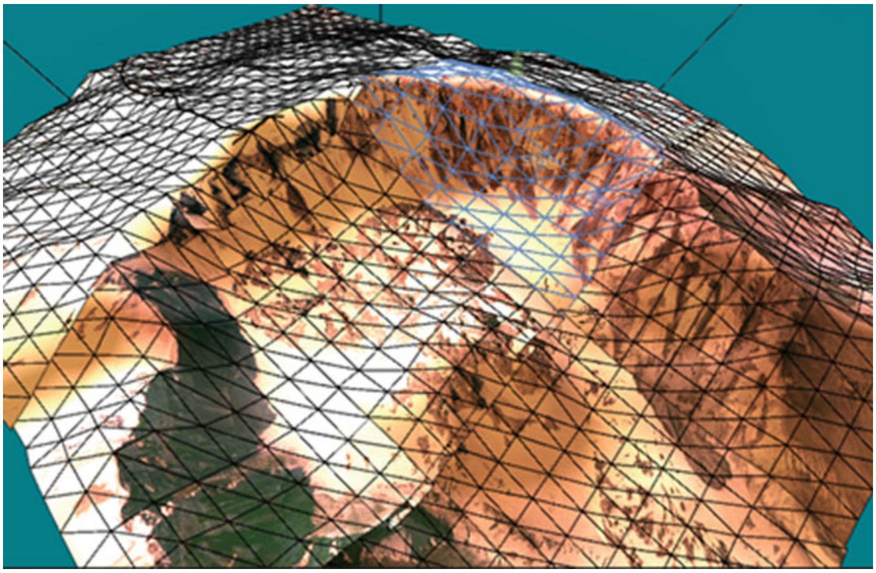


Fig. 3 Screenshot of the reconstruction of the Mount St. Helens crater

As seen in Fig. 5, a dictionary is created in design time, pairing tags with the script classes responsible for treating information related to a specific tag. When the world object is created, it will instantiate, through C# Reflection feature, the necessary scripts (all derived from the Structure Script super-class). So, whenever a structure's information with a certain tag is compiled after consulting the multiple sources, it is passed on to the respective script for treatment. In Fig. 5, we can see that "residential", "motorway", "bridge" and "highway" are all treated by the Road Script, a script responsible for generating roads with different features. This allows for further expanding the possibilities or procedurally generating other real-world objects based on the information compiled from external sources. For instance, one could add the tag "traffic light", present in the information gathered via Open Street Maps, coupled with a Traffic Light Script that would be capable of treating that specific data. It could, for example, instantiate a semaphore prefab in that structure's position. Same thing could be done with trees, lamps, and other simple and repeatable structures.

The biggest drawback of Geostream Framework, is that it is limited by the availability of the context-related sources, and the quality (or existence) of information. Performance wise, as most operations are threaded, loading of World Chunks is made in a smooth manner, albeit for areas with many structure-related information available, it can take some time to generate all the needed geometry (even so, the FPS of the simulation rarely becomes affected). In locations with abundant information and with the needed scripts performing procedural generation of structures, the results are both visually appealing and accurate (note that generated structures fit with the structures captured by the aerial photos).

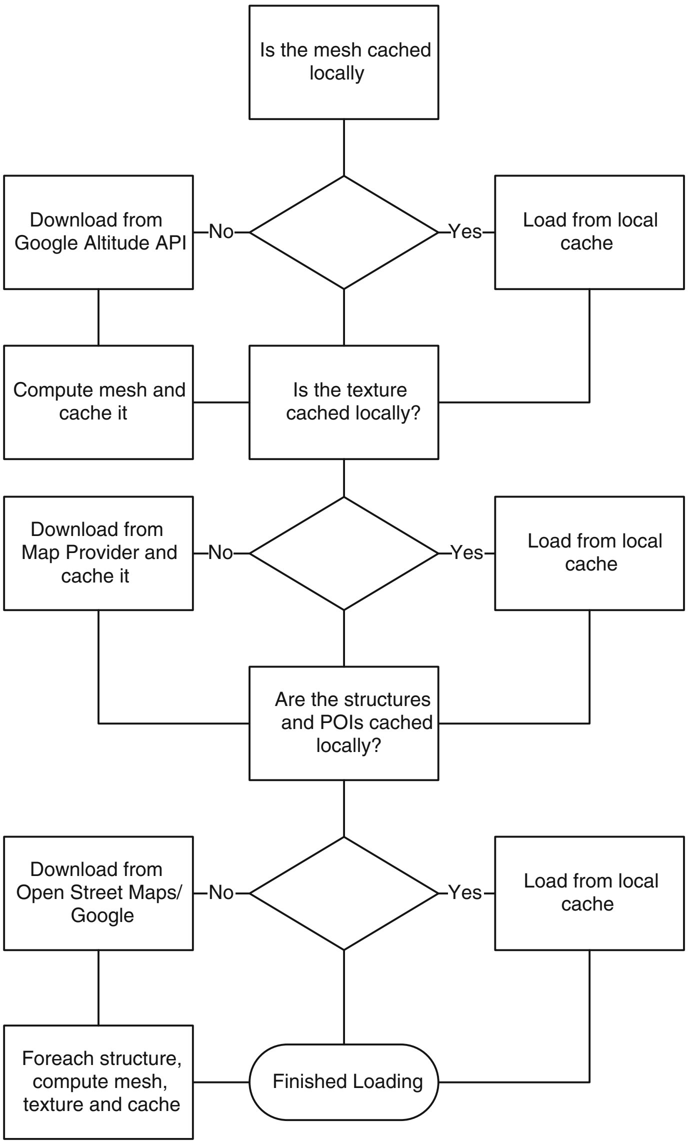
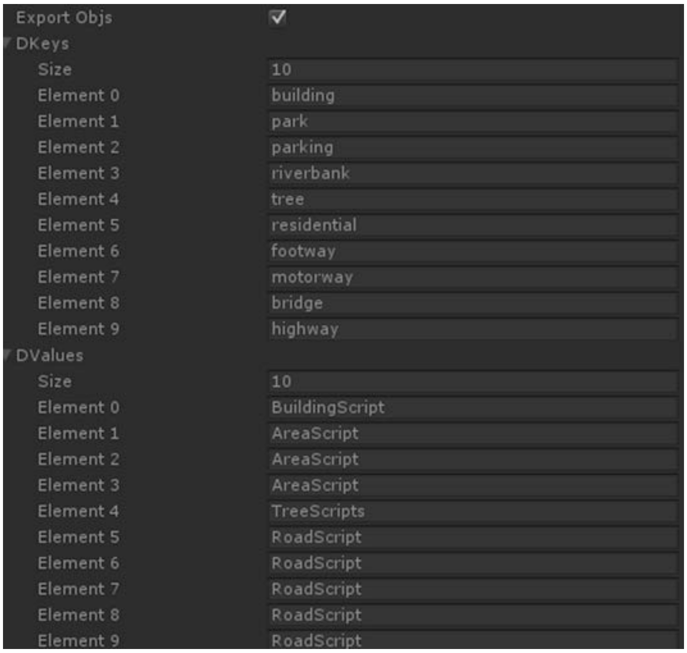

Is the mesh cached locally
Download from Google Altitude API
No
Yes
Load from local cache
Compute mesh and cache it
Is the texture cached locally?
Download from Map Provider and cache it
No
Yes
Load from local cache
Are the structures and POIs cached locally?
Download from Open Street Maps/ Google
No
Yes
Load from local cache
Foreach structure, compute mesh, texture and cache
Finished Loading

Fig. 4 Depiction of the loading behavior of a World Chunk

Notice how in Fig. 6, some buildings are red, while some are not. Red buildings are those that have all their geometry's height defined at the source, while textured buildings have their height generated based on the height of neighboring buildings and those whose heights are known from the source. Buildings with extra detail can also be computed by further developing the Building Script, associated with the creation of those structures.



Export Objs	✓
DKeys	
Size	10
Element 0	building
Element 1	park
Element 2	parking
Element 3	riverbank
Element 4	tree
Element 5	residential
Element 6	footway
Element 7	motorway
Element 8	bridge
Element 9	highway
DValues	
Size	10
Element 0	BuildingScript
Element 1	AreaScript
Element 2	AreaScript
Element 3	AreaScript
Element 4	TreeScripts
Element 5	RoadScript
Element 6	RoadScript
Element 7	RoadScript
Element 8	RoadScript
Element 9	RoadScript

Fig. 5 Subsection of the World Map behaviour script

3.3 *Mobile Device*

The Android service sets the current GPS location using *Mock Location* 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 Figs. 1 and 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 Fig. 7.

3.4 *Interaction of Driving Simulators and Android*

A typical interaction between the simulators and the mobile devices is shown in Fig. 8. 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.

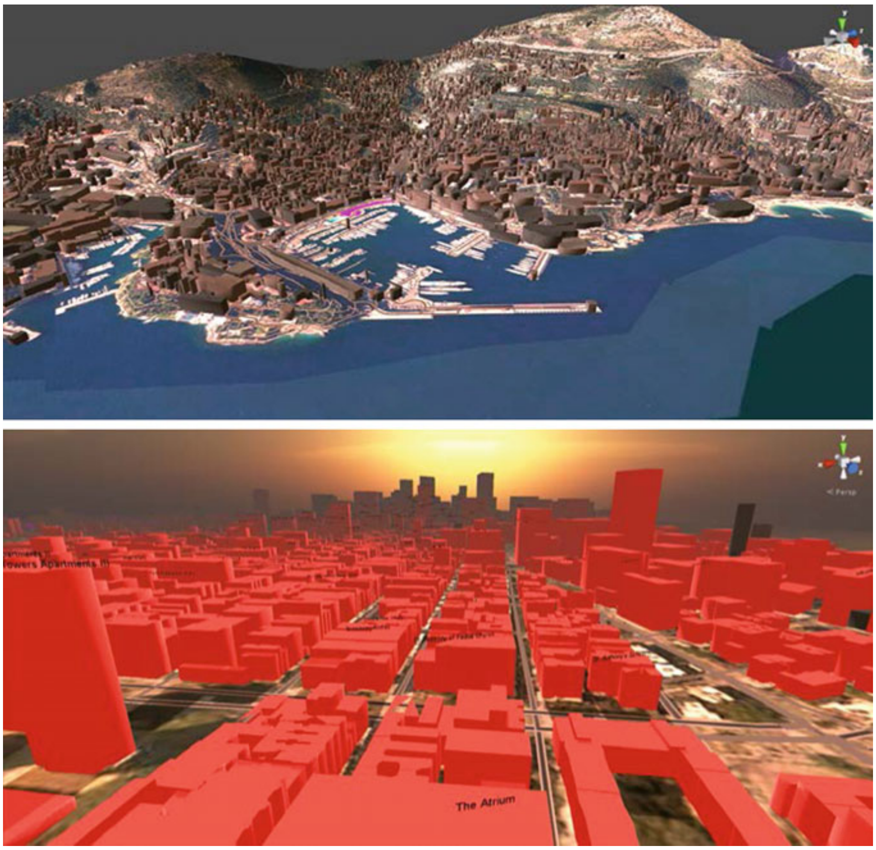


Fig. 6 Geostream procedural generation of Monaco (*above*) and New York (*below*)

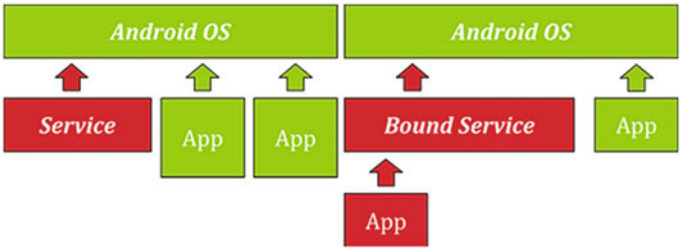


Fig. 7 Standalone mobile application (*left*) and mobile application with included library (*right*)

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.

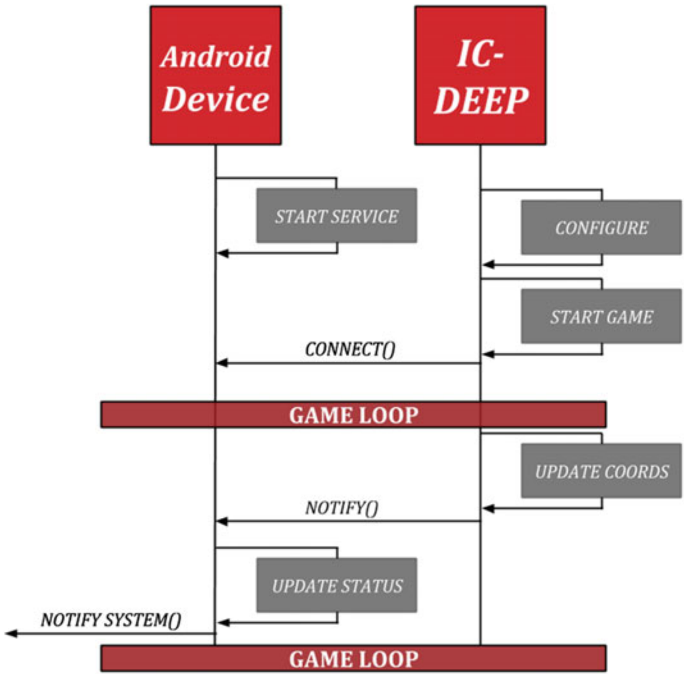


Fig. 8 Typical interaction between IC-DEEP simulator and an ADAS

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.

4 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 Fig. 1, the remainder of the system will be developed later on, as mentioned in the next section. We have divided the verification into two independent tests. Both of the tests were performed in the same geographical location, which was down-town Porto, on *Avenida dos Aliados*, as seen in Fig. 9.

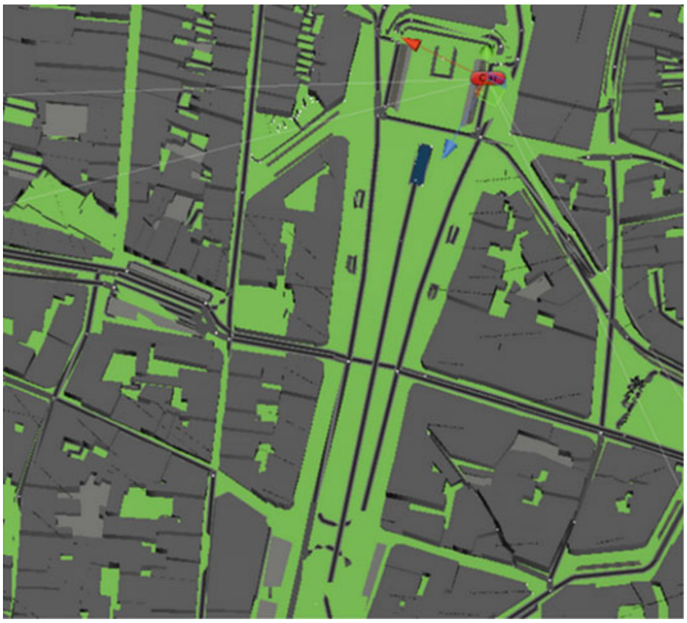


Fig. 9 Generated 3D Scene ortographic view

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.

4.1 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 overlayered a visual representation of the obtained traces on the simulator and on Google Earth, both results can be seen in Fig. 10. 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 Fig. 10 (see labels 1 and 2), 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 been seen in both the simulator and Google Earth alike.

Apart from testing the fidelity of the simulation with GPS trace logs, it is also noticeable that the orthographic view of the generated 3D scene very much resembles the satellite image of the same location, as it can be seen in Fig. 11.

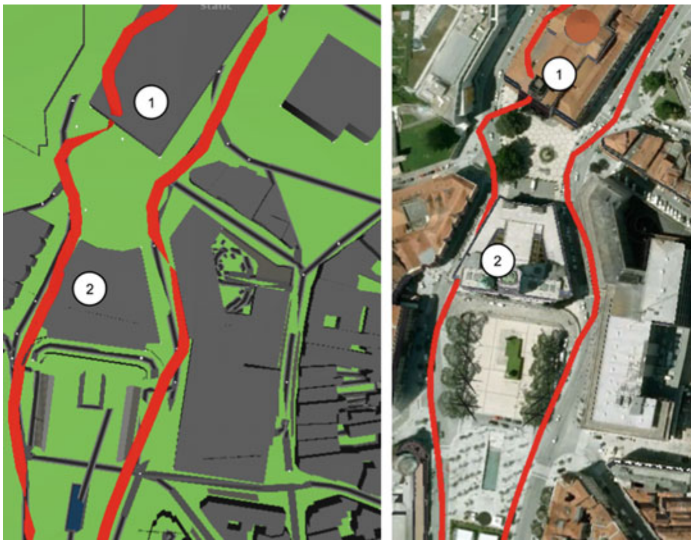


Fig. 10 GPS traces on the simulator (*left*) and Google Earth (*right*)

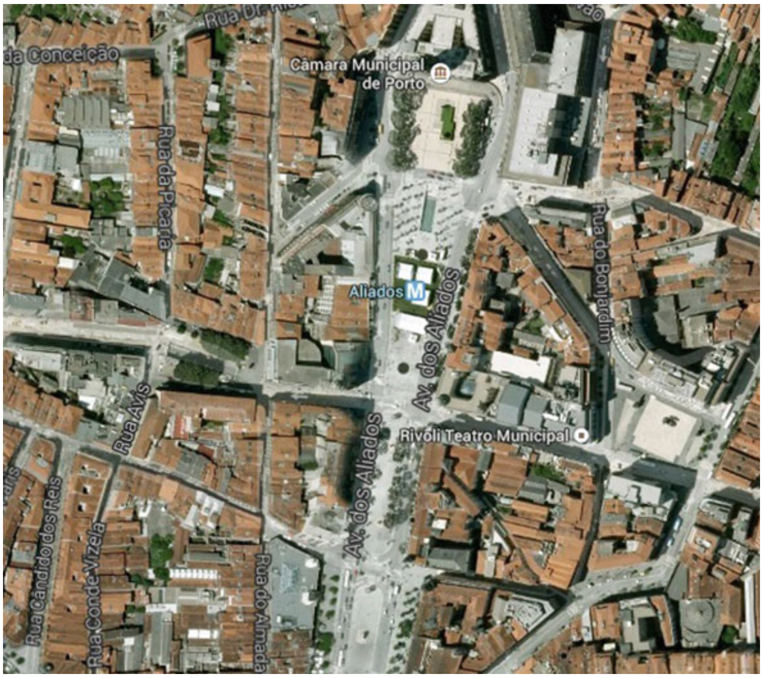


Fig. 11 Satellite image of Av. dos Aliados

4.2 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 Fig. 12.

The interaction between the simulator and the mobile application followed that of Fig. 8. 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.

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*.

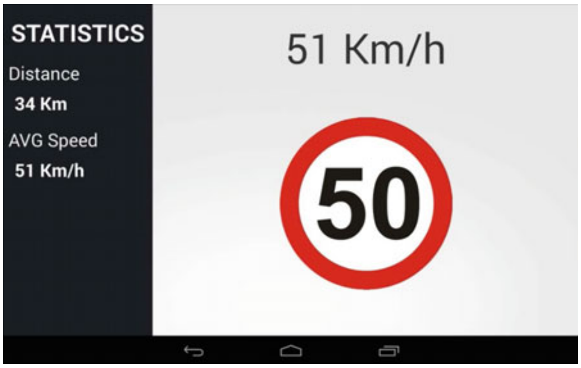


Fig. 12 Developed ADAS

5 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 behaviors 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 proprietary 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, especially regarding road generation and also importing models and textures for different buildings.

References

1. Baumann M, Keinath A, Krems JF, Bengler K (2004) Evaluation of in-vehicle HMI using occlusion techniques: experimental results and practical implications. Appl Ergon 35(3):197–205
2. Goncalves J, Rossetti RJF, Olaverri-Monreal C (2012) IC-DEEP: a serious games based application to assess the ergonomics of in-vehicle information systems. In: 2012 15th International IEEE conference on intelligent transportation systems (ITSC), pp 1809–1814
3. Wang F-Y (2003) Integrated intelligent control and management for urban traffic systems. In: Intelligent transportation systems, 2003. Proceedings 2003 IEEE, vol 2, pp 1313–1317

4. Wang F-Y, Tang S (2005) A framework for artificial transportation systems: from computer simulations to computational experiments. In: Intelligent transportation systems, 2005. Proceedings of IEEE, pp 1130–1134
5. Rossetti RJF, Liu R, Tang S (2011) Guest editorial special issue on artificial transportation systems and simulation. IEEE Trans Intell Transp Syst 12(2):309–312
6. Macedo J, Kokkinogenis Z, Soares G, Perrotta D, Rossetti RJF (2013) A HLA-based multi-resolution approach to simulating electric vehicles in simulink and SUMO. In: 2013 16th International IEEE conference on intelligent transportation systems—(ITSC), pp 2367–2372
7. Punzo V, Ciuffo B (2011) Integration of driving and traffic simulation: issues and first solutions. IEEE Trans Intell Transp Syst 12(2):354–363
8. Behrisch M, Bieker L, Erdmann J, Krajzewicz D (2011) Sumo-simulation of urban mobility-an overview. In: The third international conference on advances in system simulation SIMUL 2011, pp 55–60
9. Maia R, Silva M, Araujo R, Nunes U (2011) Electric vehicle simulator for energy consumption studies in electric mobility systems. In: 2011 IEEE forum on integrated and sustainable transportation system (FISTS), pp 227–232
10. Gomes P, Olaverri-Monreal C, Ferreira M, Damas L (2011) Driver-centric VANET simulation. In: Communication technologies for vehicles, Springer, Germany, pp 143–154
11. Kern D, Müller M, Schneegaß S, Wolejko-Wolejszo L, Schmidt A (2008) CARS-Configurable automotive research simulator. In: Mensch and computer workshop band, pp 256–260
12. Miao Q, Zhu F, Lv Y, Cheng C, Chen C, Qiu X (2011) A game-engine-based platform for modeling and computing artificial transportation systems. IEEE Trans Intell Transp Syst 12 (2):343–353
13. Hassan B, Berssenbrugge J, Al Qaisi I, Stocklein J (2013) Reconfigurable driving simulator for testing and training of advanced driver assistance systems. In: 2013 IEEE International symposium on assembly and manufacturing (ISAM), pp 337–339
14. Noth S, Edelbrunner J, Iossifidis I (2012) An integrated architecture for the development and assessment of ADAS. In 2012 15th International IEEE conference on intelligent transportation systems (ITSC), pp 347–354
15. Gruyer D, Pechberti S, Glaser S (2013) Development of full speed range ACC with SiVIC, a virtual platform for ADAS prototyping, test and evaluation. In: 2013 IEEE intelligent vehicles symposium workshops (IV Workshops), pp 93–98
16. Yu S, Lee S-Y, Kim M-S, Lee D-G (2006) Development and evaluation of ITS devices using KAAS(KATECH Advanced Automotive Simulator) system. In: International joint conference SICE-ICASE, 2006, pp 2116–2120
17. Pereira JLF, Rossetti RJF (2012) An integrated architecture for autonomous vehicles simulation. In: Proceedings of the 27th annual ACM symposium on applied computing, pp 286–292
18. El Najjar M, Bonnifait P (2005) A road-matching method for precise vehicle localization using belief theory and kalman filtering. Auton Robots 19(2):173–191
19. Rossetti RJF, Almeida JE, Kokkinogenis Z, Goncalves J (2013) Playing transportation seriously: applications of serious games to artificial transportation systems. Intell Syst IEEE 28 (4):107–112

Part III
Data Generation and Validation

TOMS—Traffic Online Monitoring System for ITS Austria West

Karl-Heinz Kastner and Petru Pau

Abstract ITS Austria West is a long-term Austrian project whose goal is to continuously generate and publish real-time traffic data. TOMS—a traffic monitoring system developed in the frame of this project—integrates sensor data, coming in real time from various data sources, into periodical snapshots of the traffic situation. 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 original 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

1 Introduction

In the frame of the "ITS Austria West" project 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.

K.-H. Kastner · P. Pau (✉)
RISC Software GmbH, Softwarepark 35, 4232 Hagenberg, Austria
e-mail: petru.pau@risc-software.at

K.-H. Kastner
e-mail: karl-heinz.kastner@risc-software.at

M. Behrisch and M. Weber (eds.), *Modeling Mobility with Open Data*,
Lecture Notes in Mobility, DOI 10.1007/978-3-319-15024-6_11

There are two categories of real-time data: floating car data (FCD) from sensors installed in roaming cars, and data coming from static detectors (vehicle detection loops—VDL—or radar detectors) installed at fixed positions on a number of roads. FCD are used to estimate the current average velocities on corresponding roads; the static detectors are basically vehicle counters that also provide velocity and vehicle type 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 obtaining results that closely resemble the real-time development of traffic conditions. The current demand model provided by the Upper Austrian authorities requires series of calibration steps so that, as much as possible, dramatic discrepancies between the reality and the simulated traffic are removed.

In this paper we give an overview of our system, with emphasis on aspects related to the use of SUMO [1] concepts and components. In Sect. 1 we describe the system architecture, at a high level of abstraction. Further, we show how the static data, fundamental to all sub-systems, is generated. Another section contains a description of the calibration process, executed as a preliminary step. In Sect. 4 we explain how the static and real-time sensor data are processed and integrated. We continue by describing TOMS, the main component of our system. Finally, Sect. 6 contains some conclusions and a few words regarding future work.

2 System Overview

Figure 1 contains the system architecture of ITS Austria West.

- In the *preprocessing* phase, a road network file, as well as a routes file, is generated from the static incoming data:
  - GIP—Graph Integration Platform, the main, most comprehensive database of Austrian roads;
  - VLSA—traffic lights description file.
  - Besides these data collections, the demand model—a file with all origin-destination relations—is taken as input by our system.
  - The generated files (the internal road network as well as the set of routes and daily trips) follow the respective SUMO formats.
- The *calibration* process 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

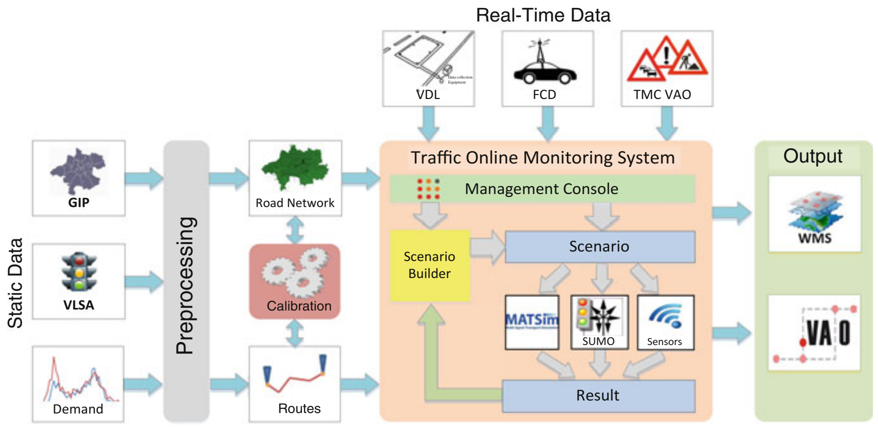


Fig. 1 Architecture of ITS Austria West

model, SUMO needs more than 5 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 static detectors) 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 Östereich", the Austrian traffic information system) and to a WMS (Web Mapping Service) layer that can be accessed by various applications. An HTML5-based web page, as well as an app for Android and IPad, integrate these WMS layers.

3 Preprocessing Static Data

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

3.1 The Road Network

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

1. Geographically: The system monitors only roads from a rectangular area which encloses Upper Austria (Fig. 2a).
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). Figure 2b contains the monitored roads in Linz city center.

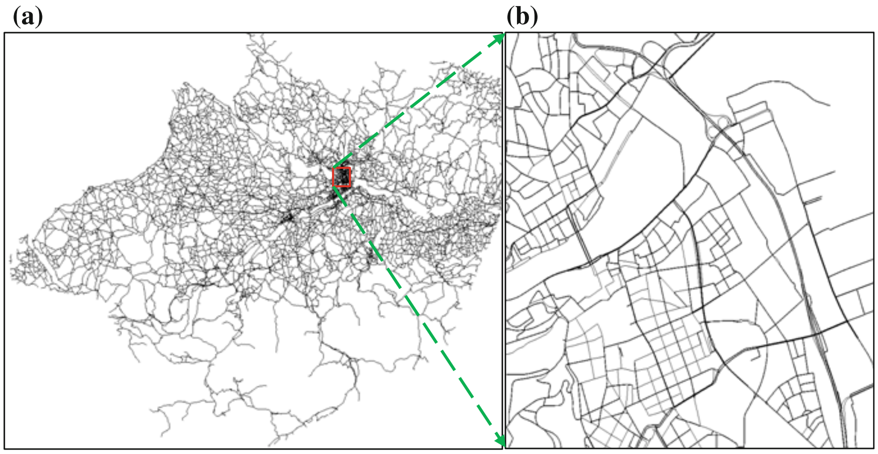


Fig. 2 **a** The whole road network. **b** Part of an urban area (Linz)

Both criteria are necessary in order to produce a manageable network of road segments. The geographical criterion is natural and self-explanatory: no foreign roads need to be maintained in the internal data structures. The functional criterion helps filtering out roads of reduced significance with respect to the traffic; while also necessary, this trimming has a few drawbacks:

- In the reality, trips normally begin and end on side-roads or alleys. Since most of these roads are eliminated, there remain fewer possible points of insertion for vehicles in the simulation. Some road segments will inevitably become congested with new vehicles, and queues of vehicles waiting to be inserted into the simulation will occur with higher frequency.
- Vehicles equipped with sensors may drive on streets that have been eliminated from the internal data structures. Interpreting measurements sent from those road segments requires carefully tailored algorithms, which we describe in some detail in Sect. 4.2.

The remaining 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.

Special care must be given to entry or exit points to and from highways. The usual 100 m of tangential driveway (the ramp) is not explicitly contained in the GIP database. We programmatically detect such situations and the computed list of ramps is given as parameter to NETCONVERT.

Our first results led to a road network that was not fully connected. We traced the problem to inaccuracies in the original data (GIP is by no means finalized, corrections and updates are continuously performed). The original information associated to some road segments marked them (mistakenly) as insignifficant, and they were consequently not inserted into the final road network; the resulting graph was not connected.

In order to address this problem we had to run an additional preprocessing phase, in which we worked with a richer road network and a demand model adapted to it:

1. Routes for each trip from the demand model are generated.
2. Road segments that will be part of the final network (the segments with a suitable functional class) are identified and marked.
3. Road segment with lower functional class contained in sufficiently many routes generated in step 1 are marked as well.
4. All marked road segments are selected for the trimmed network.

Theoretically, this algorithm does not necessarily lead to a connected network; however, in our particular case the results are quite satisfactory.

Further lists are used to refine and improve the road network, with traffic light signal systems (VLSA) and manual changes or corrections. Lastly, and very importantly, the static detectors (see Sect. 4.1) provide their own list of objects to be included into the network: Their positions define locations of induction loops for the simulated traffic. Data collected in these induction loops is used for the on-line calibration of the simulation.

Eventually, a suitable network is obtained; it can be generated in either SUMO or MATSim format.

3.2 The Routes File

The currently available demand model was generated in 2013 by the department "Gesamt-verkehrsplanung 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 hourly summary of trips, for which the source and destinations are districts. We generate a set of routes, distributed over a day, in a three-step 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, the hourly trips are randomly distributed within the hour. The exact departure and arrival positions are also distributed randomly among the nodes situated within the corresponding departure and arrival districts.

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 2 minutes.

The third step takes place in the calibration process, described in the next section.

3.3 Calibrating the Demand Model

The calibration starts with the road network and the original routes file and runs SUMO repeatedly with increasing end times, from 6:00 AM to 8:00 PM. 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 1,000), the process starts working with the next end time (the end times advances with 1 h). At the end of each run, a new routes file is generated and given as input to the next SUMO run.

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.

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 2 min 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 min), 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.

3.3.1 Alternative Routes

The alternative routes are generated with a modified Dijkstra algorithm that seeks new, not-yet-traversed routes between any given source and destination.

The method starts by increasing the weights of all edges in the already traversed routes, thus adding a handicap to the old routes. A shortest-path algorithm follows: Whenever a new node is reached, the ‘weight’ of the shortest path from the source to this node is computed, as in e.g. a normal Dijkstra algorithm. The percentage of

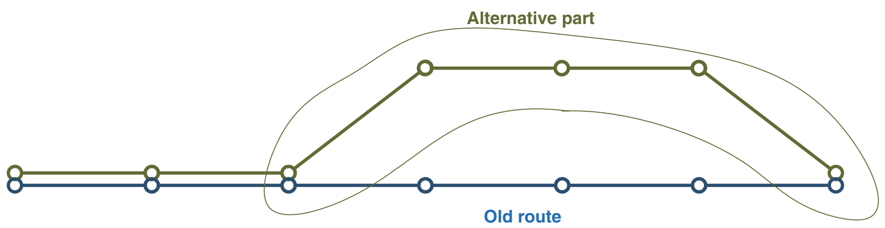


Fig. 3 Alternative route

this path that has not been traversed by any previous route (the *alternative part* of the shortest path) is also computed.

If the new node belongs to (at least) one of the old routes, the alternative part is checked to ensure that the current shortest path is significantly different—for example, 50 % of its length or 10 min drive time should differ from each of those already computed (see Fig. 3). Failure to comply with this condition leads to the current "shortest path" being discarded, and the next "shortest path" takes this place.

After SUMO ends, a new routes file is generated, in which the vehicles that arrived too late or too early are either shifted or follow different alternative routes. This new routes file is given as input to SUMO for the next run.

The goal of the calibration process is to reach a stable simulated traffic, where there are no catastrophic traffic jams and overly delayed vehicles.

In our experience, SUMO runs with the initial routes file for more than 5 days simulation time, in order to have all vehicles reach their destinations; in other words, the vehicles caught in simulated traffic jams need more than 5 days to reach their destinations. Since a lot of road segments are filled with unmoving vehicles, huge queues of new vehicles wait to be inserted into the simulation. These deficiencies lead to unrealistic and in fact unusable outcome. Last but not least, the rush hours are simulated by SUMO below real-time, quite unacceptable if we want to run SUMO alongside a real-life traffic monitoring system.

On the other hand, the calibrated route files obtained so far produced very satisfactory results: The simulation was more fluid, and the simulated traffic jams—inevitable during the rush hours—did not become persistent.

4 Integration of Real-Time Data

Real-time traffic data is made available by various providers in different formats and on different locations. We produced small applications that regularly intercept this data and save it in dedicated databases, using a uniform format.

In the following, we describe in some detail each type of real-time traffic data and we explain shortly how this information is interpreted in order to generate traffic situation snapshots.

4.1 Static Detectors

The vehicle detection loops (VDL) and radar detectors 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 static detectors 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 4 shows 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 min).

4.2 Floating Car Data

Floating car data (FCD) come from providers in intervals between 1 and 30 s. The measurements contain position information in geographic units, which is used to match the readings against the edges (road segments) of the road network.

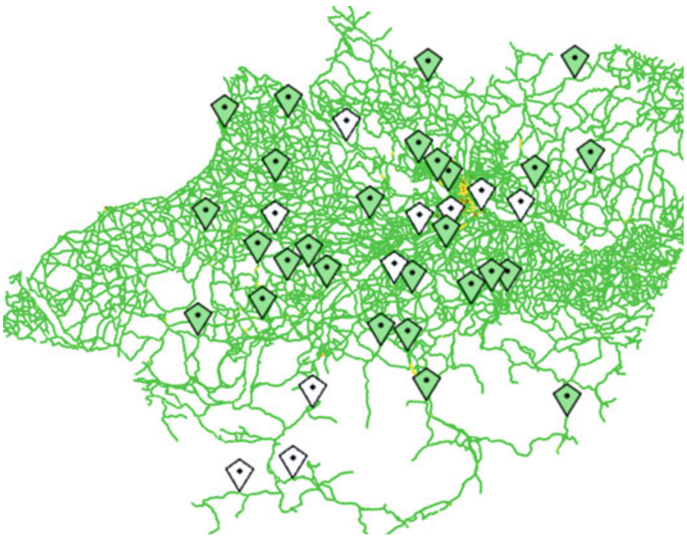


Fig. 4 Locations of vehicle counters (Upper Austria)

In order to correctly interpret FCD, it is not sufficient to determine the most probable road segments where readings were taken. Indeed, we received a lot of measurements with very slow speed values which were not due to real traffic problems 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. We had to tackle the following problems:

- For a normally running car, measurements coming with a 30 s 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.

Figure 5 shows the road segments mapped to the most recent measurements sent by a taxi cab equipped with a FCD sensor. The markers denote the exact positions of the measurements (the first and last measurement are colored in red).

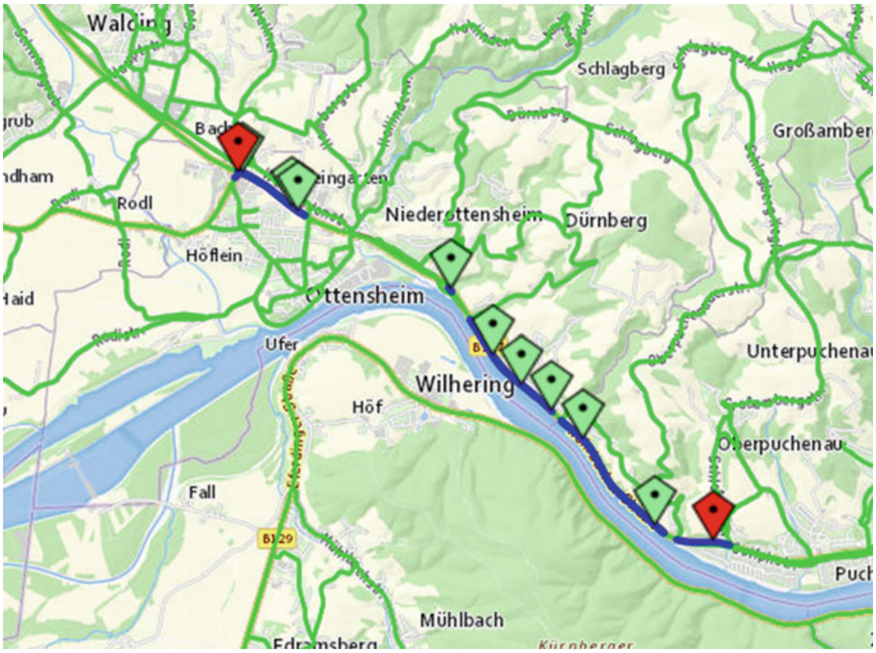


Fig. 5 The probable trajectory of a vehicle, computed from FCD

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 trajectory is eventually generated.

On this trajectory, a plausible matching of measurements to road segments can be performed. The travel time can be used to compute velocity values on all road segments contained in the trajectory (including those not matched to any measurement).

4.3 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 information from different providers is bundled into a single connection. We currently use this information to create another WMS layer, which in fact provides markers for the corresponding roadworks/roadblocks positions. Ancillary information for each roadwork can be visualized as well—see Fig. 6.

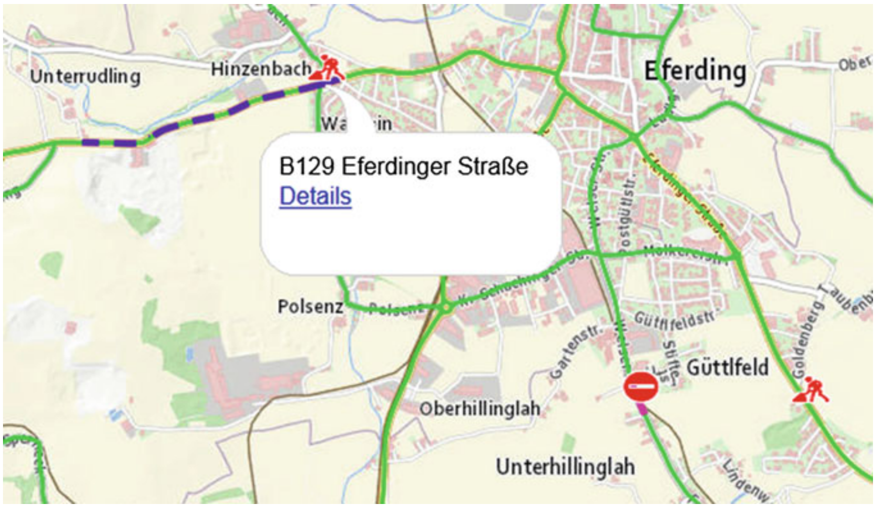


Fig. 6 Markers for roadblocks and roadworks

5 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.

TOMS starts by loading the static data and instantiating internal data structures which are fundamental for all processing steps:

- The road network—from which it produces an internal data structure (a connected graph) suitable for all graph algorithms. The edges of this graph correspond to road segments. Among other attributes, they hold the current velocity: its value is updated according to simulated or real-life data.
- The routes file—used mainly for on-line calibration of running simulations.

Further, TOMS starts the *real-time data loop*. With a 1 min periodicity, TOMS:

- downloads the latest traffic information provided by static detectors or FCD;
- interprets the measurements and computes current speed values for corresponding road segments;
- overrides the default or simulated velocities on these road segments;
- generates an LOS (Level of Service) output, which is saved as a set of WMS layers and sent to VAO as an XML file.

If so configured, or by user requests, TOMS can start a simulation and initiates a simulation loop. Within this loop, TOMS:

- collects the simulation output and extracts traffic values for all roads with simulated vehicles;
- modifies the traffic values on road segments which have simulated traffic;
- calibrates the simulation, inserting adjustments computed from the real-time data.

In Fig. 7 we show a diagram of this process.

We have to emphasize that the traffic values computed from real-time data have priority over those coming from simulation: They always override the simulated values.

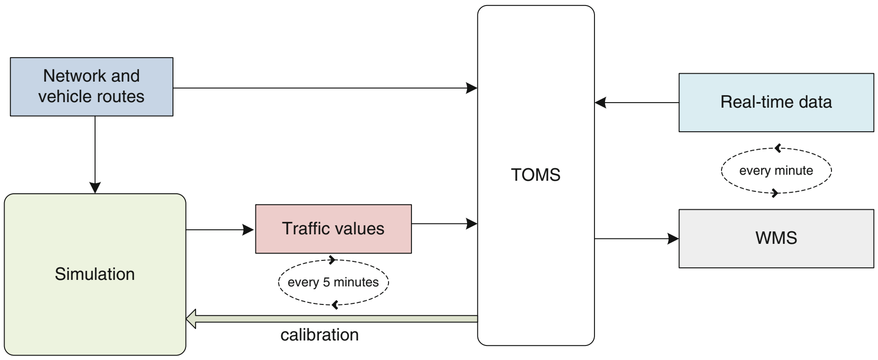


Fig. 7 The two main loops in which TOMS collects traffic information

The *scenario builder* (see Fig. 1) is responsible for the dialog with the traffic simulation applications:

- it prepares scenarios—the input for simulations—in the form of routes files;
- it starts simulations, establishes channels for communication and control;
- it sends calibration orders, etc.

The most important 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. Our first attempt to speed up the simulation, namely by parallelizing SUMO, is described in [2]. With a calibrated demand model, the simulation speed is considerably greater than the real time, even with congested traffic.

Usually, a simulation runs in 5 min steps. At the end of each step, the results are collected (as a SUMO dump) and sent to the scenario builder. These results contain two sets of information:

1. the status of every vehicle in the simulated network,
2. the statistics from induction loops.

From the first set, the average simulated velocities on each road segment are computed; they are used unmodified, for road segments not covered by sensors.

Data from the second set is compared with the current real-time information from the corresponding static detectors and adjustment decisions are accordingly taken: Vehicles whose routes contain road segments with static detectors are added to or removed from the simulation. These 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 and calibrate the simulation. The messages that TOMS currently sends to SUMO via TraCI are:

1. "Simulate To": SUMO is announced that it has to run until a given time (contained in the message) is reached.
2. "Add New Route": A new route is defined, its ID and edges constitute the body of the message.
3. "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.
4. "Remove Vehicle": A vehicle is removed from the simulation. The vehicle ID is given in the message.

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).

6 Conclusions and Future Work

In this paper, we gave an overview of the traffic 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 (microscopic edition) 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. 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.
4. 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.

References

1. Behrisch M, Bieker L, Erdmann J, Krajzewicz D (2011) SUMO—simulation of urban mobility: an overview. In: SIMUL 2011, Barcelona, Spain, pp 63–68
2. Kastner K-H, Keber R, Pau P, Samal M (2013) Real-time traffic conditions with SUMO for ITS Austria West. In: Proceedings of the 1st SUMO conference. Springer, Berlin (to appear)

Second Generation of Pollutant Emission Models for SUMO

Daniel Krajzewicz, Michael Behrisch, Peter Wagner, Raphael Luz and Mario Krumnow

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 of vehicular pollutant emissions, microscopic traffic simulations are assumed to be helpful in predicting the performance of such approaches. SUMO includes a model for vehicular emissions since 2008. In the context of the projects COLOMBO and AMITRAN, two further models were implemented. Herein, these models are presented and discussed, pointing out the progress in emissions modelling.

Keywords Vehicular emissions · Emission modelling · Environment · Traffic management

1 Introduction

Air pollution is a well-known problem that ranges from local air quality issues up to global effects the humanity is confronted with, such as global warming. Following the International Transport Forum [1], "[the] Transport-sector CO_2 emissions represent 23 % (globally) and 30 % (OECD) of overall CO_2 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 its resource consumption, often forced to do so by law. In Europe, automobile

D. Krajzewicz (✉) · M. Behrisch · P. Wagner
German Aerospace Center, Rutherfordstraße 2, 12489 Berlin, Germany
e-mail: daniel.krajzewicz@dlr.de

R. Luz
Institut für Verbrennungskraftmaschinen und Thermodynamik, Inffeldgasse 19/I, 8010 Graz, Austria

M. Krumnow
Technische Universität Dresden, Institut für Verkehrstelematik, 01062 Dresden, Germany

M. Behrisch and M. Weber (eds.), *Modeling Mobility with Open Data*,
Lecture Notes in Mobility, DOI 10.1007/978-3-319-15024-6_12

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 emission is correlated to the consumption of fuel. As fuel price has increased in the past years, the reduction of emissions is also in the focus of end users—individuals as well as (e.g. logistics) companies. This large variety of actors and customers yields in an accordingly large amount of solutions. They range from large-scale traffic management actions, such as the introduction of environmental zones, down to in-vehicle solutions that propose the driver a speed that minimizes emissions.

The development of technical solutions for critical systems usually includes a step where the solution is modelled as software and simulated. This step allows validating the assumptions about the solution's functionality and to benchmark or prove its performance a priori. In the context of evaluating on-road solutions, traffic simulations are an established tool used for this purpose by both, consultants and researchers. Academic approaches, such as the traffic simulation SUMO [4, 5] that is discussed herein, attempt to simulate large, city-wide areas using so-called microscopic models that simulate every traffic participant individually.

To evaluate a solution that was designed to reduce road traffic's impact on air quality the used traffic simulation must be capable to compute the amount of the emitted pollutants the solution attempts to reduce. 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 emission fleet, and the aggregation of the results in time and area. Therefore, according requirements must be formulated before choosing a model that shall be embedded or implemented into the used traffic simulation.

In the following, 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 are going to replace SUMO's initial emissions model that was developed within the project "iTETRIS" [10, 11]. All three projects are, or respectively were, co-funded by the European Commission.

The remainder is structured as follows. 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 emission models implemented 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.

2 SUMO's Requirements to an Emission Model

Briefly said, the emission model to choose should be capable to be used as a source of further measurements for the applications the traffic simulation is usually used for. In other words, it should not change the way the traffic simulation is used.

Instead, it should generate additional information, not available before. As SUMO's goal is to simulate real-world traffic in large areas, the model should cover the complete emission fleet 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 fleet compositions. 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. Of course, a clear distinction between passenger vehicles, heavy duty vehicles, and busses is necessary, because some regulations affect only vehicles of one of these classes, mainly the heavy duty vehicles.

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 the simulation offers. In the 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 those vehicle 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 one second or below. To achieve this, the emissions model must compute emissions at the same time scale.

Not all available models cover all pollutants emitted by road traffic. As well, not all gases emitted by road traffic are relevant. Therefore the pollutants assumed to be needed should be a part of the requirements. Within the iTETRIS project (see [12]), it was decided to model the emission of CO, CO_2, 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 (CO_2). Additionally, the fuel consumption should have been modelled.

An emission model for SUMO has to fulfil some other, non-functional requirements. It should be portable matching SUMO's overall portability. It should be fast in execution for being applicable to large-scale scenarios. And it should be directly embedded into the simulation to avoid additional interaction between programs (e.g. socket-based) or file exchange.

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

- Cover the complete vehicle fleet (in means of emission classes);
- Offer a classification of classes into Euro-norms;
- Compute certain pollutants (CO, CO_2, 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.

3 Emission Models Overview

Most of nowadays vehicles burn petroleum-derived fuel for propulsion. When regarding small time scales, fuel consumption depends on the vehicle's engine characteristics and on the current load of 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 very good explanation). Most of the fuel burns to the greenhouse gas CO_2 and to water. But other, often toxic gases are generated as well. Catalytic converters convert a major portion of some of these pollutants into non-toxic gases. The performance of the catalytic converter mainly depends on the catalyst's temperature as well as on the engine's current operating point. The amount of emitted pollutants depends on other influences, such as drive train losses, the road's slope, or the air-fuel ratio at combustion. Additionally, long-term effects of a driving style may change a vehicle's emission behavior.

In summary, every single vehicle has an individual emission behavior. But when investigating road traffic, many vehicles of different type 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 single emission class. The literature accordingly distinguishes the following classes of emission models:

- "inventory" emission models that include data for the major portion of the vehicle emission classes; their input usually consists of a vehicle population composition and the amount of driven distances, 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.

It follows that the models differ in granularity and input parameters they need as well as in the number of covered pollutants. 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 Emission Model") [15, 16], which derivate was included in SUMO as shown in Sect. 4.2, is one of such models. Its sub-modules and their inter-dependencies are shown in Fig. 1.

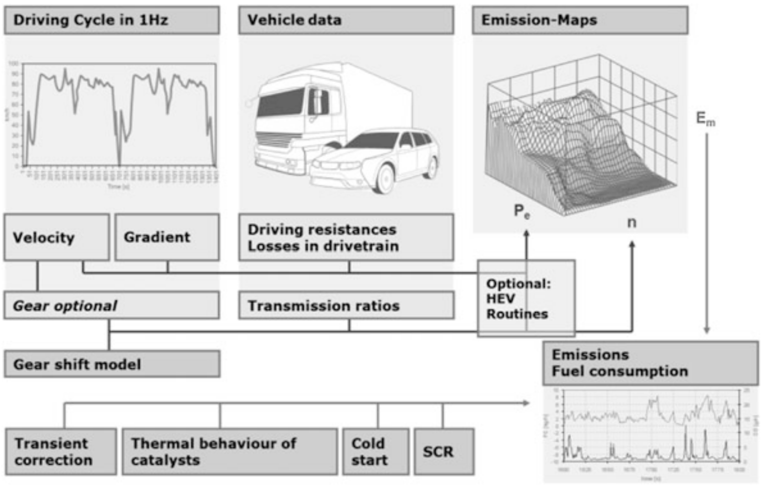


Fig. 1 Schematic representation of the PHEM emission model [17]

Within the iTETRIS project, 15 non-commercial (freely available in means of data or a document that completely defines them) emission models have been examined 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. But most instantaneous models compute only few of the required pollutants. Additionally, the needed input parameters were often not completely given. As well, most instantaneous models use parameter that are not originally covered by SUMO's simulation model and would introduce a high number of additional parameters into SUMO's vehicle type description.

As a conclusion, none of the evaluated models could be directly embedded into SUMO. Instead, one of the inventory models was chosen and reformulated to be continuous in speed and acceleration, as described in the following chapter.

4 Implemented Emission Models

As no emission model could be found that on the one hand is instantaneous in means of regarding vehicle attributes used in a microscopic simulation but on the other hand still covers a major part of the vehicle population, the decision to build an own model based on data from the HBEFA [18] database was taken. HBEFA, in version 2.1 at that time, was one of the investigated inventory models. This initial implementation of an emission model into SUMO will be described in the following section. Two recently developed emission models will be presented afterwards: "PHEMlight", which is derived from PHEM and a new approach to reformulate the emissions stored in the inventory database HBEFA, using its version 3.1. The models have been implemented in the projects "COLOMBO" and "AMITRAN", respectively.

4.1 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 (see [13, 19]). The simplified function for accordingly computing the energy consumption rate e is thereby [12]:

$$e(v, a) = c_0 + c_1 va + c_2 va^2 + c_3 v + c_4 v^2 + c_5 v^4 \tag{1}$$

This function has been used for all pollutants, only the coefficients change per emission class and pollutant. 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 only the values up to ± 0.6 m/s^2 can be determined this way, the dependency on higher acceleration/deceleration was obtained by extrapolating the 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. As a result, 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. To avoid major misbehavior, 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 ease the definition of a simulation scenario. In Fig. 2, the development of the residual sum of squares (RSS) in dependence to the cluster number is given individually for passenger (left) and heavy duty (right) vehicles. As shown, no clear thresholds in the development were found that motivate to select a certain cluster size. Therefore, the decision to define more than one cluster per passenger/heavy duty emission classes was taken. Resulting, 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.

While working with the obtained model, several issues were found, partially grounded in the decisions taken during the development. The simplification

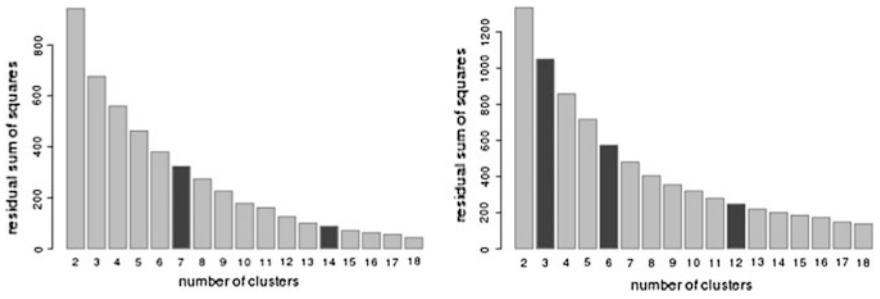


Fig. 2 The development of the error for an increasing amount of clusters. Chosen clusters are shown in *black*

attempted by clustering emission classes was found to be not beneficial. E.g., the lack of an explicitly given Euro norm does not allow to perform investigations of regulatory actions such as environmental zones that distinguish between emission classes. Additionally, the lack of a projection from the clusters back to the original emission 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 Sect. 4.3. Further information about the development of this first emission model in SUMO can be found in [12].

4.2 PHEMlight

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

The amount of emissions produced by a vehicle (as well as the amount of consumed fuel) during a simulation step are determined by computing the power needed by the vehicle, first. The overall power is computed as:

$$P_e = (P_{Roll} + P_{Air} + P_{Accel} + P_{Grad})/\eta_{gearbox} \tag{2}$$

where

$$P_{Roll} = (m_{vehicle} + m_{load}) \times g \times \left(Fr_0 + Fr_1 v + Fr_2 v^4\right) \times v \tag{3}$$

$$P_{Air} = \left(c_d \times A \times \frac{\rho}{2}\right) v^3 \tag{4}$$

$$P_{Accel} = (m_{vehicle} + m_{rot} + m_{load})av \tag{5}$$

$$P_{Grad} = (m_{vehicle} + m_{load}) \times Gradient \times .01 \times v \tag{6}$$

with:

$\eta_{gearbox}$	driver train loss (set to 0.95)
$m_{vehicle}$, m_{load}	masses of the vehicle and its load, respectively
g	Gravitational constant (6.673×10^{-11} m^3/(kg $\times$ s^2))
Fr_0, Fr_1, Fr_2	friction coefficients
v	the current vehicle velocity
c_d	vehicle's drag coefficient
A	cross-sectional area (m^2)
ρ	air density ($\sim$1.225 kg/m^3)
m_{rot}	rotational mass

PHEMlight uses so-called "Characteristic Emission curves over Power" (CEPs) which define the emission amount (g/h) as function of the current engine power of the vehicle. These curves were computed using PHEM with representative dynamic real world driving cycles. To compute the amount of an emitted pollutant, the CEPs are used as look-up tables for the previously computed power.

PHEMlight defines 112 vehicle emission classes. The major distinction is on the level of "vehicle classes". The following ones are modeled by PHEMlight: Passenger cars (PKW), Light duty vehicles (LNF), Motorcycles (MR), Scooters (KKR), Hybrid passenger cars (H_PKW), Tractor/Trailer (LSZ), Coaches (RB), Urban and inter-urban buses (LB), and Trucks (Solo_LKW). Each of those top-level classes is subdivided if appropriate, based on the type of fuel (Gasoline vs. Diesel) and the Euro norm (0–6). Light duty vehicles and Trucks are additionally subdivided by their weight.

PHEMlight is available as a commercial add-on to SUMO. The implementation itself is included in the usual, open SUMO version. But the major information is stored in CEP and vehicle attribute files. This data is included in SUMO's open source release for only two emission classes: a Euro-4 passenger car with a gasoline engine and a passenger car with the same emission class, but running on Diesel. The remaining emission class definitions have to be purchased from the Technical University of Graz. A more complete description of PHEMlight can be found in [17].

4.3 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 data on modern Euro-Norm-6 vehicles, a new attempt to build a free emission model was done in the scope of the AMITRAN project.

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

Fitting the values to the given function is a linear problem, since only the linear coefficients c_0–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 a coefficient is zero or not, a couple of slightly different models were tested in each case (one emission class and one vehicle class) where some of the coefficients of (1) were set to zero and not estimated in the according fit. By comparing these candidate functions, the best one (based on RMS and t-value) was 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 to be well represented by function (1).

In principle, emission curves could be fit to all emission classes included in HBEFA's version 3.1 resulting in some hundreds of different coefficient sets. But to keep the model lean and to ease the preparation of a vehicle population, it has been decided to use the most common emission classes only. In its current implementation, the model includes 45 emission classes: light duty vehicles (LDV) and passenger cars (PC), both sub-divided by fuel type and Euro norm and heavy duty vehicles (HDV) sub-divided by Euro norm. Additionally, average classes for LDVs, PCs, Busses, Coaches, and HDV exist. Some special classes model the emission behavior of an Eastern LDV, an Eastern HDV, and an alternative PC. Further classes may be added on purpose, by fitting the desired emission data to function (1) and embedding the so obtained coefficients into SUMO. The model as well as all obtained coefficients are publicly available as a part of SUMO's open source version.

4.4 Comparisons

In a first step, fulfilling the requirements formulated in Sect. 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 1 shows a summary of other named requirements.

The number of respectively covered emission classes requires some explanations, given in the following:

- 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 Sect. 6.2).

Table 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	45	112
Coverage	No modern (Euro 6) vehicles and other seldom classes	Major passenger, heavy duty, and bus classes	Almost complete
Euro-Norms	–	x	x
Covers chosen pollutants	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

- As discussed in Sect. 4.1, the HBEFA v2.1-derivation does not include 56 distinct emission classes but rather 56 clusters of similar emission classes.
- As mentioned in Sect. 4.3, the number of emission classes in the HBEFA 3.1-based model could be increased when necessary.

In order to verify the emission output of PHEMlight, calculations using the ERMES real world driving cycle were performed with PHEM and PHEMlight using the average EURO 4 Diesel passenger car. Figures 3, 4 and 5 show fuel consumption, NO_x and PM results for each model.

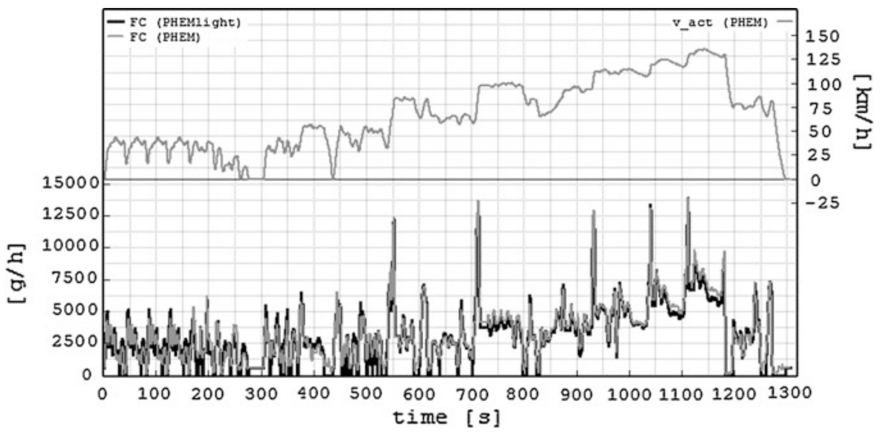


Fig. 3 Fuel consumption comparison between PHEM and PHEMlight

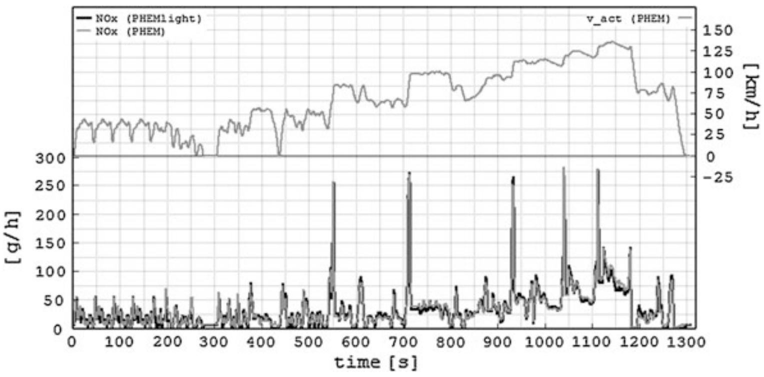


Fig. 4 NO_x comparison between PHEM and PHEMlight

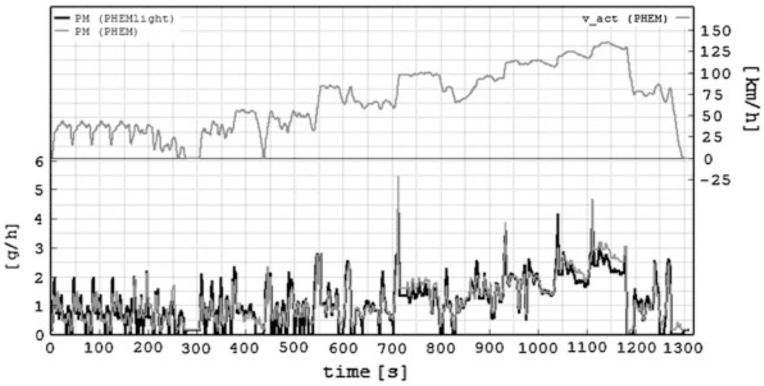


Fig. 5 PM comparison between PHEM and PHEMlight

The results present very good correlation between the two models over the whole cycle despite the fact that PHEMlight uses a significantly simpler approach with no consideration of gear shifting and engine speed. Table 2 shows the average emission results for the same components.

The deviation is to a high extent caused by the fact that PHEMlight calculates no emissions (i.e. 0 g/h) when the engine is in motoring operation. In PHEM the motoring emissions are based on measurements on the chassis dynamometer where, due to technical limitations, the measured emission level is not entirely cut off at the

Table 2 Average emissions in ERMES cycle for PHEM and PHEMlight

	FC (g/h)	NO_x (g/h)	PM (g/h)
PHEM	3352.8	33.83	1.18
PHEMlight	3183.6	33.13	1.12
Deviation (%)	−5.0	−2.1	−4.9

same moment as the engine stops injecting fuel. For PHEMlight it was decided to implement fuel cut off explicitly to depict the influence of optimized deceleration behavior on emission levels correctly.

In a further approach to compare the models, the New European Driving Cycle was applied to all comparable emission classes of the HBEFA3 and the PHEMlight model. This includes Diesel and Gasoline powered vehicles of the seven currently available Euro norms (0–6).

There is one point in the scatter plot for each emission class. Down-facing triangles describe Diesel fueled light duty vehicles (LDV), up-facing the Diesel (D) passenger cars, circles are Gasoline (G) fueled LDVs and squares Gasoline passenger cars. The brightness encodes the Euro norm. For better orientation, the diagonal line representing identical values has been drawn into the figure as well. For light duty vehicles there are up to three points for each emission class on the plot because every HBEFA3 class is subdivided by vehicle weight in up to three PHEMlight classes (Fig. 6).

While in general the values are close (please note that the axes do not start at zero), HBEFA seems to give higher fuel consumption values especially for the older gasoline-powered passenger cars. The values for Diesel engines are almost identical.

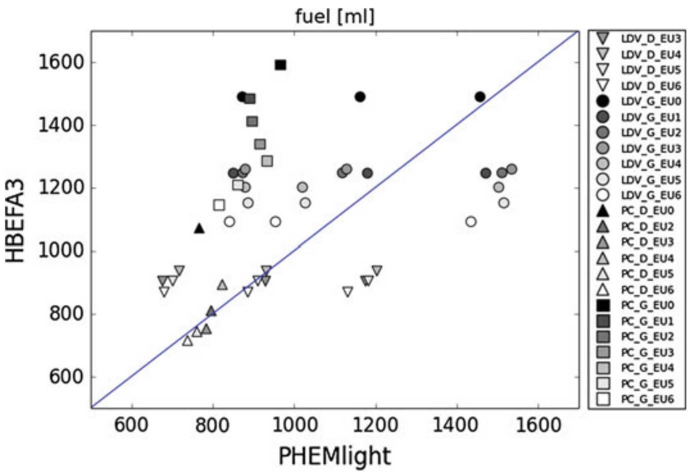


Fig. 6 Fuel consumption comparison for comparable classes between HBEFA3 and PHEMlight

5 Working with SUMO's Emission Models

Besides realizing an emission model, the work on vehicular emissions included a large variety of actions that target topics such as the implementation of proper visualisation of the generated emissions data, support to handle emissions by other tools in the suite than the simulation only, as well as opening the applications to the inclusion of further emission models. In the following subsections, the implemented features are described. Additionally, a summary on open issues in modelling emissions is given.

5.1 Simulation

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. In SUMO, so-called "vehicle types" may be defined that may be shared by an arbitrary number of vehicles to simulate. These vehicle types describe the assigned vehicles' physical and model attributes including their respective emission class. It is additionally possible to define so-called "vehicle type distributions". A vehicle type distribution is composed of several vehicle types, each having a probability to be selected. If a vehicle lists such a distribution as its vehicle type, one of the included vehicle types is selected according to the given probability.

The simulation was extended by a large variety of outputs that collect and aggregate the computed emissions in different ways. 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.

The AMITRAN trajectory format is an intermediate data exchange format 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.

In addition, SUMO's on-line interaction interface "TraCI" has been extended by methods for retrieving the emissions a single vehicle "produced" in the last simulation step, as well as aggregated emissions produced on edges or lanes. The visualization allows coloring lanes and/or vehicles by the amount of pollutants emitted on them or generated by them, respectively.

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 are supported by all implemented emission models, cover all of the modelled pollutants, and—despite the visualization of emissions—are available in both, the command line and the graphical version of the simulation.

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

5.2 Router Support

Besides enabling the traffic simulation to compute pollutant emissions, the route computation applications included in the SUMO suite were extended as well. 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. The implementations of the shortest-path algorithms were reworked to use these values as edge weights and additionally keep track of the travel time to obtain these weights from the correct time slice of the loaded emissions time line. This extension has already been used for different purposes, as outlined in Sect. 6.2.

5.3 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 driving 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 reads trajectories in the AMITRAN format mentioned above as well 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. An additional visualization script shows the so obtained matrices.

5.4 Embedding New Emission Models into SUMO

The co-existence of different emission models was realized by deriving a common "interface". This 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 on. Internally, the emission class is encoded as a 32 bit integer. The upper 16 bits are used to encode the used model 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 model [20]. When being asked to compute the amount of a pollutant's emissions, the interface determines the model to use based on the upper 16 bits, first. It then 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 parameters have to be taken into account, the interface would have to be extended.

5.5 *Open Issues*

The implemented models allow a large variety of investigations as shown in the next section. Nonetheless, some peculiarities of vehicular emissions are still neglected and may be addressed in next development steps.

The first to name are "cold start emissions"; vehicles produce more emissions as long as the engine and the catalytic converter are not at their optimal working temperature. For taking this effect into regard, the time the vehicle was driving before entering the simulated network has to be known. It should be stated that modelling this information for transit traffic—vehicles that do not start or end within the simulated network—is complicated.

The second peculiarity is the dependence on the vehicle mass. Both HBEFA-based models include this information implicitly. PHEMlight holds the average mass of vehicles of the respective emission class within the emission class definition files. Still, the mass is given as a constant value. But when fleet management, other logistics approaches or public transport shall be simulated, changes in the vehicles' masses may be an important factor. In such cases, the vehicle mass would have to be moved into SUMO's internal vehicle type definition.

The last simplification to name is the gear choice that has as well a major effect on produced emissions. Gear choice is not considered by usual car-following models and is as well not explicitly taken into regard by the emission models discussed before. Both parts would have to be extended to model gear choice properly.

6 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.

6.1 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 further 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 and were used in a large variety of evaluations.

As increasing traffic efficiency usually reduces pollutant emission, often no new insight 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 a GLOSA (Green Light Optimal Speed Advisory) implementation [21] 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. It was found that the used function to compute the speed to advice was the reason and other GLOSA algorithms are well capable to reduce emissions. Still, it shows that environmental performance indicators should be included when evaluating a new method or system.

6.2 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 the overall emissions be reduced? First investigations on this topic were performed using a real-world network [22]. To gain a deeper understanding about the dynamics of the processes, later investigations [23] were performed using synthetic scenarios. At the time being, neither a singular user nor a singular system optimum is assumed to be computable using currently available methods. The main problem in this case is that for most emissions (in most emission classes) there is some kind of an optimal speed which violates the general assumption that the cost function on a single edge is monotone in the number of vehicles driving that edge.

6.3 Evaluation of Real Traffic Management Actions

The “Directive 2008/50/EC of the European Parliament and of the Council” [3] forces European authorities to assure a certain air quality. Traffic management, usually operated by local authorities, has the duty to perform corrective actions to reduce emissions caused by road traffic, if needed. A proof-of-concept for simulating such actions that used SUMO and the HBEFA 2.1-based model was presented in [24] where three speed limit changes were investigated—30 and 60 km/h for urban areas and 80 km/h for highways.

In his Master thesis [25], Tomàs Josep Vergés investigates the MARLIS [26] database that lists actions performed by traffic management authorities, first, to evaluate which of the actions can be simulated when using a microscopic traffic simulation only. The evaluation showed that most actions target at a change in the population’s mobility behavior, mainly for using a more environment-friendly transport mode. This can only be simulated using an according population model that was not available within his research. The following traffic management actions were selected and modeled within the thesis: (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 were modeled and a new user equilibrium was computed, first. The obtained vehicle routes were then simulated using PHEMlight.

As expected, in case of a speed limit, traffic moves out of the influenced areas, yielding in an according shift in pollutant emission. Additionally, speed limits were found to not reduce emissions, as already known from the literature. After the introduction of an environmental zone, the mobility patterns change in a more complex way as prohibited vehicles have to drive around it what makes the roads within the zone more attractive to be used by allowed vehicles. The resulting changes in road usage span over a bigger area. The results related to the speed limit were similar in both investigations, independent of the emission model.

7 Summary

Recent steps in modelling vehicular emissions within the open source traffic simulation SUMO were presented. Three emission models that are currently implemented in SUMO were discussed: issues regarding the initial model derived from HBEFA 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.

As shown, the inclusion of emission models in a microscopic road traffic simulation allows gaining insights about the effects of evaluated solutions on the environment. In most cases the induction “smoother traffic $\rightarrow$ less emissions” holds. But evaluating pollutant emission behavior may offer some surprises, as named for the GLOSA example in Sect. 6.1. Besides evaluating the environmental benefit of ITS solutions the models were successfully applied to the simulation of large-scale regulatory actions.

The presented extensions cover the work defined for the projects “COLOMBO” and “AMITRAN” well. Nonetheless, several possible extensions that may be targeted in the future were identified and listed. But given the currently implemented models, it is assumed that next steps towards a further quality improvement should be performed by reworking the simulation’s representation of single vehicles’ longitudinal behavior; it is known that nowadays car-following models do not replicate the decelerations and accelerations of vehicles well. These simulation model characteristics should be addressed next.

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

References

1. International Transport Forum (2010) Reducing transport greenhouse gas emissions: trends and data, OECD
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 CO_2 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 (2012) Recent development and applications of SUMO—Simulation of Urban MObility. Int J Adv Syst Meas 5(3, 4):128–138. ISSN:1942-261x
5. DLR and Contributors (2013) SUMO homepage. http://sumo.dlr.de/
6. Krajzewicz D, Heinrich M, Milano M, Bellavista P, Stützle T, Härri J, Spyropoulos T, Blokpoel R, Hausberger S, Fellendorf M (2013) 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/. Accessed 10 April 2014
8. Jonkers E, Klunder G, Mahmod M, Benz T (2013) Methodology and framework architecture for the evaluation of effects of ICT measures on CO_2 emissions. In: Proceedings of the 20th ITS world congress, Tokyo, Japan
9. AMITRAN Consortium (2013) AMITRAN web pages. http://www.amitran.eu/. Accessed 10 April 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, Härri J, Vaz S, Lopez Y, Sepulcre M, Wetterwald M, Blokpoel R, Cartolano F (2013) 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

11. iTETRIS Consortium (2011) iTETRIS web site. http://www.ict-itetris.eu/. Accessed 8 Jan 2014
12. Krajzewicz D, Nippold R, Lazaro O (2009) iTETRIS deliverable 3.1—traffic modelling: environmental factors, public deliverable
13. Treiber M, Kesting A, Thiemann C (2008) How much does traffic congestion increase fuel consumption and emissions? Applying a fuel consumption model to the NGSIM trajectory data, presentation no 08-2715. In: Annual meeting of the transportation research board, 13–17 Jan 2008, Washington, DC
14. Cappiello A, Chabini I, Nam EK, Lue A, Abou Zeid M (2002) A statistical model of vehicle emissions and fuel consumption. In: Proceedings of the IEEE 5th international conference on intelligent transportation systems, pp 801–809. doi:10.1109/ITSC.2002.1041322
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 (2014) Pages of the institute for internal combustion engines and thermodynamics (IVT). Accessed 10 Jan 2014
17. Hausberger S, Krajzewicz D (2014) Deliverable 4.2—extended simulation tool PHEM coupled to SUMO with user guide, public project report. http://www.colombo-fp7.eu/results_deliverables.php
18. INFRAS (2013) Handbuch für Emissionsfaktoren. http://www.hbefa.net/. Accessed 06 Feb 2014
19. Schnabel W, Lohse D (1997) Grundlagen der Straßenverkehrstechnik und der Verkehrsplanung. Verlag für Bauwesen, Berlin, pp 557–577. ISBN 3-345-00565-4
20. Nota R, Barelds R, van Leeuwen H (2005) Harmonoise WP 3—engineering method for road traffic and railway noise after validation and fine-tuning. Harmonoise technical report HAR32TR-040922-DGMR20 (deliverable 18)
21. Krajzewicz D, Bieker L, Erdmann J (2012) 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
22. Krajzewicz D, Wagner P (2011) Large-scale vehicle routing scenarios based on pollutant emission. In: Meyer G, Valldorf J (eds) Advanced microsystems for automotive applications 2011, AMAA 2011. Springer, New York, pp 237–246
23. Flötteröd Y-P, Wagner P, Behrisch M, Krajzewicz D (2012) Simulated-based validity analysis of ecological user equilibrium. In: Winter simulation conference archive, 2012 Winter simulation conference
24. Krajzewicz D, Flötteröd Y-P (2013) Simulative Untersuchung abstrakter und realer Verkehrsmanagementansätze zur Emissionsreduktion. In: Kolloquium Luftqualität an Straßen 2013, pp 42–57. Bundesanstalt für Straßenwesen
25. Josep Vergés T (2013) Analysis and simulation of traffic management actions for traffic emission reduction. TU Berlin, Berlin
26. BASt (2012) MARLIS—Datenbank mit Maßnahmen zur Reinhaltung der Luft in Bezug auf Immissionen an Straßen, Version 3.1, to be found on BASt web pages

Modelling Bluetooth Inquiry for SUMO

Michael Behrisch and Gaby Gurczik

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 simplified analytic model.

Keywords Traffic simulation · Bluetooth · Inquiry modelling

1 Introduction

Bluetooth [1] is a short-range, low-power, IEEE open standard for implementing wireless personal area networks. Bluetooth operates in the globally unlicensed 2.4 GHz 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 32 available frequencies (divided in two 16 frequencies long trains A and B) 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 [1, 2, 3].

M. Behrisch (✉) · G. Gurczik
German Aerospace Center, Institute of Transportation Systems, Rutherfordstraße 2,
12489 Berlin, Germany
e-mail: Michael.Behrisch@DLR.de

G. Gurczik
e-mail: Gaby.Gurczik@DLR.de

M. Behrisch and M. Weber (eds.), *Modeling Mobility with Open Data*,
Lecture Notes in Mobility, DOI 10.1007/978-3-319-15024-6_13

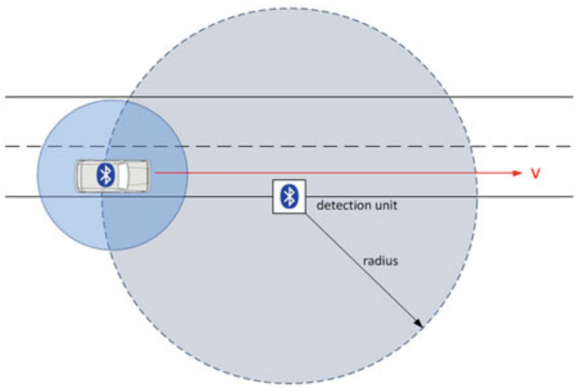


Fig. 1 The Bluetooth detection principle

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 (Fig. 1).

The German Aerospace Center (DLR) developed a traffic monitoring approach, called DYNAMIC [4, 5], 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 focuses on a simplified model for the inquiry process, describes its outcomings and the implementation of the process in the Bluetooth model of SUMO [6, 7] 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.

This paper is an extended version of the conference paper [8] with the same title.

2 State of the Art

In this paper we deal with the modelling 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 [9, 10]. 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 [11] 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), [2] try to find out whether the number of inquirer and inquiry scanners has an effect on the discovery time. In [3] 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 [3, 12, 13]).

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 inquiry scanner randomly selects the frequency on which it sends out messages from all 32 possible frequencies, that is one cannot predict if the current scan frequency is from frequency train A or B. Furthermore the inquiry scanner scans for the chosen frequency (and only this one) for 2,048 timeslots (1.28 s). Only afterwards the next frequency is scanned. Hence, it takes much longer for a device to be discovered as the receiver only sends replies to every second message received [11]. From Bluetooth core version 1.2 [1] on, so called interlaced scan mode has been introduced with which two frequencies (one from each frequency train) are scanned within the 2,048 timeslots at once. Therefore Bluetooth discovery times could be speeded up (cf. Sect. 3). Another drawback of older researches is that they are mostly based on the assumption of an ideal, error-free environment, where messages never get lost [3]. This is, especially in our special field of studies, not realistic.

For this reason, we investigate a simplified 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.

3 Analytical Modelling of the Inquiry

3.1 Modelling Based on Two Scanning Intervals

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 [1, 14] 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 (Fig. 2).

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:

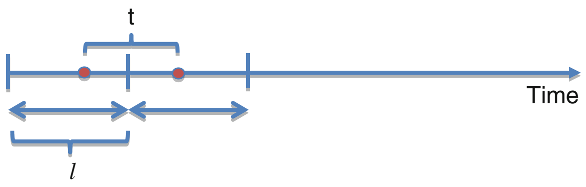


Fig. 2 Model of the inquiry process with two scanning intervals of length l

$t < l$:

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

$l \leq t < 2l$:

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

The resulting function depicting the probability depending on the travel time ratio is shown in Fig. 4. For the length of the scanning interval l we assume that each frequency train is repeated 256 times and the time to cover one frequency train takes 16 timeslots with 625 μs per slot. Therefore l is 16 × 625 μs × 256 = 2.56 s. [1]

3.2 Modelling with Simplified Exponential Approach

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 possible in principle 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. The interval between two online events (so called backoff time) is specified in [1] as being randomly chosen from range [0, 1023] time slots.

Having a duration of 625 μs to cover one timeslot, the maximum random backoff time is 0.64 s. We assume that there are on average *t*/*b* detection tries, so the simplified resulting formula is:

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

3.3 Modelling Adjusted by Laboratory Measurements

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 formula above, but to get a better fit an additional empirical exponent was introduced. Fitting to the data resulted in:

$$P_3(t) = 1 - e^{-0.24 * t^{2.68}} \tag{3}$$

A comparison of the three functions together with the results of the following section is shown in Fig. 4. Since we saw that our simplified exponential approach did not fit the real world measurement as we expected it to be, we took a closer look at the Bluetooth core specification. The following section refers to a specific inquiry mode that gives a significant reason for deviations.

3.4 Modelling Considering Interlaced Frequency Scanning

Starting with the Bluetooth specification 1.2 [1] a new interlaced scan mode was introduced which should dramatically reduce detection times at the expense of higher energy consumption on the side of the device being detected. Since this has a major influence on the average detection time we decided to redo a preciser modelling of the detection mechanism based on the analysis of Chakraborty et al. [14]. We focused only on a single case in the Chakraborty model which gives a very detailed analysis of the detection process based on the timing offsets between the detector and the device. However, for the majority of the cases it boils down to the following scenario (in [14] without interlaced mode):

1. The device sends on a frequency which is in the current train of the detector, then it gets detected more or less immediately (with an average delay less than 10 ms)
2. If not, the device may only get detected after the train switch which results in a delay of up to 1.28 s.

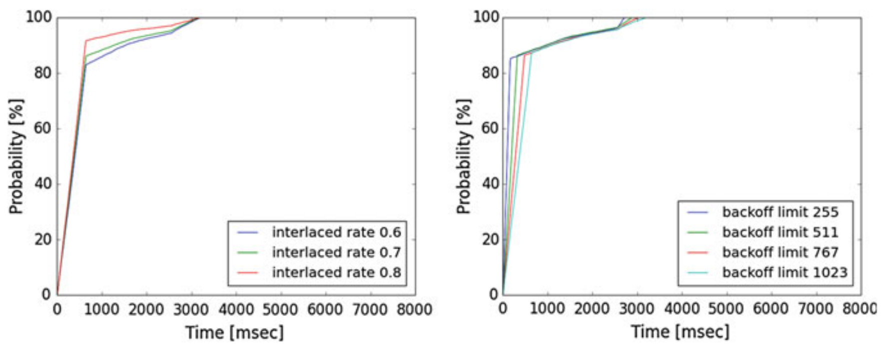


Fig. 3 Detection probability depending on interlace rate (*left Fig. 3a*) and backoff limit (*right Fig. 3b*)

3. In the rare case of an overlap of the scanning interval with both trains we might get another miss incurring another delay of up to 1.28 s.
4. In all cases a random backoff time of up to 0.64 s (1,024 time slots) is added before the second detection (resulting in final identification) can take place.

In interlaced mode there is no difference between the trains so essentially there is always immediate detection (average delay less than 20 ms) with the additional backoff. Unfortunately it is not clear in advance whether a vehicle uses interlaced mode or not so we have to assume some proportion of vehicles using that mode. The specifications recommend this mode for rather powerful devices like mobile phones. Since we assume a high proportion of those devices in our urban traffic scenarios, we did tests with detection probabilities between 60 and 80 % shown in Fig. 3a. The second parameter which allows for variation is the backoff limit which is set to 1,024 slots (0.64 s) in the original specification but may be smaller in some implementations [1]. We decided to investigate here the values recommended in the Chakraborty paper, see Fig. 3b.

In general, the influence of both parameters seems to be rather minor concerning the general shape of the function. While the interlaced rate influences the value up to which the first incline (which is solely based on the devices using the interlaced mode) raises, the backoff limit mostly influences the steepness of this first incline which is clear from the fact that it is dominated by the random backoff. As a result, we decided to use the interlaced rate of 0.7 and the standard backoff limit of 0.64 s in the further evaluations. Comparing the approaches from above we see that the "fastest" detection model is the final one incorporating the interlaced modes (cf. Fig. 4), which also has the additional property of bimodality in the higher probabilities which will be of importance in the later comparison with the real world data.

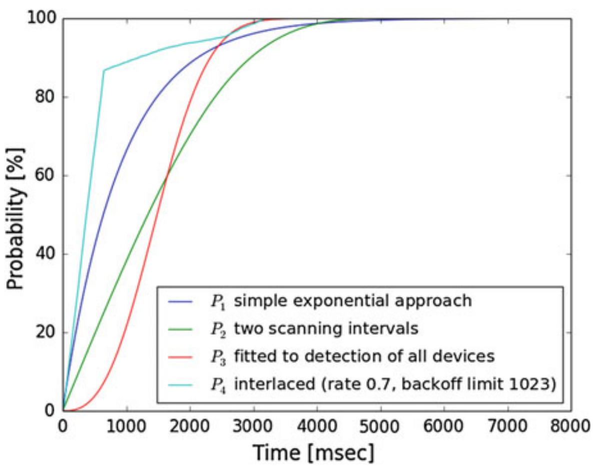


Fig. 4 Comparison of the different analytical approaches

4 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 pure 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 where 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.

4.1 Independence of Detections

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.

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

In that formula 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.

This property is also crucial for the fact that the simulation should deliver comparable results independent of the length of the time step applied in the simulation. We verified this property by running the simulation with different time step lengths without noting any difference in the detection curves (see Sect. 6.1).

4.2 The Interlaced Mode

The implementation of the interlaced mode needs a fundamental change in the architecture of the Bluetooth module in the simulation. Instead of independent decisions in every timestep we need to calculate a probable detection time on the entry in the detection radius and keep that information while the vehicle travels. Since this approach is so different it was not implemented and evaluated in the simulation yet but will be pursued in future versions.

5 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 Fig. 5) 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 11.01.2011.

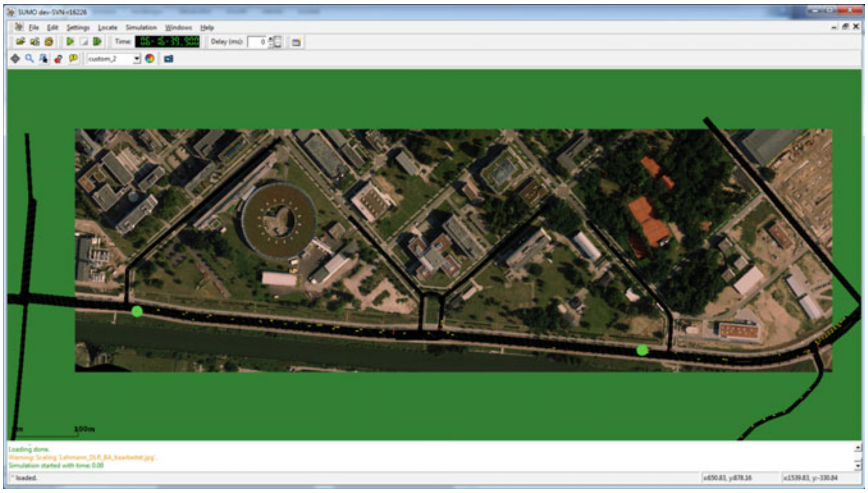


Fig. 5 Test track scenario in the final SUMO simulation (*green points* denote the Bluetooth detectors)

There is a total demand of about 30,000 vehicles including about 4 % trucks and busses. In a first step the scenario was calibrated to the detector data so 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 equipment rate of 30 %
- detection range 100 m.

6 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 one (respectively two) 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 s 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 6 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

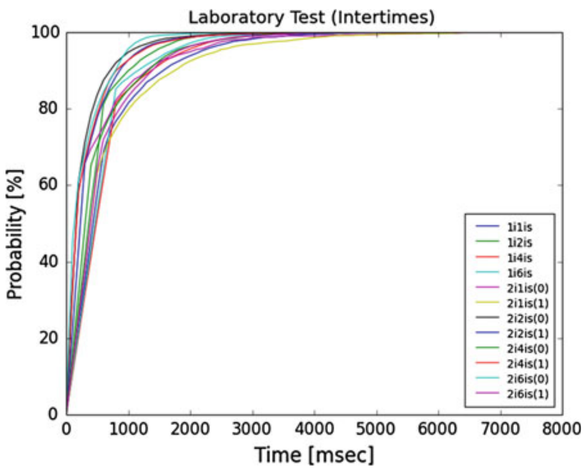


Fig. 6 Probability density for the laboratory tests

that more than 80 % of all detections are realised within a time interval of 1 s (1,000 ms). 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 providing that it is in the detection range. In the laboratory test nearly 100 % of all detectable devices were detected after 3 s irrespective of the number of BTreceivers and BTsenders.

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 2-h field test which took place on August 20th 2013 between 6 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 1.4 km) on the so called WISTA area in Berlin-Adlershof were used (see Fig. 7). 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).

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

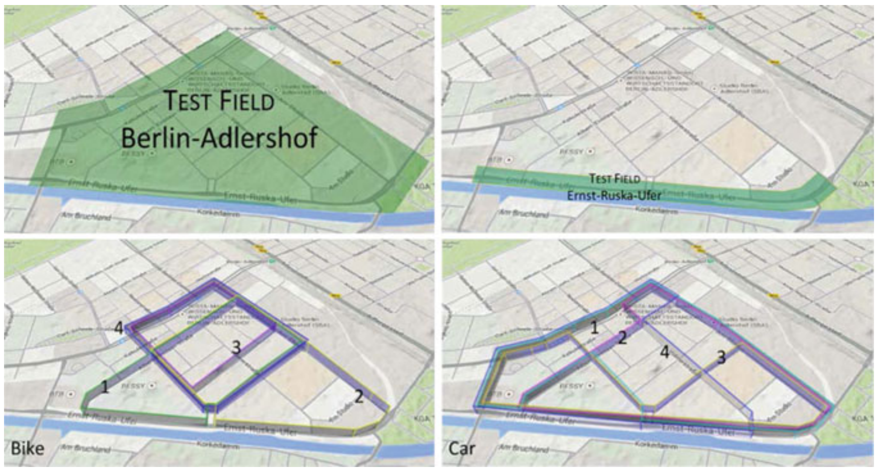


Fig. 7 Defined field test observer routes (*map source* Google): Ernst-Ruska-Ufer (*upper right*), car and cyclist routes (*lower right* and *left*) in second field test

Bluetooth detectors so that during the 2-h 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 Fig. 8a. 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. Figure 8b shows the results from the long-term measurements. It is obvious that the probability density is quite similar. Due to still undefined explicit error

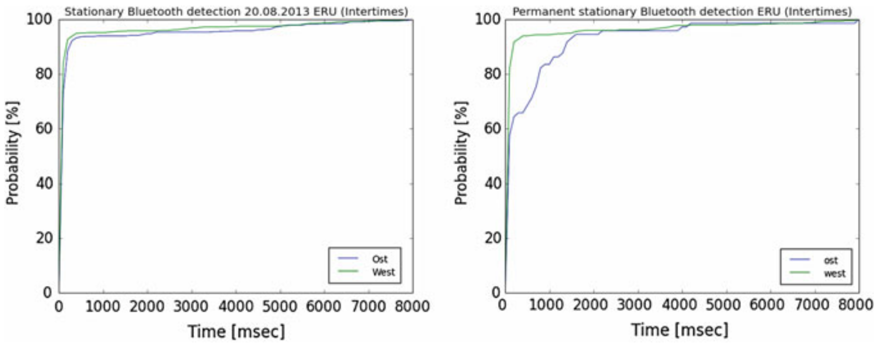


Fig. 8 Probability density for field test results on Ernst-Ruska-Ufer (*left Fig. 8a*: 2-h test on August 20th 2013; *right Fig. 8b*: permanent stationary Bluetooth detection)

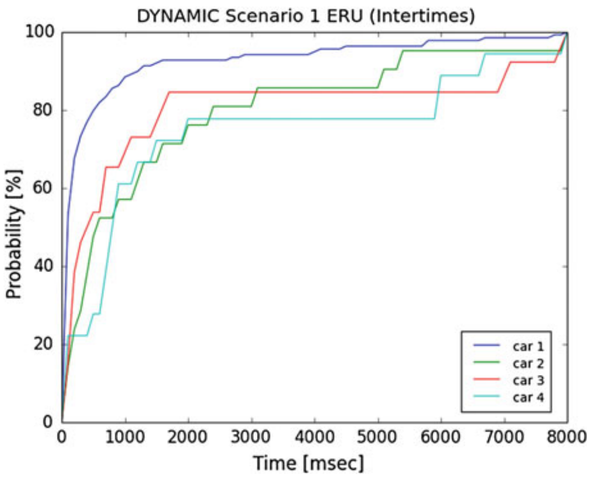


Fig. 9 Probability density for field test results on Ernst-Ruska-Ufer using moving observer (cars)

values, the increase is less sharp in comparison to the laboratory data. Especially between 1 and 6 s the course of the function is smoother than that under laboratory conditions. The effects where no inclination is observable within longer time slots (e.g. from 1,500 to 5,000 ms in Fig. 8a and from 2,000 ms to nearly 4,000 ms in Fig. 8b) can probably be attributed to the devices in non interlaced scanning mode which need considerably longer detection times (compare to Fig. 4). This topic needs further investigation however.

The results collected from moving Bluetooth observer vehicles are illustrated in Fig. 9. 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 2,000 and 7,000 ms.

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 (Fig. 10) to see whether the results affirm our conclusion. These additional field tests took even place on August 20th 2013, but from 9 to 10 a.m. and 1 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 11 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.

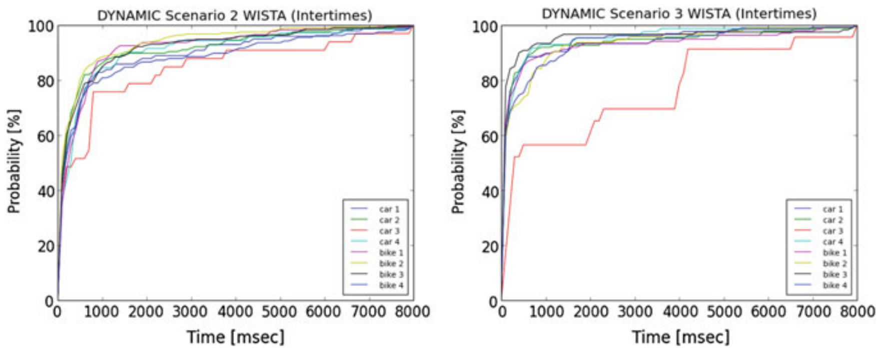


Fig. 10 Probability density for additional field test results on WISTA area using cars and cyclists as moving observers

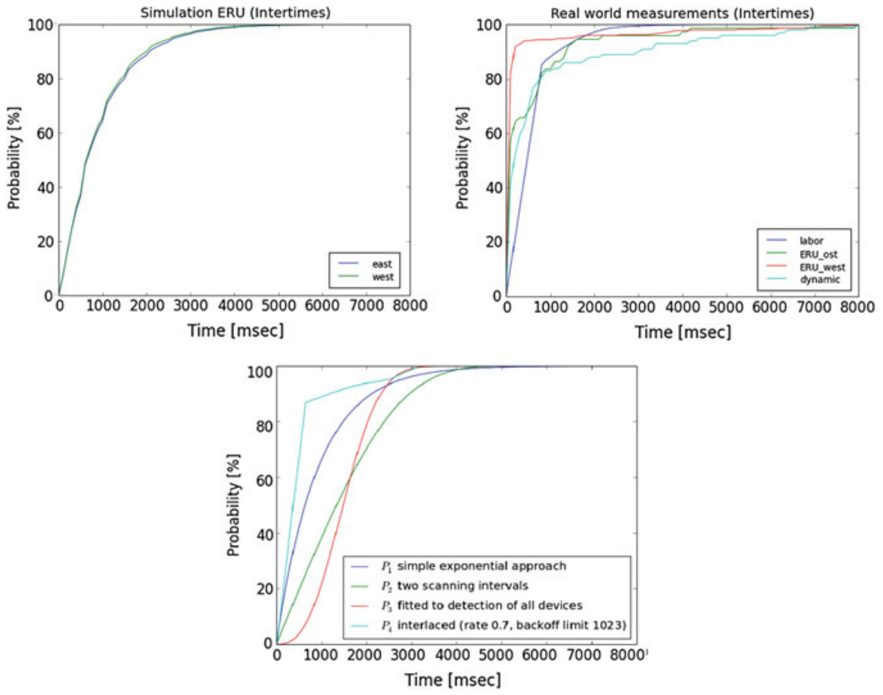


Fig. 11 Probability density for simulation results (*top left Fig. 11a*) compared to a summary of most important results from real world measurements (*top right Fig. 11b*) and the theoretical results (*bottom Fig. 11c*)

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 Fig. 8, cf. curve 'ERU_ost' and 'ERU_west' in Fig. 8b).

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 are small, so that the chances to encounter are statistically independent events. Therefore, the existence of those encounter respectively detection events can be described using an exponential distribution.

Please note that Fig. 11a was wrong in the conference version of this paper [8]. It was not calculated with a sufficient time resolution and did not incorporate all measurements and did thus not reflect the correct inter-detection times.

6.1 Validating the Independence of Detections

As we already noted in Sect. 4.1 there are some assumed properties of the detection delay function which need to be validated in the simulation. The first one is the independence concerning the length of the simulation time step. We verified this property by running the simulation with time step lengths of 0.01, 0.02, 0.05 and 0.1 s without noting any difference in the detection curves (see Fig. 12a).

Another property should be the independence of the detection delay concerning the traffic demand (or rather the number of detectable vehicles). This is not as obvious as it might seem at first sight because it includes the verification of our core assumption that the interval between two successive detections is a valid approximation for the length of the detection interval itself. While this assumption is

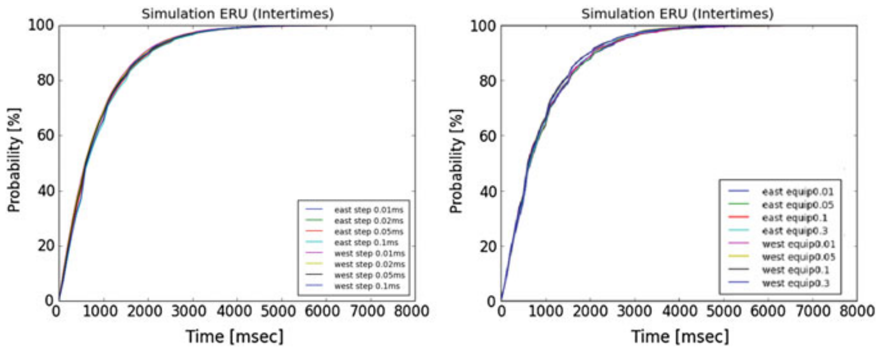


Fig. 12 Independence of detection probability of time step size (*left Fig. 12a*) and traffic demand (*right Fig. 12b*)

clearly wrong for very small traffic demands (if there is only one vehicle per minute, the interval between two detections cannot be smaller, provided redetections of the same vehicle are ignored), we still assumed that in our scenario described above we can get good data even for equipment rates below 1 %. To validate this we ran the simulation scenario with different equipment rates ranging from 0.3 to 30 %. The results are shown in Fig. 12b.

7 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 s 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 fairly 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 Fig. 8) where we often had no additional detections between second 2 and second 7. A reason might be a potential collision of so called FHS (frequency hopping sequence) packets, which are the response from detectable devices nearby to the inquirers transmitted identifier packets (send out on different frequencies within the inquiry process). That collision of FHS packets is strongly addicted to the number of potential detectable devices, that is, if the number of devices is increased, the device discovery will be delayed due to more packet collisions. But even when there are only two devices, in [14] they found out that if a device discovery is not successful within a few hundreds of time-slots, the discovery time will—more often than not—be delayed by more than 4,000 time-slots which is approximately 2.5 s.

As we find out from comparing the simulation and empirical data to our theoretical model, a pure simplified exponential approach lacks in the speed of discovery times (the incline of the curve is not steep enough). Further improvements are expected from including the preciser theoretical modelling based on the so called interlaced scan mode in the simulation. Since that needs a fundamental change in the architecture of the Bluetooth module in the simulation, the implementation will be pursued in future versions.

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

References

1. Specification of the Bluetooth system, core version 2.0 and higher, Bluetooth special interest group (SIG), 2004. http://www.bluetooth.com
2. Franssens A (2010) Impact of multiple inquirers on the Bluetooth discovery process—and its application to localization. University of Twente, Enschede
3. Peterson BS, Baldwin RO, Kharoufeh JP (2006) Bluetooth inquiry time characterization and selection. IEEE Trans Mobile Comput 5(9):1173–1187
4. Ruppe S, Junghans M, Haberjahn M, Troppenz C (2012) Augmenting the floating car data approach by dynamic indirect traffic detection. In: Proceedings of transport research arena—Europe 2012, Athens, Greece
5. Gurczik G, Junghans M, Ruppe S (2012) Conceptual approach for determining penetration rates for dynamic indirect traffic detection. In: ITS world congress 2012, Vienna, Austria
6. Krajzewicz D, Erdmann J, Behrisch M, Bieker L (2012) Recent development and applications of SUMO—Simulation of Urban MObility. Int J Adv Syst Meas 5(3, 4):128–138
7. SUMO—Simulation of Urban Mobility. http://sumo-sim.org/
8. Behrisch M, Gurczik G (2014) Modelling Bluetooth inquiry for SUMO. In: 2nd SUMO conference 2014, Berlin, Germany
9. Hoyer R, Leitzke C (2011) Verfahrenstechnische Bedingungen für die Reisezeitbestimmung mittels Bluetooth-Technologie. In: Proceedings of Heureka 2011, Kassel, Germany
10. Wasson JS, Sturdevant JR, Bullock DM (2008) Real-time travel time estimates using media access control address matching. Inst Transp Eng J (ITE J) 78(6):20–23
11. Duflot M, Kwiatkowska M, Norman G, Parker D (2006) A formal analysis of Bluetooth device discovery. Int J Softw Tools Technol Transf 8(6):621–632
12. Peterson BS, Baldwin RO, Kharoufeh JP (2004) A specification-compatible Bluetooth inquiry simplification. In: Proceedings of the 37th Hawaii international conference on system sciences (HICSS'04), Waikoloa, Hawaii
13. Kasten O, Langheinrich M (2001) 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
14. Chakraborty et al (2008) Analysis of the Bluetooth device discovery protocol. Wireless network (WINET). Springer Science+Business Media, LLC, New York