SUMO 2015 – Intermodal Simulation for Intermodal Transport

Proceedings

Berichte aus dem DLR-Institut
für Verkehrssystemtechnik

Band 28

DLR Deutsches Zentrum
für Luft- und Raumfahrt

Berichte aus dem DLR-Institut für Verkehrssystemtechnik

Volume 28

Proceedings of the
SUMO2015

Intermodal Simulation
for Intermodal Transport

May 7 + 8, 2015
DLR, Berlin - Adlershof

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

ISSN 1866-721X

DLR-TS 1.28

Berlin, May 2015

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

Preface

Dear reader,

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

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

This volume contains the proceedings of the SUMO2015 – Intermodal Simulation for Intermodal Transport Data, which was held from 7th to 8th May 2015 in Berlin-Adlershof, Germany. SUMO is a well established microscopic traffic simulation suite which has been available since 2002 and provides a wide range of traffic planning and simulation tools. The conference proceedings give a good overview of the applicability and usefulness of simulation tools like SUMO ranging from new methods in traffic control and vehicular communication to the simulation of complete cities. Another aspect of the tool suite, its universal extensibility due to the availability of the source code, is reflected in contributions covering rapid scenario prototyping and interfacing improvements to govern microscopic traffic simulation results.

The major topic of this third edition of the SUMO conference is the interaction of different types of traffic and intermodality. Several articles cover heterogeneous traffic networks as well as logistics and pedestrian extensions to the simulation. Subsequent specialized issues such as disaster management aspects and applying agile development techniques to scenario building are targeted as well. The conference's aim was bringing together the large international user community and exchanging experience in using SUMO, while presenting results or solutions obtained using the software or modeling mobility with open data. Let you inspire to try your next project with the SUMO suite. There are many new applications in your environment.

Prof. Dr.-Ing. Karsten Lemmer

All contributions have been refereed as abstract by the International Scientific Committee

International Scientific Committee

Constantinos Antoniou (National Technical University of Athens, Greece)
Ana Bazzan (Universidade Federal do Rio Grande do Sul, Brazil)
Hilmi Berk Celikoglu (Istanbul Technical University, Turkey)
Robbin Blokpoel (Imtech Traffic & Infra, Netherlands)
Quentin Bragrad (University College Dublin, Ireland)
Christine Buisson (Transport and Traffic Engineering Laboratory (LICIT), France)
Winnie Daamen (Technical University Delft, Netherlands)
Jakob Erdmann (German Aerospace Center, Germany)
Jorge E. Espinosa Oviedo (Politécnico Colombiano Jaime Isaza Cadavid, Colombia)
Gunnar Flötteröd (KTH Royal Institute of Technology, Sweden)
Ellen Grumert (Linköping University, Sweden)
Florian Hagenauer (University of Paderborn, Germany)
Marek Heinrich (German Aerospace Center, Germany)
Mario Krumnow (Technical University of Dresden, Germany)
Vincenzo Punzo (University of Napoli, Italy)
Rosaldo Rossetti (University of Porto, Portugal)
Andreas Schadschneider (University of Cologne, Germany)
Sebastian Schellenberg (Friedrich-Alexander Universität Erlangen-Nürnberg, Germany)
Jörg Schweizer (University of Bologna, Italy)

Organisation Committee

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

Table of Contents

1 Experiences with SUMO in a Real-Life Traffic Monitoring System

Karl-Heinz Kastner and Petru Pau;
RISC Software GmbH, Softwarepark 35, 4232 Hagenberg, Austria

{karl-heinz.kastner, petru.pau}@risc.software.at

Abstract

In this paper, we give an overview of a traffic monitoring system currently operating for Upper Austrian roads, which uses SUMO for providing traffic data on roads where real-time traffic information is unavailable. We describe a method for reliably integrating traffic information sent by sensors mounted on driving cars, which works by detecting the most probable routes covered by these cars. Traffic information obtained from dense networks of vehicle counters needs special interpretation, especially when used for calibrating a traffic simulation that runs in parallel.

Keywords: traffic monitoring systems, floating-car data, map-matching, simulation calibration.

1.1 Introduction

As project "ITS Austria West" enters its second year of service, we are proud to give a short report on the acquired experiences, encountered problems, new requirements and the implemented solutions. Our previous papers [2] and [3] offer detailed descriptions of various topics from the project; we focus here on very specific problems which arose since the system became operational.

The main goal of the project is to provide reliable, up-to-date traffic information for the roads of the state Upper Austria. Basis data come from the state authorities and includes a soon-to-be public database of Austrian roads. Real-time traffic data is collected from a set of data sources, graciously provided by some logistic, taxi and ambulance companies, as well as by the Austrian highway authority Asfinag.

The main component of our project, TOMS (Traffic Online Monitoring System) aggregates the real-time data and issues periodically (basically every minute) snapshots of the traffic situation. On the one hand, these snapshots are used to update a set of WMS (Web Map Service) layers that can be visualized as LOS (Level of Service) in a provided web-site; on the other hand, delays or traffic jams are packaged and sent to the central Austrian traffic information platform, VAO (Verkehrsauskunft Oesterreich).

The number of roads affected by real-time traffic data is significantly smaller than the number of considered road-segments in Upper Austria: Data packets cover less than 10 000 road segments, while the number of roads segments monitored by TOMS is greater than 180 000.

In order to generate and/or analyze traffic information for uncovered roads, we simulate the traffic with SUMO [1].

TOMS can thus start SUMO, connect to it, get periodically lane and vehicle information, and calibrate the simulation by adding or removing vehicles.

The initial demand model for the simulation has been provided by government as well and has a comprehensive set of trips on Upper Austrian roads, distributed hourly. We used this demand model to generate a set of trips along routes computed with our own algorithms. The 1.6 million trips led to huge traffic jams during rush hours; they needed more than a week simulation time to dissolve. We refined the set of trips through a (rather large) number of successive simulation runs, whose output was analyzed in order to detect delayed vehicles: They were either sent on alternative routes, or inserted earlier into the simulation. With the calibrated demand model all vehicles reach their targets before 1:00AM next day – in other words, SUMO simulates quite realistically the daily Upper Austrian traffic in less than 25 hours simulation time.

In this paper, we concentrate our discussion on specific problems occurring in the day-to-day work of TOMS:

1. Integration of real-time data: We described briefly in [2] our intended method for integrating numerical information received from sensors installed on running vehicles. The implemented solution is presented here in more detail. The results are used to calibrate the simulated traffic on road segments covered by the running vehicles.

   Data coming from static counters may need special treatment, especially if the detectors are close to each other: Lots of vehicles can be counted by more than one detector in a given time interval. The on-line calibration of the simulated traffic needs to take this into account.

2. Interfacing one or more simulations: Quite often the need arose to start more than one simulation in parallel, e.g. in order to analyze the effect an event on a specific road can have. Our system contains components that enable starting and controlling simulations remotely, with or without on-line calibration. The generated traffic situations can be integrated into specific WMS layers, which can be visualized in TOMS.

3. Addressing the efficiency problems: The microscopic simulation is too slow for the traffic during rush hours. With the mesoscopic version of SUMO, however, we obtain excellent results.

We begin by mentioning some interesting information about the day-by-day use of or system.

1.2 ITS Austria West in Service

ITS Austria West has run almost without interruption ever since becoming operational, more than one year ago. The main component, TOMS, provides traffic information with a one-minute periodicity. Additional components take care of receiving, pre-processing and forwarding the incoming real-time data.

Here are some numbers:

- The monitored road network has 248 749 nodes and 323 282 road segments (edges) with a total length of 27 437 km.
- There are 3 traffic information sources providing vehicle counter data, one of them monitoring motorways. The detectors monitor around 200 road segments.
- There are 5 traffic information sources providing floating car data. On average, around 6000 road segments are covered by the FCD.
- Every minute, the real-time traffic data from the last 15 minutes are analyzed and integrated into a traffic situation snapshot. The processing time very exceeds 30 seconds.
- The calibrated demand model for the simulation has 1.2 million routes and 1.6 million vehicles.

1.3 Floating-car data

In order to correctly integrate floating-car data, we had to come up with solution to the following problems:

1. Road-segment detection: We need to reliably decide from which road segment a floating-car data packet was sent.
2. Velocity uncertainty: For small values of the velocity, we have to assess whether the cause is a problem on the road (traffic jam), a particular, non-significant event (like waiting in front of a red light), or a malfunction of the sensor.
3. Calibrating simulated traffic: On road segments where floating-car data becomes available, some traffic values computed from FCD may differ significantly from the simulated traffic.

In the following, we begin by shortly discussing the main characteristics of floating-car data providers grouped after the behavior of vehicles. We then describe our map-matching algorithm that solves the first problem. We show how the output of this algorithm – in fact, sets of routes driven by vehicles – is used to solve the second problem, the velocity.

1.3.1 Long-distance drives

The sensors installed on vehicles driving long-distance trips deliver the most convenient, reliable and rich sets of data. Incidentally, the location devices send their readings in very small time intervals – 5 to 10 seconds.

As the long-distance drives tend to be on important (high level) streets, the driven road segments are normally present into the road network monitored by TOMS. Most of the times, a simple, geometry-based map-matching algorithm successfully pinpoints the road segment from which floating-car data packets were sent. Such packets contain both the coordinates and the heading, and they are employed in tandem to strengthen and validate the road-segment detection.

The image below (Figure 1-1) shows a segment of a typical long-distance drive.

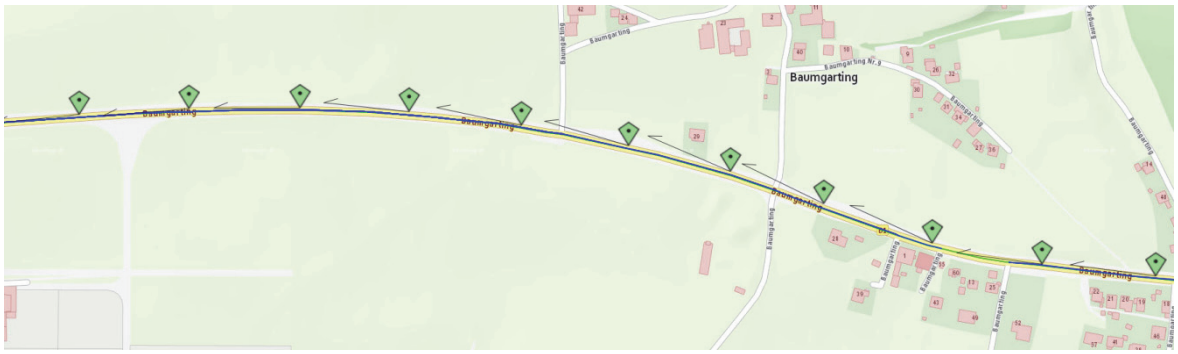


Figure 1-1: Section of a long-distance drive.

However, the long-distance drives cover mainly road segments outside of cities, so they are less important for computing traffic situations in urban areas.

1.3.2 Taxi drives

The trips driven by taxis are rather chaotic, without clear-cut begin-end points, quite often covering the same road segments more than one time. Since they happen mainly in urban areas, where lots of less-significant road segments are not monitored by TOMS, some readings cannot be properly matched.

There may also be sequences of readings with zero (or near-zero) velocity.

The data packets come at variable time intervals, between 10 and 60 seconds. Figure 1-2 contains the trajectory of a typical taxi drive.

Figure 1-2: Normal taxi drive.

1.3.3 Ambulance drives

In essence quite similar to taxi drives, the ambulance drives have some specificity:

- Expectedly, routes have an endpoint near a hospital (see Figure 1-3).
- Long sequences of floating-car data packets with 0 velocity and random heading occur at the other endpoint.
- The routes can extend outside cities.

Figure 1-3: Typical ambulance drive.

Our initial, geometry-based approach for identifying the driven road segments performed very well with data from long-distance drives and less accurately with the other sensors: We got lots of false positives, i.e., road segments where the aggregated FCD "reported" traffic problems (delays or traffic jams). Filtering out possible stops at traffic lights did too little to improve the outcome.

Specifically designed map matching methods became a necessity.

1.3.4 Map Matching

The literature concerning map matching is quite vast; we took some ideas from [4] and implemented first a fast geometry-based map matching, whose efficiency was supported by the ubiquitous R-Tree [5] which helps speed up all searches in our road network. Shortly, given a FCD packed containing coordinates, heading and velocity, we use the coordinates to find a set of nearest road segments and we use heading to identify the best match. All this happens, clearly, if the velocity information is plausible (negative values, or values unrealistically big, are known to occur every now and then).

For readings coming from long-distance drives this method works very well. Almost all road segments on the driven route are hit by at least one FCD packet and properly identified in the map-matching process. The traffic values can be reliably collected.

However, for urban drives (taxi or ambulances) the geometry-based map matching did not provide satisfactory results. This method attempts to identify road segments: It become clear that we should look for a method which detects *the most probable route* followed by the sensor.

To solve this problem we used a modified Dijkstra shortest-path algorithm.

1.3.5 Route-Based Map Matching

In order to detect the route followed by a sensor, we start with all available readings sent by this sensor and execute first a geometry-based map matching, in which we identify, for each reading, all road segments ("edges") within a certain distance.

> Recall that a normal Dijkstra algorithm works by associating some numerical values – "weights", e.g. edge length or drive time – to each edge in graph and then proceeds by looking for a path with minimal sum of weights between two given nodes. This

algorithm can be easily modified to look for shortest paths between two given edges. (Although the term "shortest path" applies literally only when the weight is a distance measure, we will use it in our presentation regardless of the nature of this weight.)

The edge weights, in our algorithm, are adjusted according to how well they fit the readings: If, in the first step, an edge was found to be close to a reading, then the weight of this edge will be decreased, proportional to its distance to the coordinates of the reading, and to the angular difference with the reading's heading.

Since the first and last reading can match to more than one road segment, our algorithm works with multiple start and end edges. When an end-edge is reached, a shortest path from a specific start-edge has been found. The algorithm computes first paths to every end-edge. During this computation, edges where readings can be mapped are considered and the corresponding readings are marked.

If the computation of start-end paths finishes but too few readings are marked, our algorithm continues by looking for shortest paths between the start edges and edges matching some of the unmarked readings. The computation stops when enough readings are marked (currently we need 2/3 marked readings).

The main result of this computation is essentially a data structure that offers fast access to shortest paths between start edges and edges on which readings are map-matched.

In the next phase, we effectively compute the best matching route (or routes) for our set of readings. This takes place incrementally, by counting the number of readings that can be mapped on the shortest path between the first reading and the third, then between the first reading and the fourth, and so on.

If, at such a step, the number of mapped readings decreases, we stop the process and split the set of readings: Somewhere along the way the vehicle has made a turn, and has reached a point with a different shortest path from the first reading. We record the last path that we found, isolate the mapped readings, and continue the whole process with the remaining readings.

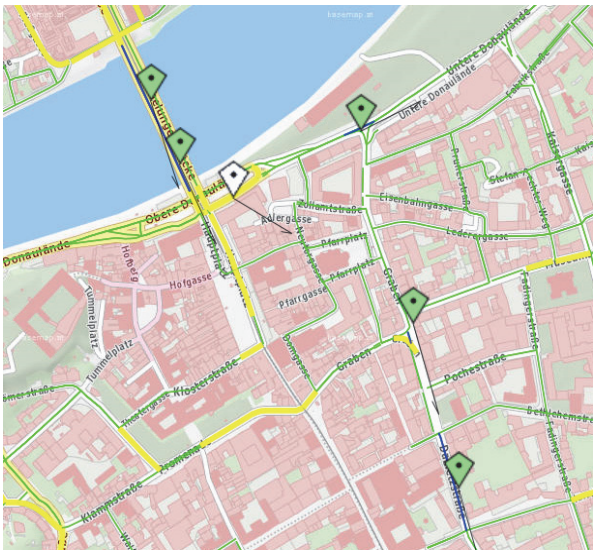


Figure 1-4: Geometry-based map matching.

Figure 1-4: Route-based map matching.

The algorithm ends when there are no more readings to consider. The output is a partitioning of the original readings, and for each part, a route containing the best matching edges.

The difference between the geometry-based and route-based map matching is evident from the above images: In Figure 1-, the geometry-based approach identifies (correctly, in this case) the driven edges, but there is no information on the road-segments in between. Figure 1-4, on the other hand, shows that the route-based approach has been able to identify two routes from the given readings.

1.3.6 Traffic Velocity from FCD

Once the routes driven by sensors are identified, computing the average velocity along them is immediate. From each such route we get an average velocity value on each of its road segments; we use this average also for road segments, belonging to the route, that are not hit by any FCD.

Eventually we are provided with a set of velocity values for each driven road segment (in cities, it is very often the case that routes driven by different sensors overlap, and hence road segments common to two or more routes get more than one velocity value). From this, we compute the velocity either as maximum, or by following the "85%" rule: We sort the values, discard the smallest 84%, and take the minimum of the remaining set.

The second problem mentioned at the beginning of this chapter, the velocity uncertainty, loses its importance: Indeed, zero-velocity readings can now be ignored if the average velocity along the route followed by the respective sensor is significantly greater.

1.3.7 Addressing Simulation Deviations

The treatment of differences between simulated traffic and readings from vehicle counters was mentioned in our paper [3], with an addendum in the next section.

Now that we have reliable traffic data on road segments spanned by FCD, we can analyze the simulated traffic on these segments and try to calibrate the simulation, if the discrepancies are too flagrant: We add or remove vehicles into, respectively from the simulation, on the corresponding lanes. The number of such vehicles is proportional to the speed difference.

1.4 Dense Networks of Vehicle Counters

Ever since we began receiving real-time data from the highway authority Asfinag, we noticed that the number of vehicles inserted into the simulation increased by an order of magnitude: At the beginning of each leg of simulation run (see [2] for a description of the interplay between TOMS and SUMO), between 2 000 and 10 000 new vehicles were generated. Without these counters, the number of new vehicles did not exceed 800.

The reason is that the Asfinag vehicle counter network is quite dense (see Figure 1-5) and lots of vehicles were read by more than one counter. By neglecting this, we artificially overloaded the simulated traffic on monitored road segments. The outcome was an increase in simulated traffic jam occurrences on highway segments.

In order to solve this problem, we created from our internal road network a dedicated graph structure which reflects transfer possibilities from one vehicle counter to another. Basically, each node in this graph represents a counter. Two nodes A and B are connected if the counter at B is the first one that can be reached from A by a vehicle running along the highway, from the road segment containing A to the road segment containing B. This transfer graph is thus a directed graph, non-transitive (if the graph contains edges [A,B] and [B,C], then edge [A,C] is not contained) and in Upper Austrian case has no loops.

Figure 1-5: Asfinag VDL in Linz Area.

We adapt the number of vehicles inserted into the simulation by taking into account the number of nearby nodes – i.e., reachable by a vehicle in a given amount of time – in the transfer graph. In our current approach, we divide the difference between the measured and the simulated traffics by the number of nearby nodes that come "before" the current node.

The results were more than satisfactory. The artificial traffic jams did not occur anymore, and the calibration values became less and less dramatic.

1.5 Handling Multiple Simulations

Besides the main goal of the project – providing reliable traffic information – we wanted to be able to study various exceptional scenarios, like accidents on specific road segments, or road works in sensitive areas (e.g. bridges). A possible solution was by running in parallel more traffic simulations, each with its own scenario (road network, demand model, calibration methods).

Our system currently provides mechanisms for starting and communicating with traffic simulation applications, on more than one computer:

- A small software component needs to be installed on each remote computer. It is responsible with starting/stopping the simulation and configuring its communication with TOMS, which takes place over a TCP/IP channel. It acts as a server, handling requests received from TOMS.
- A traffic simulation program (e.g. SUMO) must be installed on each remote computer. The road network, as well as the demand model, need to be there as well. In SUMO case, TraCI is locally configured to communicate with TOMS.
- TOMS centrally manages the remote simulations:
    - It connects via TCP/IP to the remote machines able to run simulations;
    - Via TOMS, the user can start and stop the remote simulations;
    - TOMS connects to running remote simulations and communicates with them using the same principles as with the local simulation;

- If calibration of a remote simulation is enabled, TOMS uses the same mechanisms as for the local simulation;
- Remote simulations' output can be integrated in specific WMS layers, thus ensuring that visual inspection is always possible.

1.6 Microscopic vs. Mesoscopic Traffic Simulation

The standard simulation engine for SUMO is microscopic: It simulates every car in detail, with breaks, accelerations, lane changes, turns, etc. It provides a faithful modelling of the reality, but it requires both road models and demand models to be very accurate. Small errors in the road network can have catastrophic effects for the whole simulation. Moreover, the computation effort is considerable: As mentioned in our paper [2], the rush hour traffic cannot be simulated faster than real-time, even on powerful machines (we work with computers having Intel Core I7 processors, running at 3.4 GHz).

Due to these reasons, we tried the mesoscopic simulation engine of SUMO (not open source), which handles the traffic in priority queues. The benefit is enormous: It speeds up the simulation with a factor of 50 to 100. The input data for the microscopic SUMO can be taken as-is as input for mesoscopic SUMO; some functions and outputs for vehicles are missing, as well as the treatment of some TraCI commands, but with the support of SUMO developers we were able to provide the required implementations.

The offline calibration of the demand model, for which the microscopic core needs several months, took less than two weeks with the mesoscopic. In online mode, where we always simulate the traffic for the next 5 minutes at a time, even during the rush hours it takes meso-SUMO less than 20 seconds to perform.

We have to mention, though, that the demand model calibrated with the mesoscopic simulation engine is not appropriate for the microscopic simulation: Traffic jams occurring at rush hours tend to persist for days (simulation time) with the microscopic SUMO, although in the mesoscopic simulation they dissolve easily.

We adapted all our systems to work with meso-SUMO: For our purposes, where we have to handle very large scenarios (see the numbers in the first section of our paper), this was the only acceptable solution.

1.7 Conclusions and Future Work

We are now in a phase where the real time and the static data are very well processed with our described methods. We can generate a realistic traffic situation with the real time data and the simulation provides a good illustration of the Upper Austrian traffic.

Statistically, the simulated traffic is still "not close enough" to the values collected from sensors. We plan to work on intelligent mechanisms for continuous on-line calibration of the demand model, so that eventually we can reach a point where the adjustments are negligible.

We currently ensure the anonymity of our data providers; however, in order to be prepared for all possible requirements our future providers may have, we plan to strengthen our anonymisation procedures. On the one hand, incoming data will be filtered, in order to

reduce the possibilities of detecting specific drive patterns; on the other hand, there will be a delay in the data flow (latest readings will be kept for a minute or two before being saved into our databases, or sent to TOMS).

We foresee significant negative influences of this floating-car data pre-processing, especially for the route-mapping algorithms; the consequence will be a decrease in the quality of our traffic situations. We are currently analyzing in detail all possible drawbacks, and are looking for ways to minimize them.

1.8 References

[1] Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D.: SUMO - Simulation of Urban MObility: An Overview, Barcelona, Spain, SIMUL 2011, 2011, 63-68

[2] Kastner, K.-H., Keber, R., Pau, P., Samal, M.: Real-Time Traffic Conditions with SUMO for ITS Austria West, "Simulation of Urban Mobility", LNCS 2014, pp 146-159, Springer Verlag.

[3] Kastner, K.-H., Keber, R., Pau, P.: TOMS – Traffic Online Monitoring System for ITS Austria West, Proc. of the 2nd SUMO Conference Berlin 2014, Springer Verlag, to appear.

[4] White, E.C., Bersntein, D., Kornhauser, A.: Some map matching algorithms for personal navigation assistants, Transportation Research Part C 8 (2000) 91-108, Elseveier Science Ltd.

[5] Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*-tree: An Efficient and Robust Access Method for Points and Rectangles+, Proceedings of the 1990 ACM SIGMOD international conference on Management of data, Pages 322-331, Praktische Informatik, Universität Bremen, D-2800 Bremen 33, West Germany.

2 LuST: a 24-hour Scenario of Luxembourg City for SUMO Traffic simulations

Lara Codeca, Raphael Frank, Thomas Engel;
Interdisciplinary Centre for Security, Reliability and Trust
University of Luxembourg, 2721, Luxembourg

{lara.codeca, raphael.frank, thomas.engel}@uni.lu

Abstract

Various vehicular communities ranging from telecommunication to infrastructure are working on problems related to traffic congestion, intelligent transportation systems, and mobility patterns using information collected from a variety of sensors. In order to test the solutions, the first step is to use a vehicular traffic simulator and an appropriate scenario. Many mobility simulators are available, but a common problem is finding a realistic traffic scenario. The aim of this work is to provide a scenario able to meet all the common requirements in terms of size, realism and duration, in order to have a common basis for the evaluations. In the interest of building a realistic scenario, we decided to start from a real city with a standard topology common in mid-size European cities, and real information concerning traffic demands and mobility patterns. In this paper we show the process used to build the Luxembourg SUMO Traffic (LuST) Scenario, and present a summary of its characteristics together with an overview of its possible uses.

Keywords: Vehicle-to-X Simulation, Infrastructure-to-X Simulation, Scenario Generation.

2.1 Introduction

This Many vehicular communities are working in a variety of areas that range from crowd-sourcing of information to safety applications. They are also making structural studies concerning infrastructure communications, traffic light systems, and more generally intelligent transportation systems. In order to study mobility patterns, traffic congestion or new communication protocols, we need a vehicular traffic simulator and an appropriate scenario to evaluate new proposals. For the vehicular networking community, the behaviour of the single vehicle is usually important and needs to be modelled. For this reason a microscopic mobility simulator is generally chosen. In our case, we chose the Simulator of Urban MObility (SUMO) [1]. In this context, the common problem that we must face is the lack of properly-working and freely-available scenarios for the community.

In 2014, during the SUMO User Conference, realistic traffic scenarios from the city of Bologna [2] were released to the community. The scenarios, built in the iTETRIS [3] framework, give a very good starting point for the community, but they present some limitations such as the traffic demand, which is only defined over one hour; also, the size of the scenario is relatively

small. Alternatively, the SUMO community provides the TAPASCologne1 scenario package, which includes road networks imported from OpenStreetMaps (OSM) [4] and the traffic demand for the period between 6:00 and 8:00 in the morning. Unfortunately this scenario is usable only with difficulty, requiring a lot of work to be done to improve the network quality, and to verify how routes are mapped onto both the network and the traffic demand.

In order to focus on car-to-X problems and solutions, the community needs a scenario that fulfils the following requirements: (1) It has to be able to support different kinds of traffic demand such as congested or free-flow patterns. (2) It should support different scenario dimensions. (3) It has to include different road categories (e.g. residential, arterial and highway). (4) It should allow multi-modal evaluations. (5) Is should describe a realistic traffic scenario over one day (i.e. avoid gridlocks and teleportations2).

Due to the lack of scenarios that meet all these requirements, the usual approach is to build a simple scenario that fulfils the purpose of the application. This approach results in several problems that are well known to the community, the most prominent being the lack of repeatable experiments allowing the comparison of different solutions or approaches to solving the same problem. Another problem that may be encountered is the specificity of the scenario and the consequent lack of generalization or realism.

We decided to use the road network of a real city as basis for our scenario in order to reproduce real traffic demand and mobility patterns. We chose the City of Luxembourg because its topology is comparable to that of many of European cities and because we have access to its traffic statistics.

In this paper we present the process used to build the Luxembourg SUMO Traffic (LuST) Scenario, a summary of its characteristics and an overview of its possible uses.

2.2 LuST Scenario

Topology. In order to create a realistic scenario we decided to start from a real mid-sized European city. The topology of European cities consists of a central downtown area, surrounded by all its different neighbourhoods, which are linked by arterial roads [5]. Another important characteristic is the presence of a highway in the outskirts that surrounds the city. The size of the city is another very important property: the scenario must be big enough to show the standard congestion pattern visible in modern cities, but it must be small enough to permit simulations in a reasonable amount of time. The City of Luxembourg meets these requirements.

After choosing the city, we used OpenStreetMap (OSM) to extract its road topology. An OSM file contains all the necessary information about the environment and is trustworthy [6]. We used JOSM3 to extract and manually select and change points of interest and road segments. In this phase we returned information about roads (of any kind), traffic light, locations and

[1] This scenario can be found at http://sumo.dlr.de/wiki/Data/Scenarios/TAPASCologne.

[2] In SUMO a teleport occurs when a vehicle is blocked for too long in front of an intersection or collided with another vehicle.

[3] The JOSM Editor is available at https://josm.openstreetmap.de/

names of bus stops, and we also saved information about schools (i.e. location and kind) to be used in the activity generation process.

As the aim of this scenario is to have a working SUMO simulation, all the intersections were checked manually for correctness using an iterative process with JOSM, netconvert4 and SUMO to ensure that they did not represent an unrealistic bottleneck for the traffic flows.

To provide more flexibility, we decided not to impose any vehicle restrictions on any edge or lane. In order to maintain the traffic patterns close to reality, we modified the number of lanes in some segments of the roads. We tried to standardise the roads in order to obtain a scenario that could easily be modified in the future.

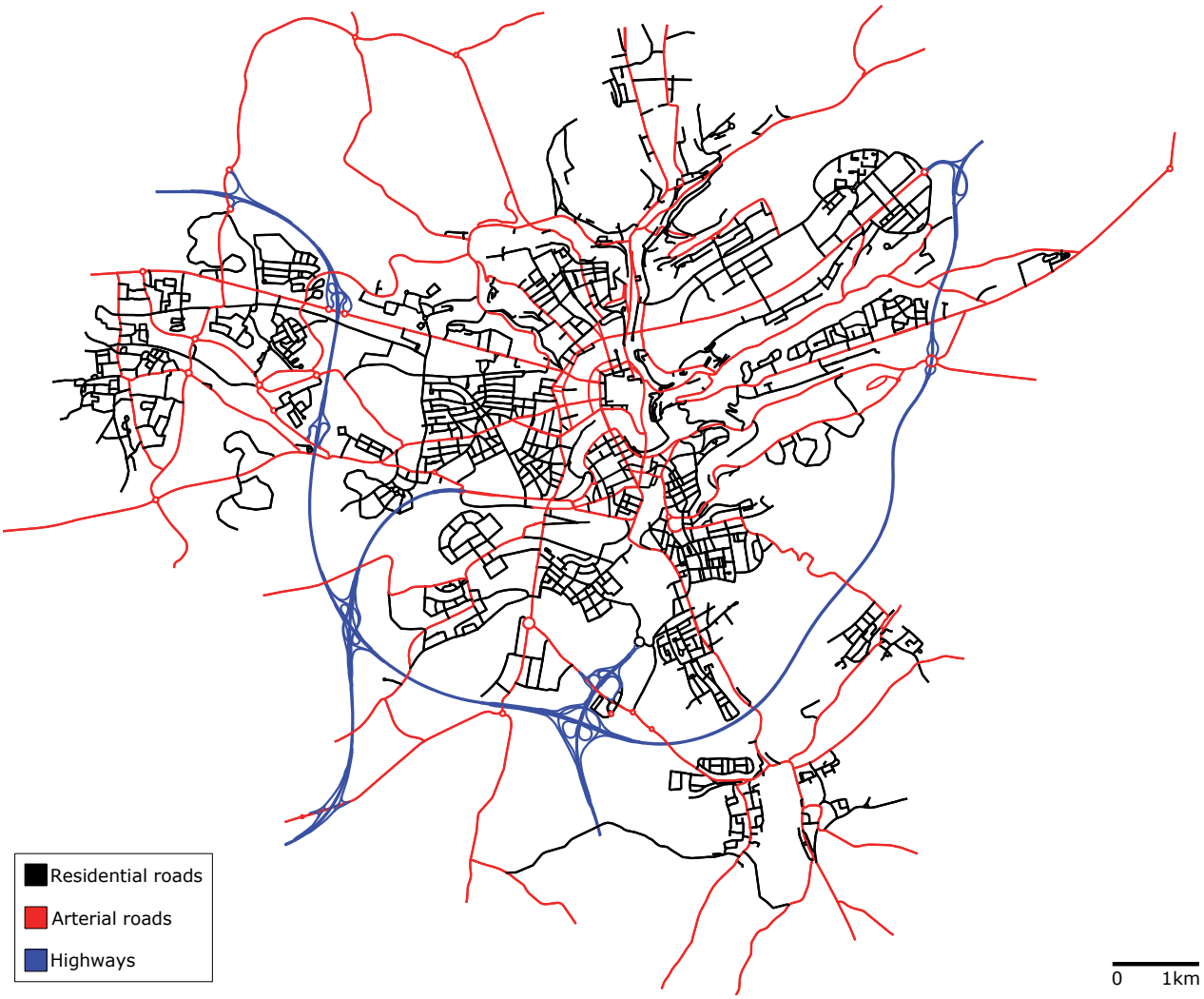


Figure 2-1: LuST Scenario Topology.

Figure 2-1 shows the topology of the LuST Scenario, with streets coloured by type. The highway is depicted in blue, the main arterial roads in red and the residential roads in black. The static information contained in the **net.xml** file is summarised in Tables 2-1a and 2-1b. The scenario covers an area of almost 156 km2 with a total of 931 km of roads of different types. In the SUMO network file an edge is defined as a segment between two nodes, it may have a shape, and it is divided in one or more lanes.

[4] See the netconvert wiki page at http://sumo.dlr.de/wiki/NETCONVERT

Demographics. In order to achieve realistic traffic patterns we used the data published by the government, which is available on the Internet site of the Luxembourg National Institute of Statistics and Economic studies (STATEC[5]) (e.g. population, age distribution) to generate the activity demand for the ACTIVITYGEN[6] tool. The activity demand required by the tool must contain information concerning schools, workplaces and residential areas. All of these are retrieved from OSM and STATEC.

Table 2-2a: LuST Scenario in numbers. Topology information.

Topology	
Area	155.95 km^2
Total nodes	2,376
Total edges	5,969
Total length edges	931.11 km
Total length lanes	1,571.4 km
Edges with 1 lane	3,944
Edges with 2 lanes	1,188
Edges with 3 lanes	764
Edges with 4 lanes	78

Table 2-1b: LuST Scenario in numbers. Intersections information.

Intersections	
Roundabouts	39
Total junctions	4,341
Traffic lights	203
Unregulated	16
Priority	1,914
Internal	1,969
Dead end	239

Mobility. We used a mobility study that describes traffic characteristics over recent years[7]. Based on this report we decided to tune the traffic demand around 140,000 vehicles per day. The public transport database was used to retrieve the information about bus routes[8]. A total of 563 bus stops was added to the scenario. As shown in Table 2-2, we added 38 bus routes inside the city for a total of 2,336 night and day bus movements per day. The location of the bus stops in the LuST Scenario is not the same as the one in the OSM file, although we tried to keep it as close as possible. For this reason, we had to rebuild the bus routes to match the new bus stop locations. Figure 2-2 shows an intersection located in the city centre; the yellow

[5] The STATEC internet site is https://www.statistiques.public.lu

[6] See the ACTIVITYGEN wiki page at http://sumo.dlr.de/wiki/ACTIVITYGEN

[7] The LuxTram Internet site is https://www.luxtram.lu

[8] The Mobility Internet site is https://www.mobiliteit.lu

squares are the inductive loops positioned 5m [7] from the intersection and the green boxes annotated with an H are the bus stops. We positioned the inductive loops at every intersection with a traffic light, on the highway, and on the on and off ramps (see Table 2-3). We fixed the location of each inductive loop close to the intersection to allow dynamic adjustments of the traffic light system using the information provided by the detectors.

Table 2-2: Bus information.

Buses	
Number of lines	38
Bus stops	563
Buses per day	2,336

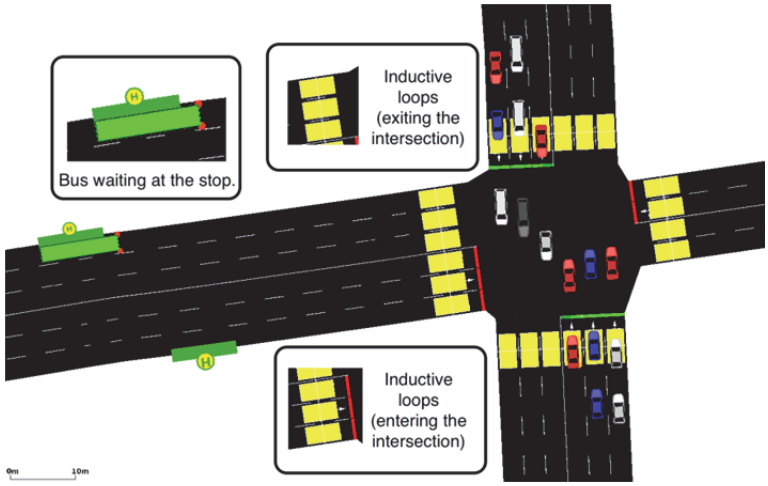


Figure 2-2: Intersection with bus and inductive loops.

Table 2-3: Inducrive loop information.

Inductive Loops	
Total number	3,161
Highways	94
Highway ramps	225
Intersections	2,842

Traffic demand. The ACTIVITYGEN tool utilises the definition of a network and the description of the population. It uses an activity-based traffic model that relies on a multi-modal trip planner including buses, cars, bicycles and pedestrians to derive the daily activities such as work, school, and free time. The output represents the traffic demand for the scenario. We separated the routes provided by the tools between vehicles and buses and optimised them using the SUMO **duarouter** tool. In order to allow the buses to wait at each stop, we added the **stop tag** to the routes proposed by ACTIVITYGEN.

Table 2-4a: LuST Scenario simulation in numbers: SUMO simulation short report.

Simulation Information		
	Total	Percentage
Inserted vehicles	138,361	
Teleports	39	0.0282
Collisions	3	0.0022
Jam	5	0.0036
Yield	10	0.0072
Wrong lane	21	0.0152
Emergency stops	7	0.0051

The short report provided by SUMO at the end of the simulation is shown in Table 2-4a. In the percentage column we see that all the issues (e.g. teleportation, collisions, emergency stops, etc.) that may be experienced by a vehicle during the simulation are lower than 0.05%, allowing us to assume that the scenario is running smoothly without bottlenecks and gridlocks. The traffic demand over the entire day is depicted in Figure 2-3. We can clearly see the morning rush hour peak at 08:00. Two smaller peaks are visible around lunchtime and in the evening, this being typical for city traffic scenarios. Figures 2-5a and 2-5b show respectively the traffic situation in the city centre during morning and evening rush hours.

Table 2-4b: LuST Scenario simulation in numbers: SUMO trip information report.

	Trip Information	
	Total	Average
Total vehicles	138,259	
Departure delays [s]	9,101,390	65.81
Waiting time [s]	39,744,745	287.47
Travel time [s]	98,070,337	709.32
Travel length [km]	1,016,033.78	7.35
Vehicle travel speed [m/s]		10.36

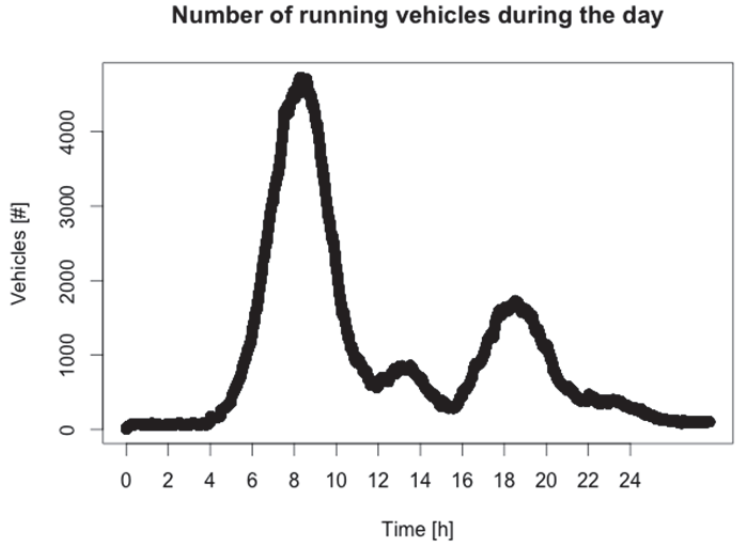


Figure 2-3: Traffic Demand over a day.

In Table 2-4b we present the data processed from the **tripinfo.xml** file. Here we see that the average length of a trip is around 7 km and that the average vehicle speed is around 10 m/s. Figure 2-4 present the distribution of average speed of each trip. In a mid-size city like Luxembourg, the average time required to move from a neighbourhood to the city center is around 10 minutes and this value is respected in the simulation.

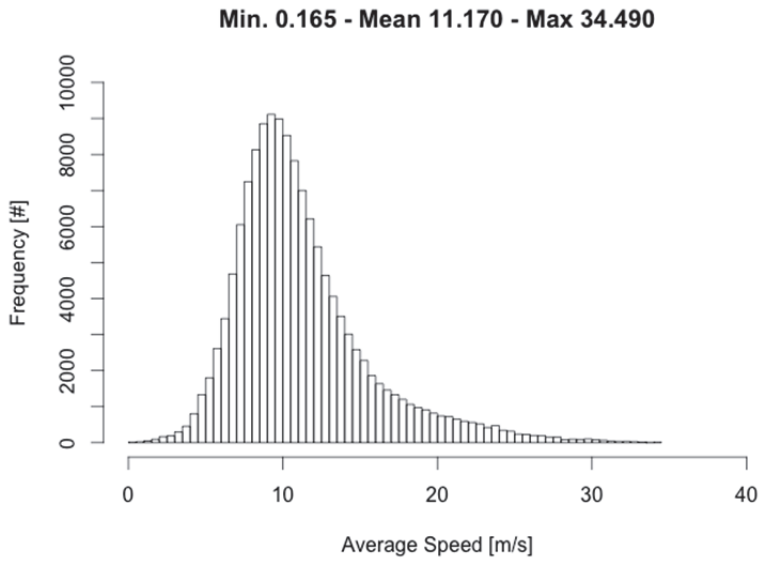


Figure 2-4: Distribution of average speed of the trips.

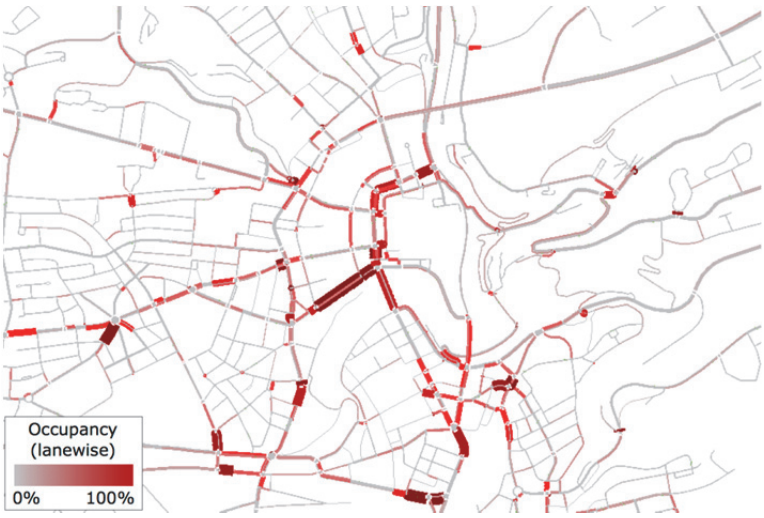


Figure 2-5a: Street occupancy (lanewise) during morning and evening rush hours. Morning rush hour (8:00).

Figure 2-5b: Street occupancy (lanewise) during morning and evening rush hours. Evening rush hour (18:00).

Buildings. In order to use the scenario with other network simulator such as NS3 [8] or OMNet++ [9] it is necessary to have information regarding the shape and position of the buildings. The information is extracted from OpenStreet Map and refined with JOSM to match the modified network topology. Table 2-5 summarize the polygons imported in the scenario. We decided to have only two different categories of polygons for the moment, the parking lots and everything else (ranged from apartments, houses and construction sites). In Figure 2-6 is possible to see the location of the buildings (in red) and the parking lots (in grey).

Table 2-5: Polygons information in the LuST Scenario.

Polygons	
Total	14,173
Buildings	13,555
Parking lots	618

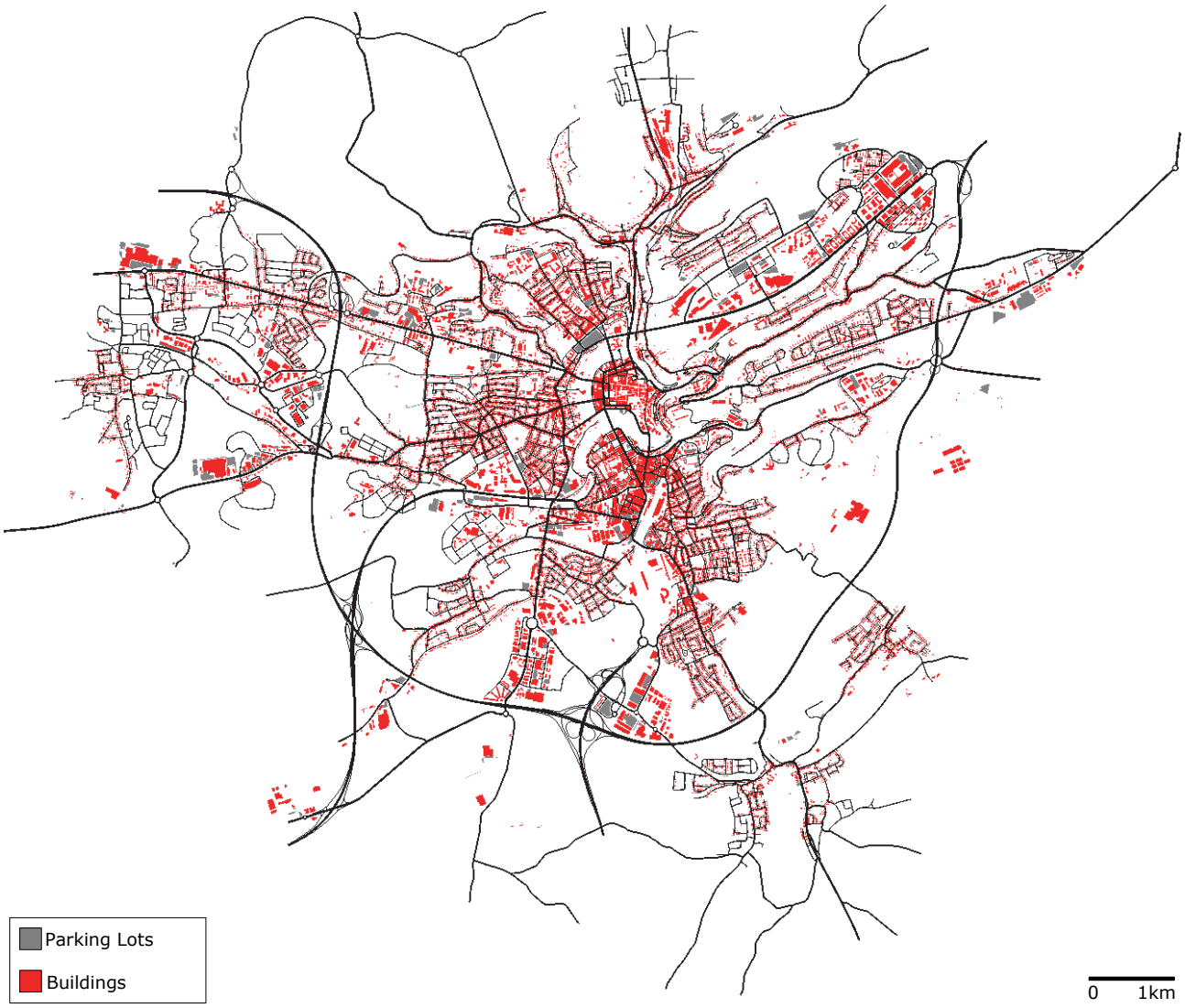


Figure 2-6: Buildings and parking lots information in the LuST Scenario.

2.3 Use Cases

The LuST Scenario is a framework that provides realistic mobility patterns in a mid-size city. The mobility provided by this scenario can be used as an input for other types of simulators such as NS3 or OMNet++ in order to investigate network protocols. When the evaluation of a proposal requires the interaction between a traffic simulator and a connectivity simulator, it is possible to use the LuST Scenario in association with VEINS [10] to obtain a closed- loop feedback between SUMO and OMNet++. Using LuST in combination with those tools allows study of both the performance of VANET protocols and of related applications. Further, it allows evaluation of different multi-modal strategies for commuters. Using the SUMO toolset it is possible to simulate smaller traffic scenarios that only use a subset of the available road network, allowing testing of protocols and applications on different scales. Among the features provided by SUMO there is the possibility of providing an on-board routing system for the vehicles. In our case, we decided to provide to the 70% of the vehicles this routing mechanism, since the commuters are used to checking traffic information and are likely to modify their route when there is a serious congestion ahead. Using this on-board system, it is possible to change the percentage of cars that react to surrounding congestion, obtaining

scenarios with different levels of congestion to test different routing strategies. In our urban area there are 203 intersections managed by the traffic light system. These intersections can be used to test optimisation algorithms for a main arterial road (e.g. green waves) or emergency protocols to allow firefighters, ambulances or the police to be prioritised.

2.4 Conclusion and Future Work

In this paper we have introduced a traffic scenario built for the research community, the Luxembourg SUMO Traffic (LuST) Scenario. This scenario meets all the common requirements needed to have a common basis for the evaluation of various protocols and applications. To build this scenario we started from a real mid-size city and with a typical European road topology and its mobility patterns. The LuST scenario covers an area of 156 km^2 and 932 km of roads. There are 38 different bus routes with 563 bus stops. All intersections with traffic lights and all highway ramps are equipped with inductive loops. We used ACTIVITYGEN to generate the traffic demand using real information provided by various data sources. We have discussed several use cases for the LuST Scenario. Among them are the evaluation and testing of network protocols, and applications for intelligent transport systems.

Future work consists mainly of the maintenance of the scenario with new versions of SUMO and the implementation of additional tools. As new features are provided by the SUMO simulator, the scenario can be enriched with other transportation modes (e.g. pedestrian, bicycle). At the moment the traffic light system uses a static scheduler; among the additional functionality we want to provide, is a dynamic version of the traffic light system.

The scenario is freely available under the MIT licence to the whole community. The scenario is hosted on GitHub (https://github.com/lcodeca/LuSTScenario). Your contribution is highly appreciated!

2.5 Acknowledgements

This AFR project with the project number 5761149 is funded by the Fond Nationale de la Recherche (FNR), Luxembourg.

2.6 References

[1] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO–simulation of urban mobility," International Journal On Advances in Systems and Measurements, vol. 5, no. 3 and 4, pp. 128– 138, 2012.

[2] L. Bieker, D. Krajzewicz, A. Morra, C. Michelacci, and F. Cartolano, "Traffic simulation for all: a real world traffic scenario from the city of Bologna," in SUMO2014 - Modeling Mobility with Open Data, Deutsches Zentrum fur Luft- und Raumfahrt e.V. Institut fur Verkehrssystemtechnik, 2014.

[3] D. Krajzewicz, R. J. Blokpoel, F. Cartolano, P. Cataldi, A. Gonzalez, O. Lazaro, J. Leguay, L. Lin, J. Maneros, and M. Rondinone, "iTETRIS - A System for the Evaluation of Cooperative Traffic Management Solutions," in Advanced Microsystems for Automotive Applications 2010, pp. 399–410, Springer, 2010.

[4] M. Haklay and P. Weber, "OpenStreet Map: User-generated street maps," Pervasive Computing, IEEE, vol. 7, no. 4, pp. 12–18, 2008.

[5] P. Crucitti, V. Latora, and S. Porta, "Centrality measures in spatial networks of urban streets," Physical Review E, vol. 73, no. 3, p. 036125, 2006.

[6] M. Haklay, "How good is volunteered geographical information? A comparative study of OpenStreetMap and Ordnance Survey datasets," Environment and planning. B, Planning & design, vol. 37, no. 4, p. 682, 2010.

[7] P. Koonce, L. Rodegerdts, K. Lee, S. Quayle, S. Beaird, C. Braud, J. Bonneson, P. Tarnoff, and T. Urbanik, "Traffic signal timing manual," tech. rep., 2008.

[8] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in Modeling and Tools for Network Simulation, pp. 15–34, Springer, 2010.

[9] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in Proceedings of the 1st international conference on simulation tools and techniques for communications, networks and systems & workshops, p. 60, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[10] C. Sommer, R. German, and F. Dressler, "Bidirectionally coupled network and road traffic simulation for improved IVC analysis," Mobile Computing, IEEE Transactions on, vol. 10, no. 1, pp. 3–15, 2011.

3 Developing tools for building simulation scenarios for SUMO based on the SCRUM methodology

Andrés F. Acosta[1], Jairo Espinosa[1] and Jorge E. Espinosa[2]

[1]Universidad Nacional de Colombia , Cra 80 No. 65-223, Medellín, Colombia
[2]Politécnico Colombiano Jaime Isaza Cadavid, Cra 48 No. 7-151, Medellín, Colombia

{afacostag, jespinov}@unal.edu.co, jeespinosa@elpoli.edu.co

Abstract

This article describes the development process of tools aimed to simplify the creation of simulation scenarios for the Simulation of Urban Mobility (SUMO) package, based on the SCRUM agile software methodology. The SCRUM methodology allows incremental software development, meaning that the desired functionality of the product can be classified and prioritized so that it can be delivered in several software releases or sprints. The application of this methodology considering the re-engineering of open-source packages that led to TraCI4Matlab, sumolib4matlab and several modules for Network Editor for SUMO is described. Furthermore, the utility of these tools is demonstrated in a small case study in the city of Medellín.

Keywords: Simulation of Urban MObility (SUMO), SCRUM, software re-engineering.

3.1 Introduction

There is an increasing interest on performing realistic traffic simulations for testing and validating new mobility models and traffic control strategies. One of the most popular packages is the SUMO [1] road traffic simulator. The remarkable features of SUMO, openness and reliable outputs, have allowed academics and researchers to achieve important results. Although SUMO has excellent documentation and support system, many users agree that the process for creation and edition of multimodal traffic scenarios can be improved, since it considerably depends on XML files and command line applications such as NETCONVERT [2] or DUAROUTER [3]. The SUMO project includes the NETEDIT [4] graphical network editor, but unfortunately it is currently not an open source tool. The lack of open-source user-friendly interfaces can result on a slow and strenuous learning curve.

Despite several efforts made by the community in order to simplify the creation of such scenarios, which have been gratefully received by the SUMO developer team and have led to spaces such as the SUMO contributed tools and the annual SUMO conference, shortening the time for setting up simulation scenarios is still an issue to be resolved. Furthermore, many of these initiatives are under documented and got into an out-of-maintenance period. Reusing these packages and extending them in order to improve the creation of SUMO simulations can benefit the community, potentially save costs and support the creation of academic knowledge and human capital. For this reason, one of the main goals of the MOYCOT project

[5] is the development of a multimodal traffic simulation tool, based on existing open-source packages, that allows to validate new models and control strategies for traffic.

The selection of the software methodology to be used took into account several characteristics of the team, the stakeholders and the project, including:

- Initially, both simulation and control strategies were being performed in Matlab [6]. Thus, there was know-how on the use of this tool and the added value was on the control component.
- Guarantee of open communication among team members and researchers.
- Involvement of the research group, testers and developers (a small team size) for catching defects and making changes throughout the development process, instead of at the end.
- The requirement to speed up the time spent on evaluations since each evaluation is only on a small part of the whole project.
- The need to ensure changes quicker and throughout the development process by having consistent evaluations to assess the product with the expected outcomes requested.
- There were regular and consistent meetings with the researchers. Additionally, there are systems that allow everyone involved to access the project data and progress.
- Although there were requirements that demonstrated the desire of having into account multimodal scenarios and a rich Graphical User Interface, the tasks required to achieve them were unclear and there were no prior experience on estimating the related effort, including the required for open-source program comprehension and re-engineering.

The above characteristics supported the use of agile methodologies in the MOYCOT project. Agile methodologies have been successfully applied in small teams where requirements are not clear enough and it is not necessary to strictly follow the traditional plan-based project management approach. Hence, agile methodologies reduce the effort spent on planning, thus favoring productivity, as stated by Dyba and Dingsoyr [7]. In the same review, Dyba and Dingsoyr [7] mentioned some limitations of agile methodologies. For example, in one of their reviewed papers, the authors mentioned the "lack of attention to design and architectural issues". This issue was overcome in this project by establishing a pre-process of design and architecture previous to the developing phase. Additionally, Dyba and Dingsoyr [7] highlighted the difficulty of accomplishing the customer´s on-site role. Finally, an important finding of their investigation is that empirical studies suggest that agile methodologies "can be combined with overall traditional principles".

This article presents results and lessons learned on the application of the SCRUM agile software methodology for the development of three tools that support the setup of simulations in SUMO, namely TraCI4Matlab, sumolib4matlab and an extension of the Network Editor for SUMO. Unlike most reported cases in the literature, the influence of reverse engineering and re-engineering of existing open-source packages is taken into account. TraCI4Matlab allowed migrating the simulation component from Matlab without migrating the control component. sumolib4matlab allows reading SUMO networks and generating traffic demands using the Matlab command line. Finally, the extensions made to Network Editor for SUMO allow to import maps and to insert traffic lights to a SUMO

network through a Graphical User Interface made in Qt/C++ [8]. Furthermore, design artifacts and refactoring tasks are presented for each of these tools.

This article is organized as follows: Section 3.2 describes SCRUM methodology. Section 3.3 describes the SCRUM and design artifacts achieved in the development process of TraCI4Matlab, sumolib4matlab and the extensions made to Network Editor for SUMO, based on the re-engineering of existing open-source tools. Section 3.4 demonstrates the usefulness of these tools in a small case study in the city of Medellín. Finally, section 3.5 presents conclusions.

3.2 The SCRUM methodology for software development

SCRUM is an agile software methodology characterized by being simple, lightweighted and being able to work well in projects with high complexity and uncertainty, requiring creativity and adaptability. Moreover, SCRUM allows the involvement of the customer in periodical meetings, called sprint meetings, and potentiates the incremental software development approach, thanks to the prioritization of functional requirements, organized in a list called the Product Backlog, which favors software releases.

Three main roles can be clearly distinguished in the SCRUM methodology:

- The scrum master, who acts similarly to a project manager, coordinates the sprint meetings (which are normally carried out monthly) and daily meetings, called SCRUMS.
- The developer team is a self-organizing and autonomous entity, as explained in [9]:

  > *"No one (not even the Scrum Master) tells the Development Team how to turn Product Backlog into Increments of potentially releasable functionality;"*

- The product owner: A person in charge of providing the requirements and supporting the prioritization of the product backlog. He/She can be a customer representative or any person interested in the software product.

People exerting these roles are in charge of executing the SCRUM main activities, being:

- Defining the Product Backlog, which as stated before, is a prioritized list of tasks required to fulfill the software requirements.
- Defining Sprints: In the sprint meetings, the team selects a subset of tasks of the product backlog to be completed within a Sprint: a software increment to be completed within a time period, which is normally defined as one month, but can also be defined in terms of one or more weeks.
- Carrying out the daily SCRUMS: Short time meetings (15 to 30 minutes) aimed to track the progress of the development process.

3.3 SCRUM in the MOYCOT[5] project

Table 3-1 shows the initial requirements set up in a workshop held in the International Seminar in Transport and Traffic in the city of Medellín-Colombia [10], where parties from the academic and the government sector showed their interest in developing software products to support the assessment and improvement of mobility in the city. These requirements were refined and organized by priority, according to the SCRUM methodology.

Table 3-1: Initial requirements for the MOYCOT simulation software.

Requirement ID	Type	Description	Priority
H1	Integration	Users can import maps from Open Street Maps (OSM) format. Additionally, users can define traffic lights' control programs using Matlab	1
H2	Network design	Users can create graphical elements that comprise the network including road sections, nodes, detectors, Variable Sign Panels and public transport stops	2
H3	Network parametrization	Users can define a control plan for traffic lights, public transportation plans, traffic state, centroids	2
H4	Network design	Users can choose in the network, the following road types: arterial, highway, on and off ramp, pedestrian zone, ringroad, two-lane roadway, roundabouts. Additionally, urban and rural roads can be distinguished and properly signalized. Finally, users can define if a lane is exclusive	3
H5	Traffic demand	Users can define the following types of traffic: car, public transport (bus, BRT, etc.), motorbikes and bycicles. Also, users can generate demand using O/D matrices and probabilistic definitions	4
H6	Other	Users can measure traffic state	4

Thus, the software required in the MOYCOT project belong to the scientific software application domain: a kind of specialized software frequently used in research tasks to simulate complex systems, which typically involves high computational power.

Three important facts to be taken into account in the development of the software were:

- Instead of developing the software package from scratch, it should be based, as far as possible, in existing open-source packages. Additionally, software should be user-friendly, i.e. it should be based in a Graphical User Interface (GUI) with editing capabilities.
- There is a strong knowledge in simulation and control using Matlab and Simulink. Therefore, the highest priority was assigned to the integration of Matlab and the road traffic simulator so that Matlab could act as the Traffic Control Center and the traffic simulator as the plant, in terms of control theory
- It was equally important to be able to import maps from Open Street Maps, since the final aim of the MOYCOT project is to achieve a simulation scenario that represents the city of Medellín. In this sense, Open Street Maps, along with the open-source editor JOSM [11], can offer enough simplicity and openess so that maps can be easily obtained and edited to achieve the desired accuracy

It was found, in a benchmark study, that SUMO satisfied almost all the requirements listed in table 3-1. Additionally, SUMO can be integrated with other platforms through the TraCI API [12], but an implementation for the Matlab programming language was missing. On the other hand, although the SUMO team has developed a graphical network editor, namely NETEDIT, it is currently not an open source tool, then it was necessary to extend SUMO in order to achieve the graphical editing capabilities. In this regard, SUMO includes the sumolib library, which is a python module that allows to parse a SUMO network, storing the

information in an object-oriented hierarchy and including methods for adding nodes, edges, connections and traffic lights. Therefore, the initial idea was to build a GUI on top of sumolib that compliments the editing capabilities of JOSM and preserves familiarity with Matlab. This GUI could be incrementally extended so that the dependency on JOSM is reduced.

Thus, the process of building simulation scenarios for SUMO would follow a sequence shown in figure 3-1. Note that this process is similar to the explained in the SUMO user documentation [13] except for the final step where, instead of editing the SUMO network using the command-line tool NETCONVERT, the proposed graphical editor is used.

Having this in mind, the major tasks to accomplish the requirements took the form showed in table 3-2, where the resulting product´s name is showed in parenthesis. These tasks are related to different SCRUM sprints, which are explained in the following subsections.

Figure 3-1: A general methodology for creating simulation scenarios for SUMO, using information from Open Street Maps (OSM).

Table 3-2: Tasks required to accomplish requirements listed in table 3-1.

Item description	Available in sprint
Develop an implementation of TraCI for the Matlab programming language (TraCI4Matlab)	1
Develop an implementation of sumolib for the Matlab programming language (sumolib4matlab)	2
Build a GUI based on sumolib4matlab to visualize and edit (including creating elements) SUMO networks, and include a tool for importing maps from OSM	3

3.3.1 Sprint 1: TraCI4Matlab

The first sprint was focused on taking advantage of the existing knowledge around the traffic control strategies developed in Matlab so that the simulation component could be performed in SUMO. After identifying that this could be achieved through the TraCI interface, which is

part of the SUMO package, an implementation of TraCI for the Matlab platform was developed, thus giving birth to TraCI4Matlab.

It is worth noting that a considerable effort was assigned to a re-engineering process that included enough comprehension of SUMO at a user level and at a developer level. The successful achievement of the first sprint that led to TraCI4Matlab, has been documented in [14, p. 4].

3.3.2 Sprint 2: sumolib4matlab

Following the re-engineering experience acquired in the first sprint and after identifying the sumolib tool, which is a python library of the SUMO package that allows to read SUMO networks, it was concluded that, continuing the integration of SUMO with Matlab, the second sprint should focus on implementing sumolib in Matlab and performing a suitable re-engineering process to develop a Graphical User Interface on top of it for editing tasks. Because sumolib is based on a SAX parser [15], which has not been comprehensively implemented in Matlab to date, the component was developed in java and added to the Matlab Java path. Additionally, an object-oriented component was developed for the generation of traffic demand that wraps the DUAROUTER application using turning ratios. The demand component accepts a Matlab vector representing the demand in vehicles and distributes it along a user defined interval, thus allowing to define variable demands. Figure 3-2a, shows the sumolib4matlab package UML diagram, note its dependency on the java sax implementation. Figures 3-2b and 3-2c show the UML class diagrams corresponding to the net and the demand sub-packages, respectively.

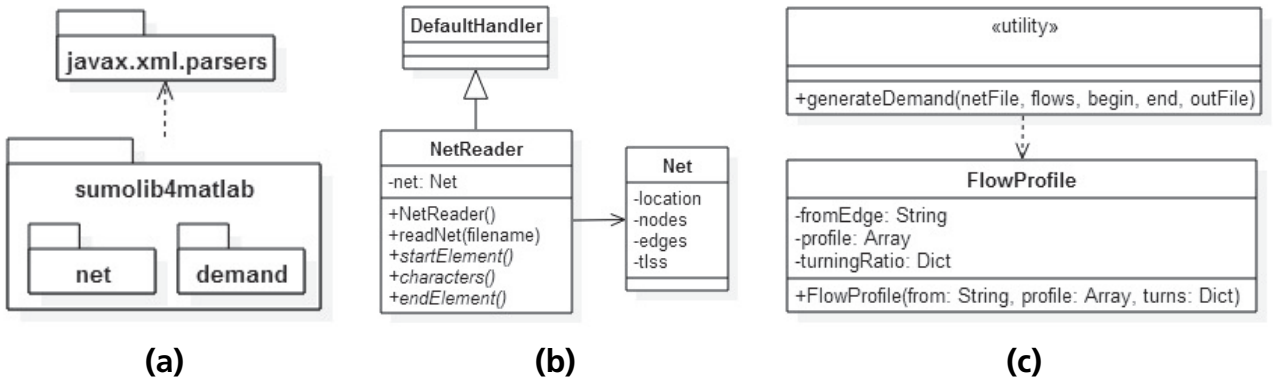


Figure 3-2: Components of sumolib4matlab. (a) Packages, (b) net module class diagram and (c) demand module class diagram.

3.3.3 Sprint 3: Extending network editor for SUMO

As stated previously, the objective was to develop a GUI for sumolib4matlab that allows to visualize SUMO networks and edit objects interactively, featuring suitable dialog boxes and drag-and-drop functionalities. These features are closely related to the Model-View-Controller (MVC) software pattern [16], which consists in separating the way data is represented (Model) from the way it is displayed (View), and providing an interface between them and the user (Controller).

The MVC pattern has gained increasing popularity in the last two decades, both in desktop and web applications. Many software libraries for GUI development (i.e. GUI toolkits) have

adopted the MVC pattern along with widget classes that greatly simplify the development process. However, it was found that Matlab lacks from a native MVC implementation and the development of interactive graphs with drag-and-drop capabilities is quite difficult. Therefore, it was necessary to find alternative approaches to satisfy the requirements of sprint 3. Such approach could be found in Network Editor for SUMO (NES) [17], an open-source effort developed in Qt/C++, released in 2014.

By testing NES, it was found that its main features are:

- **Visualization of SUMO networks.** As showed in figure 3-6(b), NES can display the main SUMO elements: junctions, edges, lanes and connections in a main graph view, taking into account their geometry. These elements can be selected so their properties are displayed in the "Properties View".
- **Some editing capabilities.** Graphical editing capabilities include the modification of junction, edges and lane shapes. One also can edit the element's properties in the "Edit View".

Thus, NES had to be extended to incorporate the creation of new elements and a component for importing maps from OSM, according to sprint 3 (see table 3-2).

Again, a re-engineering process was performed on NES in order to understand its design and implement the new components adequately.

Figure 3-3 shows the extracted class diagram of NES, resulting from a reverse engineering process. Note that the Qt framework provides the MVC pattern, due to the presence of a `Model` class and several views in the `MainWindow` class. In this case, the model has a Document Object Model (DOM) instance that represents the XML definition of the SUMO network, where XML elements are, in turn, instances of a node class. Furthermore, NES follows the typical way of a MVC implementation in Qt using an item-based model class, which can be used to represent hierarchical data structures, and the corresponding Tree View.

The Model class, which inherits from Qt's `QAbstractItemModel`, requires the implementation of an Item class, which in this case aims to represent a Node instance, including the `XMLNode` and `XMLSubnode` attributes for its location in the DOM. Additionally, its important to note that the NES model implements its own methods to build the item hierarchy, being `loadModel()`, `loadJunctions()`, `loadEdgesAndLanes()`, `loadConnections()` and `loadSignals()`. This approach differs from the one described in the Qt documentation [18], where the item hierarchy is built by taking advantage of the Model's `index()` (which is called for each item) and `rowCount()` methods.

Figure 3-4 shows the class diagram resulting from a forward engineering process performed on NES, taking into account the requirements of sprint 3. It can be seen that the requirement related to the creation of SUMO elements (junctions, edges and so on) was initially addressed through the creation of traffic lights. This functionality had the highest priority in the MOYCOT project. Thus, a wizard and a dialog box were developed for importing maps from OSM and building traffic lights, respectively. Basically, both components take advantage of the NETCONVERT command-line application, which is part of the SUMO package, by calling it through the `netconvertProcess` attribute. `netconvertProcess` is an instance of the `QProcess` class, and is executed whenever the `accept()` method is called (i.e. when the user clicks the 'Accept' button).

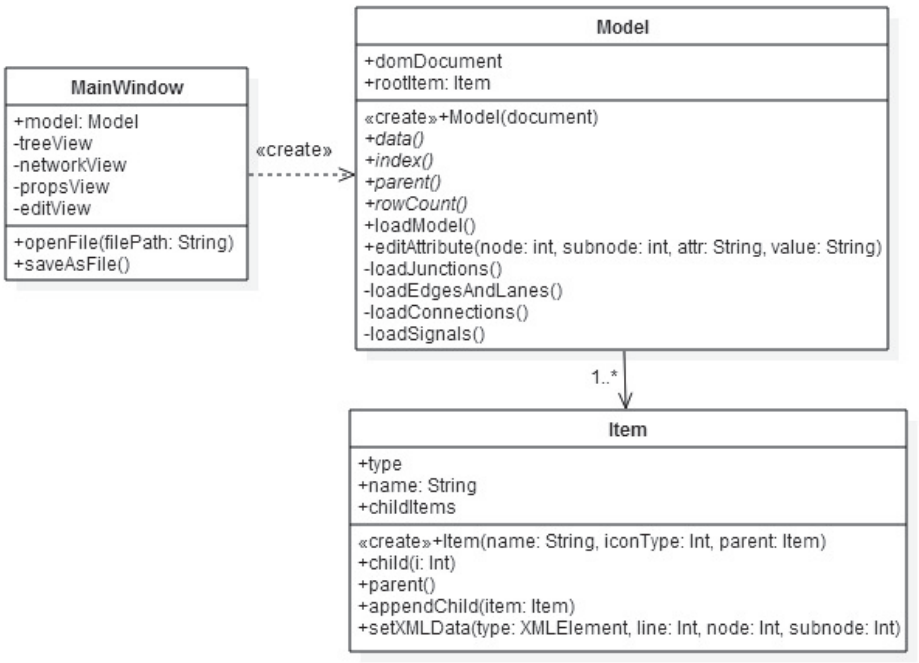


Figure 3-3: Network Editor for SUMO (NES) class diagram obtained from a reverse engineering process. Methods inherited from Qt's `QAbstractItemModel` interface are showed in italics.

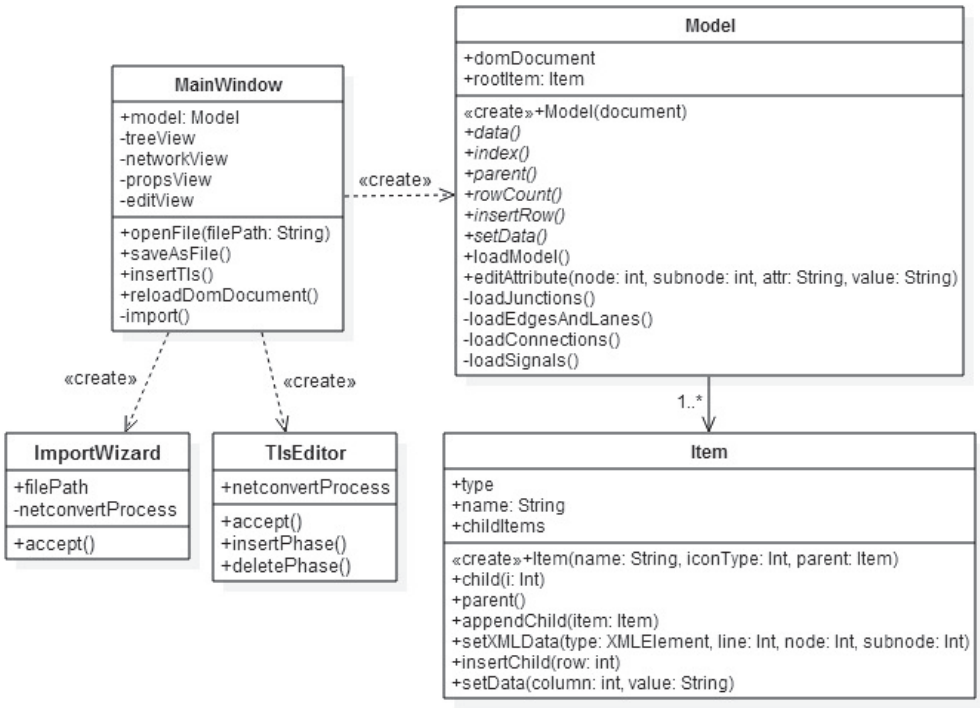


Figure 3-4: Class diagram showing the extensions made for NES through a forward engineering process. Methods inherited from Qt's `QAbstractItemModel` interface are showed in italics.

In the Model class, the methods `insertRow()` and `setData()` were implemented in order to refresh the tree view after inserting or editing traffic lights. These methods rely on the implementation of the `insertChild` and `setData` methods in the Item class, correspondingly.

The following section demonstrates the capabilities of TraCI4Matlab, sumolib4matlab and the extensions made to Network Editor for SUMO in a small case study.

3.4 Results

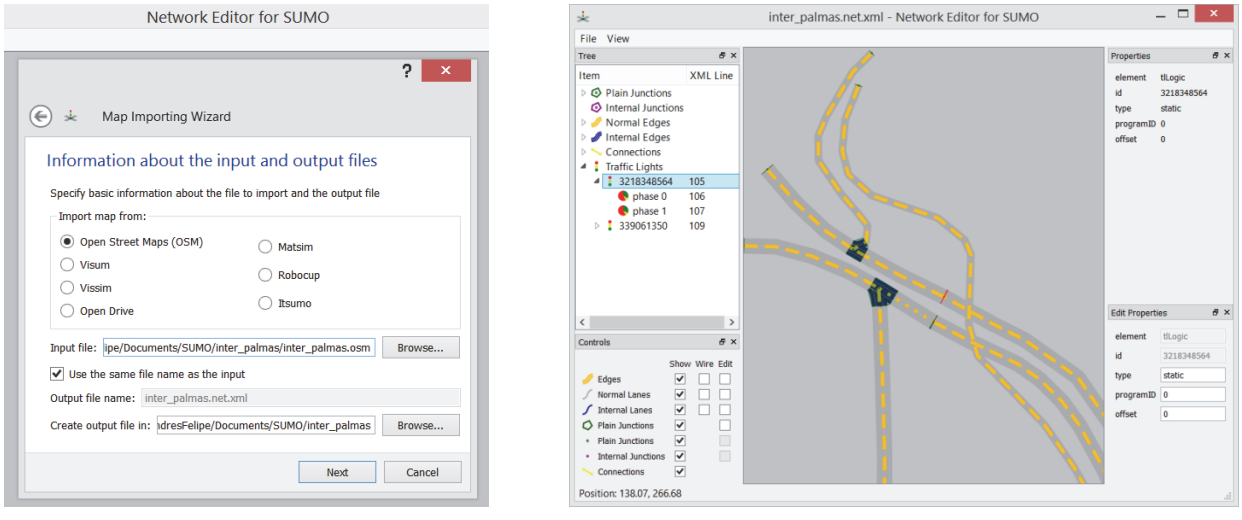
Following the methodology proposed in figure 3-1, TraCI4Matlab, the demand generation component of sumolib4matlab and the modules developed for NES (Importing Wizard and Traffic Lights Editor) were tested in the scenario showed in figure 3-5(a), which corresponds to the first kilometer of Via las Palmas: an important road in Medellín-Colombia that connects this city with the José María Córdova's airport. In this section of the road, two traffic lights have been built in the last year to allow the crossing of pedestrians.

(a) **(b)**

Figure 3-5: Vía las Palmas, 1st kilometer. (a) In Open Street Maps, (b) In the JOSM editor.

In this case, the area of interest was extracted directly from the Open Street Maps site, then the number of lanes was verified against Google Street View through the JOSM editor, as showed in figure 3-5(b).

The next step was to use NES and the Importing Wizard described in section 3.3.3, to convert the map from OSM to the SUMO format (.net.xml), as showed in figure 3-6(a). The resulting SUMO network can be seen in figure 3-6(b). Note the successful importing of traffic lights, thanks to NETCONVERT. Figure 3-7(a) shows one of the traffic lights program automatically generated in the importing process, using SUMO GUI. This program was modified using the Traffic Lights Editor developed for NES, as showed in figure 3-8. The new traffic lights program was verified, again, in SUMO GUI, as showed in figure 3-7(b).



(a) **(b)**

Figure 3-6: Importing a map from OSM to SUMO in Network Editor for SUMO. (a) Importing Wizard, (b) Resulting SUMO network.

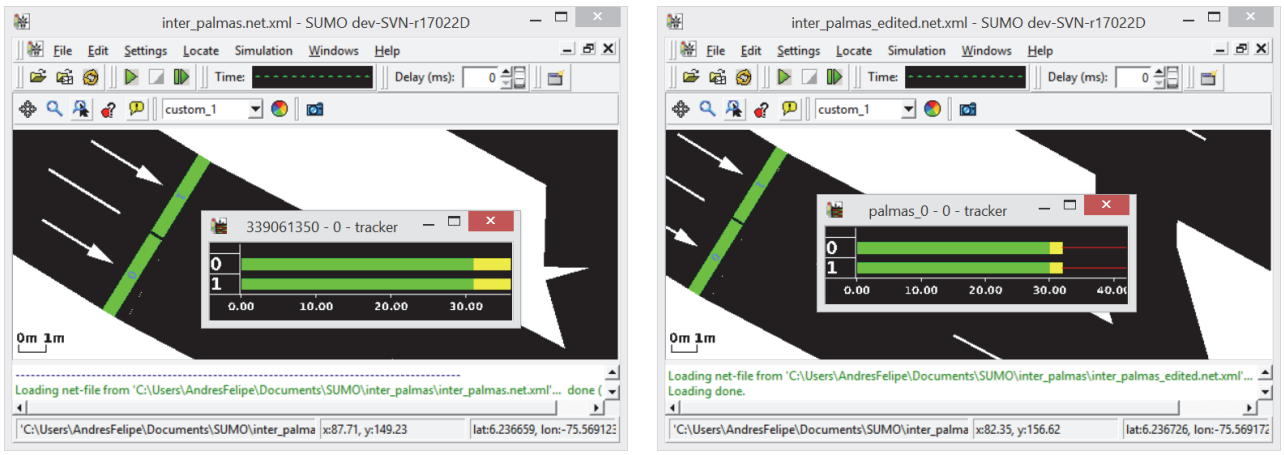


Figure 3-7: Verifying one of the traffic lights program in Las Palmas scenario. (a) Traffic Lights program generated automatically during the importing process. (b) Modified traffic lights program, using the Traffic Lights Editor implemented for Network Editor for SUMO.

Finally, some traffic demand was generated for Las Palmas scenario, using the demand package of sumolib4matlab, then the simulation was set up in order to test TraCI4Matlab. Particularly, the traci.edge.getCO2Emission was used for obtaininig and plotting the generated CO2 emissions in Las Palmas road in both directions. Figures 3-9 and 3-10 show a screenshot of the Las Palmas simulation and the obtained CO2 emissions, respectively.

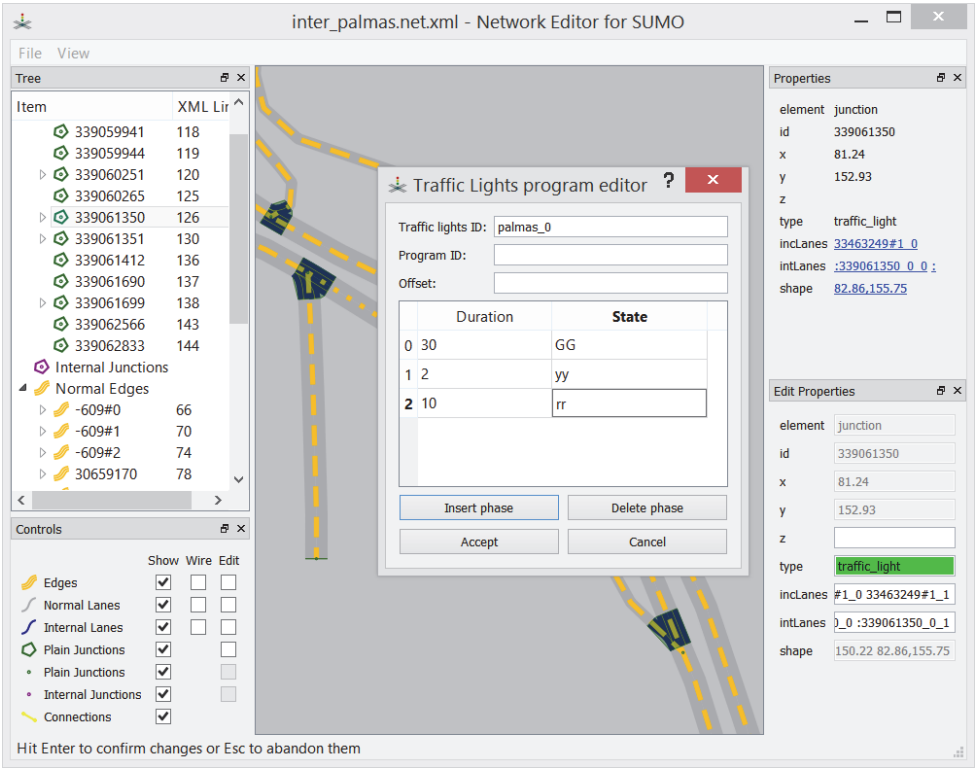


Figure 3-8: Modifying a traffic lights program using the Traffic Lights Editor developed for Network Editor for SUMO.

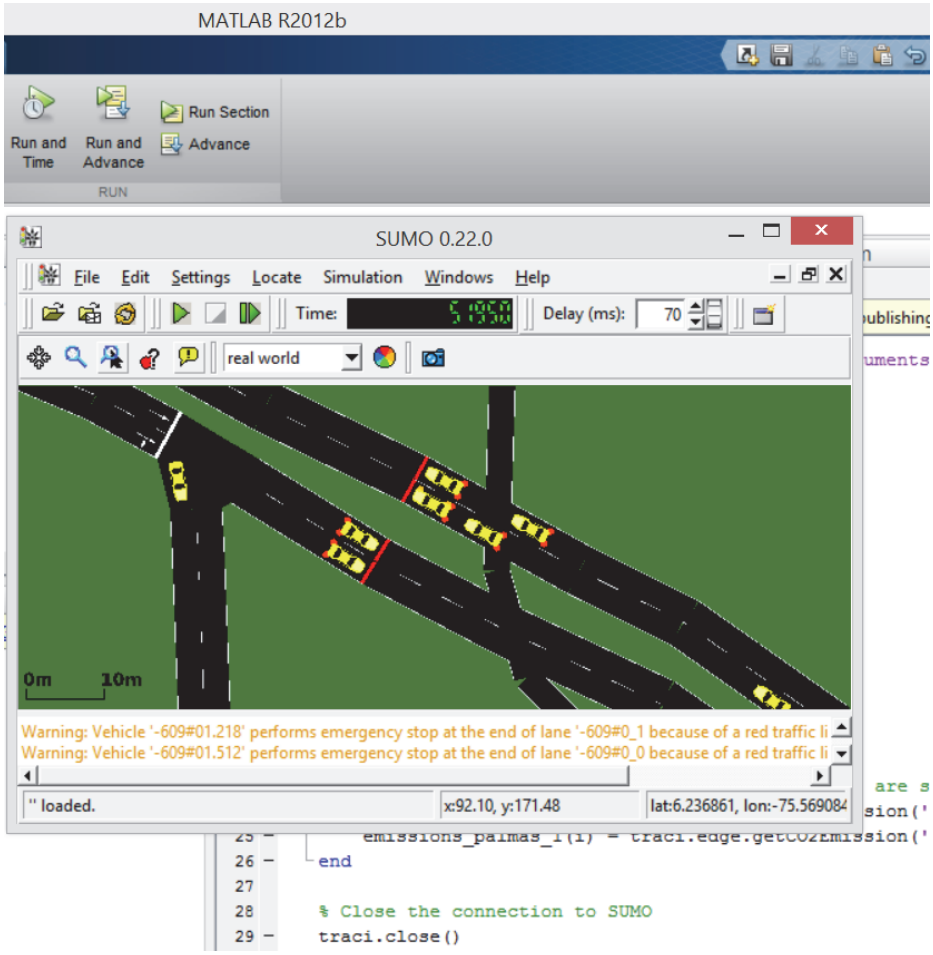


Figure 3-9: Screenshot of the Las Palmas simulation scenario.

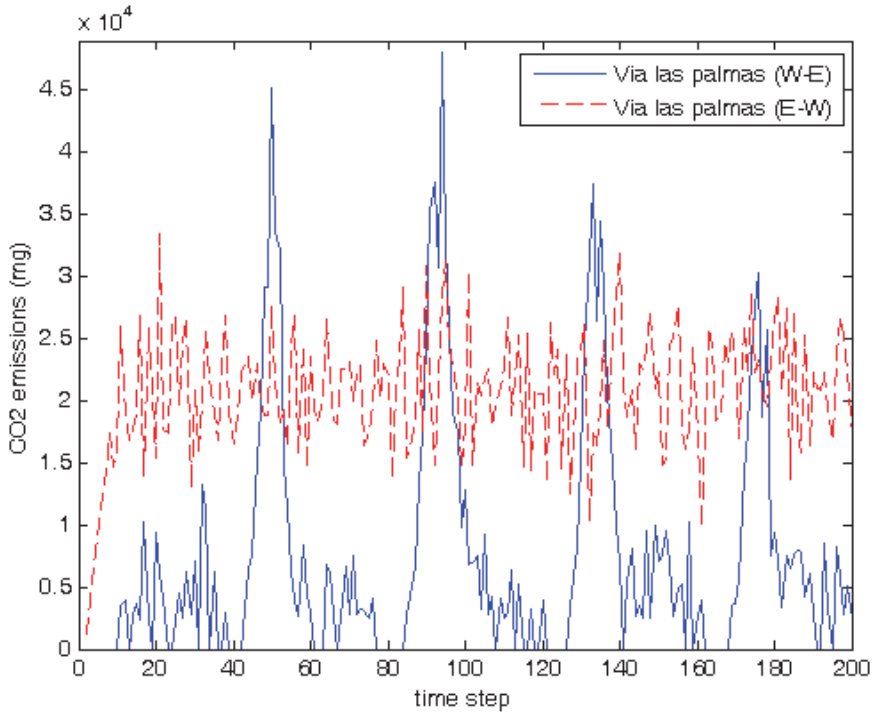


Figure 3-10: CO2 Emissions obtained for Las Palmas road in both directions using TraCI4Matlab.

3.5 Conclusions

This article presented the development process of tools aimed to simplify the creation of simulation scenarios in SUMO, based on the SCRUM agile software methodology and the re-engineering of existing open-source packages. The effort required to accomplish the requirements stated in the MOYCOT project was divided in three sprints, resulting in three software products: TraCI4Matlab, sumolib4matlab and two modules for Network Editor for SUMO that allow importing maps from OSM and editing traffic lights programs. These tools were incorporated in a methodology for setting up simulation scenarios in SUMO that includes open tools such as those offered by the Open Street Maps project and the JOSM editor, so that their functionalities are complementary. Finally, the proper functionality of these software products was demonstrated in a small scenario in the city of Medellín.

For future work, Network Editor for SUMO (NES) could be extended further to allow the creation and edition of any SUMO object. Additionally, the graphical demand generator should be implemented in NES and improved to define multimodal traffic and incorporate the different strategies for demand generation allowed by SUMO such as O/D matrices and Traffic Assignment Zones.

The development of MOYCOT project plans to deploy the entire solution in a parallel execution. The proposal is to allow multiple chained simulations, dealing with complex and high concurrent scenarios that involves an entire traffic congestion in the metropolitan area.

3.6 Acknowledgements

This work was supported by Proyecto Colciencias 111856934640 contrato 941-2012 y – FP44842-202-2015: Modelamiento y Control de tráfico urbano en la ciudad de Medellín Fase 1 y Fase 2. Convocatoria 569 y 669.

3.7 References

[1] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent Development and Applications of SUMO - Simulation of Urban MObility," Int. J. Adv. Syst. Meas., vol. 5, no. 3&4, pp. 128–138, Dec. 2012.

[2] "NETCONVERT - Sumo." [Online]. Available: http://sumo.dlr.de/wiki/NETCONVERT. [Accessed: 27-Mar-2015].

[3] "DUAROUTER - Sumo." [Online]. Available: http://sumo.dlr.de/wiki/DUAROUTER. [Accessed: 27-Mar-2015].

[4] "NETEDIT - Sumo." [Online]. Available: http://sumo.dlr.de/wiki/NETEDIT. [Accessed: 27-Mar-2015].

[5] "MOYCOT," MOYCOT. [Online]. Available: http://www.moycot.org/. [Accessed: 27-Mar-2015].

[6] "MATLAB - The Language of Technical Computing." [Online]. Available: http://www.mathworks.com/products/matlab/. [Accessed: 31-Mar-2015].

[7] T. Dyba and T. Dingsoyr, "What Do We Know about Agile Software Development?," IEEE Softw., vol. 26, no. 5, pp. 6–9, Sep. 2009.

[8] "Qt | Cross-platform application & UI development framework," Qt. [Online]. Available: http://www.qt.io/. [Accessed: 27-Mar-2015].

[9] K. Schwaber and J. Sutherland, "The scrum guide," Scrum Alliance, 2011.

[10] "SEMINARIO INTERNACIONAL DE TRÁFICO Y TRANSPORTE." [Online]. Available: https://sites.google.com/site/moycot1/home. [Accessed: 03-Feb-2015].

[11] "JOSM." [Online]. Available: https://josm.openstreetmap.de/. [Accessed: 27-Mar-2015].

[12] "TraCI - Sumo." [Online]. Available: http://sumo.dlr.de/wiki/TraCI. [Accessed: 27-Mar-2015].

[13] "Tutorials/Import from OpenStreetMap - Sumo." [Online]. Available: http://sumo.dlr.de/wiki/Tutorials/Import_from_OpenStreetMap. [Accessed: 31-Mar-2015].

[14] A. F. Acosta, J. E. Espinosa, and J. Espinosa, "TraCI4Matlab: Enabling the Integration of the SUMO Road Traffic Simulator and Matlab® Through a Software Re-engineering Process," in Modeling Mobility with Open Data, M. Behrisch and M. Weber, Eds. Cham: Springer International Publishing, 2015, pp. 155–170.

[15] D. Megginson and others, "Sax 2.0: The simple api for xml," SAX Proj., 2001.

[16] G. E. Krasner, S. T. Pope, and others, "A description of the model-view-controller user interface paradigm in the smalltalk-80 system," J. Object Oriented Program., vol. 1, no. 3, pp. 26–49, 1988.

[17] "editor4sumo," SourceForge. [Online]. Available: http://sourceforge.net/projects/editor4sumo/. [Accessed: 28-Aug-2014].

[18] "Simple DOM Model Example | Qt Widgets 5.4." [Online]. Available: http://doc.qt.io/qt-5/qtwidgets-itemviews-simpledommodel-example.html. [Accessed: 26-Mar-2015].

4 Multi-Resolution Traffic Simulation for Large Scale High Fidelity Evaluation of VANET Applications

Manuel Schiller[1], Marius Dupius[2], Daniel Krajzewicz[3], Andreas Kern[4] and Alois Knoll[1];

[1]Lehrstuhl für Echtzeitsysteme und Robotik, Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany
[2]VIRES Simulationstechnologie GmbH, Grassingerstraße 8, 83043 Bad Aibling, Germany
[3]Institute of Transportation Systems, German Aerospace Center, Lilienthalplatz 7, 38108 Braunschweig, Germany
[4]AUDI AG, Ettinger Straße, 85057 Ingolstadt, Germany

manuel.schiller@in.tum.de

Abstract

This paper presents an approach for coupling traffic simulators of different resolutions in order to conduct both large scale and high fidelity virtual evaluations of Advanced Driver Assistance Systems based on Vehicular Adhoc Networks. The emphasis is put on the need for such an attempt to satisfy the constraint of performing simulations in real time. Both, the methods to accomplish this as well as the resulting performance are described.

Keywords: V2V Communication, ADAS, Multi-Resolution Simulation

4.1 Introduction

Vehicular Adhoc Networks (VANETs) have attracted a lot of research attention over the last years due to the potential improvements in traffic safety, efficiency and driver comfort. A high variety of applications, commonly referred to as Advanced Driver Assistance Systems (ADAS), such as cooperative driving and subsequently automated driving, can only be enabled through wireless communication between the vehicles on the road.

Before deployment of such systems, which often exhibit safety-critical features, in series production on a large scale a lot of effort has to be put into testing and validation. Although real test drives using physical testbeds of prototype vehicles offer the highest degree of realism, the large amount of financial, material and human resources needed to perform large-scale and extensive testing of vehicular networks render their use rather impossible. Due to this, simulations are employed for obtaining a view of the performance of such solutions in large-scale virtual environments. As well, simulation-based evaluation techniques particularly allow testing of those complex systems in a wide variety of dangerous and critical scenarios without putting humans and material at risk while at the same time being less resource-intensive.

In the automotive industry the use of simulation is well established in the development process of traditional driver assistance and active safety systems. However, the current emphasis is primarily on the simulation of individual vehicles at a very high level of detail [1].

When investigating and evaluating the performance of ADAS based on vehicular communication, this isolated view of a single vehicle alone or a small number of vehicles in the simulation is not sufficient anymore. Potentially every vehicle equipped with wireless communication technology is coupled in a feedback loop with the other road users participating in the vehicular network and therefore the number of influencers which need to be taken into account is drastically increased.

These considerations lead to a trade-off between the accuracy in terms of the simulated level of detail of each vehicle and the scalability in terms of the number of vehicles that can be simulated with the computing resources available. In this paper we present an approach on how to solve this trade-off by coupling multiple resolutions of traffic simulations to get highly accurate simulation results where it is necessary and simultaneously achieving an efficient simulation of large scale scenarios.

The rest of this paper is organized as follows. The testing and evaluation of real-world implementations of ADAS imposes a certain set of additional requirements, which are discussed in section 4.2 before giving an overview of the related work. Section 4.3 describes the concept of our multi-resolution traffic simulation approach. In section 4.4 we evaluate the performance by means of an exemplary scenario. In section 4.5 we discuss the limitations of the approach and conclude the paper in section 4.6.

4.2 Background and Related Work

Before we proceed to the discussion of related work, it is essential to illustrate our scope and area of application. In order to evaluate and validate real implementations of ADAS in a simulated, virtual environment, both state and behavior of the vehicles must be modeled and simulated in high fidelity. We define the term *high fidelity* as a three-dimensional problem:

To substitute the real vehicle by its simulated counterpart, the virtual vehicle must provide its state variables in a *sufficient range*, in *sufficient precision* and in a *sufficient temporal resolution*. The concrete manifestations of these three requirements depend on the respective use case. For example a Cooperative Adaptive Cruise Control (CACC) system will need, among others, the dynamic state of the vehicle (e.g. speed, acceleration, position), powertrain state, RADAR sensor values as well as the input from the wireless communication channel at different sampling rates [2]. If any of these requirements is not met, e.g. a necessary state variable is not covered by the simulation model, simulative testing can not be performed.

Since the range of wireless communication technology is higher than what can be achieved through conventional sensors, even vehicles which are farther away can act as relevant information sinks and sources and therefore influence the driving assistance system as well as the road traffic system as a whole. In order to capture these effects in the simulation, a naive approach would be to simply scale existing, high detailed simulations by increasing the amount of vehicles in the simulated area. However, these high-detail simulations are extremely computationally intensive and hence are not suitable to perform evaluations of large-scale scenarios in a reasonable amount of time. This also prohibits their use in hardware-in-the-loop simulations where a real time constraint must be fulfilled [1].

This additional timing requirement conflicts with the aforementioned three dimensions of simulative high fidelity. There are numerous references that deal with similar issues in the

fields of traffic simulation, VANET simulation and multi-resolution simulation. In the following we give a brief overview of those research areas in order to state the fundamentals of this investigation:

4.2.1 Traffic Simulation

Road traffic simulations can generally be subdivided into the following four categories according to the level of detail [3,4]: Macroscopic, mesoscopic, microscopic and nanoscopic. While macroscopic flow models describe traffic at a high level as aggregate flows, microscopic simulations model the behavior and interactions of each simulated entity individually with specific state variables such as position, speed and acceleration. Mesoscopic models are medium-detailed models where traffic is usually represented by queues of vehicles. In nanoscopic models, which are also referred to as submicroscopic models, an even higher level of detail is achieved through the subdivision of each vehicle in multiple subunits. This allows to model for example the vehicle dynamics, complex decision processes of the driver or the interaction with the vehicle surroundings more accurately. The necessary amount of computation time for the traffic simulation rises considerably with the increasing degree of detail.

4.2.2 VANET Simulation

VANET SimulationThe usual strategy to simulate VANETs found in literature is to bidirectionally couple a network simulator and a microscopic traffic simulation. Following this approach the interactions between road traffic and network protocols are represented and the mutual impact can be explored [5,6]. The majority of research publications focuses on the investigation of low level networking subjects such as medium access [7] or rather high level concepts of applications such as reducing CO_2-emissions [8]. For this kind of studies it is sufficient to apply realistic mobility patterns originating from the microscopic traffic simulation. In this bird's eye view of the overall system it is not necessary to model individual cars in the high level of detail mentioned above because the accuracy of the vehicle representation has a negligible influence on the simulation results of the vehicular network. Therefore the three requirements of high fidelity in terms of modeling and simulation individual vehicles need not be considered when simulating VANETs on such an abstract level.

4.2.3 Multi-Resolution Simulation

Multi-Resolution Modeling (MRM) is defined as the combination of different models of the same phenomenon at different levels of resolution which are then executed together [9]. This methodology allows to find a good balance between simulation accuracy and computing resources. High resolution models, which provide accurate simulation results at the cost of high computational efforts, are only applied in limited areas of interest whereas the major part of the simulation is handled by less acurate but also less resource consuming low resolution models. However, not only the difference in execution speed can be exploited but also the fact that low resolution models tend to give a better overall understanding of the system under examination because of their rather abstract view of the big picture.

MRM has been successfully applied in road traffic simulation. In [10] a combination of a microscopic simulation modeling the inter-vehicle interactions and a computationally less

expensive macroscopic simulation applied to freeways is described. This allows a scalable, yet accurate investigation of traffic flow in large scale networks. This approach is not applicable for VANET simulations since due to the flow-based simulation in macroscopic models accurate vehicle positions are missing which is crucial when simulating vehicular networks. In [11] a coupling of two different microscopic traffic simulators of different accuracy is implemented for VANET simulation to exploit the difference in execution speed. In [11] and in [12] the areas of interest are fixed throughout the simulation, for example road intersections are simulated at a higher level of detail than freeways. In contrast to that, the areas of interest are not statically fixed in [13] but rather depending on the simulation context.

4.2.4 Combining Traffic and Driving Simulation

A rather recent research direction is the combined simulation of both traffic and driving simulation. A co-simulation of a microscopic traffic simulation and a driving simulator [14] respectively a robotics simulator [15] has been investigated. In these approaches the behavior of a fixed subset of all vehicles is remote controlled through an external simulator, which allows to have a predefined number of detailedly simulated vehicles to be surrounded by a large number of microscopically simulated vehicles.

4.3 Novel approach for Multi-Resolution Traffic Simulation

4.3.1 Dynamic Spatial Partitioning of Simulated Area

Our approach aims to couple traffic simulation models of different resolutions at dynamic regions of interest. Contrary to conventional traffic simulation we are not interested in investigating a large number of vehicles from a bird's perspective but the focus is rather on a single vehicle or a limited number of vehicles which are used to conduct a test drive in a virtual environment. In the following we will refer to this kind of vehicle as the EGO car. The ADAS under investigation is imagined to be on board of such an EGO car. The simulated measurements and sensor values are fed into the ADAS. Depending on its type and its use case, the respective ADAS directly or indirectly influences the vehicle's state and behavior.

As stated before, all surrounding vehicles both near and far away from the EGO car need to be taken into consideration because of the wide range of transmission on the wireless communication channel. However, we can distinguish between highly relevant and less relevant vehicles. This distinction is based on the criterion of the respective distance between the vehicles to the EGO car. Nearby vehicles are inherently of more relevance because they pose a higher danger in terms of possible collisions and because their messages transmitted on the vehicular network are of higher importance due to the vicinity of their origin.

Based on this distance criterion an area of interest is defined which is centered around the EGO car and in which the defined simulative high fidelity requirements must be fulfilled. Since the EGO car is driving continuously through the virtual environment, this area of interest is being moved along likewise. We therefore partition the global area of the simulation dynamically into a High Resolution Area (HRA) and a Low Resolution Area (LRA). Figure 4-1 shows a schematic view of the dynamic spatial partitoning. There the HRA is defined as the area of a circle which is centered around the EGO vehicle. Red vehicles are within that circle and are therefore simulated in high resolution by the nanoscopic simulator, whereas the

green vehicles are outside of the circle and are consequently simulated in low resolution by the microscopic simulation. All vehicles exist in the microscopic simulation but in the nanoscopic simulation only the high resolution vehicles are contained and their movements are applied to their proxy counterparts in the microscopic simulator.

Due to the dynamic nature of road traffic the EGO car, the high resolution vehicles as well as the low resolution vehicles are permitted to move continuously. The classification of the assigned resolution mode is therefore performed after each time step of the simulation. Vehicles for which the classification has led to a change in resolution are transferred to the appropriate simulator. This change of resolution is possible in both directions at every time step. However, since the HRA is defined to be centered around the EGO car, it is always simulated in high resolution.

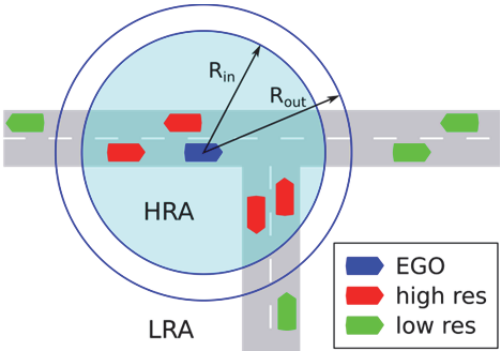


Figure 4-1: Dynamic Partitioning of Simulated Area

In order to prevent vehicles which are close to the boundary between HRA and LRA from oscillating very frequently between the two resolution areas, a hysteresis controller as depicted in figure 4-2 is applied in the classification process. As shown in figure 4-1 the two thresholds R_{in} and R_{out} are defined. A vehicle is transferred into the high resolution simulation only if its distance to the EGO car falls below the value of R_{in}. The exchange back to the low resolution simulation is carried not out until the threshold R_{out} is exceeded.

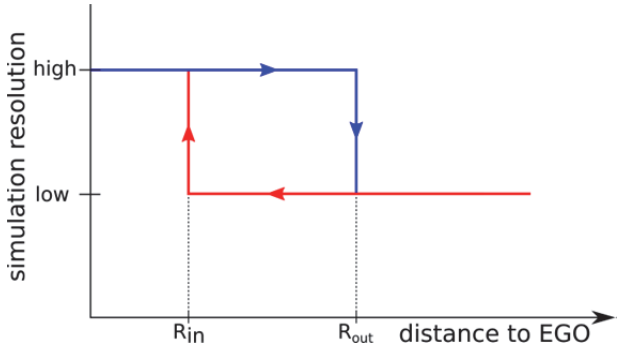


Figure 4-2: Hysteresis control of the simulation resolution

The difference in simulation resolution switching is shown in figure 4-3 by an exemplary trajectory. Without the hysteresis the change in resolution is performed multiple times, whereas when applying the hysteresis controller the vehicle stays in the high resolution model while being close to the boundary.

The extent of R_{in} defines the circumcircle in which vehicles are simulated by the nanoscopic simulator. This value needs to be determined separately for each application scenario, for

example in an urban scenario due to the expected low traffic speed a lower value can be used than what is suitable for a freeway scenario. As an alternative to the definition of a fixed size the value of R_{in} can be determined dynamically at simulation runtime, for example based on traffic speed, volume of traffic or even depending on the currently available computing capacities.

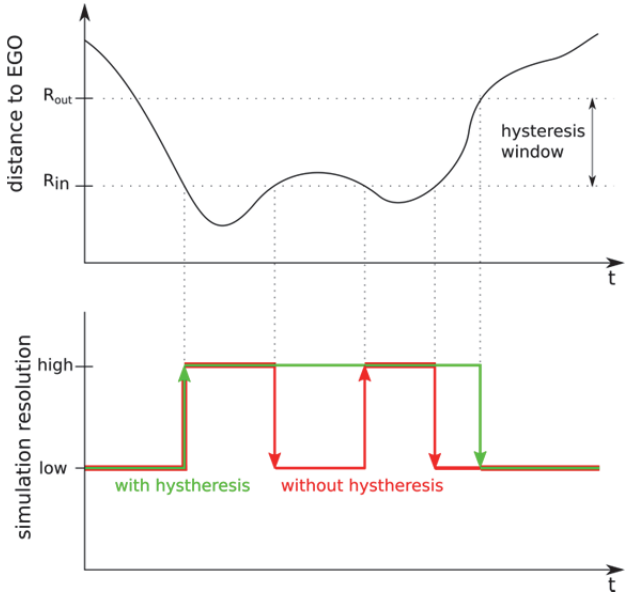


Figure 4-3: Comparison of simulation resolution switching

The gap $R_{out} - R_{in}$ defines the size of the hysteresis window for the dynamic change between the two simulators. R_{out} can be selected in both absolute and relative terms in relation to R_{in} and exhibits a lower dependence with regard to specific scenarios and traffic speeds.

Our approach of dynamic spatial partitioning of the simulated area enables us to allocate the processing resources between the simulators. As the focus is only on a relatively small region of the simulated area we achieve both a simulative high fidelity in this region of interest and the simulation of large scale scenarios.

4.3.2 Utilized Simulators

In the following we describe the concrete manifestations of the simulators which we combine using the above described concept to achieve a multi-resolution traffic simulation.

4.3.2.1 Microscopic Traffic Simulator - SUMO

We chose to use Simulation of Urban MObility (SUMO) as the traffic simulator responsible for the simulation of the LRA. SUMO is a microscopic, space-continuous and time-discrete simulator. While it is employed in a wide range of research domains, its most notable use is shown in a high number of research papers regarding VANET simulations [16]. SUMO is well known for its high execution speed as well - being open source - its extensibility. In [17] it is reported that it can handle 200,000 vehicles in real-time when using timesteps of 1 second. Due to its efficiency, which is partly achieved through its simplified driver model [18], SUMO is ideally suited to simulate a high number of vehicles residing in the LRA.

4.3.2.2 Nanoscopic Traffic And Vehicle Simulator - VIRES Virtual Test Drive

We employ the nanoscopic traffic and vehicle simulator VTD for the simulation of the high resolution vehicles. VIRES Virtual Test Drive (VTD) has been developed for the automotive industry as a virtual test environment used for the development of ADAS [19]. Its focus lies on interactive high-realism simulation of driver behavior, vehicle dynamics and sensors. VTD is highly modular, so any standard component may be exchanged by a custom and potentially more detailed implementation. Its standard driver model is based on the intelligent driver model [20], however an external driver model may be applied if necessary. The same concept applies to the vehicle dynamics simulation, where the standard single-track model can be substituted by an arbitrarily complex vehicle dynamics model adapted for specific vehicles. Each simulated vehicle can be equipped with arbitrary simulated sensors, for example a RADAR sensor, which is shown in figure 4-4.

While VTD is designed for online operation, it is however not suited to simulate a large number (i.e. thousands) of vehicles with respect to real time due to the details and complexity of the simulation. Therefore only the EGO car as well as the vehicles residing in the HRA are simulated by VTD.

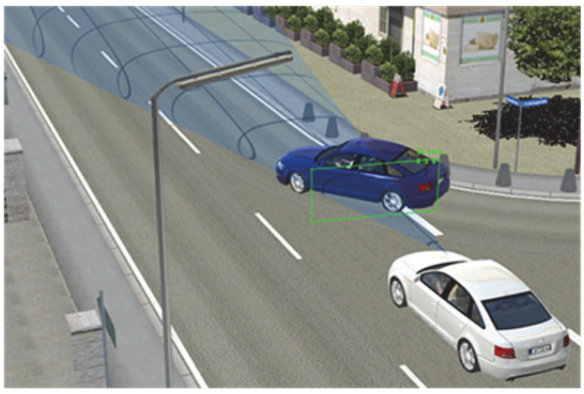


Figure 4-4: 3D visualization of a simulated RADAR sensor in VTD

4.3.3 Coupling concept

4.3.3.1 offline preprocessing

Both simulators rely on different data formats representing the modeled road network. In order to be able to run a co-simulation of both simulators the underlying data basis has to match. VTD uses the OpenDRIVE format to specify the road network. This specification models the road geometry as realistically as possible by using analytical definitions. SUMO on the other hand approximates the road network by line segments. There are additional differences in the modeling of intersections and lane geometries. To achieve a matching database we convert the road network in an offline preprocessing step from OpenDRIVE to the file format SUMO supports.

4.3.3.2 online coupling and synchronization

The coupling of the simulators at simulation runtime is based on the master-slave-principle. Figure 4-5 shows this sequence of operations during a single simulation step, in which VTD und SUMO can operate with different temporal resolutions. s_{VTD} is the length of a time step for the HRA, whereas s_{SUMO} is the length of a time step for the LRA. Typically, the nanoscopic simulation is run at a higher frequency than the microscopic simulation. t_{VTD} respectively t_{SUMO} denote the local simulation time in each simulator. At the beginning of each simulation step a new time step is simulated in VTD. If the next time step has been reached for SUMO and therefore the condition $t_{VTD} \geq t_{SUMO} + s_{SUMO}$ is fulfilled, the state of the high resolution vehicles is sent to SUMO through a gateway. It then triggers the simulation of the next timestep in the low resolution model and as a result the positions of the low resolutions vehicles are passed back. These vehicles are now classified according to section 4.3.1 and, if applicable, the change of resolution is performed for individual vehicles. When an exchange of a vehicle between the simulators happens, the previously mentioned inherent difference in the underlying road network may cause problems if a vehicle can not be mapped based on its position on a specific lane due to difference in accuracy. This is especially true for complex intersections which are modeled quite differently.

After all resolution changes have successfully been completed the simulation is unblocked again and the next time step can be simulated. This synchronization is very important to ensure reproducible simulation results across multiple simulation runs.

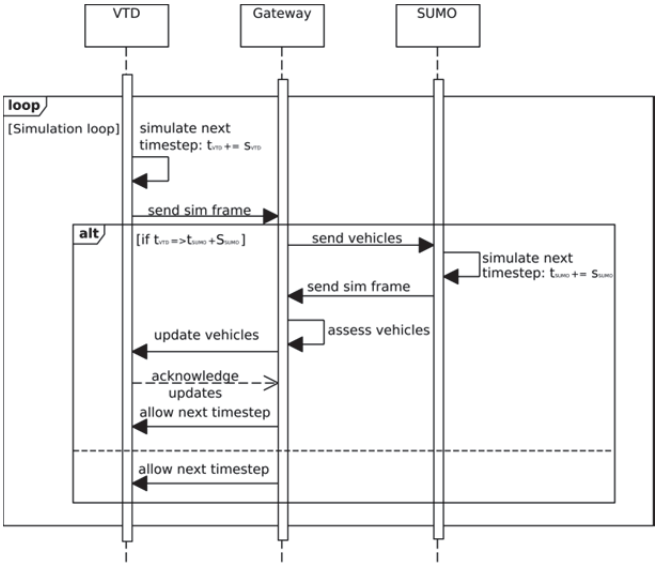


Figure 4-5: Synchronization

4.3.4 Implementation

The gateway depicted in Figure 4-5 is implemented as a dynamic plugin for VTD written in C++. It uses the TraCI network interface [21] provided by SUMO to control the vehicles of the microscopic simulation.

4.3.5 Generalization of the Approach

While the description in section 4.3.1 focuses on the simulation of a single EGO car, the concept can be generalized as follows. Generally speaking multiple EGO cars can be simulated analogously and can exist either in separate or in joined areas of high resolution. The dynamic spatial partitioning concept is not necessarily limited to having only two different simulators forming the multi-resolution simulation but even more simulators could be added to the synchronization scheme in figure 4-5.

The definition of the area of interest is also not limited to a circle. The circle was chosen due to its simple definition and fast distance calculations in the classification process, but the HRA could be represented by arbitrary complex shapes. As another generalization, the classification process need not only be based on pure geometrical calculations but could also be enriched with logical conditions, for example on a freeway vehicles driving on oncoming lanes could be excluded due to their limited relevance to the EGO car.

4.4 Evaluation

4.4.1 Scenario and Simulation Setup

A synthetic scenario was created for testing the coupling concept and evaluating its performance. It consists of a single straight road running west to east with a length of 50 kilometers and two lanes, one for each direction. Each lane is configured to have a constant inlet of 1000 vehicles per hour heading either east or west. The EGO car is located near the start of the road. It drives from west to east and is followed by a traffic flow and heading to the oncoming traffic flow. This artificial road was first modeled in the OpenDRIVE format and was then converted to the SUMO road network format.

We performed two series of experiments. In the first series, the nanoscopic traffic simulator VTD was applied to the whole simulated area. In the second series we used the described multi-resolution concept to partition the simulation area between VTD and SUMO. We chose a timestep of $s_{VTD} = 20\ ms$ for the high resolution area in VTD and a timestep of $s_{SUMO} = 1\ s$ for the low resolution area in SUMO. The hysteresis thresholds which define the dynamic area of interest were set to $R_{in} = 500\ m$ and $R_{out} = 550\ m$.

Both simulators are executed on the same computer which is equipped with an Intel Xeon CPU E5-1620 at 3.60 GHz and 16 GB of RAM. The operating system is Ubuntu Linux 14.04 with a 3.13 64bit kernel. We used VTD version 1.4.1 with 3D rendering disabled and a custom branch of SUMO v0.21 that incorporated an in-work version of the methods needed to exchange vehicles between VTD and SUMO.

4.4.2 Performance Evaluation

We measured the duration it takes to perform each simulation step over the simulation period of 1800 seconds while the number of vehicles is constantly being increased. Each series consists of five separate simulation runs to account for fluctuations in the measured execution times. To illustrate the trends of the measurements more clearly the moving average is also displayed in the following figures.

Figure 4-6 shows the performance development of the nanoscopic simulation while increasing the simulated vehicle count over the simulation period. The duration of each simulation step is almost constant up to a count of 70 vehicles. Until then, the duration is around 12 ms, which is less then the timestep length of 20 ms and therefore yet fulfills the real time constraint. At around 150 vehicles, the duration is beyond these 20 ms and real time simulation is not possible anymore. With increasing vehicle count the duration for each timestep also considerably increases and reaches 180 ms at the end of the simulation period. This results in an increase of factor 15 compared to the amount of computation time at the beginning of the simulation. The overall simulation took over 120 minutes to complete, which is 4 times more than the simulated time.

Figure 4-7 shows the performance development of the multi-resolution simulation in the same simulation scenario over the same simulation period. While the total vehicle count is increased the same way as in the pure nanoscopic simulation, the separately plotted nanoscopic vehicle count illustrates the amount of cars which are within the high resolution area. It shows that reducing the nanoscopic model's area of interest fulfills the aim of reducing the overall simulation time. After a local maximum of 11 nanoscopic cars is reached this count decreases slowly since slower vehicles are left behind the faster moving EGO car. At around simulation time 1350 seconds the two traffic flows from each end of the road meet in the middle of the road, which then increases the nanoscopic vehicle count. However, due to the limited extent of the HRA the nanoscopic vehicle count does not exceed a certain limit, which for the given configuration is at around 27 vehicles. The duration for the time steps stays on average constant around 12 ms, so it can be seen that the overhead resulting from the coupling of the two simulators is negligible as is the execution time of the microscopic simulator due to its less detailed yet much more efficient simulation model. The overall simulation took less than 18 minutes to complete, so the simulation is faster than real time by factor 1.66. The real time constraint is fulfilled throughout the whole simulation period.

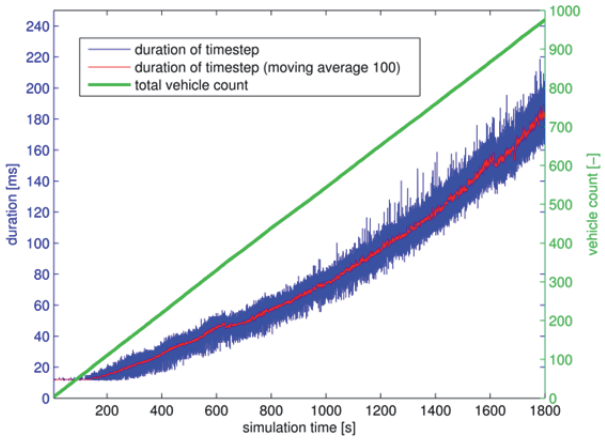


Figure 4-6: Simulation Performance - nanoscopic simulation only

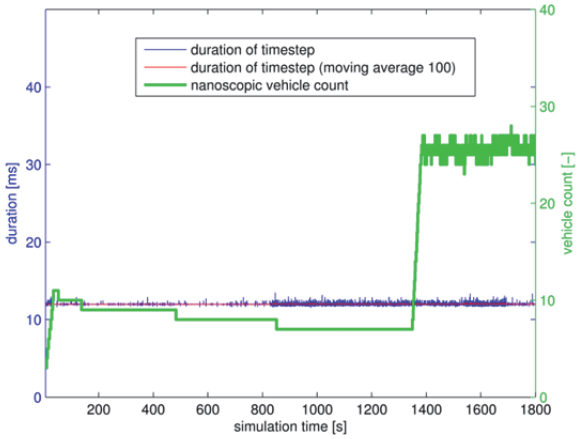


Figure 4-7: Simulation Performance - multi-resolution simulation

4.5 Discussion

The above shown performance evaluations state that the multi-resolution approach leads to the desired goal of a highly detailed simulation in the region of interest while maintaining a high overall performance. However, there are additional concerns, which we cover in the following.

When switching between resolutions and simulators, maintaining simulation consistency is of utmost importance [22]. The change of resolution from high to low can generally be handled rather simply by losing information through a transformation function. The opposite direction though is problematic. The presented approach will face a consistency problem when using vehicle dynamics models with a higher level of detail. When switching from low to high resolution only a minimal subset of the state variables (position and speed) is available. The remaining state variables (pitch and roll angle of the car body, engine torque, current gear, etc.) must be somehow interpolated to reach a valid state in the simulation model.

Additionally, simulating using the nanoscopic traffic simulator naturally requires a high resolution representation of the simulated area, especially the road network, but furthermore a 3D model of the environment if this is necessary for the employed sensor models. Microscopic simulation usually works with rather coarse road networks, which are available publicly from sources like OpenStreetMap. Nanoscopic simulation however has additional requirements, e.g. the continuity between road segments due to the vehicle dynamic simulation, which need to be satisfied. Obtaining the data basis in the necessary detail is not always directly possible and can cost an additional amount of time and money.

4.6 Conclusion

In this paper we proposed a concept for coupling traffic simulators of different simulation resolutions to achieve a multi-resolution traffic simulation which focuses on a dynamically determined area of interest. The presented methodology partitions the simulation area into a variable, highly detailed region of interest represented by a nanoscopic model and the surrounding area simulated at low resolution by a microscopic model. The evaluation shows a

dramatic reduction of computation time in comparison to a pure nanoscopic simulation of the same simulation dimensions, which even makes real time simulation possible. This divide-and-conquer strategy enables accurate, realistic and large scale testing and validation of real implementations of driver assistance systems based on vehicular networks in a virtual environment. As the next steps, we are investigating the application of the multi-resolution simulation methodology for the other domains relevant for the simulation of vehicular networks, namely network simulation and application emulation, to model the whole system across all domains efficiently at high fidelity.

4.7 References

[1] O. Gietelink, J. Ploeg, B. De Schutter, and M. Verhaegen, "Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations," Vehicle System Dynamics, vol. 44, no. 7, pp. 569–590, 2006.

[2] G. Naus, R. Vugts, J. Ploeg, M. van de Molengraft, and M. Steinbuch, "String-stable cacc design and experimental validation: A frequency-domain approach," Vehicular Technology, IEEE Transactions on, vol. 59, no. 9, pp. 4268–4279, Nov 2010.

[3] S. P. Hoogendoorn and P. H. Bovy, "State-of-the-art of vehicular traffic flow modelling," Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, vol. 215, no. 4, pp. 283–303, 2001.

[4] D. Ni, "A framework for new generation transportation simulation," in Proceedings of the 38th Conference on Winter Simulation, ser. WSC '06. plus 0.5em minus 0.4emWinter Simulation Conference, 2006, pp. 1508–1514.

[5] C. Sommer, R. German, and F. Dressler, "Bidirectionally coupled network and road traffic simulation for improved ivc analysis," IEEE Transactions on Mobile Computing, vol. 10, no. 1, pp. 3–15, January 2011.

[6] F. J. Ros, J. A. Martinez, and P. M. Ruiz, "A survey on modeling and simulation of vehicular networks: Communications, mobility, and tools," Computer Communications, vol. 43, pp. 1–15, 2014.

[7] M. J. Booysen, S. Zeadally, and G.-J. Van Rooyen, "Impact of neighbor awareness at the mac layer in a vehicular ad-hoc network (vanet)," in Wireless Vehicular Communications (WiVeC), 2013 IEEE 5th International Symposium on. plus 0.5em minus 0.4emIEEE, 2013, pp. 1–5.

[8] C. Sommer, R. Krul, R. German, and F. Dressler, "Emissions vs. travel time: Simulative evaluation of the environmental impact of its," in 71st IEEE Vehicular Technology Conference (VTC2010-Spring). plus 0.5em minus 0.4emTaipei, Taiwan: IEEE, May 2010, pp. 1–5.

[9] P. K. Davis and R. Hillestad, "Families of models that cross levels of resolution: Issues for design, calibration and management," in Proceedings of the 25th conference on Winter simulation. plus 0.5em minus 0.4emACM, 1993, pp. 1003–1012.

[10] G. Flötteröd and K. Nagel, "High speed combined micro/macro simulation of traffic flow," in IEEE Intelligent Transportation Systems Conference, ITSC, 2007, pp. 1114–1119.

[11] D. Rieck, B. Schünemann, I. Radusch, and C. Meinel, "Efficient traffic simulator coupling in a distributed v2x simulation environment," in Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, ser. SIMUTools '10. plus 0.5em minus 0.4emICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010, pp. 72:1–72:9.

[12] T. Potuzak, "Issues of parallel hybrid nanoscopic/microscopic road traffic simulation," in EUROCON, 2013 IEEE, July 2013, pp. 614–621.

[13] L. Navarro, F. Flacher, and V. Corruble, "Dynamic level of detail for large scale agent-based urban simulations," in The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2, ser. AAMAS '11. plus 0.5em minus 0.4emRichland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 701–708.

[14] J. Olstam and R. Elyasi-Pour, "Combining traffic and vehicle simulation for enhanced evaluations of powertrain related adas for trucks," in Intelligent Transportation Systems - (ITSC), 2013 16th International IEEE Conference on, Oct 2013, pp. 851–856.

[15] J. L. F. Pereira and R. J. F. Rossetti, "An integrated architecture for autonomous vehicles simulation." in SAC, S. Ossowski and P. Lecca, Eds. plus 0.5em minus 0.4emACM, 2012, pp. 286–292.

[16] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," International Journal On Advances in Systems and Measurements, vol. 5, no. 3&4, pp. 128–138, December 2012.

[17] D. Krajzewicz, M. Bonert, and P. Wagner, "The open source traffic simulation package sumo," in RoboCup 2006, June 2006.

[18] S. Krauss, P. Wagner, and C. Gawron, "Metastable states in a microscopic model of traffic flow," Phys. Rev. E, vol. 55, pp. 5597–5602, May 1997.

[19] K. von Neumann-Cosel, M. Dupuis, and C. Weiss, "Virtual test drive - provision of a consistent tool-set for [d,h,s,v]-in-the-loop," in Proceedings of the Driving Simulation Conference Monaco, 2009.

[20] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," Physical Review E, vol. 62, no. 2, p. 1805, 2000.

[21] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux, "Traci: An interface for coupling road traffic and network simulators," in Proceedings of the 11th Communications and Networking Simulation Symposium, ser. CNS '08. plus 0.5em minus 0.4em New York, NY, USA: ACM, 2008, pp. 155–163.

[22] P. F. Reynolds, Jr., A. Natrajan, and S. Srinivasan, "Consistency maintenance in multiresolution simulation," ACM Trans. Model. Comput. Simul., vol. 7, no. 3, pp. 368–392, Jul. 1997.

5 How does the traffic behavior change by using In-Vehicle Signage for speed limits in urban areas?

Laura Bieker;
German Aerospace Center , Rutherfordstraße 2, 12489 Berlin, Germany

Laura.Bieker@DLR.de

Abstract

The impact of In-Vehicle Signage (IVS) for speed limits via Car2Infrastructure communication was investigated in a test field in Tampere (Finland). The test field results show that IVS has a positive influence on the speed of the drivers of the equipped vehicle. Different scenarios for the estimated penetration rates of equipped vehicles were set-up to see the overall effects on the traffic efficiency by IVS. The effects on the traffic efficiency were simulated in SUMO. The simulation results show that no benefit in traffic efficiency for all traffic participants could be reached by IVS.

Keywords: C2X, In-Vehicle Signage, Traffic efficiency

5.1 Introduction

The purpose of the In-Vehicle Signage (IVS) function is to display the traffic signs on an onboard unit inside the vehicle to improve the driver's perception of the sign. The main idea was that many drivers would reduce speed if they would be more aware of the current speed limit. An implication of the reduced speed could be an effect on traffic safety and/or traffic efficiency.

The IVS function is implemented via Car2Infrastructure communication (C2I). This means that the car is sending and receiving messages from/to a road side unit along the road. In this study the IVS for speed limits in urban areas was investigated. The onboard unit reminds the driver of speed limits and warns if he/she is violating it. The visual warning is given at the location of the traffic sign and was visible for 100-200 meters (depending on the relevance and physical environment).

IVS informs the driver under the following circumstances: when driving just under (90%–100%), just above (100%–110%) or significantly above (over 110%) the speed limit. Here, the purpose of this system is to make the driver aware of how he/she is driving with respect to the speed limit. The purpose of the IVS is to encourage the driver to slow down when exceeding the speed limit.

5.2 Test field in Tampere (Finland)

In the EU project DRIVE C2X 7 test sides have been built up [1]. A basic set for C2X-services has been tested at these locations, see Figure 5-1.

Figure 5-6: Test sides of DRIVE C2X [1]

For this study the data from the test field in Tampere (Finland) were used, see Figure 5-2. The test-field has roads with different speed limits. VTT was operating the test-field and was supported by the city of Tampere. The test field includes 8 km of urban roads and normally 1000-2500 vehicles per hour are driving on the streets.

Figure 5-7: Finnish test field

5.3 Analysed Data

The data were collected from 24.April to 08.May 2013. The cars of the test drivers were equipped with Car2Infrastracture communication (C2I). Along the streets 4 road side units were installed to collect the data of the equipped vehicles and send information about the traffic signs. Two test drive scenarios exist:

(1) the baseline scenario: the vehicle is driving like a normal vehicle without IVS only the vehicle data is collected via C2I.

(2) Treatment scenario: the vehicle is driving with IVS. The dataset includes 544 events, with equally 277 for baseline and treatment events being ready for the analysis.

Table 5-2. Data from the finish test field

Indicator	Mean (baseline)	Std. Dev. (baseline)	Mean (treatment)	Std. Dev. (treatment)
Average speed (km/h)	54.551	3.988	53.738	3.607
Average speed 30km/h speed limit) (km/h)	34.651	4.402	33.144	3.819
Average speed 40km/h speed limit (km/h)	39.017	3.510	38.568	3.286

The analysed data show that the speed limit warnings have a small positive effect on the driving behavior in terms of reduction of speed (of 4% at 30km/h speed limit and of 1% at 40km/h speed limit). A reduced speed also implies often reduced fuel consumption and produced emissions. These positive results are only for the equipped vehicles, but it would be interesting to see how these results could influence the traffic behavior overall.

5.4 Simulation

To scale up the results of the equipped vehicles on a level of a whole vehicle population (including equipped and non-equipped vehicles) a simulation in SUMO was performed [2].

Simulation Scenario

An urban road of 1 km length was used as simulation scenario, see Figure 5-3. The same scenario is simulated with a speed limit of 30 km/h and 40 km/h to analyze the difference on traffic efficiency in both cases. A road with two lanes was chosen so that all vehicles are driving in the same direction. The opposing traffic is assumed to have no effects on the driving behavior. Simulation runs with only one lane showed that the vehicles have to adapt their speed to the leading vehicle with the lowest speed because no overtaking is possible.

Figure 5-8: Urban scenario with speed limits 30 and 40 km/h, two lanes in the same direction.

IVS is not expected to have an effect on route choice. Hence it is sufficient to analyse corridor networks without route alternatives. In SUMO every driver has an own speed which is randomly computed for every street and the corresponding speed limit. For every vehicle type a speed factor and a standard deviation can be given. The desired speed of each vehicle is randomly given according to a lognormal distribution with the average speed and the standard deviation from the field test. Additionally, the standard deviation is also given for the field test data. Given these figures a lognormal distribution can be calculated and the desired speed of the vehicles is set according to the speed distribution 100 meters in front of the speed limit so the vehicle has time to adapt its speed.

Penetration rate

For the simulation different scenarios of penetration rates of the vehicle within the simulation were investigated within the DRIVE C2X project [3]:

- a main estimate,
- a pessimistic estimate and
- an optimistic estimate.

Figure 5-4 shows graphs of these estimates over time. Four cases will be considered for the penetration rates in the simulation scenarios.

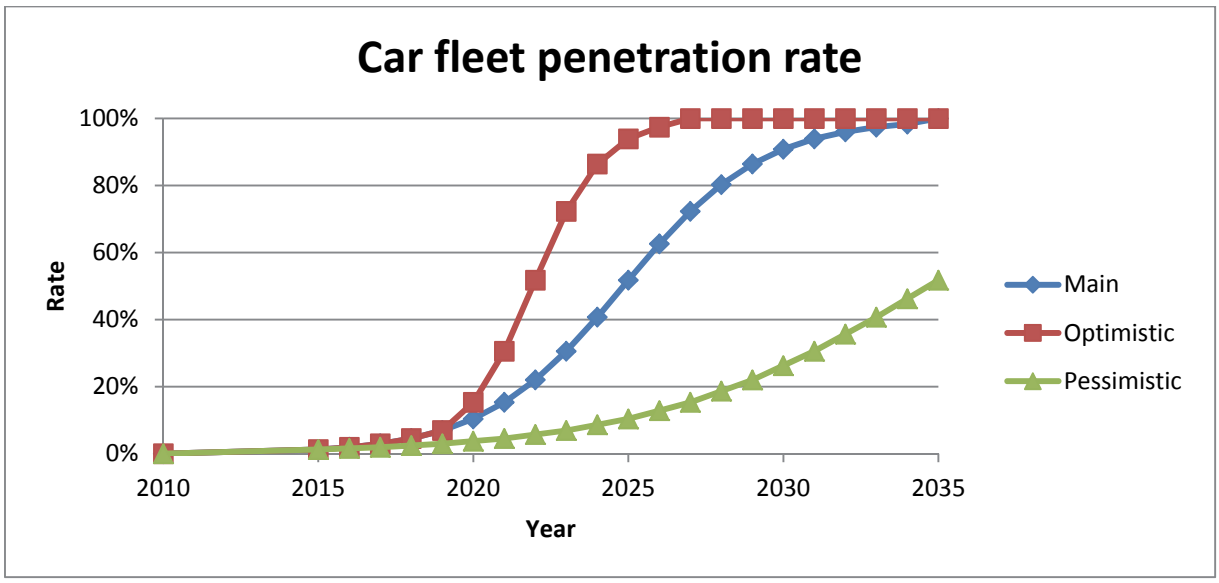


Figure 5-9: Fleet penetration rate estimates for all Drive C2X systems.

Table 5-3: Fleet penetration rate

Scenario	Car penetration rate
Baseline (2010)	0.00%
Low (2020, Main)	10.39%
Medium (2030, Pessimistic)	26.29%
High (2030, Main)	90.84%

Every simulation scenario was run 10 times with a high traffic demand (peak) and low traffic demand (off-peak). The results of the simulation runs can be found in the next section.

5.5 Results

For analyzing the results of the simulation 3 induction loops were included within the scenario to measure the local mean speed. The first speed detector was 200 meters before the speed limit sign, one was directly at the location of the speed limit sign and one was 200 meters after the speed limit sign. The travel time and the delay time was calculated for the whole trip of each vehicle within the simulation.

Table 5-4: Results of simulation scenario with off-peak demand

Indicator	Unit	Baseline	Low	Medium	High
Travel time (av)	s	94.01	93.65	93.76	93.99
Travel time (std)	s	12.88	12.52	12.56	12.23
Delay (av)	s	17.09	16.74	16.87	17.06
Delay (std)	s	12.88	12.52	12.56	12.23
Speed detector 1	km/h	51.51	52.16	51.56	51.65
Speed detector 2	km/h	38.79	38.89	38.99	38.69
Speed detector 3	km/h	36.78	36.89	37.12	36.48

The differences in travel time, delay and speed are in almost all scenarios very small and not significant compared to the baseline scenario. Some strange behavior can be seen for example the average travel time decreases in the low scenario while it was expected to increase because a larger percentage of drivers would reduce the speed. An explanation could be that the measured speed differences in the test fields are too small to see the difference in a stochastic simulation. Namely the changes are 1.507 km/h for speed limit 30km/h, and 0.449 km/h for speed limit 40 km/h. These small changes seem to be neglected in the simulation with many traffic participants. One way would be to have even more simulation runs to see whether it is possible to produce significant results. Another way would be to check whether significant results could be produced if the speed difference for equipped and non-equipped vehicles would be larger. In the full paper the simulation scenario is run also with artificial numbers of speed changes to see the effects on traffic efficiency then.

Table 5-5: Results of the simulation scenario with peak demand

Indicator	Unit	Baseline	Low	Medium	High
Travel time (av)	s	94.15	94.23	94.24	94.10
Travel time (std)	s	12.91	12.60	12.99	12.07
Delay (av)	s	17.19	17.27	17.28	17.14
Delay (std)	s	12.91	12.60	12.99	12.07
Speed detector 1	km/h	49.48	49.62	49.55	49.63
Speed detector 2	km/h	38.71	38.57	38.60	38.45
Speed detector 3	km/h	36.55	36.38	36.57	36.38

5.6 Summaries

The results of the test field indicate that IVS has positive effects on the speed of equipped vehicles. IVS can influence traffic efficiency by displaying speed sign information in car and thereby adapting the drivers' desired speed. The effects found in the field trial were used to model the change in desired speed in the traffic simulations: for urban roads the desired speeds were lowered by 1% to 4%. The results are not significant. It could be presumed that the influence of IVS in Urban areas is too small to see benefits in traffic efficiency of the whole traffic.

5.7 References

[1] DRIVE C2X http://www.drive-c2x.eu/test-sites. last visited 23. Januar 2015.

[2] Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.: Recent Development and Applications of SUMO - Simulation of Urban MObility. International Journal On Advances in Systems and Measurements, 5 (3&4):128-138, December 2012.

[3] DRIVE C2X Deliverable D 11.4 Impacts of cooperative systems and user perception. http://www.drive-c2x.eu/tl_files/publications/Deliverables%20and%20abstracts/DRIVE%20C2X_D11.4_Impact_Assessment_v1.0_full%20version.pdf last visited 23. January 2015.

6 A Hybrid Approach to Large Scale Simulation Based Traffic Assignment

Michael Behrisch;
German Aerospace Center , Rutherfordstraße 2, 12489 Berlin, Germany

Michael.Behrisch@DLR.de

Abstract

Two of the basic methods of traffic assignment being static assignment of link costs and a dynamic assignment based on microscopic traffic simulation results are combined to derive a good starting solution for the more precise microscopic approach from the coarse macroscopic solution. In addition a new tool from the SUMO suite and some first results of applying the schema to the city of Berlin are presented.

Keywords: SUMO, Macroscopic Assignment, Dynamic User Assignment, Microscopic Traffic Simulation

6.1 Introduction

In the context of the VEU project a coupling between an agent based traffic demand model (TAPAS [3]) and a microscopic traffic simulation (SUMO [1]) is being developed. The objective of this connection is to give realistic feedback about the expected travel times to the agents of demand model. This allows for an iterative refinement of the planned activities as well as the expected mobility costs in terms of time but also emission. Since both models are based on the simulation of individual agents, which is a potentially time consuming task, and they should be applied to scenarios of larger cities like Berlin (3.5 million inhabitants), improving runtime performance is almost as important an issue as the quality of the solution. This paper will focus mostly on the runtime aspect by describing an approach to traffic assignment which differs considerably from the current state of the art in SUMO while retaining or even improving the quality of the user assignment.

The next section describes the current state of dynamic user assignment in SUMO together with some drawbacks and limitations especially concerning large scenarios. Section 3 describes the new MARouter tool recently introduced into the SUMO suite and how it can be applied to solve some of the limitations of the pure microscopic approach. Section 4 and 5 describe the Berlin scenario and give some first results of the approach.

6.2 Microscopic Traffic Assignment with SUMO

The current microsimulation approach in SUMO involves the repeated execution of a routing step and a detailed microsimulation on the resulting routes leading to a new traffic situation. This traffic situation is measured edgewise mostly based on the single measurement of travel time but sometimes also on other measurements such as fuel consumption [2]. The resulting

edge costs are fed back into a router which calculates new route costs and probabilities (based on a model by Gawron[4]) trying to achieve a user equilibrium state, where no user can reduce its individual route cost by changing to a different route. As the description already indicates this process involves a large number of microscopic traffic simulations which in itself is already a time consuming task (simulating a city like Berlin for a whole day can easily take up to several hours). Usually there are constraints applied to limit the number of iterations either to a fixed amount or to some limit in the variation of the travel times among the available routes for each car.

Still there are major complications in using this approach because it tends to give unstable results for large scenarios where after a relative period of stability later iterations show very different travel times (deviating in both directions). Furthermore it tends to recover only very slowly from iterations in which the network was completely jammed. For our scenario we took the most basic approach, evaluating an assignment as being successful, when the number of cars being locked (a car is considered locked when it does not move for more than five minutes) is below 5% of the total number of cars. In the conventional microscopic approach we failed to fulfil this criterion using the Berlin scenario described below, so we needed to plug in another method leading to two different strategies, one being the MARouter approach described in the next section. The other strategy is based on direct routing in the simulation and will be described elsewhere.

The traffic assignment process itself is still subject to a number of improvements since it can for instance easily get confused if there are a lot of routes between an origin and a destination travelling the same jammed edge, because it stores by default only travel times for (a limited number of) whole routes and not for single edges. It is also expected that a macroscopic preprocessing can help in avoiding an early breakdown of the iterative assignment process due to such events.

6.3 The Macroscopic Assignment Tool (MARouter)

The MARouter (short for macroscopic assignment router) is a tool which was introduced into the SUMO suite starting with the release of version 0.22.0. It implements standard algorithms for macroscopic traffic assignment such as the iterative assignment algorithm and the stochastic user assignment based on Lohse.

The input for a macroscopic assignment run is either an origin-destination matrix (O-D matrix) containing aggregated information about trip counts between traffic zones or a list of trips which is then aggregated internally but disaggregated before output. The routing is currently available for individual cars only so there is no public transport involved.

The MARouter is (for large scenarios) much faster than the iterative microscopic assignment since it replaces the microscopic simulation step by a simple calculation of expected travel times based on the measured demand. This allows for a fast iteration but loses all the benefits of microscopic simulation as detailed junction modelling and interaction of traffic flows. In order to keep the microscopic advantages in the result, we coupled the approaches to start with a macroscopic assignment and then feed the results into the microscopic loop mentioned above.

6.3.1 Macroscopic Traffic Assignment

The MARouter implements a traditional static approach to traffic assignment. The input consists of an input area (a SUMO network) subdivided into so called traffic analysis zones (TAZ) and an origin destination matrix (OD matrix) which is given for a fixed time interval (usually a day) and a mode of transport (e.g. passenger cars). This matrix has one entry for each pair of origin A and destination B defining the amount of vehicles which move (or rather start moving) from A to B. A second input into the modell is a set of capacity restraint (CR) functions which model how the travel time on an edge increases with the volume of traffic on that edge. The MARouter currently uses functions of the following form:

$$t(x) = t_0 * (1 + c * \frac{x}{l})$$

where t_0 is the travel time in the empty network, l is the number of lanes and c is a constant depending on features like the road class (highway or urban road) and the maximum allowed speed. At the moment those functions are hard coded into the SUMO codebase and cannot be altered by the user (except by modifying the source code). They can be found in the file src/marouter/ROMAAssignments.cpp. The MARouter calculates from those OD matrices and functions an assignment which is a set of routes for each OD pair together with a number of vehicles driving this route. In order to do so several basic algorithms can be employed which calculate those mappings iteratively. The most basic approach called iterative assignment assigns a fixed percentage of the demand to the net, recalculates the expected travel times given the CR functions above and assigns the next part. For further algorithms we refer to an upcoming extended version of this paper or to the excellent descriptions in [5].

6.3.2 Input formats

SUMO accepts OD matrices in different formats which originate either from VISUM or from the AMITRAN project [6]. An OD matrix in the standard VISUM format looks like the following:

```
$V
* From-Time  To-Time
0.00 24.00
* Factor
1.00
*
* some
* additional
* comments
* District number
2
* names:
         1          2
*
* District 1 Sum = 100
         0          100
* District 2 Sum = 0
         0          0
```

All lines starting with asterisks are comments so the important information is the time span in the beginning, the number and names of the TAZ' (districts) and then the real demand given matrix line by line. In the given example 100 vehicles drive from "1" to "2" during a whole day. In the AMITRAN format the same input looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<demand xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/amitran/od.xsd">
  <actorConfig id="0">
    <timeSlice duration="86400000" startTime="0">
      <odPair amount="100" destination="2" origin="1"/>
    </timeSlice>
  </actorConfig>
</demand>
```

In addition to the VISUM format the AMITRAN format contains information about the mode (or rather the vehicle class) it describes as well. This is encoded in the actorConfig which refers to an exisiting vehicle type for SUMO. Since the format only allows for numeric ids, the mapping from the actorConfig to the vehicle type is done in a separate input file.

6.3.3 Output format

The MARouter creates SUMO route files in the standard SUMO format describing the routes and the percentages of usage between the different OD relations. For the example given above the output would look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<routes xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/routes_file.xsd">
    <routeDistribution id="0">
        <route cost="90.24" probability="100.00000000" edges="middle end"/>
    </routeDistribution>
</routes>
```

MARouter can also directly output flows which can be fed into the simulation or the DUARouter / dualterate.py procedure:

```
<?xml version="1.0" encoding="UTF-8"?>
<routes xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/routes_file.xsd">
    <flow id="0" begin="0.00" end="86400.00" number="100" fromTaz="1" toTaz="2">
        <routeDistribution>
            <route cost="90.24" probability="100.00000000" edges="middle end"/>
        </routeDistribution>
    </flow>
</routes>
```

In this example MARouter determined only a single route between the source and the destination. The probability in the described output is not normed to 1 in order to allow for easy recalculation of the number of vehicles travelling each of the routes.

6.4 The Berlin scenario

The test scenario is the individual traffic demand (only passenger cars, no public transport and no delivery / goods traffic) for the city of Berlin consisting of approximately three million trips a day between 1100 traffic zones. The TAPAS model sets up a virtual population for the city of Berlin and determines its transportation demands for an average weekday. Each person has an assigned activity plan representing destinations like work and school places and the time they spent there. The input data is based on various sources mainly statistical data from the Berlin Microcensus and the "Mobilität in Deutschland" study. For a detailed desciption, see [3]. The model generates trips for all available modes of transportation, but in the current

implementation only the individual traffic is used for the later microsimulation. The input from the TAPAS simulation consisting of individual trips was aggregated on the traffic zone level for running the MARouter. The resulting edge loads are fed into the microscopic dynamic user assignment procedure as described in section 6.2.

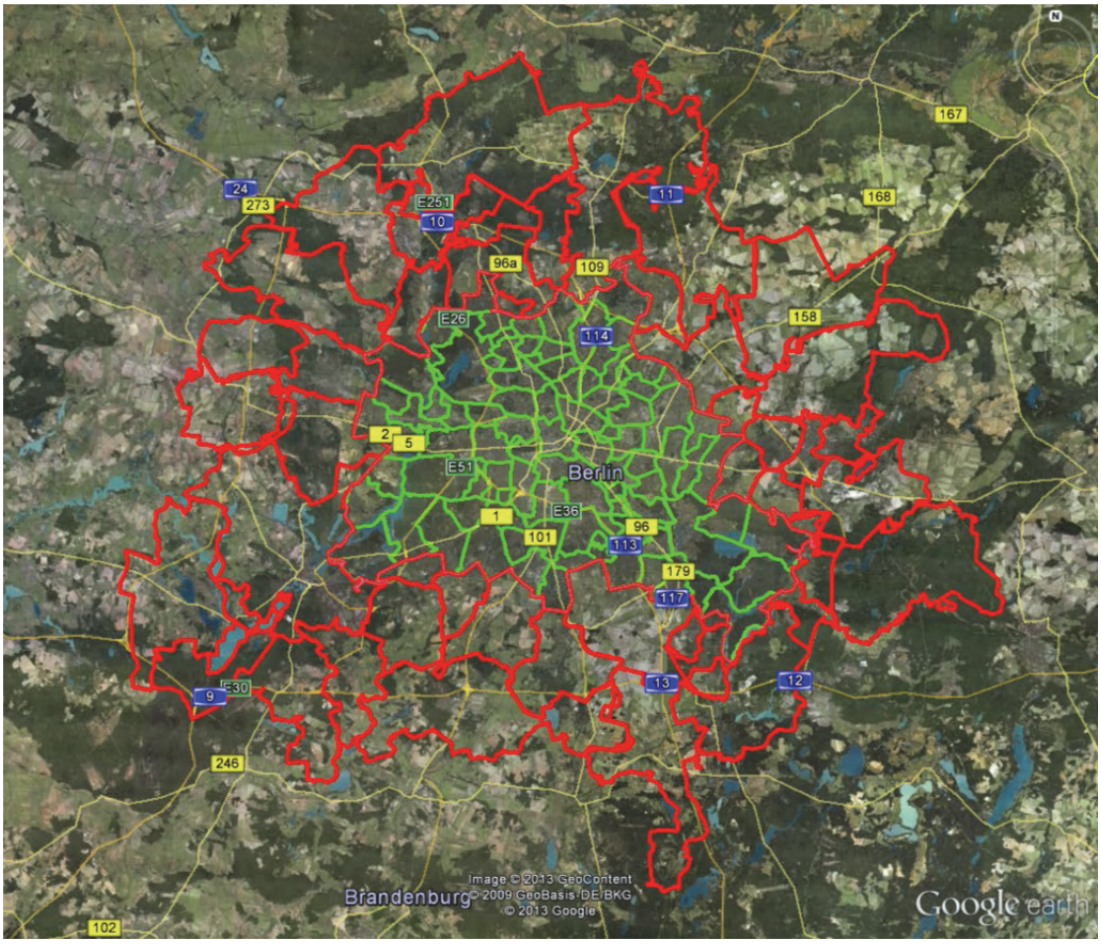


Figure 6-10. The districts of the City of Berlin together with the surroundings (courtesy Google Earth) Each of the depicted districts corresponds to roughly 20 traffic analysis zones.

Afterwards a comparison between the raw microscopic process and the process with macrospcopic preprocessing could be performed. Unfortunately the whole process still takes in the order of a day to complete so that there are only preliminary results available yet.

6.5 Conclusion and discussion

The first results show a speedup of the user assignment runs. To reach a comparable result the runtime decreases in a range of 15%-20%. From the small sample set it is hard however to judge the final results especially concerning the quality of the solution. More detailed results are expected in further research and will be presented at the conference.

While the speedups are currently not as high as expected the MARouter tool itself seems to be of high value because now for the first time we can directly compare macrospcopic and microscopic assignment strategies on the same dataset. Further results also on abstract networks will be presented in an extended version of this paper.

6.6 References

[1] Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D.: SUMO – Simulation of Urban MObility: An Overview. In: SIMUL 2011, The Third International Conference on Advances in System Simulation (2011)

[2] 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, 09. Dez. - 12. Dez. 2012, Berlin, Deutschland.

[3] Justen, Andreas und Cyganski, Rita (2008) *Decision-making by microscopic demand modeling: a case study.* In: Transportation decision making: issues, tools, models and case studies. genesidesign.com. Transportation decision making: issues, tools and case studies, 2008-11-13 - 2008-11-14, Venedig.

[4] Gawron, C. (1998). Simulation-based traffic assignment – computing user equilibria in large street networks. Ph.D. Dissertation, University of Köln, Germany.

[5] Lohse, D., Schnabel, W. (2011). Grundlagen der Straßenverkehrstechnik und der Verkehrsplanung: Band 2 – Verkehrsplanung, Beuth Verlag

[6] AMITRAN consortium (2014): AMITRAN web pages, http://www.amitran.eu/, last visited on 10.04.2015

7 A SUMO Extension for Norm based Traffic Control Systems

Jetze Baumfalk, Barend Poot, Bas Testerink and Mehdi Dastani;
Department of Information and Computing Sciences, Universiteit Utrecht, The Netherlands

{J.T.Baumfalk,B.W.A.Poot,B.J.G.Testerink,M.M.Dastani}@uu.nl

Abstract

Autonomous vehicles will most likely participate in traffic in the near future. The advent of autonomous vehicles allows us to explore innovative ideas for traffic control such as norm based control systems. A norm is a violable rule that describes correct behavior. Norm based traffic control systems monitor traffic and effectuate sanctions in case vehicles violate norms. In this paper we present an extension of SUMO that enables the user to apply norm based control systems to traffic simulations. In our extension, vehicles are capable of making an autonomous decision on whether to comply with the norms or not. We provide a description of the extension, a summary on its implementation and an experimental evaluation.

Keywords: Normative systems, distributed control.

7.1 Introduction

In the near future, smart roads with autonomous vehicles will become prevalent. Vehicles will no longer have a driver, but will drive themselves and communicate with the infrastructure and other vehicles. This allows, and gives rise to the need, for the investigation of new traffic control measures [11]. Traffic control systems can exert some level of influence on vehicles in order to improve traffic flow and safety. In this paper, we shall focus on control systems where the infrastructure instructs vehicles on how they ought to behave.

Current control techniques include for instance ramp metering where a traffic light, or adaptive speed limit displays, can be used in order to control traffic flows. However, with new autonomous vehicles (see [14],[15],[16]) that have the ability to pursue goals, follow regulations, and perform an array of actions, we can give more individualized directives. We do not necessarily have to rely on drivers to observe road side units, but can communicate directly with vehicles, since the vehicles can communicate with their environment, reason to decide which actions to perform, and execute their decided actions. In the use case shown in this paper we can for instance assign target velocities to individual vehicles. Just like current traffic regulations, it is possible that vehicles violate these regulations by speeding or driving on an emergency lane in order to arrive at their destination at an earlier time. We observe however that microsimulation platforms such as SUMO do not currently allow for vehicles to reason about regulations, the consequences of violations and their personal preferences.

In this paper we present the implementation of a new traffic control system for SUMO. This control system is based on the notion of norms and is hence reffered to as a norm based

control system. Norms can be seen as violable rules that specify desirable behavior. Norm based control systems are a proposed technique for solving control problems where the subjects of the regulations are capable of making autonomous decisions on whether to violate the regulations. They lend themselves very well for the cause of traffic control. A norm based traffic control system is a more natural way of modeling roads with different properties and regulations [13]. We also provide a new vehicle driver model so that vehicles can reason about norms. Our implementation is a plug-and-play extension for SUMO. This allows researchers to use our extension out of the box.

We illustrate our framework by applying it to the common example of merging traffic streams. For a schematic overview, see Figure 7-1 where two traffic streams have to be merged together. There is one main traffic stream and one secondary traffic stream that joins the main stream, resulting into a single output stream. The goal is to make optimal use of the output capacity of the network whilst not causing unnecessary traffic jams for the secondary traffic stream or compromising safety. A solid analysis of this scenario can be found in [11]. Our approach is to use a norm based control system that assigns individual norms to agents. In particular, agents are obligated to move or stay on a lane and/or adopt a certain target speed until they are released of this obligation.

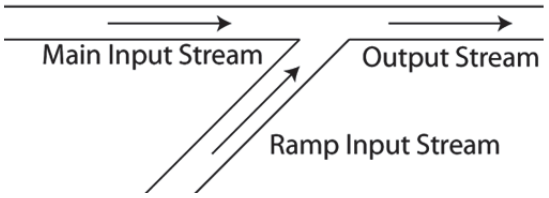


Figure 7-11. Example setting where two traffic streams must merge

Our paper is structured as follows. We first give a brief introduction in the field of norm based control systems. We will then show how the norm based control system for SUMO is designed, and discuss an application. Following that, we describe how vehicles can reason about the norms that they receive. In the next section we evaluate our extension through a series of experiments that highlight different aspects of our contribution. After that, we briefly look at related work and see how our approach differs. Finally, we summarize the main points and contributions of our work.

7.2 Norm Based Control Systems

The theory of norm based control systems originates from the Multi-Agent Systems (MAS) community [17]. In the MAS paradigm, the main concepts and abstractions are the environment, agents, norms, and control system. The environment is the system in which agents can act. For our purposes we see SUMO without driver models as the environment. The agents in a MAS are autonomous entities that exhibit their own behavior. We see the drivers of vehicles as the agents in a traffic MAS. Norms specify how agents ought to behave under what circumstances. For instance, traffic regulations could be seen as norms. The control system monitors agents' behaviors and the environment, and checks whether norms are violated. If a norm violation is detected, then the control system can impose a sanction to the violating agent. These sanctions are imposed on agents in order to discourage the agent to violate any traffic regulations. We refer to the agents that operate the vehicles when we refer to vehicles throughout this paper.

Background on Norm Based Control Systems

Multi-agent systems consist of autonomous interacting agents whose behavior is not always predictable or controllable. In addition to reactively responding to environmental changes, autonomous agents are assumed to have their own objectives (goals) for which they proactively initiate actions in order to achieve them. The behavior of individual agents needs to be controlled and coordinated in order to ensure the desirable properties at the system level [17]. One approach is to design and implement hard constraints at both individual agent and multi-agent system levels. This approach may not always be desirable or feasible, since limiting the autonomy of individual agents can be costly or even impossible. For instance, consider future traffic on smart roads as a multi-agent system where vehicle drivers (autonomous cars) are modeled as individual agents and the road infrastructure is modeled as the multi-agent system environment. In such a scenario, it is undesirable, if not impossible, to fully control and determine the behavior of all individual autonomous cars, as these vehicles are designed to make their own decisions.

Norm based control systems are widely proposed as an effective mechanism to control and coordinate the behavior of individual agents [4]. In norm based control systems, norms are considered as specifying the standards of behavior that can be used to govern the interaction between autonomous agents (cf. [10],[5],[2]). Across various theories and frameworks there is no general consensus on what a norm is. In this paper we use the term norm as a reference to both norm schemes and norm instantiations. With the concept norm scheme we refer to the specification of the circumstances after which a specific agent is obliged to achieve a system state, and the sanction that will be imposed should this obligation not be met before a certain deadline (cf. [10]). With the concept norm instance we refer to a specific obligation and sanction that holds for a specific agent. Norms can take the form of an obligation, prohibition or permission, but the scope of this paper concerns only obligation norms. The application of norms in multi-agent systems, called norm based multi-agent systems, requires continuous monitoring of the behavior of individual agents, evaluation of their behavior with respect to the specified norms, and assurance that norm violating agents are sanctioned. This approach maintains the agents' autonomy and can still promote desirable behavior. We observe that traffic regulations fit the normative approach. For instance one can represent the regulation “if you approach a ramp, then you ought to have a safe speed for merging into the traffic stream” as a norm.

Finally, norm instantiations in norm based control systems are announced to autonomous agents, just like drivers are assumed to be aware of traffic regulations. Norm awareness allows an agent to reason and decide whether or not to comply with the norms (cf.[1],[7]). Norm awareness is crucial for future traffic on smart roads as autonomous vehicles need to know the consequences of norm violations before deciding whether or not to comply with the norms.

A Norm Based Control System for SUMO

A norm based traffic control system monitors vehicle behavior and reacts to it. In our extension of SUMO, the monitoring functionality of traffic control systems serves two purposes. The first purpose is to detect violations of norms, such as speeding, tailgating or driving on a priority lane without a permit. If such a violation is detected, then a sanction coupled with this violation will be issued towards the violating vehicle. The second purpose of

monitoring is to issue new norms. If the traffic control system continues to observe situations where either the throughput or safety of vehicles declines, then a norm might be issued to improve the situation. This norm might be global, or tailored to a specific vehicle. This allows the control system to be very adaptive to traffic dynamics. Furthermore, the severity of a sanction might be increased if the coupled violation either occurs an excessive amount of times, or seems to be the cause of problematic situations. In our framework for norm based traffic control, both autonomous vehicles and the control system show adaptive behavior towards the ever-changing traffic flow and enable the system to cope with difficult and dynamic traffic situations. An improvement might also be possible in common situations like ramp-merging. In this scenario, norms issued by the traffic control system are aiding in improving the throughput of vehicles.

The structure of the presented norm based control system for SUMO is depicted in Figure 7-2. As stated before, we use the native SUMO application as the environment. However, we do not use SUMO's driver models. Instead we communicate commands from our own agent model to the vehicles. We have also implemented our own sensor business logic. These sensors are connected to lane area detectors as implemented in SUMO. The communication between the extension software and SUMO is provided by the TraaS library. We use and provide a new version of TraaS. In addition to improved performances, the new version has some extra functionalities that are needed for our extension.

The software in our extension is composed mainly of the agent models and the control system. The control system monitors vehicles through its sensors. It then checks the norms to determine whether new obligations hold for specific agents (if so, then they are also communicated to said agents) and/or whether norms are violated. The software executes in lockstep with SUMO, i.e. each simulation tick in SUMO is also a tick in the extension.

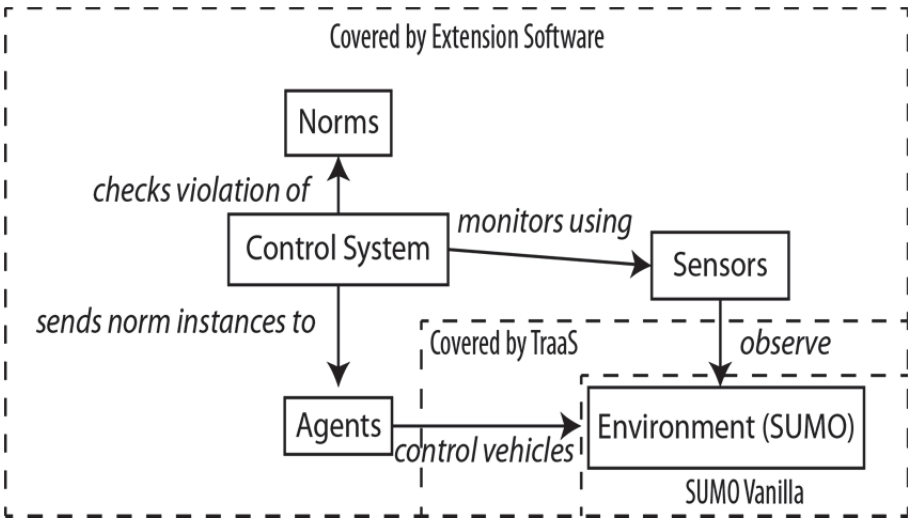


Figure 7-2. Overview of the presented extension.

We chose to provide our work as an extension to SUMO, rather than modifying the source code of SUMO directly. There are several reasons for this. First of all, vehicles in SUMO are goal directed only in a limited way. Their goal is to follow a certain route, as opposed to having a specific location as the destination. Second, SUMO agents are preprogrammed to follow a specific route. They only respond reactively to their environment, instead of deliberating on what action would best suit them. Third of all, vehicles in SUMO are not inherently capable of deciding to break the rules. For example, they always stop for a red light and they obey to the right of way. This is because vehicles in SUMO move according to specific car-following and lane-changing models and these models do not allow

reasoning about norms for action decision. The car-following model is not created with norm aware agents in mind. The most commonly used car-following model made by Stefan Krauss is designed purely to create realistic traffic flows in general, since in most traffic simulations individual movement on the microscopic level is not interesting [6]. In contrast, we aim at designing futuristic autonomous cars with a more fine grained sense of control and most importantly, the ability to violate norms.

There are also pragmatic reasons to propose a SUMO extension rather than altering its code. For example, we can now support multiple versions of SUMO, starting from SUMO 0.20 and upwards. This also makes our extension more accessible to users, since it eliminates the need for users to recompile SUMO before they can use the framework. Our extension is thus usable out of the box.

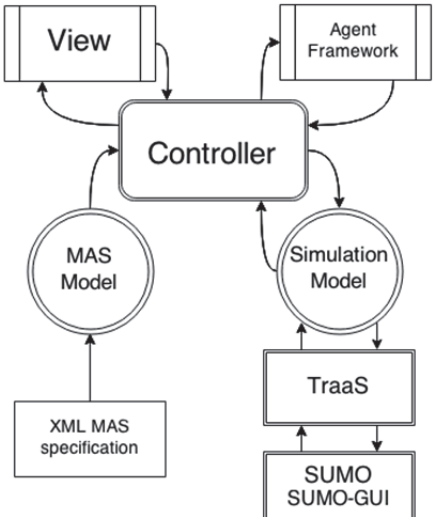


Figure 7-3. The Model-View-Controller structure.

The framework is developed in Java since most norm-based autonomous agent frameworks are written in Java and Java programs are easy to use on a variety of platforms. We have designed the framework using the Model View Controller pattern, as can be seen in Figure 7-3. A multi-agent system can be specified in XML and is converted to a MAS-model. We furthermore use TraaS to retrieve the simulation state of SUMO. These models can be manipulated by the controller software (note that the controller is a control component in the software engineering sense; it is not the same as the conceptual control system for traffic). The agent framework simulates our driver models. The view of the system allows the user to see the state of the MAS side of the simulation. This decomposition allows other researchers to easily create their own front end by changing the view implementation. It is also possible to create a new data format for scenarios by changing the data model implementation, or to change the simulation package by providing a different simulation model implementation. Each part can be changed without the need to change other parts of the framework.

7.3 Application of Norm Based Traffic Control in SUMO

In our use-case scenario we illustrate a fairly simple norm based control system. On the control system side it is calculated when individual vehicles would arrive on the merge point of the traffic streams. The algorithm used is described in [12] and returns an ordering of vehicles. Next, the target velocity is calculated for each individual vehicle that ensures that it a) crosses the merge point at least two seconds after the vehicle that will cross the point before it, and b) the maximum safe velocity is still maintained. In case the main stream road has two lanes, then a vehicle can also instead be obliged to move to the left lane. This happens when target velocity of a vehicle on the main road is below a predefined threshold. The sanctions of violating norms are captured by a low or high fine.

On the agent side of the system we model vehicles as individual agents with a target arrival time. Aside from this, an agent has a personal profile that specifies how desirable it is for the agent to be on time and how undesirable it is for the agent to receive a sanction. This can lead to different behaviors such as an affluent agent in a hurry opting to take a high fine by

increasing its speed, in contrast to an equally hasty but poor agent not being able to afford the fine which then obeys its assigned target speed. Our main goal is to model future traffic on smart roads as norm based multi-agent systems and simulate such traffic scenarios in SUMO. In order to achieve this objective, we extended SUMO to simulate i) norm-aware autonomous cars, and ii) norm based traffic control systems consisting of monitor and control mechanisms.

Scenario details

In Figure 7-4, a schematic representation of the aforementioned ramp merging scenario is given. Triangles are vehicles that travel in the direction towards they point. White vehicles are the vehicles that have not yet received their obligation from the control system. In this case those are vehicles A, B, C and D. On the road there are lane sensors (s_1 to s_5) which can detect the status of vehicles that are residing on them. For the scenario to work correctly, it is necessary that either the sensors are sufficiently long, or the vehicles sufficiently slow, so that no vehicle can pass by undetected. The sensors should also be placed at a distance far enough from the merge point m, so that vehicles have enough time to comply to the obligation before the deadline. There are two important points on the road, point m where the two roads merge, and point e where the vehicles exit the scenario. Distances d_A and d_C are agent's A and C's distances to m, and d_{exit} is the distance from the mergepoint to the exit point, and d_{safe} is the distance between vehicles that is deemed safe, the minimal gap. Ideally A and C traverse d_A and d_C s.t. they arrive at m with a distance $d_{safe}$and can accelerate to their maximum speed within the distance d_{exit}.

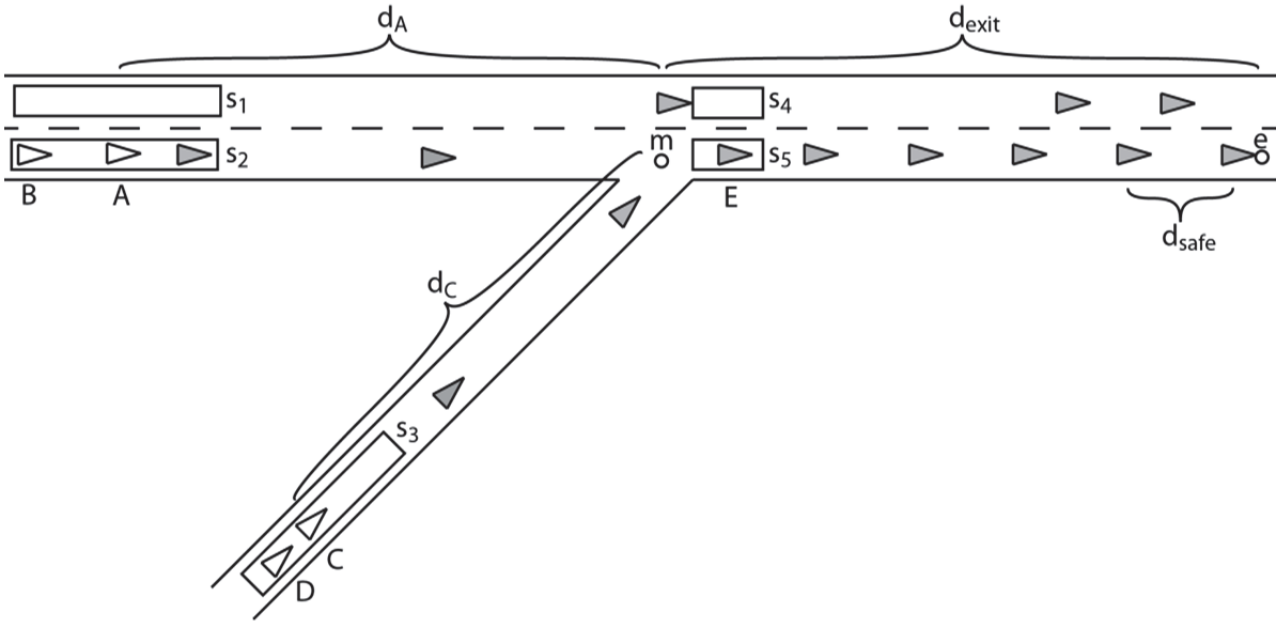


Figure 7-4. Use case with two lanes.

Monitoring happens through interpreting the observations of sensors. In the case of SUMO we use lane detectors that can sense the vehicles that driving along the area they cover. Specifically, each sensor can detect the identity, velocity and position of each vehicle on the sensor's area. Futher parameters such as the maximum velocity, acceleration and deceleration capabilities can be assumed within reasonable margins. The control system uses the sensor data to check given norm schemes whether new norms are instantiated, as well as checking whether instantiated norms are violated. If a new norm is instantiated, then the subjected

vehicle is notified of the obligation that it has to fulfill, and the associated sanction that will be imposed if the obligation is not fulfilled.

The standard traffic rule is that the stream that originates on the main road has priority over the ramp road's stream. However, if the main road is busy, this may lead to large traffic jams on the secondary road. Therefore the organisation uses the traffic data from sensors 1 to 3 calculate how fast vehicles must go so that on the merge point the traffic streams smoothly merge together. More specifically, the organisation tells vehicles passing sensor 1 to stay on the left lane. Sensor 2 and 3 are used by the organisation to coordinate the scheduling of vehicles on the merge point. If a vehicle passing sensor 2 has to slow down too much, i.e. to a velocity less than the threshold, then it receives the obligation to move to the left lane.

In order to not overcomplicate the scenario we decided to simplify some aspects of the control system. The sanction that an agent can receive for violating a norm is modelled by either a low or high fine. In general the set of possible sanctions is S, which for our scenario is: $S = \{fine_{low}, fine_{high}\}$. The set of possible obligations is O. An obligation that a vehicle can receive is an obligation to be on the left or right lane of the main road at a certain target velocity. For instance (l_{right}, v_{10}) is read as the obligation to be on the right lane at 10 m/s. For our scenario the possible obligations are: $O = \{l_{left}, l_{right}\} \times \{v_x | x \in \mathbb{R}\}$. A pair of an obligation with a sanction is called a norm instantiation. The set of all norm instantiations is denoted with $N = O \times S$. For each simulation step the new norm instantiations are created. Recall that the specification of how norms are instantiated is referred to as a norm scheme.

Example Norm Scheme

For our scenario, an organization instantiates a norm scheme into agent specific norms, so that two vehicles cannot arrive on the merge point at the same time, if they comply with the norm. This will cause vehicles to slow down considerably on the main road. If they have to slow down too much then they will be obliged to move to the left lane which is assumed to be more free flowing. The exact explanation of this norm is provided in Appendix A.

We also have another norm scheme that obliges vehicles that enter on the left lane of the main road to stay on their lane at a preset maximum velocity v_{MAX}. We shall illustrate the different aspects of a norm scheme according to this scheme's pseudo code that is given in Table 7-1. We assume that there is a current set of norm instantiations NI, a function sanction that given an agent and fine issues the fine for that agent, and a function read that returns the vehicles on a sensor's area that have not been seen before by that sensor. Though not a forced pattern, we do encourage future users of our extension to use the same code structure as in Table 7-1. Every norm scheme has a specification of when norm instances are created, when they are retracted, and when a sanction should be issued.

The code in Table 7-1 is executed after every simulation tick. We begin with creating new norm instances (lines 0-3). In this case sensor 1 is read. In Figure 7-3 it can be seen that this sensor detects all vehicles that enter the scenario on the left lane of the road. Hence, we give each agent on the left lane the obligation to stay on the left lane, and also obtain a preset maximum velocity to ensure that the flow stays high (line 2). We then continue by checking sensor 4 (lines 4-10). Vehicles with instantiations of this norm that pass sensor 4 fulfilled their obligation to stay on the left lane. However, if their velocity is not optimal, then they get a low fine anyway (lines 6-8). After passing this sensor the vehicles are relieved of their obligation (line 9). The same holds for sensor 5 (line 16). However, if a vehicle passes sensor 5

then it means that it switched to the right lane. Hence, each vehicle that has received an obligation to stay left but passes sensor 5 is fined (line 14).

Table 7-6. Pseudo code for the stay-on-lane norm scheme.

```
0:  L ← read(s_1)
1:  for each agent ∈ L do {
2:          NI ← NI ∪ {(agent, ((l_left, v_MAX), fine_low))}
3:  }
4:  L ← read(s_4)
5:  for each agent ∈ L do {
6:          if (agent, ((l_left, v_MAX), fine_low)) ∈ NI & agent.s < v_MAX then {
7:                  sanction(agent, fine_low)
8:          }
9:          NI ← NI \ {(agent, ((l_left, v_MAX), fine_low))}
10: }
11: L ← read(s_5)
12: for each agent ∈ L do {
13:         if (agent, ((l_left, v_MAX), fine_low)) ∈ NI then {
14:                 sanction(agent, fine_low)
15:         }
16:         NI ← NI \ {(agent, ((l_left, v_MAX), fine_low))}
17: }
```

7.4 Norm-Aware Vehicle

Our goal is to build autonomous norm aware vehicles that operate on smart roads where vehicles' behaviors are automatically monitored, evaluated with respect to some traffic norms, and possibly sanctioned. This means that individual vehicles have their own objectives (e.g., destination, arrival time, travel cost, etc.) and are able to deliberate on how to achieve their objectives. This implies that a vehicle can choose to obey/violate a norm, when this helps with achieving its objectives.

A Norm-Aware Driver Model

In Figure 7-5 the general Sense-Reason-Act cycle of an autonomous vehicle is illustrated. In order to model norm awareness we deviate in our scenario from the standard driver models that SUMO provides [6]. A vehicle perceives its road environment by processing the information it receives from SUMO. That information also includes which obligations are newly instantiated or retracted. Using its knowledge of the the road network, the vehicle then estimates the utility of each of its actions by balancing between the value of achieving its objectives and possible sanctions that will incur if it performs the action. For example, the

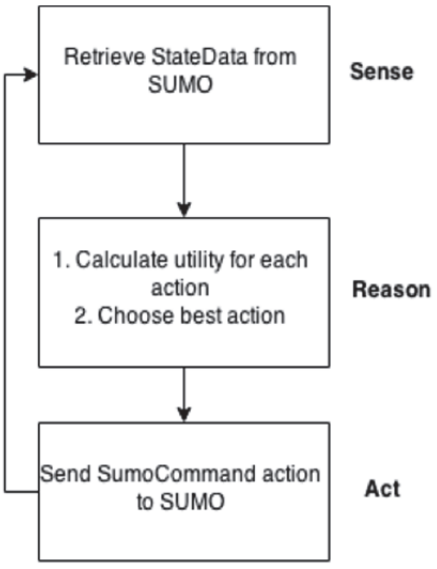


Figure 7-5 The Sense-Reason-Act cycle of an agent

arrival time and additional fine payments are used to calculate the utility for each action. The action with the highest utility is chosen, using a priority based tie-break mechanism for ties. One autonomous vehicle differs from another through its personal profile. A personal profile states not only its goals and current norm norm instances, but also how severe sanctions are for the vehicle and how important it is to arrive on time at the goal location. Since each driver profile is unique, different emergent behavior can be observed between different drivers. For instance, a wealthy driver can at times choose to violate speeding norms, since it can afford the fines. However, a poor driver cannot afford the fines, so it will not break the speed limit.

The required components and concepts for a norm aware agent in our system are:

- A set of actions to choose from.
- A description of the physical state of the driver's vehicle.
- A function that estimates the next physical state given a physical state and an action.
- A personal profile that contains:
    - A goal location and time.
    - A function to calculate how good an expected arrival is compared to the goal time.
    - A function that grades how bad sanctions are.
    - The current norm instances of the agent.
- A function, called the obligation distance measure, that given a physical state of the vehicle and a norm instance, returns how likely it is that the sanction will be imposed upon the driver.
- A utility function that given a possible actions, the current physical state and the personal profile returns the expected utility of the action.
- An action selection function that picks the best action given the current physical state, and personal profile.

Note that the personal profile and the obligation distance measure are not fixed. While it is recommended to let the functions have certain properties (e.g. the grade functions should be monotonic), our framework allows for arbitrary implementations of these functions.

In the next section we give examples of each of these components. Before a simulation tick in SUMO each agent determines what the next best action is by calculating for each action the utility. This is done by determining for each action the next expected physical state and then look at how likely sanctions will be incurred in the future given that next state. If a norm instantiation's sanction is severe, then moving away from its instantiated obligation will decrease the utility. Also if

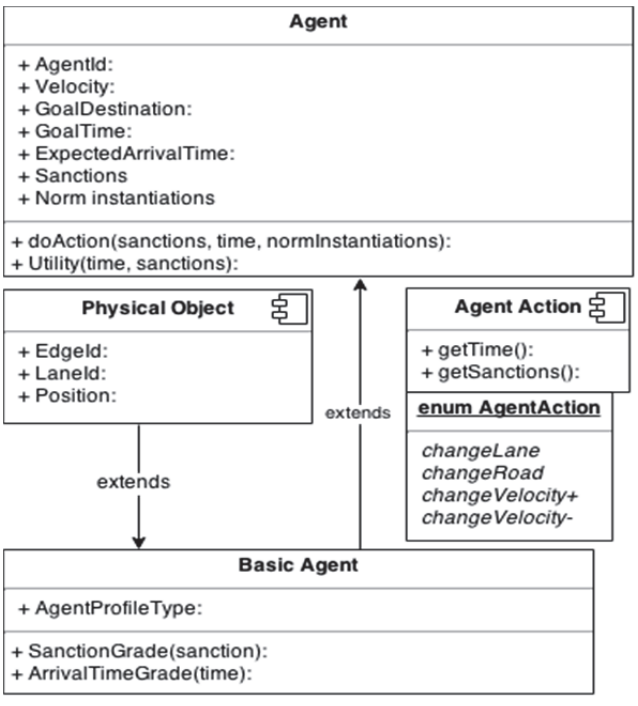


Figure 7-6 An UML overview of the Agent structure

the target arrival time is less likely to be met, then the utility also decreases.

Norm Aware Driver Model for SUMO

An UML overview of the agent structure is shown in Figure 7-6. Agents are an extension of physical objects since they occupy a certain space and have a certain position in the world. A Basic Agent is a prototype AgentProfile. The agent profile is specified by the SactionGrade and ArrivalTimeGrade functions which are the sanction and arrival time appraisal functions discussed in section 1.5.2. In the doAction function the obligation distance measure is used to calculate the utility for each individual action.

In this manner, the agent profile of the agent together with the current state decides what action the agent will choose towards achieving its goal. These design choices were made to model the agents in such a way that new agent profiles can easily be created to model norm-aware human or driverless autonomous vehicles, while still keeping the agents as simple as possible.

With each deliberation cycle agents can pick the an action most suited to their goals and the current state, one of the actions listed by the AgentAction enumeration in Figure 7-6. The displayed *changeRoad* action is not used in the experiments, but is used to change a road in the current route of the agent. In our current implementation this action is used when the agent receives information about traffic jams at locations he still plans to visit. The agent can then plan a new route, make an estimation to check if the new route might be faster, and change its current route. This is one feature that show the pro-active attitude that the TrafficMAS agents possess.

7.5 Application of Norm Aware Driver Models

In this section we shall go into detail how norm awareness in achieved in our example scenario.

Agent Specification

For our application the set of actions are de-/accelerating a certain amount (denoted as a_x) or switching a lane to the left or right (denoted as l_{left}, l_{right}). More specifically, the set of actions A that a vehicle driver can possibly perform are:

$$A = \{a_x | x \in \{0{,}0.01, -0.01, 0.1, -0.1{,}1, -1{,}5, -5{,}10, -10{,}20, -20{,}50, -50\}\} \cup \{l_{left}, l_{right}\}$$

A vehicle's physical state s is specified by $\langle v_{max}, v, l, d\rangle$, where v_{max} is the maximum velocity, v is the current velocity, l is the current lane, and d is the distance from the startpoint of that lane. Finally, B is used to denote the set of all possible physicial states.

An agent's personal profile p is specified by $\langle g_s, g_t, n\rangle$, where $g_s: S \to \mathbb{R}$ is the sanction grading function that returns how important it is to avoid a given sanction from S, $g_t: \mathbb{N} \to \mathbb{R}$ is the arrival time grading function that returns how good it is to exit the scenario at a given time, $n \subseteq N$ are the current norm instantiations. With P we denote the set of all possible personal profiles.

Agents need to reason about the future in order to determine the utility of actions. We assume that there is a given function $f: B \to \mathbb{N}$ that returns given a physical state an expected arrival time. This function reflects for instance the GPS planning tools that vehicles have

available. This function is uniform for all agents, but can be parameterized in the future in order to make more optimistic/pessimistic agents. We also assume agents have a local action effect function $e: B \times A \to B$ that returns given a physical state and an action the next expected physical state. For instance: if the current velocity is 20 m/s and a vehicle accelerates by 5 as an action, then it expects for the next simulation tick to be at 25 m/s if this is possible within its acceleration capabilities.

Due to physical contraints, it may not be the case to adhere to obligations immediately after they are received. Hence, an agent must be able to deliberate whether an action will bring it closer or further away from fulfilling an obligation. To this end they use an obligation distance measure $\delta: B \times O \to \mathbb{R}$ which returns given a physical state and an obligation, a positive expectancy of whether the agent can fulfill the obligation in time before it is sanctioned. The precise definition of this function depends on the possible obligations and agent specifics in an application. However, a high distance should mean that it is likely that the sanction will be incurred in the future whereas a distance of zero should indicate that the current state fulfills the obligation.

In the case of our scenario the factors that an agent considers to compute δ are the time it takes to fulfill the obligation when doing so as soon as possible (based on the vehicles acceleration, local traffic data etc), the current time and the latest expected time that the sanction will be issued if the obligation is not met. We have implemented vehicles in such a way that they expect that the control system will check whether an obligation is fulfilled somewhere between the current time t and the current expected exit time t_{exit} for the vehicle. The minimal amount of steps needed to adhere to the norm is denoted δ_t. For instance, if a vehicle at time step t is being instructed to drive 25 m/s, and can accelerate to this speed in minimally 3 time units, then $\delta_t = 3$. If we need zero steps to adhere to the norm, the distance is zero. Otherwise the distance proportionally moves to 1 given the current time. If the control system will check obligation fulfillment before the agent can achieve adherence (i.e. $t_{exit} - t < \delta_t$), then δ should be 1.

The exact function given the current time t, the expected exit time t_{exit} and the minimal number of steps needed for adherence δ_t is:

$$\delta\big(\langle v_{max}, v, l, d\rangle, (l_o, v_0)\big) = \frac{\min(\delta_t, t_{exit} - t)}{t_{exit} - t}$$

Note that this means that the agents expect the control system to issue a sanction if an obligation is not met between now and $t_{exit} - t$ time units after that. This distance measure can be modified easily in our framework.

Agents use an action selection function $\alpha: B \times P \to A$ that given a physical state and agent profile returns an action. This selection is based on the utilities of actions. For each action it is simulated what the new expected arrival time and physical state will be. The result is used to calculate how good a new state is given the distance to fulfilling obligations (and thereby avoiding sanctions) and also how good the new arrival time is. The utility function $u: B \times P \times A \to \mathbb{R}$ is:

$$u(b, \langle g_s, g_t, n\rangle, a) = g_t\big(f(b')\big) + \sum_{(o,s)\in n} \big(\delta(b', o) \cdot g_s(s)\big)$$

where $b' = e(b, a)$.

Example Utility Calculations

For example, suppose we have two agents, a poor one and an affluent one in an identical situation. They currently drive 20m/s, their maximum speed is 30 m/s, they can accelerate or decelerate with 10m/s and their travel distance is 1080 meters. Both are in a hurry, so their g_t is defined as $g_t(time) = bestTime/time$. Here, $bestTime$ is defined by the minimal travel time, i.e. the time it would take the agents to travel the distance if they could go their maximum speed all the time. In this case, $bestTime = \frac{1080}{30} = 36.$

Table 7-1. Example of a poor and affluent agent.

	a_0	a_{10}	a_{-10}
Speed after action	20m/s	30m/s	10 m/s
Norm speed	10 m/s	10 m/s	10m/s
Exit time with this speed	54	36	108
g_t	0,67	1,00	0,33
steps needed to oblige to norm	1	2	0
δ	0,02	0,06	0,00
u poor agent	**0,30**	**-0,11**	**0,33**
u rich agent	**0,66**	**0.99**	**0.33**

However, the road the agents travel on has a speed norm, with the maximum speed being 10 m/s. Not obliging to this norm gives a $fine_{high}$. The poor agent cannot afford this fine, so it has $g_s(fine_{High}) = -20.$ The affluent agent can easily afford this fine, so it has $g_s(fine_{High}) = -0.2.$ Suppose for this example, that agents only can take the actions a_0, a_{10} and a_{-10}. In Table 7-2, we see the utilities for each of these actions and both agents. Here we see that the highest rewarded action for the poor agent is to indeed oblige to the norm, since it cannot afford the fine. The affluent agent is in a position to not decrease its speed to oblige to the norm, since it can afford the fine. In fact, it will even increase its speed, since it then maximizes its time grade, yielding a higher utility.

7.6 Experimental Evaluation

We tested the performance of our normative agent based traffic approach using three experiments. In each of these experiments variations of the ramp-merging scenario as explained in section 7.3 is used. The first experiment considers a ramp-merging scenario where the main road consists of a single lane, while on the second and third experiment the main road has two lanes. In the sec0nd experiment this second lane is accessible for all agents, but in the third experiment this lane is marked as an "emergency only" lane.

The experiments were set up as follows. Each experiment has a length of one hour (3600 ticks). The spawn rate shown in the tables of the experiments is defined as the chance of an agent spawning every tick. If there is not enough room to spawn an agent at a certain time, then SUMO puts the agent on hold and spawns it at the earliest possible time there is space avaliable. The maximum speed at the merge point, $\mathbf{v_{max}}$,was set to 80 km/h. Furhermore, all three experiments consists of comparing two scenario's, both are run one hundred times. The values displayed in the tables are the averages over those hundred runs. The Throughput is

defined as the number of vehicles leaving the simulation every tick. Average speed is the average speed over all runs in m/s, and finally the Average gap is the average distance between two cars in meters. Also defined for each experiment is the maximum (expected) throughput, this is the expected throughput if each vehicle could keep driving its maximum speed thoughout the scenario and can be calculated by the following formula: $\mathrm{throughput}_{\max} = 60 * \mathrm{p}$, where p is the total chance of a car entering the simulation on that tick. Since SUMO does not spawn vehicles when space is too limited, and this is the case when traffic jams occur, we believe a significant difference in throughput is a sufficient validation of improvement.

Experiment 1: SUMO and TrafficMAS

The first experiment illustrates the distinction in behavior between the default SUMO vehicles and the norm-aware agents implemented in the TrafficMAS extension. A single norm based traffic control system observes the vehicles in the simulation and communicates tailored norm instantiations to each vehicle. In this experiment a norm instantiation is simplified to only a target velocity since there is no choice of lanes on the main road. The expected result is that these personalized norms will result in a higher average velocity and a better throughput of vehicles since traffic jams will be prevented. In this scenario a classic ramp-merging situation is implemented, where both the main road and ramp consist of a single lane. The spawn rate of the vehicle input stream will be slightly higher on the main road to resemble a realistic traffic situation. In the *TrafficMAS* scenario three sensors are placed on the road, one on the main road, one on the ramp and a control sensor on the output road. The traffic control system can only communicate (instantiate norms, remove norms or apply sanctions) when the vehicles are driving on said sensors. In the *SUMO* scenario the main road has priority over the ramp, comparable to an authentic merging situation.

Table 7-3: results for experiment 1.

	SUMO	**TrafficMAS**
Main road Spawnrate	20%	20%
Ramp Spawnrate	15%	15%
Throughput	16,16	21,01
Average Speed	3	20,97
Average Gap	13,81	101,821
Max throughput	21	21
Throughput %	76,95%	100,05%

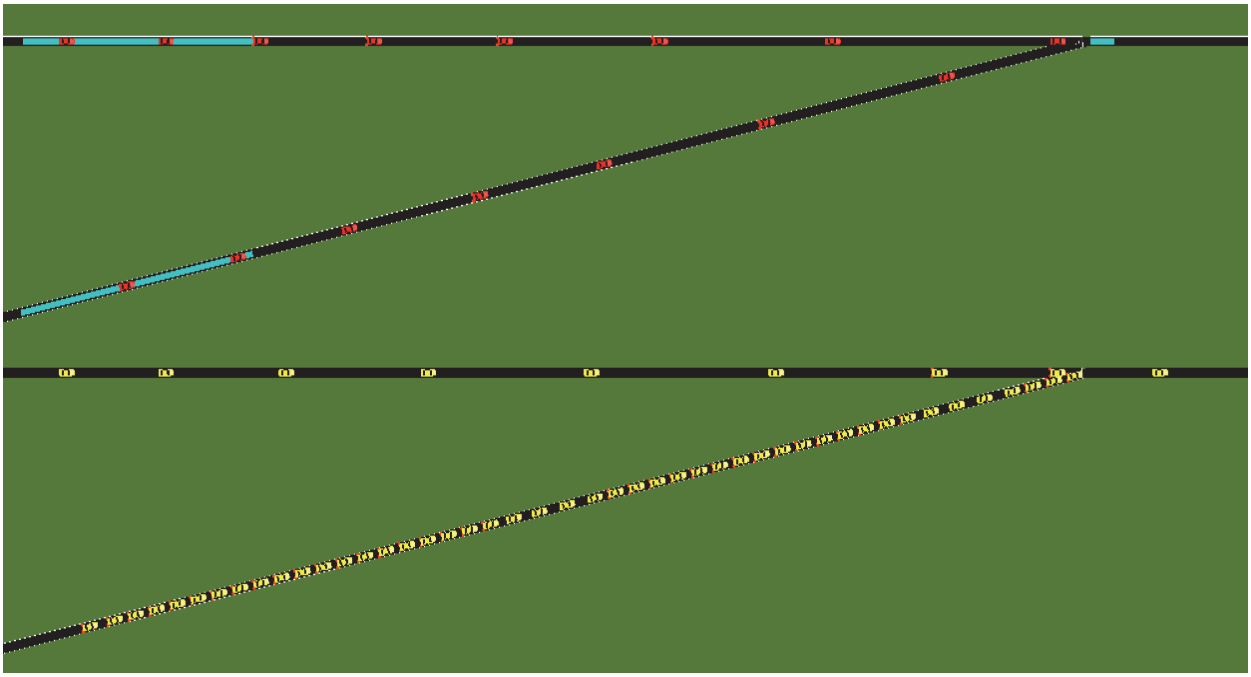


Figure 7-7: Screenshot depicting the difference in performance in experiment 1. The top scenario uses our framework and merge norm. The bottom scenario uses the default SUMO driver models.

As is clear from the results in table 7-2, there is an increase in both throughput, average speed and the average amount of space between the vehicles. This is the case since in the SUMO scenario a traffic jam instantly forms on the ramp, because of the relatively high density of cars on the main road. These results confirm our expectation of coordination by a norm-based traffic control system improving on classic ramp-merging scenarios. Note that the *throughput %* value exceeds a hundred percent, this is possible because the spawnrate is probability based and thus can exceed the maximum expected throughput.

Experiment 2: Simple norms and Advanced Norms

Table 7-4: results for experiment 2

	SingleNorm	**AdvancedNorm**
Main road Spawnrate	30%	30%
Ramp Spawnrate	20%	20%
Throughput	20,38	29,91
Average Speed	3,31	14,91
Average Gap	14,2	61,49
Max throughput	30	30
Max throughput %	67,93%	99,7%

This experiment is a comparison which demonstrates the distinction between norm-based traffic control systems using simple or advanced norms. Each scenario is observed by a traffic control system, where the control system in the *SingleNorm* scenario observes and controls the same norm as in experiment 1. In the *AdvancedNorm* scenario the control system effectuates a more advanced norm, advanced in the sense that the obligation that the agent receives can consist of different goals instead of just a target velocity. In this case the other obligation that can be issued is for the agent to change lanes in order to relieve the rightmost lane traffic and prevent congestion. This lane change obligation will be given to an agent when its calculated velocity on the mergepoint is below a certain threshold. For this experiment the threshold was set to $\frac{1}{2}v_{max}$. Our expectation is that in this multi lane scenario, the control system with the

advanced norm can successfully cope with a higher input stream of vehicles; the traffic control system which effectuates the simple norm, will not.

The setup for the *SingleNorm* scenario is almost an exact copy of the *TrafficMAS* scenario in experiment 1, the two distinctions are that the main road has two lanes instead of one and the input stream of vehicles of both roads are increased. The *AdvancedNorm* scenario also implements extra sensors on the sec0nd lane, but is exactly the same in every other aspect.

As can be observed from the results, the simple norm cannot cope with the increased spawn rate of vehicles in this scenario. The average speed is has dimished severly, as well as the average gap between vehicles. This means congestion is abundant in the *SimpleNorm* scenario. However, the *AdvancedNorm* seems to cope very well with the increased input stream of vehicles. In this scenario the throughput approximates the maximum expected throughput by a factor 0.3%, which indicates that the vehicles move throughout the simulation without much congestion.

Experiment 3: Sanction severity

The third experiment illustrates that agents are able to reason about norms. Experiment 1 has shown that agents are norm-aware. However, TrafficMAS agents also have the capabilities to not comply with certain obligations if these are not of significant importance to them. In this experiment the leftmost lane is an emergency lane, reserved for certain traffic in order to help with accidents and other emergencies. Therefore regular agents will get sanctioned if caught driving on this lane. Since this lane remains mostly empty, this is a viable option for agents who greatly value a faster arrival time and are in a financial position which makes them willing to take a fine. We expect that the more affluent agents will choose to take a sanction and win some time, resulting in distinct behavior between the two groups of agents.

Table 7-5: results for experiment 3

	Poor Agents	**Affluent Agents**
Main road Spawnrate	20%	20%
Ramp Spawnrate	15%	15%
Throughput	20,48	20,88
Average Speed	12,95	14,42
Average Gap	48,37	69,29
Sanctions	0	133,12

This experiment is set up in the same way as experiment 2, with the exception that the leftmost lane is reserved for emergencies and the spawn rates are lowered because of this. In the *Poor Agents* scenario, the input stream consists of agents who are hasty, but in a substandard financial position. The *Affluent Agents* scenario spawns agents who care about being sanctioned, but are willing to take a fine if by doing so they can arrive closer to their goal arrival time with a significant amount of time.

With this experiment, the difference in throughput, average speed and gap is much smaller, and not significant enough to lead to any conclusions about improvement. However, a significant distinction in the number of sanctions can be seen. This indicates a difference in behavior between the groups of agents. On average about 133 affluent agents decide to drive on the emergency lane in an hour of simulation. This shows a clear difference in behavior from the poor agents, who never decided to change lanes, since no sanctions are given to them throughout the 100 runs.

7.7 Contributions and comparison to other work

The contributions of our work are as follows. First of all, we have created a lightweight framework for autonomous norm-aware vehicles and norm-based traffic control system on top of SUMO. This framework is easy to extend with different types of driver profiles. It also allows for the easy usage of a different simulation package. Secondly, with our framework, we have created the possibility to conduct traffic experiments and measuring the impact of a norm-based traffic control system. Finally, we have improved the TraaS performance, yielding a performance increase of up to four times over the original TraaS library in certain scenarios.

Our approach has some similarity with Baines et al [3] since they employ autonomous agents, and also use norms. However, Baines et al concentrate on agents' internal architecture and the communication between agents. Our driver model is deliberately kept simple in order to focus on the interaction between traffic control systems on the one hand, and between agents and traffic control systems on the other hand. Finally, our framework supports decentralized traffic control system while Baines et al. focus on a single, all knowing, institution.

7.8 Future Work

Our framework could be extended in a number of ways. First of all, one could add support for contrary to duty norms. Contrary to duty norms consist of a hierachy of norms. An agent should comply to all hierachies, but if it doesn't comply to the first norm level (and thus incurring a sanction), it should at least comply to the second norm level, or get an even higher sanction. An example is the norm "You shouldn't break the speed limit, but if you do, you should drive on the leftmost lane."

Another extension would be to include communicating decentralized traffic control, as described in [8] and [9]. In our current framework, multiple control systems are possible (though not shown in this paper). However, they cannot yet communicate information about vehicles and sanctions with each other. If this infrastructure is created, more complex norms and scenario's can be implemented.

7.9 Conclusion

Our goal was to create a traffic simulation Multi-Agent System where agents should generally follow traffic regulations, yet are able to ignore these regulations in certain circumstances without implementing hard constraints on the agents themselves. We used norm based control systems, since they deal well with these kinds of situations.

Our work is an extension to SUMO. It features a norm based traffic control system which monitors and possibly sanctions the vehicles, as well as deliberative pro-active drivers which make autonomous decisions according to their goal and received sanctions. The extension features i) driver profiles which model different types of behavior, ii) traffic control systems and norms to control these agents and iii) an easy way to add new driver profiles, control systems and norms. This plug and play extension to SUMO can serve as a testing suite for all experiments concerning normative systems.

We showed in our experiments that the performance of normative systems is be better than the default behavior in a ramp-merge scenario. Furthermore, we showed that complex norms allow for more fine grained steering of behavior in complicated scenarios. Finally, we

illustrated the autonomy of agents, by demonstrating a different in behavior between poor and affluent agents.

7.10 References

[1] Alechina, N., Dastani, M., Logan, B.: Programming Norm-Aware Agents. In proc. of the 11th int. conf. on autonomous agents and multi-agent systems, vol. 2, 1057-1064 (2012)

[2] Alechina, N., Dastani, M., Logan, B.: Reasoning About Normative Update. In proc. of the 23rd int. joint conf. on A.I., 20-26, AAAI Press (2013)

[3] Baines, V., Padget, J.: A Situational Awareness Approach to Intelligent Vehicle Agents. In SUMO2014 – Modeling Mobility with Open Data, 1-17 (2014)

[4] Boella, G., Van Der Torre, L.: Introduction to Normative Multi-Agent Systems. Computational and Mathematical Organization Theory, vol.12, 71-79 (2006)

[5] Hübner, J., Boissier, O., Bordini, R.: A Normative Programming Language for Multi-Agent Organisations. Annals of Mathematics and A.I., vol 62, 27-53 (2011)

[6] Krauss, S., Wagner, P., Gawron, C.: Metastable States in a Microscopic Model of Traffic Flow. Physicial Review E, vol. 55, 5597-5602 (1997)

[7] Meneguzzi, F., Vasconcelos, W., Oren, N., Luck, M: Nu-BDI: Norm-aware BDI Agents. In proc. Of the 10th Europ. Workshop on Multi-Agent Systems (2012)

[8] Testerink, B., Dastani, M., Meyer, J-.J.: Norms in Distributed Organizations. In AAMAS'13: Workshop on Coordination, Organization, Institutions and Norms in Agent Systems (2013)

[9] Testerink, B., Dastani, M., Meyer, J-.J.: Norm Monitoring Through Observation Sharing. In A. Herzig, E. Lorini (eds), Proc. Of the Europ. Conf. on Social Intel., 291-304 (2014)

[10] Tinnemeier, N., Dastani, M., Meyer, J-J., van der Torre, L.: Programming Normative Artifacts with Declarative Obligations and Prohibitions. In IEEE/WIC/ACM Int. Joint. Conf. on Web Intel. and Intel. Agent Technologies, vol. 2, 145-152 (2009).

[11] Baskar, L. D., De Schutter, B., Hellendoorn, J., & Papp, Z.: Traffic control and intelligent vehicle highway systems: a survey. IET Intelligent Transport Systems, 5(1), 38-52 (2011).

[12] Wang, Ziyuan, Lars Kulik, and Kotagiri Ramamohanarao. "Proactive traffic merging strategies for sensor-enabled cars." Proceedings of the fourth ACM international workshop on Vehicular ad hoc networks. ACM, 2007

[13] Baines, Vincent, and Julian Padget. "On the benefit of collective norms for autonomous vehicles." Proceedings of 8th International Workshop on Agents in Traffic and Transportation. 2014.

[14] http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/tesla-working-towards-90-autonomous-car-within-three-years

[15] J. Markoff. Google cars drive themselves, in traffic. http://www.nytimes.com/2010/10/10/science/10google.html, October 2010.

[16] Abdelkader, G.: "Requirements for achieving software agents autonomy and defining their responsibility." Proc. Autonomy Workshop at AAMAS 2003. Vol. 236. 2003.

[17] Dastani, Mehdi, Davide Grossi, John-Jules Ch Meyer, and Nick Tinnemeier. "Normative multi-agent programs and their logics." In Knowledge Representation for Agents and Multi-Agent Systems, pp. 16-31. Springer Berlin Heidelberg, 2009.

7.11 Appendix A: Merge Norm Scheme

For the merge norm scheme we use the same pseudo code structure (Table 7-2) as for the stay-on-lane norm scheme. As in the other norm scheme we begin with the instantiation of the norm (lines 0-8). Initially we read sensors 1 and 3 and merge the readings according to [12] (line 0). The result is an ordered list of agents, which, if they continue as they are, will arrive at the merge point in the same order. We maintain a global variable t_{free} that indicates the next moment in time that the merge point is free. With $optimalVelocity$ we calculate the optimal speed for an agent s.t. it will arrive at t_{free} on the merge point plus some safe margin, or later if the agent cannot make it in time physically (line 2). If the agent is at the right lane of the main road and the optimal velocity is below a predefined threshold, then it is obliged to move to the left lane (line 4), otherwise it is obliged to adapt its velocity to the optimal velocity and pass the merge point on the right lane (line 6). An agent is sanctioned if it is not passing the merge point on the correct lane (lines 12-13, and 19-20). Otherwise an agent can also be sanctioned if it did not achieve its predetermined velocity (lines 23-24).

Table 7-6. Pseudo code for the merge norm scheme.

```
0:  L ← merge(read(s1), read(s3))
1:  for each agent ∈ L do {
2:          s ← optimalVelocity(agent, t_free)
3:          if agent.lane = l_right & s < v_threshold then {
4:                  NI ← NI ∪ {(agent, ((l_left, v_MAX), fine_high))}
5:          } else {
6:                  NI ← NI ∪ {(agent, ((l_right, s), fine_high))}
7:          }
8:  }
9:  L ← read(s4)
10: for each agent ∈ L do {
11:         NI ← NI \ {(agent, ((l_left, v_MAX), fine_high))}
12:         if (agent, ((l_right, s), fine_high)) ∈ NI then {
13:                 sanction(agent, fine_low)
14:                 NI ← NI \ {(agent, ((l_right, s), fine_high))}
15:         }
16: }
17: L ← read(s5)
18: for each agent ∈ L do {
19:         if (agent, ((l_left, v_MAX), fine_high)) ∈ NI then {
20:                 sanction(agent, fine_high)
21:                 NI ← NI \ {(agent, ((l_left, v_MAX), fine_high))}
22:         } else if (agent, ((l_right, s), fine_high)) ∈ NI then {
23:                 if s ≠ agent.v then {
24:                         sanction(agent, fine_high)
25:                 }
26:                 NI ← NI \ {(agent, ((l_right, s), fine_high))}
27:         }
28: }
```

8 Extensions for logistic and public transport in SUMO

Andreas Kendziorra and Melanie Weber;
German Aerospace Center , Rutherfordstraße 2, 12489 Berlin, Germany

{Andreas.Kendziorra, Melanie.Weber}@DLR.de

Abstract

Disasters or major events affect the efficiency of passenger and freight transportation. The project VABENE++ considers the question what happens when the road network is disturbed due such an event. Within this project, the German Aerospace Center developed a decision support tool that provides consolidated information and operation recommendations regarding individual motor car traffic based on traffic simulation performed by SUMO. Recently, it was aimed to realise simulation scenarios in which incidents impairing the road network occur, and that focus on multimodal transportation systems. To enable this, the implementation of public transport and logistics in SUMO were extended. The present paper presents these extensions in detail and outlines its usefulness in examples.

Keywords: SUMO, Logistic, Transport, Public Transport, Railway

8.1 Introduction

Transport and traffic is one of the identified critical infrastructures in Germany [1]. A disturbance of the traffic network can have serious influences on passenger transportation, freight transportation and the supply of necessary goods and services. These disturbances can be manifold. They may be plannable, like major events, or unpredictable, like accidents, natural disasters (e.g. floods, earthquakes) or from human malevolence, such as acts of sabotage, terrorism or war. To support decision makers, like authorities and emergency forces, in such critical situation, the decision support system *EmerT Portal* was developed within the project VABENE [2]. Until recently, the system considered only motorised individual traffic. Due to the fact that not only this means of transport is affected by an extraordinary occurrence, this system was expanded by more means of transport. One part of this system is the traffic simulation software SUMO [3]. Public transport (bus, tram and train) and pedestrians were already a part of SUMO prior to the expansion, however in a basic fashion. The implementation of logistic transport is novel in SUMO. To set up a simulation scenario with a multimodal transportation system, SUMO had to be extended.
The implemented extensions will be presented in this paper in the following order. In Section 8-8.2, the extensions regarding public transport will be presented. The implementation of the logistic concept will be explained in Section 8-8.3. In Section 8-8.4, some examples using the presented extensions will be shown. Finally, the conclusions and an outlook will be given in Section 8-8.5.

8.2 Public transport Extension

8.2.1 Existing concept

So far, simulating public transport within SUMO is possible, however, with limited capability (see [1, 2]). The most necessary concepts already implemented for this kind of simulations are *persons*, *vehicles* and *stops*.

Persons are objects that can *walk*, use a vehicle (*ride*) and *stop*. Each person must have a plan which is a sequence of these stages. For walking, a person is following a given sequence of edges. Thereby, the walking behavior depends on the chosen *pedestrian model*. The most advanced pedestrian model implemented in SUMO so far is the model "striping"; a 2D-model that enables walking side by side on sidewalks and includes a collision avoidance algorithm for persons that walk towards each other [3]. For a ride, a person has a list of *lines* (vehicles can be assigned to lines which can be used particularly for buses, trams and trains) and *ids* of vehicles to use. At the beginning of a riding stage, a person is positioned at an edge. Whenever a vehicle that corresponds to a line or an id of the given list stops at this edge, the person will board the vehicle. When the transporting vehicle stops at the destination edge of the ride, the person will leave the vehicle and will proceed with the next step of its plan. Stopping can be used to simulate activities such as working, shopping or doing sports. For that, a duration or a fixed end time, as well as a lane, have to be defined. For example:

```
<routes>
   <person id="1" depart="0">
      <walk from="edge1" to="edge2" departPos="10.0" arrivalPos="20.0"/>
      <ride from="edge2" to="edge3" lines="a"/>
      <walk from="edge3" to="edge4" arrivalPos="30"/>
      <stop lane="edge4_0" duration="20" startPos="40"/>
      <ride from="edge4" to="edge5" lines="b"/>
   </person>
</routes>
```

Vehicles have the capability to transport persons. For that, there exist no restrictions on how many persons a vehicle can transport. Moreover, there is no duration for a person to board a vehicle and any number of persons can board a vehicle in one time step. Also, there are no requirements regarding the distance between the vehicle and the boarding person. The only requirement is that both have to be positioned at the same edge.

There exists a very particular procedure of departure for vehicles, called *depart triggered*. A depart triggered vehicle has no depart time but will depart when a person boards the vehicle. This can be used to simulate parking vehicles.

As stated above, vehicles can be assigned to lines. This is in particular useful for buses, trams and trains. For example, buses that have the same route can be assigned to one line. Therefore, one can tell a person to use a bus of a certain line without choosing a particular bus. Moreover, it is possible to define *flows* of vehicles. That means, vehicles that have the same attributes, except their id and their depart time, are inserted periodically with a fixed frequency into the simulation. In particular, bus and tram lines can be defined via flows.

As already mentioned, persons can have stops as stages of their plan. However, stops can be used for vehicles and routes in general as well. In these cases, one can define further attributes. Amongst others, the start and end position at the lane can be defined. That means, a vehicle will stop within this interval. A more advanced concept of stops is a *bus stop*. The difference is a more elaborated approaching behavior of the vehicles (mostly buses) towards the bus stop.

8.2.2 Person capacity and person number Extension

To resolve the unrealistic fact that a vehicle can transport an unlimited number of persons, the parameters *person capacity* and *person number* were introduced. Person capacity is a vehicle type parameter that specifies how many persons (excluding an autonomous driver) a vehicle can transport. For every vehicle class, there exists a default value for the person capacity (see [7] for a list of the default values), but it can also be set in the definition of a vehicle type (e.g. <type vClass="passenger" id="pkw1" personCapacity="4"/>). The person number states how many persons (excluding an autonomous driver) a vehicle is actually transporting. If the person number of a vehicle is equal to its person capacity, no further person can board this vehicle.

8.2.3 Boarding behavior Extension

Some modifications regarding the boarding behavior of persons were implemented. As already mentioned, a person can board a vehicle only if the person number of the vehicle is smaller than its person capacity. Moreover, some restriction about the distance of the vehicle and the boarding person were incorporated. If a vehicle is stopping at a stop and a person wants to board the vehicle, the person's position has to lie within the stop. That means, the person's position on the lane has to be larger than the start position and smaller than the end position of the stop. For depart triggered vehicles, a person positioned outside of the stop of the vehicle can still board the vehicle if the person's position has a distance of at most 10 meters to the vehicle.

Another enhancement is the implementation of the vehicle type parameter *boarding duration*. This parameter states how long it takes one person to board the vehicle. Only one person can board the vehicle at a time. Therefore, if n persons want to board one vehicle for an $n \in \mathbb{N}$, and $t \in \mathbb{R}$ is the boarding duration of the vehicle, the time required to let all persons board the vehicle equals $n \cdot t$. For example, if 12 persons waiting at a bus stop want to board a bus whose boarding duration equals 0.5 seconds, it takes 6 seconds until the last person boarded the vehicle. If the duration of the boarding of all persons exceeds the stop duration of the vehicle, the stop duration will be extended by the necessary amount of time.

Due to the changes regarding the boarding behavior and the implementation of the parameters person capacity and person number, scenarios using public transport can be simulated more realistically. For example, bottlenecks due to low capacities of vehicles (e.g. buses and trams) or extended travel times due to long boarding durations can be considered or may be identified in simulations.

8.3 Logistic Extension

A concept for freight and goods was implemented to enable the simulation of goods traffic. This concept consists basically of the new objects called *containers* which represent goods of all kinds. For example, one can represent an ISO container, a tank container, an arbitrary amount of bulk material, an arbitrary amount of animals etc. as a container.

The concept of containers is very similar to the one of persons. Containers are objects that can be transported by a vehicle (*transport*), *stop* and that can be transhipped between two places (*tranship*), e.g. to simulate the transhipping of a container from a rail station to an adjacent harbor (which can be represented as a stop at a waterway). The mode *transport* is equivalent to the mode *ride* for persons, and *stop* is equivalent to *stop* for persons. *Tranship* defines a direct transhipping of containers between two points. Thereby, containers do not move along edges. They move in a straight line with constant velocity, no matter if the line is crossing buildings or anything else (see Figure 8-12). Therefore, the time required for a tranship-stage depends on the Euclidean distance between the two points and the chosen velocity (default velocity is 5 km/h).

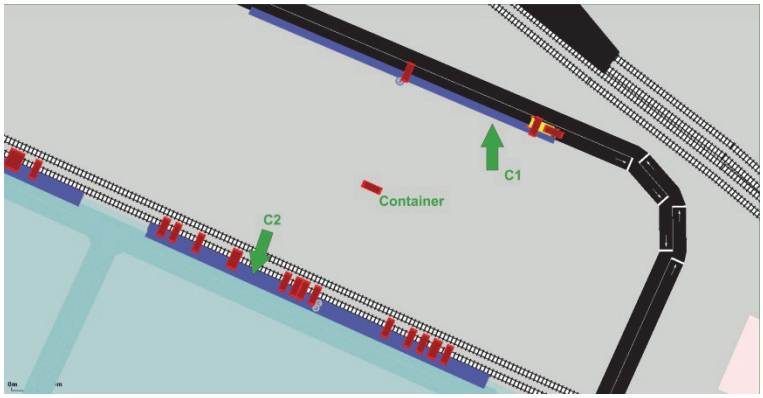


Figure 8-12. A container gets transhipped from container stop C1 to container stop C2; thereby, the container moves straight from its depart position on C1 to its arrival position on C2

As with the concept of persons, containers must have a plan which is a sequence of these stages. For example:

```
<routes>
   <vType id="DEFAULT_VEHTYPE" sigma="0" containerCapacity="1"/>
   <container id="container0" depart="0">
        <tranship from="edge1" to="edge2" departPos="80" arrivalPos="55"/>
        <transport from="edge2" to="edge3" lines="train0"/>
        <tranship from="edge3" to="edge4" arrivalPos="30"/>
        <stop lane="edge4_0" duration="20" startPos="40"/>
        <transport from="edge4" to="edge5" lines="truck0"/>
   </container>
</routes>
```

A complete description of all available parameters of the three stages of containers can be found at [8].

Similarly to the concept of bus stops, *container stops* were introduced at which containers can be loaded onto or unloaded from a vehicle. Vehicles use the same advanced approaching

behavior at container stops as at bus stops. They can be used to simulate transhipment stations, harbors and other places for transhipping and storing containers/goods.

To enable a realistic loading behavior, analogously to the new boarding behavior of persons, the parameters *container capacity*, *container number* und *loading duration*, which correspond to the parameters *person capacity*, *person number* and *boarding duration* for persons, were introduced.

8.4 Examples

8.4.4 Public transport

A scenario involving persons and public transport was used to test the enhancements regarding public transport. More precisely, it was examined if unfavourable relations between the amount of people using public transport and the capacity of the public transport system affect the state of the traffic. As expected, one could find situations in which bus stops get crowded due to low frequencies of bus lines such that the busses cannot transport the amount of people intending to ride a bus of this line (see Figure 8-13). This results clearly in longer travel times for persons.

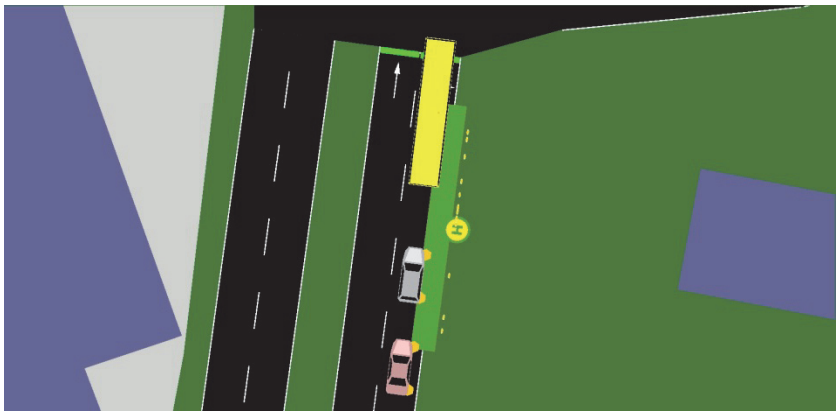


Figure 8-13. A fully occupied bus leaving people behind at a bus station due to a too low capacity of the bus line with respect to the amount of people; travel time of people left behind extends accordingly

Other expected consequences are congestions due to blocking busses at bus station. This occurs when many people board a bus such that the bus blocks the corresponding lane for a long time. In Figure 8-14, a situation can be seen at which 80 people board a bus with a boarding duration of 0.5 seconds. Consequently, the bus blocks the right lane for 40 seconds, which results in congestion on the right lane, as all cars intending to turn right are blocked.

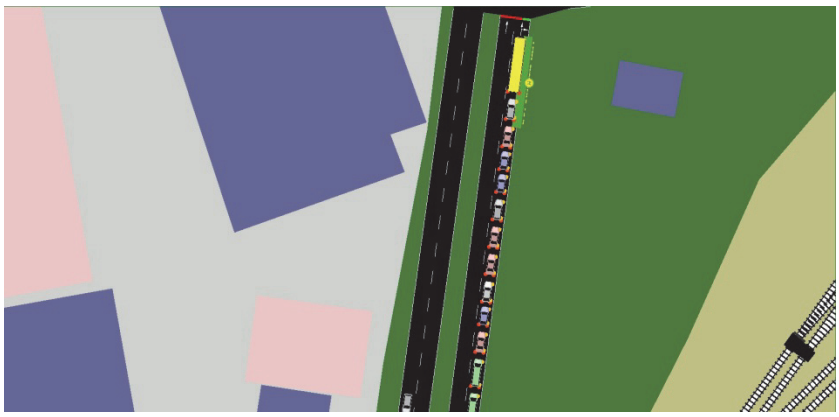


Figure 8-14. 80 people boarding a bus cause the bus to block all right turning vehicles for 40 seconds, resulting in a congestion on the right lane

8.4.5 Logistics

A scenario was developed to test and demonstrate the functionalities of the logistics extensions. The scenario considers the goods traffic around the harbor of Brunswick, Germany. Goods are transhipped, stored and transported by trains, ships and trucks within the scenario. An overview of the harbor area of the scenario can be seen in Figure 8-15.

The harbor consists of seven landing stages, and for each landing stage there exists a corresponding goods station for trains, as well as a container stop for trucks next to the road. In this scenario, almost all the goods are transported by a vehicle (truck, ship or train) to the harbor area, where the container gets transhipped to another container stop, gets stored for a while and finally transported by another vehicle (see Figure 8-16 and Figure 8-17).

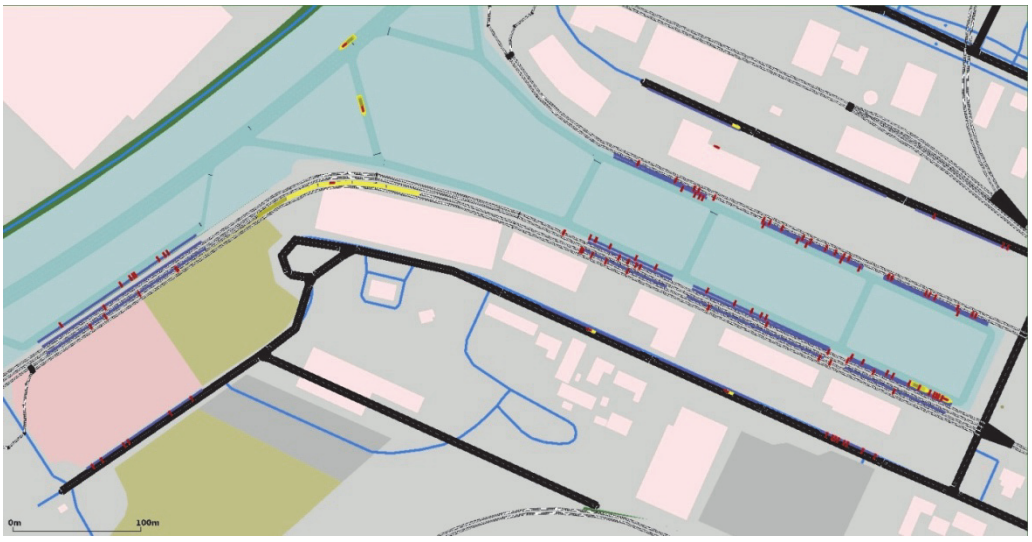


Figure 8-15. The harbor of Brunswick in the SUMO-simulation; containers are displayed in red, vehicles in yellow and container stops in blue

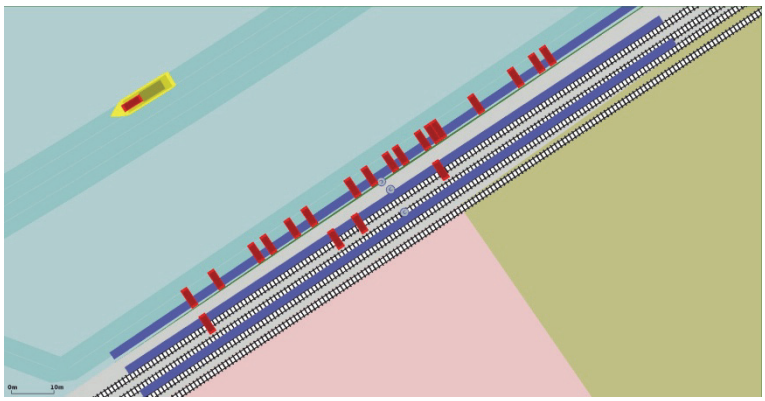


Figure 8-16. Stored containers at a landing stage (container stop) of the harbor

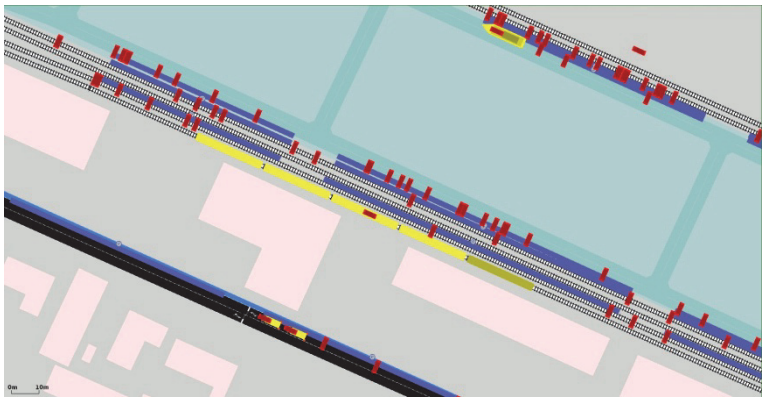


Figure 8-17. Containers get loaded/unloaded onto/from ships, trains and trucks

8.5 Conclusion and discussion

It was aimed to expand the existing public transport concept and to implement a first version of a logistic transport concept in SUMO to create the possibility to set up simulation scenarios with a multimodal transportation system. The enhancements allow simulations of public transport with a realistic boarding behavior. Buses block the traffic when many people to board them, and people also wait for the next bus when the current one is fully occupied. In addition, major events and disaster involving persons can be better simulated. For example, in the case of a major accident, replacement buses are often inserted. With the help of the simulation, one can identify bottlenecks due to a too low number of replacement buses. The implementation of public transport reached a quite elaborated state and only less essential extensions are thinkable. For example, an intermodal router for passengers could be implemented.

The implemented concept for freight and goods enables the simulation of goods traffic. The concept is very similar to the extended public transport concept. Due to the implementation, intermodal logistic chains can be realised now. For the future, several extensions are possible. These include a goods router or the enhancement of the goods concept, such that not only container but also pallets or individual items can be transported. One could also enable that certain goods, such as food, bulk or liquids/gases, can be transported only by special vehicles. Furthermore, one could enable restrictions for the route of vehicles transporting hazardous material or heavy load. Another extension that is needed for more realistic freight transport is

the implementation of a rail signal concept. So far, trains behave on rail networks as cars behave on road networks. In reality, the rail network is divided into blocks and trains can enter a succeeding block only if they get signalised that there is no train in the succeeding block.

It can be summarised that the extended SUMO is able to simulate logistic and public transport which was proven in example scenarios. Further expansions are possible, however, mostly with respect to logistic as the implemented concept for public transport is quite advanced.

8.6 References

[1] Bundesministerium des Inneren, „Nationale Strategie zum Schutz Kritischer Infrastrukturen (KRITIS-Strategie),“ 2015. [Online]. Available: http://www.bmi.bund.de/cae/servlet/contentblob/544770/publicationFile/27031/kritis.pdf. [Zugriff am 27 1 2015].

[2] German Aerospace Center, „VABENE++ Traffic Management for Large Scale Events and Disasters,“ [Online]. Available: http://www.dlr.de/vabene/en/desktopdefault.aspx. [Zugriff am 4 2 2015].

[3] M. Behrisch, L. Bieker, J. Erdmann und D. Krajzewicz, „SUMO - Simulation of Urban MObility: An Overview, Proceedings of SIMUL 2011,“ in The Third International Conference on Advances in System Simulation, Barcelona, 2011.

[4] M. Behrisch, J. Erdmann und D. Krajzewicz, „Adding intermodality to the microscopic simulation package SUMO,“ in MESM 2010, Alexandria, 2010.

[5] J. Erdmann, „Multimodalität und Nachfragegenerierung mit SUMO,“ in Workshop Intermodalität, Berlin, 2013.

[6] J. Erdmann und D. Krajzewicz, „Modelling Pedestrian Dynamics in SUMO,“ preprint, 2015.

[7] German Aerospace Center, „SUMO Wiki,“ [Online]. Available: http://sumo.dlr.de/wiki/Vehicle_Type_Parameter_Defaults. [Zugriff am 4 2 2015].

[8] G. A. Center, „SUMO Wiki - Container,“ [Online]. Available: http://sumo.dlr.de/wiki/Specification/Containers. [Zugriff am 25 03 2015].

9 Accurate Vehicle Simulation in Logistic and Manufacturing Planning

Dr. Stefan Roth[1], Franz-Joseph König[2], Christian Drischl[2] and Marek Heinrich[3]

[1] Volkswagen AG, ITP Digitale Fabrik Logistik- und Behälterplanung, Brieffach 1832, 38436 Wolfsburg
[2] ZIP Ingenieurbüro Industrieplanung und Organisation, Wolfratshauser Straße 288, 81479 München, Germany
[3] German Aerospace Center, Rutherfordstraße 2, 12489 Berlin, Germany

Stefan.Roth@Volkswagen.de, Franz-Josef.Koenig@ZIP.de, Marek.Heinrich@DLR.de

Abstract

This paper outlines the need of accurate vehicle and traffic simulation in the field of logistic and manufacturing planning. Requirements, existing solutions and their enhancements will be described and their abilities will be discussed. Finally the results of bringing together the planning software suite MALAGA with SUMO, in order to provide a logistic planning suite enabled with accurate vehicle movement, will be described.

Keywords: MALAGA, SUMO, digital manufacturing, logistic planning, traffic jam, in-house logistic, congestion

9.1 Introduction to Simulation in Logistic Planning

Planning of production systems and their logistic supply systems originates as one of the traditional disciplines and strongholds in the automotive industry. Developments in the past have led to the inevitable capabilities of digital manufacturing. Digital manufacturing, focuses on merging available information from the description of manufacturing processes and factory layout information. Yet being widely accepted in the field of production planning, methodologies of digital manufacturing also emerge into material flow and logistic planning.

Continuous effort in developing standardized, high performing and user friendly software for layout based logistic and material flow planning has been carried out though the last ten year, leading to remarkable results. For instance from the cooperation between BMW, Daimler, VW and ZIP Industrieplanung has emerged the standard planning suite MALAGA, which dedicatedly incorporates the needs of the automotive industry and provides versatile bidirectional interfaces e.g. between the CAD-System Microstation and the Simulation tool kit Plant Simulation. Layout based logistic and material flow planning utilizes technologies such as CAD and simulation; advanced visualization capabilities of the engaged software is a key factor, flanking and driving the planning process.

Simulation studies have emerged as an indispensable investigation method in the field of industrial planning and layout development. Simulation studies in manufacturing planning vary over a broad range in their detail granularity.

Logistic layout planning in the automotive industry provides concepts and solutions for the organization of material transportation within manufacturing sites, e.g. planning how many vehicles are needed to reliably supply an assembly station in time with the needed manufacturing components.

As logistic planning evolves early in the process of projecting new facilities, when only few reliable data is available, 'static' design models are frequently used for evaluating the required transportation capacity and effort. Nevertheless, simulation studies are used, typically in the later phases of planning for testing and evaluating the developed layout.

Ongoing developments urge to bring the simulation into the planning process, becoming a constant tool in the workflow of a modern industrial planer. Enriching the static approach by ad hoc simulation studies, running in the background, validating dynamically the logistic model unveils planning discrepancies betimes. System inherent effects such as traffic densities, congestions and their recovering must be part of such a system.

When simulating the logistical network of a manufacturing site, often simplified models for vehicle movements are applied, due to the high modelling and computation effort and even more due to the lack of available and suitable vehicle and network modules in the leading industrial simulator software. These simplifications usually do not sufficiently consider the dynamic interactions between vehicles and with their environment.

When substituting the shortfall of embedded vehicle modules in industrial software, the desired module should provide a set network elements such as multi-lane road segments, controlled and uncontrolled intersection elements in arbitrary shapes and with complex turning relations. The behavior of the vehicle types must be realistic/natural and be compliant with standard traffic rules. Further it has to accurately resemble vehicle movements and driving maneuvers in pure car following or overtaking modes even when overtaking encounters intersections.

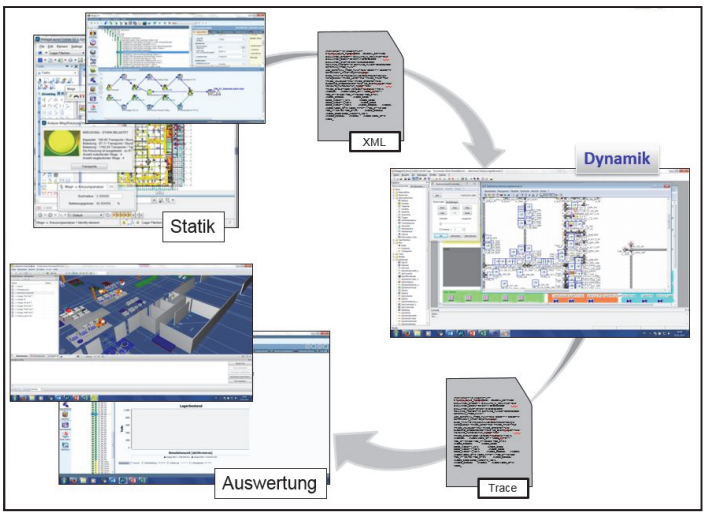


Figure 9-1: Schematic Work Flow in MALAGA

With the simulation of traffic dynamics logistic planning gains the opportunity to identify and investigate (potentially) congested traffic and blockades due to an overload or partly a blockade in the transportation network.

Investigations on accurate vehicle movement in logistic simulations carried out by ZIP, VW and DLR have fostered over the last three years to the development of an experimental software solution which leading to a deployable prototype in 2014.

9.2 Simulation Tools and Software Landscape

This paper is based on the developments that have been undertaken in a consortium of industrial partners and a research organization using the logistic planning and evaluation software suite MALAGA.

MALAGA is a planning tool kit suite assisting the logistic planner to design the inhouse logistic system and material supply of a manufacturing site. MALAGA has continuously been developed since 1990. MALAGA is a distributed software suite, enabling planers to cooperatively develop their projects. MALAGA's core software components are backboned by an object data base (Oracle) server operating both in online or in mobile mode.

The user interface is integrated into a CAD-software for layout design. MALAGA's strength is to visualize the material flow of a production system, its interdependencies and planning related analyses directly in the CAD-layout.

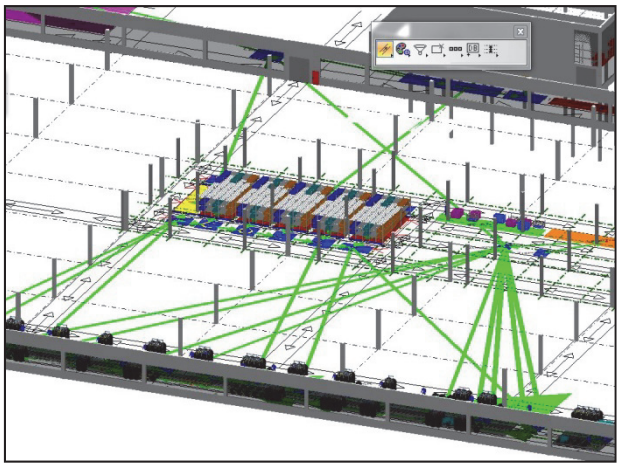


Figure 9-2: Relations of Material Flow

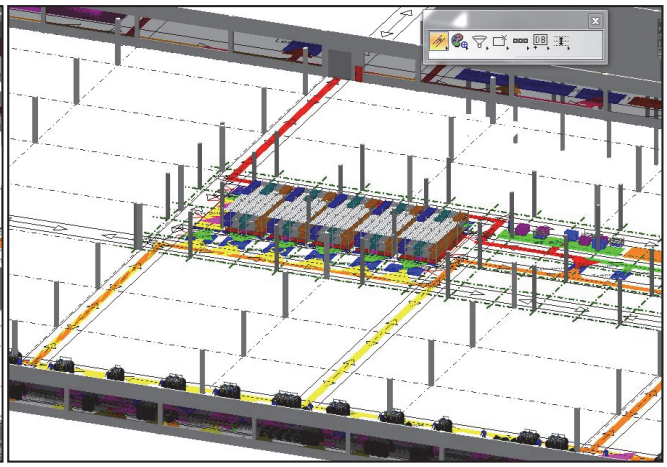


Figure 9-3: Type-specific Segment Occupation

When setting up an inhouse logistic system with MALAGA, the required data must be imported first. Since MALAGA is integrated into a CAD software, existing facility layout documents and floor plans can be easily be utilized.

The layouts shall be augmented with path elements and floor space information, e.g. about warehouses and productions facilities. The corresponding elements are placed directly into the CAD-layout.

To assess the material flow relations, information about the quantitative structures can be imported from a broad range of supported BoM management system. Additionally MALAGA needs process information, which can either be automatically imported or manually amended.

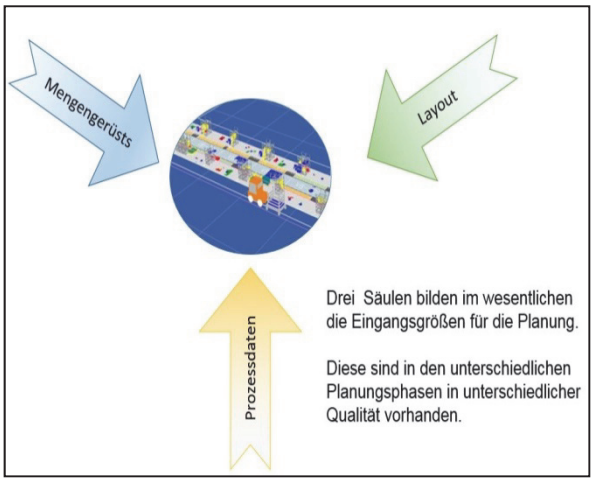


Figure 9-4: Information types used by MALAGA

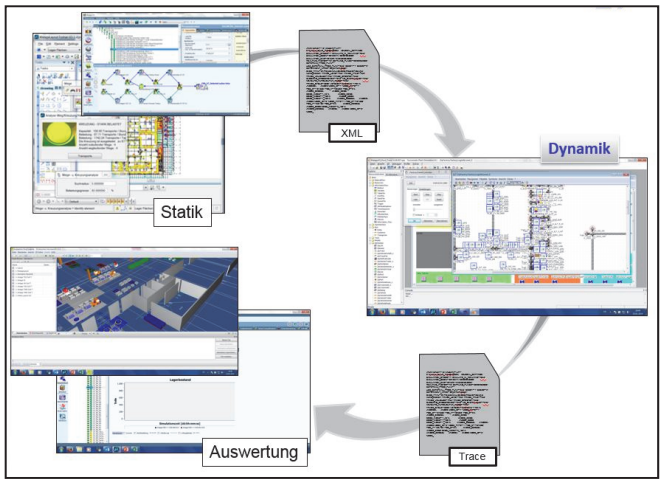


Figure 9-5: Planning Workflow in MALAGA

Together with the layout information and the quantity structure, the material flow relations can be assessed and are displayed in the layout by arrows of different strength indicating the relation's volume.

Importing floor layouts or BoM into MALAGA is a rather automatic process, whereas defining processes, especially logistical processes, is the design task of the planner shaping and analyzing the new inhouse logistic system.

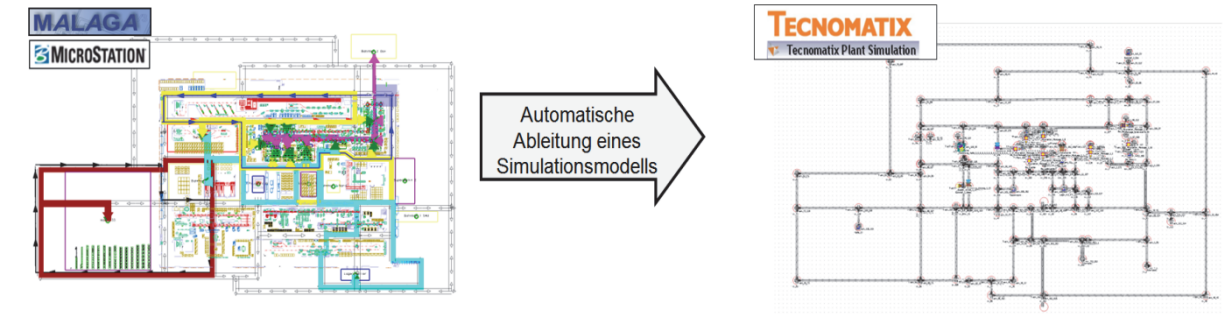


Figure 9-6: Converting a facility layout into a simulation model

For logistic supply planning, MALAGA breaks the variety of processes down into two main classes, indicating whether the process belongs to manufacturing entity (rectangular icon), or whether the process belongs to the logistic system (circular icon), e.g. certain parts need to be transported to the next machining center.

As it is inevitable important for logistic planning to know where parts are needed spatially, every manufacturing process is linked to a corresponding element (area) in the CAD-layout to define its geographic position.

Yet the transport processes must be assigned. Information sets about transportation processes consist of the transport action type, transport vehicle type, its route, part handling properties and others.

MALAGA assigns shortest paths between manufacturing stations and warehouses when routing vehicles, respecting restrictions such as transport type restriction, one way limitations or geometric properties of the path.

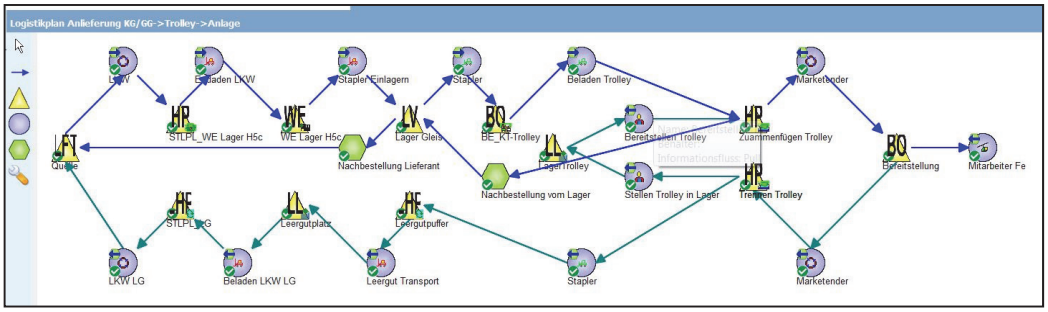


Figure 9-7: Process Chart in MALAGA

Based on the available data, MALAGA computes automatically the material flow and its characteristics. The results such as the visualization of the material flow, resource utilization and occupation rates for rolling stock and for the stores and area consumption can easy be displayed within the layout in a 3D fashion.

When calculating the material flow, MALAGA performs both a static estimation of the material flow, based on statistical analyzes and heuristics and a set dynamic simulation studies, coping with the dynamic nature of logistic and manufacturing processes.

Therefore since the early days, MALAGA is co-powered by a simulation engine, Plant Simulation, formerly known as Simple++ by AESOP. The simulation framework of the MALAGA Planning Suite is customized to runs on the same data. MALAGA automatically builds up the simulation model from the available data and performs the simulation runs autonomously in the background. Relevant events and data generated during the simulation are tracked and stored for post processing. Generally MALAGA's simulation module is used in a computational manner and no interactive user input or specific simulation knowledge required. After terminating the simulations, the results are evaluated by MALAGA and information e.g. on the stock inventory fluctuation and order management can be visualized.

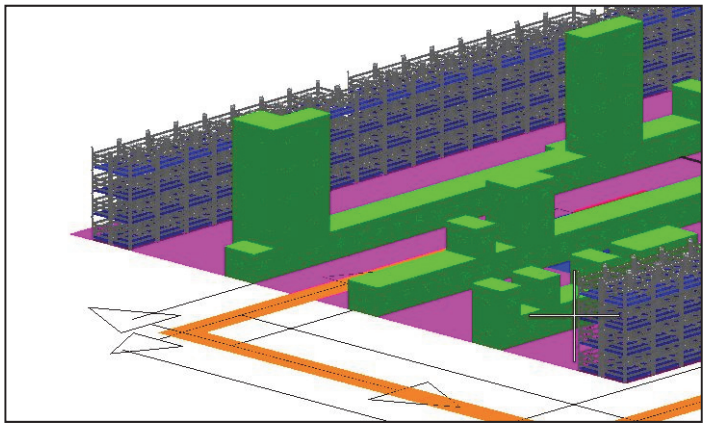


Figure 9-8: 3D visualization of a Warehouse's occupation

The outcome of the planning step, the intermediate layout of the material flow system is directly visualized in the CAD layout, showing the transport network's load, the material flow volume and the impact of measurement onto the logistic grid, highlighting stress zones and bottle neck areas for improvement in the next planning iteration.

9.3 Challenges in Embedding Aspects of a Road Traffic Simulation

Investigations such as [4] and their use cases show that there has always been a drive of integrating 'real-world-traffic' in material flow simulation models. Often Plant Simulations internal interpreter language is used when developing an in-model traffic simulation.

Yet no commonly used, widely spread, rich functionally equipped and even more, fast performing in-model solution could be found. Performance issues while handling intersection states and overtaking processes may have reduce the (re-) usability of the concepts in large scale simulation studies.

When enhancing MALAGA with accurate vehicle movement, we have preferred the traffic simulation suite SUMO over the option of developing a single use or constrained intra-model traffic simulation module, for its rich abilities in simulating accurate vehicle movements, its mature and well tested development state and its large user community, to be coupled to the planning tool's simulation core.

SUMO has been developed as scientific tool to simulate vehicles movements in a road network. SUMO provides fast and realistic road vehicle behavior. SUMO further provides all elements and traffic participants / user types of a real road network in accordance to most central European traffic rules.

SUMO is a testified simulation environment for simulating and analyzing traffic scenarios and traffic management interventions.

SUMO has been used in countless projects for the purpose in investigating mobility and transportation, as well as for researching traffic flow and traffic management concepts. SUMO’s stronghold is the proper representation of vehicles, their interactions with other vehicles and the 'world' in a coherent simulation suite. Nevertheless for various purposes its flanked with numerous tools providing solutions in traffic planning related tasks.

Due to its flexibility and its rich interfaces, SUMO has been linked to other simulation software and science application, eg. tapas, ns3, vtd, various driving simulators and others more. Still so fare SUMO has not been used in a manufacturing context and no attempt ever, to our knowledge, has been undertaken to connect SUMO to one of the leading tools in industrial planning.

Nevertheless it's well known, that linking two simulators can lead to performance draw backs.

9.4 Separation of Logical Responsibilities between the Simulators

In the existing Logistic Planning Software Suite MALAGA, Plant Simulation is used as a plug in to provide event driven simulation studies. Any simulation logic concerning the manufacturing processes, such as the dependencies among parts, process order, resource occupation and other are handled within Plant Simulation. Plant Simulation also keeps the sovereignty over the transport control logic. The transport control logic determines which parts need transportation at which time with constrains to the current state of the manufacturing system. It reserves vehicles, assigning transportation requests to transportation capacities,

directs milk-run organized vehicles and finally sends transportation units into their route or to their parking position.

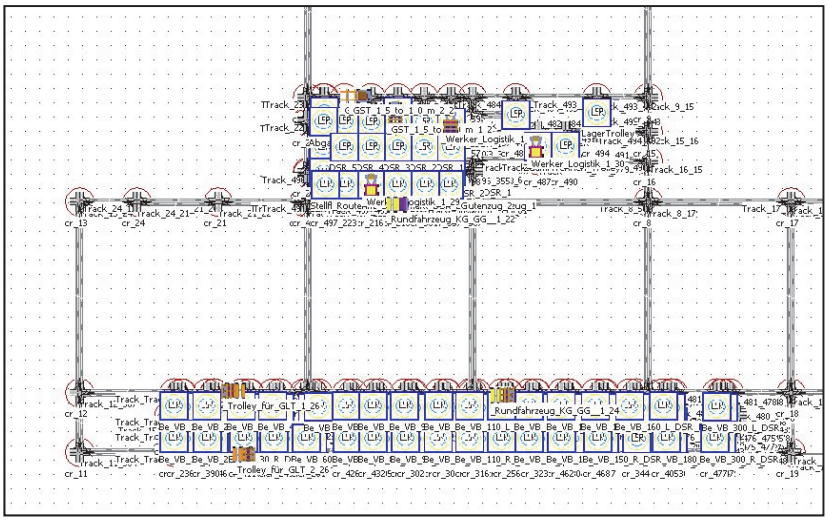


Figure 9-9: A Plant Simulation Model generated by MALAGA

When MALAGA sets up the simulation model, it also provides layout information, the configuration of the transport network and production resources, process data and the production plan. The simulation network contains the transportation gird and the production stations, each maintaining its own registries on assembly orders, parts and the local storage.

The Simulation is performed accordingly to the production plan, parts consumption is tracked at the production stations and once a critical amount of parts withdrawal has been reached, new part orders are released, leading to transportation requests. Transportation requests are handled by the Transport Control Module, providing and directing the vehicles in the simulation. If a transportation request had been scheduled, it must be assigned to a free transportation vehicle, fulfilling the transportation request restrictions.

MALAGA provides different, usually coexisting, transportation strategies, such as exclusive and direct fork lifter supply, transporting homogenous parts, 'Milk Run' supply vehicles operating on a fixed schedule, which might be carrying containers with mixed or homogenous parts and finally taxi service vehicles, carrying order specific containers on a predefined relation.

Depending on the transportation strategy, the transport control module gathers and merges transport requests and finally assigns new transport orders to the vehicles accordingly to the current available transportation capacities.

In unison with Plant Simulation, SUMO keeps out of any production logic, but takes over the control of any vehicle in route. SUMO simulates the vehicles movement through network and all their interactions. For instance, SUMO receives from Plant Simulation information that a certain vehicle should move from its current position to the next manufacturing position. While moving the vehicle there, respecting traffic rules and other vehicles SUMO frequently resubmits the vehicle's motion state and its position. Traffic jams, local congestions or speed drops related to the occupation of traffic network are handled within SUMO - natively. The effects are reported either back to transportation control, which means to Plant Simulation, or directly to the evaluation components of the planning suite.

9.5 Requirements, Enhancements and Realization

The task of integrating an accurate vehicle behavior into a highly specialized, mature and battle tested, industrial planning software suite, such as MALAGA, which is already in use at many customer sites, leaves no space for major changes in the principle software appearance, its interfaces or its fundamental functionality. Moreover the changes have to happen "behind the scenes".

For finding the most suitable component alignment with reasonable few alternations to the existing planning work flow, we deemed firstly to develop a prototype, which provides all crucial components and covers the essential functionality of the simulator coupling up to early deployment. The prototype should outline the interaction between MALAGA's internal simulator and SUMO and even more should already include the main characteristics of the full featured interface between both simulators.

In the prototype development phases we used a medium scale scenario, native in the automotive industry, as reverence with about 50 different manufacturing stations, 12000 parts and variants, 20 different transport vehicle types for logistic services and a total network length of nearly 10km.

The prototype and its functionality must be integrated into MALAGA's toolkit environment, using MALAGA's interfaces and providing access to the simulation data. Therefore it must contain a number of high level software modules:

A module to convert available data to SUMO's native formats

When MALAGA builds up the simulation model for Plant Simulation, it must now in parallel configure the SUMO simulation, using the same information available. This mainly concerns the topology of the transportation network including the shape and properties of edges and intersections. Further must be provided the positions of manufacturing stations as input for OD-Matrices, as well as routes, paths sequences and vehicle types and classes with their dimensions and their restrictions in the network.

A module synchronizing the two simulators

Once properly initialized both simulators must perform their simulations synchronously. Hence a fundamental functionality of the prototype must be to provide synchronously the head beat stimulus to the simulators.

Generally both simulators operate independently. Plant Simulation simulates any manufacturing events and handles all the processes logic and even orchestrates the logistic system. SUMO is supposed to focus on the vehicle movement simulation purely. The simulators are brought together in order to complement each other, each operating in its own, disjunctive sphere. But nevertheless, to a certain extend both may also simulate the same situations overlapping. It's important to keep track of these parallel situations to make sure both simulators do find the same results or are aligned to each other properly. This situation can be found for example when a vehicle leaves a production facility and hence for a moment is in the sphere of both simulators.

A communication module

Since both simulators do depend on each other's simulation state and intermediate results, the exchange of information and the performance of the communication interface is crucial to the prototype.

Enhancements to the Transport Control Module

The control over the transport system remains the domain of Plant Simulation, but has strong effects in both simulations. The transport control must be able to direct SUMOs vehicles between manufacturing stations and warehouses, accordingly to the transportation requests. Vehicles must be able to resume and pick up new transport orders or being sent to their parking position.

Where possible specific behavior of vehicles should be resembled, e.g. certain types entering a manufacturing cell, whereas others don't.

As the Transport Control Module is already available for the vehicles in Plant Simulation, it hence must be enhanced to be compatible with SUMO.

A data extraction module

Both simulators, Plant Simulation and SUMO are designed to collect and evaluate data during the simulation run.

Within the prototype development, SUMO must be enhanced to track specific simulation events, the same way as they would have been tracked within Plant Simulation. Further the tracked information must be consolidated and being provided back to MALAGA. Additionally, SUMO then does provide much more data and even more information which, yet could not have be generated by Plant Simulation, in particular concerning the vehicles motion and the dynamics in the road network. The software must be enabled to process this new data and to incorporate their evaluations into MALAGA's visualization capacities.

Besides the pure technical co-alignment of the simulators, certain traffic situations and vehicle behavior was deemed to be part of the prototype.

The prototype's main objective is to bring correct vehicle behavior into the material flow simulation. Therefore typical aspects of a traffic simulation where defined in order to find them in MALAGA's vehicle visualization.

First, vehicles should never run through each other while they are on the same track or lane. This must always be the case, regardless of the vehicles speed, the traffic density, the network topology. This can only be assured when vehicles know about each other, behave strategically and cooperatively and even do plan their behavior into the future, considering the movement of other vehicle, even if they are on different segments.

Vehicles must properly respect traffic rules, and vehicles should not violate the right of way at intersections when other vehicles approaching intersection, regardless weather the intersection is unregulated, controlled or prioritized. The complexity of the intersections shape must not have any impact on the reliability and the compliance of the vehicles.

Intersections do reduce the free flow of a road network, we therefore deemed to find vehicles jamming up at over-crowed intersections. On the other hand congested intersection should never be completely blocked or dead locked by vehicles, leading to a collapse of the entire traffic system.

Vehicles should block other vehicles, in situations where this can happen in reality too, e.g. when a unit is unloading in front of a manufacturing station, occupying parts of the road network. If the road segment is not sufficiently broad for other vehicles to pass, they should wait behind the blocking vehicle.

Insertion of the vehicles should already consider the actual traffic density on the edge. Vehicles should only be inserted into the simulation, if there is enough space for them to enter, without causing collisions.

The co-alignment of Plant Simulation and SUMO provides information about dynamic aspects of the vehicle flow within a production site, leading to congestions and in the worst case outlining disruptions or delays in the part supply of the inhouse logistic system. The prototype also provides information about the occupation of the network, its most stressed segments and critical intersections tending to be congested. Loss time can be computed individually or globally for the entire network.

The evaluation emphasis of the prototype focuses on vehicles and parts, providing capabilities to analyze delay times of single vehicles and parts, the total delay of orders, individually or grouped, allowing to analyze the tradeoff between fleet size and utilization vs. delay of part orders.

9.6 Runtime and Communication Efforts

Existing models of extensive manufacturing sites can currently perform a simulation run for a complete production year within hours using static logistic planning approaches.

It has been expected that the simulation of vehicle movements and their dynamics will increase the computational effort. Nevertheless serious real live and large scale models must still perform a simulation run within hours, the worst with in a working day, which is perceived as the user's run time threshold.

When developing the prototype of the integrated SUMO for MALAGA, information exchange and synchronization between the simulators was found to be one of the major bottlenecks.

Static information exchange, such as the layout of the network and it's and basic vehicle descriptions, their serviced area which will not be altered during the simulation run, can for ease, be converted into a SUMO-readable or SUMO-like xml-based file. Dynamic information exchange, such as new transportation orders, vehicle's motion state or task completion information can only be delivered during the simulation run and hence must use one of Plant Simulation and SUMO online exchange protocols.

Within the prototype two pre-existing interfaces of Plant Simulation and SUMO have be utilized, but were found not to meet the performance criteria.

For an early proof of concept, with no concessions towards performance speed, a purely file based approach for dynamic data exchange was implemented and immediately discarded.

Information was written into text files accessed by both simulators, the files lived in a ram-drive. Besides the file handling overhead, performance also heavily suffered from the limited efficiency in Plant Simulation's string parsing capacities.

With the shift to encode information into a binary format, SUMOs native purely socket based interface TraCI was used to establish the communication interface to SUMO. Again TraCI comes with some overhead, leading to acceptable performance rates for medium scenarios. Large scale scenarios including multiple manufacturing sites, perform the simulation of a production year within 24h. Since the additional computational time SUMO needs is about 20% of the total time consumption, still communication and synchronization remains the main bottleneck. In order to overcome the communication time gap new customized and performance tweaked interfaces for both SUMO and Plant Simulation have been under developed and successfully tested.

9.7 Outlook and Discussion

With the development of the prototype for the integration of SUMO into MALAGA, ZIP, VW and DLR have proven on a technical level that both simulators may symbiotically be coupled. Yet not all obstacles have been removed, especially further investigation is needed in the field of handing unutilized vehicles and on manufacturing specific artefacts in the network and its elements.

The prototype, even though yet not fully featured in all details the prototype has proven its usefulness already in real life consultant projects flanking the existing evaluations of milk run systems, leading to a reduction in the estimated vehicles fleet and containers.

Moreover the ongoing development promises an industry rollout of the new technology in 2015, becoming available to planners in early 2016.

9.8 References

[1] Mayer, Gottfried, Pöge, Carsten: Quo vadis Ablaufsimulation – Eine Zukunftsvision aus Sicht der Automobilindustrie, 15. ASIM Fachtagung 2013

[2] Krug, Wilfried; Marz, Lothar; Simulation und Optimierung in Produktion und Logistik: Praxisorientierter Leitfaden mit Fallbeispielen (VDI-Buch); 2010

[3] Staab, Tobias, Galka, Stefan, Klenk, Eva, Günthner, Willibald A.: Effizienzsteigerung für Routenzüge - Untersuchung des Einflusses der Routenführung auf die Auslastung und Prozessstabilität; 15. ASIM Fachtagung 2013

[4] A.E. Rizzoli, R. Montemanni, E. Lucibello L.M. Gambardella; Ant colony optimization for real-world vehicle routing problems; Swarm Intelligence, Springer December 2007, Volume 1, Issue 2,

[5] Juan, A.A.; Rabe, M.: Combining Simulation with Heuristics to solve Stochastic Routing and Scheduling Problems. In: Dangelmaier, W.; Laroque, C.; Klaas, A. (Hrsg.): Simulation in Produktion und Logistik - Entscheidungsunterstützung von der Planung bis zur Steuerung. Paderborn: HNI-Verlagsschriftenreihe 2013

10 Modelling Pedestrian Dynamics in SUMO

Jakob Erdmann and Daniel Krajzewicz
German Aerospace Center , Rutherfordstraße 2, 12489 Berlin, Germany

{Jakob.Erdmann, Daniel.Krajzewicz}@DLR.de

Abstract

Walking is the most natural way of human mobility. The microscopic simulation package SUMO has supported an intermodal person-based simulation including pedestrian since 2010 (version 0.12.0). However, movements along a road were resembled using a linear interpolation only and pedestrian dynamics at an intersection were not modelled at all. Within the scope of the COLOMBO project, SUMO was extended to simulate pedestrian dynamics in more greater detail. This includes extensions to the road network format, movement models and routing tools.

Keywords: pedestrian dynamics, human locomotion, inter-modality.

10.1 Introduction

Especially in Europe, a long-term shift towards putting soft modes of transport, mainly bicycles and pedestrians, into the focus can be observed. Several reasons motivate this. First, these modes of transport are environment friendly. Second, they are healthier than using vehicles, both for the user himself as well as for other persons. Both circumstances do not only motivate individuals to change to non-motorized traffic. They are as well targeted by authorities and societies that try to avoid penalties for not keeping pollutant density thresholds and avoid long-term costs of an unhealthy population. On the other hand, pedestrians and bicycles require special care due to being more vulnerable than motorized traffic.

Conventionally, traffic simulations are helping in the design and development of both, strategic actions as well as road infrastructure changes. Consequently, established commercial traffic simulations have incorporated pedestrian and/or bicycle dynamics in recent years. Until version 0.21.0 SUMO lacked a model of pedestrian dynamics. Albeit its inter-modal person routes [Behrisch et.al, 2010] include a "walking" stage, pedestrians were moved along roads with a constant speed and jumped over the intersections. No interactions between pedestrians or between pedestrians and traffic were modelled.

Extending SUMO by pedestrian and bicycle dynamics was scheduled within the COLOMBO project. The main goal of COLOMBO is to develop traffic surveillance method and traffic light controls that use data from V2X-enabled vehicles assuming a low equipment rate. In this context, the dynamics of pedestrians was assumed to be necessary for two reasons:

- Pedestrians may deliver additional information that may be used by the developed traffic surveillance and, in case of a sparse connectivity, may be used as additional relay nodes.
- Vehicles which turn left or right at an intersection typically have to yield to pedestrians. Thus pedestrians need to be included in a simulation to correctly model urban intersection capacity.
- The development of environment-friendly traffic light controls should take pedestrians into account to allow research on control strategies which prioritize the environment-friendly transport modes.

The remainder is structured as following. At first, the requirements for pedestrian dynamics are listed. Then, the implementation is described and finally some measurements from pedestrian simulations are given.

10.2 Requirements

As outlined, the major goal was to correctly replicate the behavior of pedestrians at (traffic-light controlled) intersections. This implicates the following functional requirements:

- vehicles need to wait for pedestrians which are crossing the road in front of them;
- pedestrians need to cross the road in order to continue on the other side;
- right-of-way rules observed normally between the different modes at different types of intersections should be respected;
- pedestrians dynamics should be sufficiently detailed to model the time required for passing a pedestrian crossing including the following aspects:
  - width of the available walking space,
  - bidirectional movement,
  - positioning in front of the crossing while having to wait,
  - density of pedestrian traffic,
  - route choice when passing an intersection diagonally;
  - simulation outputs should allow tracking of pedestrians.

The according adaptations had to be performed to the simulation modules (SUMO and SUMO-GUI). One should note that the implementation of the needed models and data structures has to be accompanied by according extensions in the used data files. As well, to match SUMO's philosophy of offering a high level of user support, the supporting tools had to be extended. Thereby, regarding the implementation of pedestrians within SUMO, additional requirements relate to the application chain used in scenario creation:

- The application for network building should be enabled to support the necessary networks structures for meeting the above functional requirements
  - by using explicit input specifications;
  - by using heuristics to generate the necessary structures from context.
- The tools for demand generation should be able to support the creation of multi modal demand

Additionally, there were non-functional requirements related to the architecture of the simulation suite SUMO:

- the implemented models should be modular enough to make them replaceable,
- the applications should be backwards compatible with the input data formats of previous versions,
- input data formats should be changed as little as possible,
- the implementation should be fast enough to allow the simulation of city-sized scenarios (at least for some models),
- the visualization of pedestrians should be sufficiently detailed to allow diagnosing the simulation behavior,
- pedestrian dynamics should be included in the existing inter-modal trip chains.

10.3 Implementation

This section describes the implementation steps performed for obtaining the needed functionality. At first, the changed in representation of the infrastructure are given followed by a presentation of the implementation of the pedestrian dynamics themselves. Finally, additional work performed on the network and routes preparation modules is described.

10.3.1 Infrastructure

One of the most challenging work steps was to find a representation of the road infrastructure used by pedestrians that on the one hand is capable of representing complex intersections, but on the other hand fits well with the existing structures and is efficient enough for large simulations. Different alternatives for modeling intersections were tested as reported in [COLOMBO D5.2, 2014].

In the implemented architecture, road vehicles, bicycles and pedestrians move on separate network elements. Each mode only interacts with members of its own mode while traveling along a road and the interaction between modes happens at intersections only.

SUMO simulates movements along unidirectional roads (also called edges) consisting of one or more lanes where each lane allows as many vehicles as it's longitudinal length permits but only ever allowing a single vehicle in the lateral direction. At intersections (also called nodes) vehicles regard traffic lights and right of way rules before passing. To allow for pedestrians and bicycles, additional lanes are added to existing edges which represent sidewalks and bicycle lanes. Furthermore, "blind" lanes which do not allow any traffic can be added to model green verges between these mode-specific lanes. To model the paths of pedestrians at intersections, specialized edges (and lanes) are added for modeling pedestrian *crossings* and for modeling sidewalk corners where crossings and sidewalks meet (called *walkingarea*). Figure 10-18 shows the previous and the extended network model in the simulation GUI.

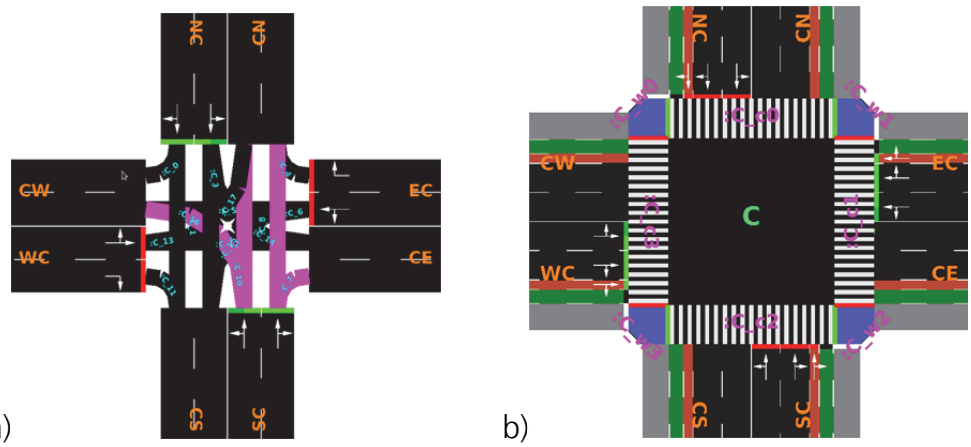


Figure 10-18 a) Previous network model with "normal" edges labelled in orange and "internal" edges labelled in cyan. The internal edges outgoing from edge "SC" are highlighted in magenta; b) 4-arm intersection with bicycle lanes (brown), sidewalks (grey), walkingareas (blue), crossings (striped), green verges (green). IDs are shown for all edges which may be used by pedestrians.

Each crossing is defined to either give priority to the pedestrian or to the vehicle (The latter case is distinguishable in the GUI by having black/grey stripes instead of black/white). The connectivity among sidewalks, walkingareas and crossings is modelled using unidirectional connections as in the previous network model. However, these connections as well as the edges may be traversed by pedestrians in either direction. Edges of the type "walkingarea" have the unique property of being connected to edges in multiple directions so as to make the question of their direction ambiguous. Resolving this ambiguity is left to the pedestrian model (see 10.3.4). When drawing a *walkingarea* in the GUI its "shape"-attribute is interpreted as the polygonal border around the space, rather than as a polygonal line in the direction of the edge.

Some features of real world traffic such as heterogeneous lane use and bidirectional lane use during overtaking cannot by modelled by this architecture. This is a consequence of keeping the existing vehicular model and only extending the intersection model. However, we expect the extended intersection model to be able to accommodate future extensions along these lines.

10.3.2 Pedestrian Dynamics

One of the initial goals of the traffic simulation SUMO was to support researchers in comparing and validating different traffic models. While this was mainly stated having car-following models in mind, it should as well count for models of pedestrian dynamics. Therefore, not a single dedicated model of pedestrian dynamics was implemented, but rather an API (application programmer interface) for embedding different models. The interface is minimalistic to give the model developer a high degree of freedom. A pedestrian dynamics model has to support the following functionality:

- Return whether a given lane is currently blocked by any pedestrians from being passed at a certain location (*function blockedAtDist*)
- Add a new pedestrian and return a *PedestrianState* object which must be able to report on the position, angle and speed of that pedestrian

When adding a pedestrian to be controlled by the Pedestrian model, the following information must be supplied:

- A sequence of (normal) edges to define the "skeleton" of the walking route
- The starting position relative to the first edge
- The destination position relative to the last edge
- The maximum speed

These attributes are all contained in the definition of a <walk> which is part of a person's plan, just as in older versions of SUMO. It is the responsibility of the pedestrian model to select the sequence of walkingareas and crossings which are needed to connect the given normal edges when passing an intersection.

Currently, two pedestrian dynamics models are included in SUMO. They are presented in the following subsections.

10.3.3 Model "nonInteracting"

The initial "dynamics model" where pedestrians move with a constant speed, disregard interactions with other pedestrians and "jump" across intersections can be selected using the option `--pedestrian.model nonInteracting`. It may be useful if the pedestrian dynamics are not important and a high execution speed is desired. One enhancement that has been made is that pedestrians may use edges in both directions. The walking direction on a given edge is computed based on the topology of the edge sequence.

10.3.4 Model "striping":

The "striping" model implements detailed pedestrian dynamics according to the requirements in section 10.2. It is selected using the option `--pedestrian.model striping` and also serves as the new default model. In the following, the main functionalities of the model are described.

Routing within an intersection

When passing an intersection, a sequence of walkingareas and crossings must be used to reach the next "normal" edge in the pedestrians route. When there are multiple routes available, the *PedestrianRouter* described in 10.3.7 is used with its scope limited to the current intersection. The signal states of the traffic lights are used to select a path which avoids waiting when possible.

Interactions of pedestrians with each other

The model assigns 2D-coordinates within a lane (of type sidewalk, walkingarea or crossing) to each pedestrian. These coordinates which are defined relative to the leftmost side of the start of the lane are updated in every simulation step. This is in contrast to the coordinates of vehicles, which (generally) only have 1D-coordinates within their respective lane. Pedestrians advance along a lane towards the next node which may either correspond to the natural direction of the lane (forward movement) or it may opposite to the natural direction (backward movement). Thus, the x coordinate monotonically increase or decreases while on a lane. Once the end of a lane has been reached, the pedestrian is placed on the next lane (which may either be unique or determined dynamically with a routing algorithm).

The most important feature of pedestrian interactions is collision avoidance. To achieve this, the "striping"-model divides the lateral width of a lane into discrete stripes of fixed width. This width is user configurable using the option `--pedestrian.striping.stripe-width` and defaults to 0.65 m. These stripes are similar to lanes of a multi-lane road are used by vehicles. Collision avoidance is thus reduced to maintaining sufficient distance within the same lane. Whenever a pedestrian comes too close to another pedestrian within the same stripe it moves in the y-direction (laterally) as well as in the x-direction to change to a different stripe. The y-coordinate changes continuously which leads to situations in which a pedestrian temporarily occupies two stripes and thus needs to ensure sufficient distances in both. The algorithm for selecting the preferred stripe is based on the direction of movement (preferring evasion to the right for oncoming pedestrians) and the expected distance the pedestrian will be able to walk in that stripe without a collision.

During every simulation step, each pedestrian advances as fast as possible while still avoiding collisions. The updates happen in a single pass for each walking direction with the pedestrian in the front being updated first and then its followers sorted by their x-coordinate. The speed in the x-direction may be reduced by a random amount with the maximum amount defined as a fraction of the maximum speed, using the option `--pedestrian.striping.dawdling <float>` (defaulting to 0.2).

As a consequence of the above movement rules, pedestrians tend to walk side by side on sidewalks of sufficient width. They wait in front of crossings in a wide queue and they form a jam if the inflow into a lane is larger than its outflow.

The division into stripes in the lateral direction is straightforward for walking areas and crossings which have two main directions of walking. In contrast, walkingareas are used in multiple directions. To apply the above movement rules additional processing takes place. For every combination of sidewalk and crossing adjacent to a walkingarea, a unique path is computed at the start of the simulation. During the simulation each pedestrian uses the unique path which allows it to follow the sequence of walkingareas and crossings prescribed by the *PedestrianRouter*. Each of these paths is computed separately according to the above movement rules. To avoid collisions between pedestrians on different walkingarea-paths, the pedestrians from other paths are mapped into the coordinate system of the current path beforehand.

The "striping"-Model can be seen as a compromise between space-discrete and space-continuous pedestrian models due to combination of continuous positions and discrete stripes. The model captures qualitative dynamics when there are two main directions of movement such as is found on sidewalks and crossings but is not well suited to describe the dynamics in other cases (i.e. pedestrians cannot back up in order to clear space in a crowded area). As an advantage over other more detailed models it allows for a computation time which is linear in the number of simulated pedestrians. More specifically the running time for executing a single simulation step is in the order of $O(n \times k)$ with n being the number of pedestrians and k being the maximum number of parallel stripes for all lanes. This is achieved by using only a very limited set of surrounding pedestrians to compute pedestrian interactions (Since the coordinate-remapping on walkingareas only happens per path, the effort is linear in the number of pedestrians) .

Interactions between pedestrians and other modes

In SUMO there are two concepts for modelling the influence of a conflicting traffic stream on a vehicle:

a) Each vehicle registers its approach to an intersection along with an expected time slot for passing the intersection. A vehicle approaching the intersection must yield to any vehicle with higher priority which wants to use the same time slot.
b) Each vehicle must cross certain set of "foe" lanes which are used by conflicting streams. The vehicle must yield regardless of priority whenever such a "foe"-lane is occupied by another vehicle (and the vehicles are not geometrically past the conflict point).

Concept a) is used for modelling uncontrolled crossings. A pedestrian wishing to cross the street at an uncontrolled intersection can only do so if its expected time slot for using the intersection does not interfere with that of an approaching vehicle. It should be noted that the dynamics at unprioritized crossings are conservative in estimating the time required gap. In the simulation, pedestrians will only use such a crossing if the whole length of the crossing is free of vehicles for the whole time needed to cross. In reality, it can be observed that pedestrians start to cross while vehicles are still occupying the far side of the crossing.

Concept b) is used for preventing vehicles from driving across a pedestrian crossing which is occupied by pedestrians. Pedestrians themselves never register for a time slot. While they have not moved onto the crossings, vehicles are free to drive. The influence on vehicles is implemented via the interface method *blockedAtDist* which is called to request whether a "foe"-lane in the vehicles path is blocked at specified distance due to the presence of pedestrians. The given distance value corresponds to the geometric intersection between the crossing and the vehicles trajectory measured as distance from the start of the crossing. The "striping"-model computes its results by iterating over all pedestrians on the lane and returns "blocked" status if a pedestrian is found which is not yet past the intersection point but within a threshold distance to that point (currently fixed at 10m). For "foe"-lanes other than crossings the check always returns false since pedestrians do not walk there.

Concept b) could also be used to prevent pedestrians from walking into vehicles which occupy the crossing but this is currently not implemented.

10.3.5 Further Pedestrian Models

Both presented models have not been published before and are thereby not known to the scientific community. Within the COLOMBO project, a further model was implemented that has been discussed in literature intensively [Antonini et al., 2006], [Antonini, Berlaire, Schneider, Robin, 2009]. Being currently implemented within a standalone application, the model has not yet been included into SUMO. This is planned to be done during the next time. As well, several open source implementations of established pedestrian dynamics models exist, e.g. "pedsim" [pedsim] which uses a social force model (see [Helbing, Molnár, 1995]), that are planned to be included.

10.3.6 Extensions for network generation

The NETCONVERT application is part of the SUMO application suite. It is responsible for preparing the simulation network (net.xml) from a wide range of input data formats such as

OpenStreetMap (OSM), VISSIM or shape files. Another important input format is a set of simple xml inputs (called plain XML) which describe the nodes, edges and optionally the connections of the road network. NETCONVERT enriches its inputs by computing connectivity, right-of-way rules, and the geometry of intersections with configurable heuristic models.

To support intermodal simulations, NETCONVERT was extended to create sidewalks as well as the new edge types "walkingarea" and "crossing" described in the previous sections. The crossings must be included in the generated right-of-way rules. Furthermore, heuristically generated traffic-light programs are adapted to include pedestrian signals. We describe these procedures in the sequence in which they are executed. For a usage description of the new functionality refer to [3].

Generating Sidewalks

NETCONVERT supports multiple ways of defining sidewalks which are appropriate in different usage scenarios:

- In plain XML input when describing edges (*plain.edg.xml*). This is done by defining an additional lane which only permits the vClass "pedestrian" and setting the appropriate width or by including the new attribute `sidewalkWidth`
- When importing edges with defined types (i.e. from *OpenStreetMap*), sidewalks may be added for selected types
- Heuristically for all edges with a speed limit within a defined range
- Based on permissions: edges which allow pedestrians receive a sidewalk.

Below is an example edge definition with two vehicular lanes, a green verge, a bicycle lane and a sidewalk:

```
<edge id="x" from="A" to="B numLanes="5" speed="13.89">
    <lane index="0" allow="pedestrian" width="3.00"/>
    <lane index="1" allow="bicycle" width="1.00"/>
    <lane index="2" disallow="all" width="1.50"/>
    <lane index="3" allow="passenger"/>
    <lane index="4" allow="passenger"/>
</edge>
```

Generating Crossings

Crossings may be defined explicitly in plain XML input when describing connections (*plain.con.xml*). This is done using the new XML element `<crossings>` with the mandatory attributes `node="<nodeId>" edges="<listOfEdgesToCross>"` and the optional attributes `width="<widthInM>"` and `priority="true/false"`. This defines a crossing at the given node across the given list of edges. Crossings at TLS-controlled nodes are always prioritized. Since crossings are always associated with nodes, a node must be present if a crossing somewhere along an edge is to be modelled. This fits well into the existing simulation architecture which only recognizes conflicting traffic streams at nodes. Below is an example crossing defininition:

```
<crossing node="A" edges="x y" width="4.5"/>
```

The second available method adding crossing information to a network is with the option `--crossings.guess`. This enables a heuristic which adds crossings wherever sidewalks with similar angle are separated by lanes which forbid pedestrians. If the edges to be crossed have sufficient distance between them or vary a by a sufficient angle, two crossings with an intermediate walking area are generated. Such split crossings can be seen in Figure 10-19.

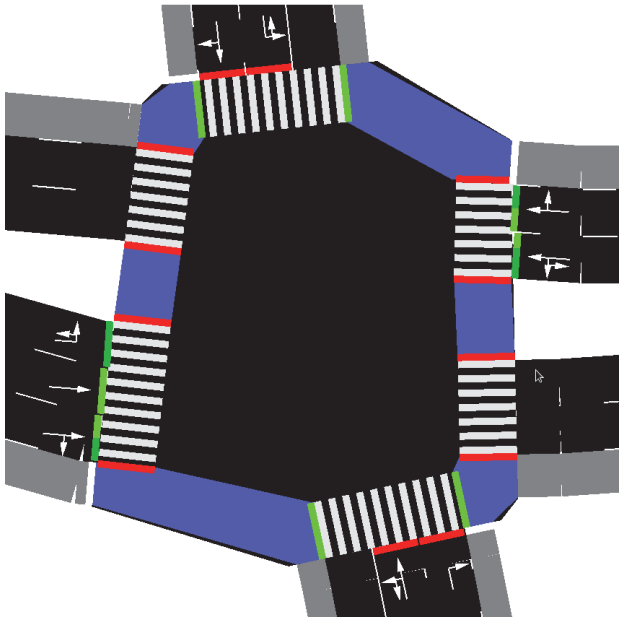


Figure 10-19 Intersections with split pedestrian crossings

Generating Right of Way Rules

The Intersection model described in [Erdmann, Krajzewicz, 2014] extends naturally to pedestrian crossings. Crossings are simply another set of *internal lanes* which must be considered in the conflict-matrix and the right-of-way matrix.

The first matrix (called "response") defines for each connection, the set of foe connections to which it must yield in case of registered approaches. The second matrix (called "foes") describes for every connection, the set of foe lanes which may not be crossed in case of occupancy. These matrices are extended to incorporate crossings depending on whether they are prioritized or not. In the former case, all connections which have trajectories intersecting the crossing must yield to pedestrians occupying the crossing whereas the crossings themselves are only flagged in "foes" matrix which means pedestrians are free to walk. In the case of unprioritized crossings, the right of way varies depending on the properties of the road connection: Vehicles which perform a left or right turn must yield if there is a pedestrian crossing on their target edge. Otherwise the pedestrians must yield to all vehicles.

Generating Signal Plans for Crossings

A controlled intersection with pedestrian crossings needs to incorporate information about the signal states for pedestrians. In the previous versions of SUMO all connections entering an intersection are generally controlled by the traffic light. When adding pedestrian structures, this no longer holds true. Connections between sidewalks and walkingareas are never controlled. On the other hand connections from walkingareas to crossings are always controlled. Connections from crossing to walkingarea are uncontrolled as it is always possible to leave the crossing. When entering a crossing in the backward direction (relative to its

natural direction), the traffic light state for the forward entering connection is substituted instead of using the (uncontrolled) connection from the crossing to the walking area in reverse.

The additional controlled connections are indexed in clockwise directions starting in the north following the connections from normal edges. Thus, signal plans for such intersections can be given explicitly by defining phase states of the appropriate size.

When signal plans are generated heuristically, the signal state for pedestrian crossings is set to "red" whenever any intersecting straight connections are set to "green major" (being able to drive with absolute priority). Otherwise the crossing is set to the "green major" state itself. This ensures that pedestrians are only allowed to walk when they do not disturb straight-going traffic. TLS signals for vehicles with a green state are set to "green minor" if the destination edge of that connection intersects a pedestrian crossing which also has a green state. This models the fact that vehicles turning right or left at an intersection need to yield to pedestrian crossings when leaving an intersection. The "green minor" state ensures a slow approach which allows vehicles to brake for pedestrians in time. Additional phases are generated to allow pedestrians to leave an intersection before giving vehicles the green light.

NETCONVERT can apply these heuristics to existing signal plans in order to "upgrade" vehicular networks in to intermodal networks.

Generating Walkingareas

Whenever at least two sidewalks are adjacent at an intersection or a sidewalk is adjacent to a crossing, a walkingarea which connects these structures is generated. Unidirectional connections following the existing schema for regular road connection are generated according to the following rules:

- sidewalks of edges incoming to the current node have a connection to the walkingarea
- walkingareas have a connection to sidewalks of outgoing edges
- Connections between walkingareas and crossings are generated in a counter-clockwise fashion around the node.

Currently walkingareas are only generated if the network is built with the option `--crossings.guess` or at least one crossing is specified in the input files. This was done to still allow the generation of networks without pedestrian structures but could be made configurable in the future.

10.3.7 Additional Extensions

Various other additions to the SUMO code base were implemented in order to meet the requirements listed in section 10.2. They are described in the following.

Pedestrian router

A routing application was implemented which allows routing bidirectionally on sidewalks, walkingareas and crossings. This is accomplished for constructing a special routing graph which can be processed with the existing implementation of the Dijkstra routing algorithm. One particular feature where this adapted graph differs is the dynamic treatment of TLS-controlled crossings. In reality pedestrians which need to cross the street twice to reach the

diagonally opposite corner of an intersection will usually select the crossing which first shows a green light, using the knowledge that the second crossing will be green soon after they reach it. To achieve this type of behavior, the travel times which are returned by each edge in the routing graph take into account whether access to an edge is regulated by a traffic light which is currently in its red phase. The travel time for passing an edge behind a red light is computed using the following formula:

$$traveltime = length / speed + max(0, 20 - (t - t_D))$$

where $(t - t_D)$ is time offset to reach that edge from the current moment. Thus, red lights in close proximity are avoided while far away red lights are not.

Output

The existing methods for retrieving simulation data where extended to cover pedestrians

- Option `–fcd-output` now includes positions, speeds and angles of pedestrians
- Option `–nestate-output` now includes positions, speeds and angles of pedestrians
- TraCI allows retrieving 2D-position, edge, edge-position, angle and speed of persons (and thus pedestrians)

Demand Generation

The tool randomTrips.py was extended with the options `--pedestrians` which generates persons with a single walk between random locations. The new option `--max-distance` can be used to limit the distance of walks.

10.4 Simulations

The main goal of the described extensions was to model the interactions between vehicles and other modes of traffic. To obtain a quantitative assessment of these interactions some experiments were conducted. These are described in the following.

10.4.1 Interactions between right-turning vehicles and crossing pedestrians at a single intersection

In this experiment, a saturated flow of right-turning vehicles arrives at a single intersection. To complete the right turn, this flow must pass a pedestrian crossing which is frequented by a binomially distributed pedestrian flow of variable strength. The synthetic intersection used for this experiment is shown in Figure 10-20. The simulation was conducted with a traffic light as well as with a prioritized pedestrian crossing. The traffic light was following a fixed cycle of 90 seconds with 26 seconds combined green time for vehicles and pedestrians and 5 seconds exclusive green for the vehicles (the rest of the cycle being reserved for other directions of traffic). **Fehler! Verweisquelle konnte nicht gefunden werden.** shows the vehicular flow behind the crossing in dependence on pedestrian density. At low and medium pedestrian flows, the uncontrolled intersection allows for higher flows due to the absence of "red" phases. However, at high pedestrian flows the TLS-controlled intersection allows for higher vehicle flows because vehicles already waiting within the intersection may drive each time, pedestrians have to wait at the red light.

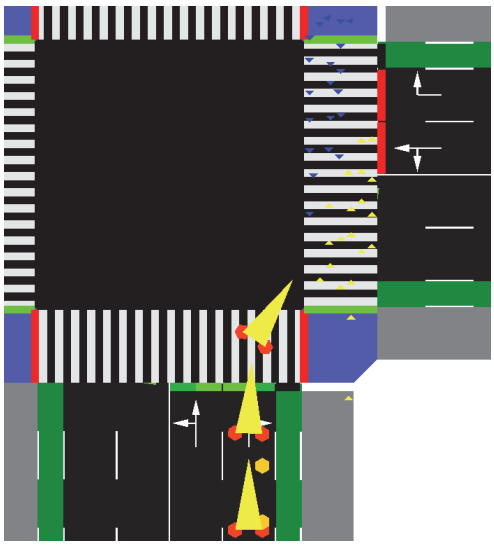


Figure 10-20 Simulation experiment for measuring the relationship between pedestrian flow and right turning vehicle flow (TLS-controlled intersection).

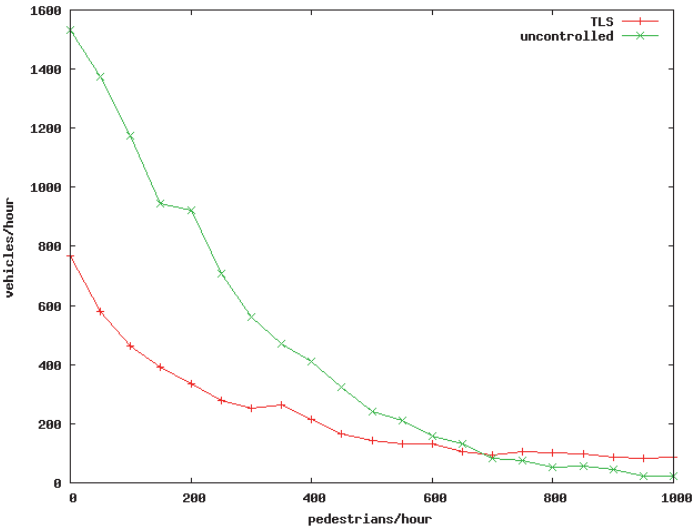


Figure 10-4 Flow of right-turning vehicles as a function of increasing pedestrian flow

10.4.2 Influence of pedestrians on an urban vehicular scenario

In this experiment an urban vehicular simulation scenario was extended with pedestrian traffic. The simulation scenario named *ACOSTA* [Bieker et al., 2013] comes from the iTETRIS project and models a part of the city of Bologna. It contains 9045 vehicle movements within the space of about 90 minutes in an area of 1.5km² and is characterized by high traffic density. The network model consists of 179 nodes and 182 edges. To extend this scenario, sidewalks and pedestrian crossings were added to the network model using the NETCONVERT options `--sidewalks.guess` and `--crossings.guess`. A total of 182 sidewalks (1 for each edge) and 164 pedestrian crossings were generated. Of these crossings, 52 are controlled by traffic lights. The existing traffic light programs were modified automatically to also cover the generated crossings. The green time allotted to vehicles remained unchanged. Pedestrian demand was generated randomly using the tool *randomTrips.py* described in section 10.3.7. 3600 pedestrians were generated which enter the network with a spacing of 1 second and then proceed to their destination along the shortest route. The scenario is shown in Figure 10-5.

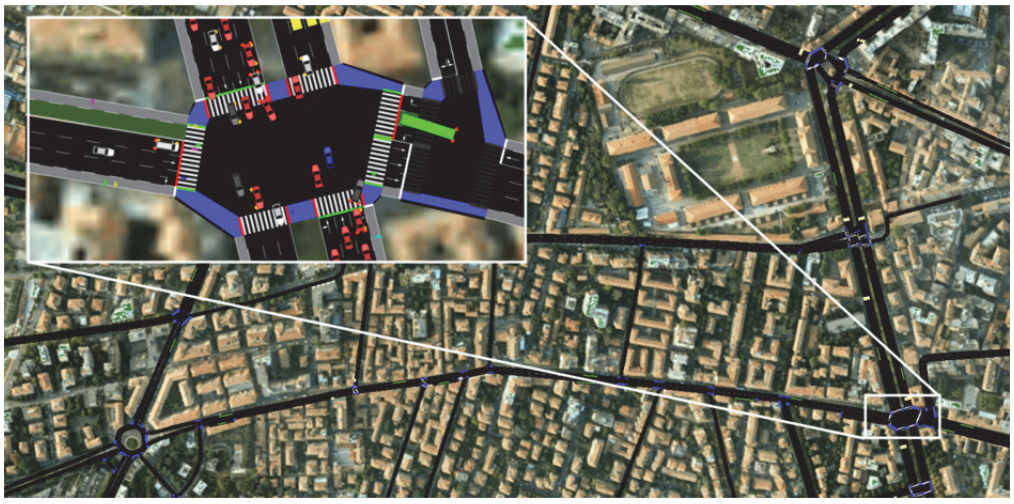


Figure 10-21 ACOSTA scenario with pedestrian enhancements. Pedestrians are shown at exaggerated size to increase visibility.

To measure the influence of the pedestrians on vehicular traffic, we compared the duration of vehicular trips in both versions of the scenario. Figure 10-6 shows the histogram of trip durations with a binning size of 60 seconds. It can be seen that the overall shapes of the distributions are similar but a small number of trips with much higher durations exist in the pedestrian scenario. The increased travel times were seen to be caused by the conflict of turning traffic with crossing pedestrians.

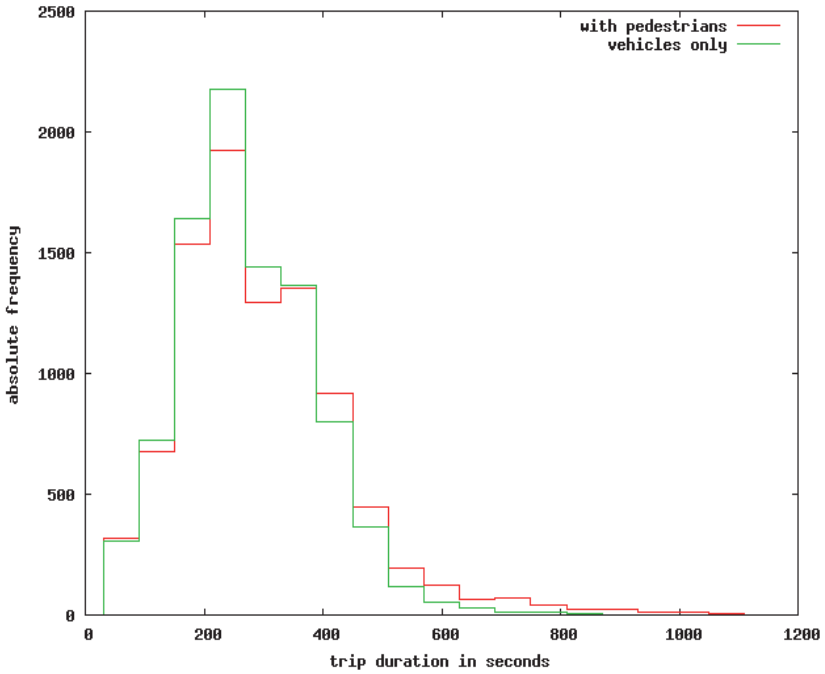


Figure 10-22 Histogram of vehicle trip durations in both versions of the ACOSTA scenario

We also compared the running times of both scenarios and noted that the average real-time factor (the fraction of simulated time over running time) decreased from 800 to 460 when adding pedestrians. When increasing the number of pedestrians to 9000 the real-time factor decreased further to 330. However, simulating 9000 pedestrians without vehicles still has a factor of 630. This shows that the pedestrian model has similar complexity to the vehicular model and a surprisingly large fraction of the time is spent on the interaction of vehicles and pedestrians.

10.5 Conclusion

An extension of SUMO for modelling pedestrians was presented. The work included modifications to several tools included in the SUMO package, which support the generation, simulation and analysis of multi-modal traffic scenarios.

The presented extensions allow a number of new investigations. While the major focus was put on evaluating the behavior of pedestrians at traffic lights, the implementation allows simulating pedestrians on a city-wide level. Being integrated into SUMO's inter-modal trip chains, it enhances SUMO by allowing microscopic modelling and evaluation of all (common) modes of urban transport.

The currently available models for both vehicle and pedestrian traffic are not yet fine-grained enough to address traffic safety research questions, but the inclusion of pedestrians into a

traffic simulation with a modular architecture is assumed to be an important step towards that goal

At last, one should point out that the inclusion of pedestrians – and the infrastructure (crossings) they use – influences the performance of motorized traffic as well. Therefore, the extensions not only extend SUMO's capabilities, but also improve its quality when the focus is on vehicular traffic.

10.6 Acknowledgements

The authors want to thank the European Commission for co-funding the work in the context of the "COLOMBO" project under the grant number 318622.

10.7 References

[1] Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D.: SUMO – Simulation of Urban MObility: An Overview. In: SIMUL 2011, The Third International Conference on Advances in System Simulation (2011)

[2] DLR and contributors: SUMO homepage. http://sumo.dlr.de/ (2015)

[3] SUMO online documentation on pedestrians 2015 http://sumo.dlr.de/wiki/Simulation/Pedestrians

[4] Krajzewicz, D.: Traffic Simulation with SUMO – Simulation of Urban Mobility. In: Barceló, Jaume (Ed.): Fundamentals of Traffic Simulation, Series: International Series in Operations Research & Management Science, Vol. 145, Springer, ISBN 978-1-4419-6141-9 (2010)

[5] [Antonini et al., 2006] Antonini, Gianluca, Michel Bierlaire and Mats Weber: Discrete choice models of pedestrian walking behavior, Transportation Research Part B: Methodological 40.8: 667-687, 2006.

[6] [Antonini, Berlaire, Schneider, Robin, 2009] Antonini, Berlaire, Schneider, Robin: Specification, estimation and validation of a pedestrian walking behavior model, In: Elsevier Transportation Research Part B: Methodological, Volume 43, Issue 1, Pages 36–56, January 2009.

[7] [pedsim] http://pedsim.silmaril.org/

[8] [Helbing, Molnár, 1995] Helbing, D., Molnár, P.: Social force model for pedestrian dynamics. Physical Reviews E,

[9] [Behrisch et al., 2010] Behrisch, Michael and Erdmann, Jakob and Krajzewicz, Daniel (2010) Adding intermodality to the microscopic simulation package SUMO. MESM 2010, 01-03.Dez. 2010, Alexandria, Agypten.

[10] [COLOMBO report D5.2, 2014] Erdmann, Jakob and Niebel, Wolfgang and Krajzewicz, Daniel and Härri, Jérôme and Bensator, Saleh (2014) COLOMBO: Deliverable 5.2: Traffic Simulation Extensions; Public Deliverable of the COLOMBO project; http://www.colombo-fp7.eu

[11] [Erdmann, Krajzewicz, 2014] Erdmann, Jakob und Krajzewicz, Daniel (2014) SUMO's Road Intersection Model. In: Simulation of Urban Mobility Lecture Notes in Computer Science, 8594. Springer. Seiten 3-17. ISBN 978-3-662-45079-6

[12] [Bieker et al., 2014] Bieker, Laura und Krajzewicz, Daniel und Morra, Antonio Pio und Michelacci, Carlo und Cartolano, Fabio (2014) Traffic simulation for all: a real world traffic scenario from the city of Bologna. SUMO 2014, 15.-16. Mai 2014, Berlin, Deutschland

11 Flood Impacts on Road Transportation Using Microscopic Traffic Modelling Technique

Katya Pyatkova[1], Albert S. Chen[1], Slobodan Djordjević[1], David Butler[1], Zoran Vojinović[2], Yared A. Abebe[2], Michael Hammond[2]

[1] Centre for Water Systems, University of Exeter , Exeter, United Kingdom
[2] UNESCO-IHE, Institute for Water Education, Delft, The Netherlands

{K.Pyatkova, A.S.Chen, S.Djordjevic, D.Butler}@exeter.ac.uk
{Z.Vojinovic, Y.Abebe, M.Hammond}@unesco-ihe.org

Abstract

The research presented in this paper proposes a novel methodology for modelling the impacts of floods on traffic. Often flooding is a complex combination of various causes (coastal, fluvial and pluvial). Further, transportation systems are very sensitive to external disturbances. There is insufficient knowledge on the interactions in these complex and dynamic systems. This paper proposes a methodology for integrating a flood model (MIKE Flood) and a traffic model (SUMO). Traffic on inundated roads will be interrupted or delayed according to the manner of flood propagation. As a consequence, some trips will be cancelled or rerouted and other trips will be indirectly affected. A comparison between the baseline and a flood scenario yields the impacts of that flood on traffic, estimated in terms of lost business hours, additional fuel consumption, and additional CO2 emissions. The outcome suggests that the proposed methodology can help to quantify the flood impact on transportation.

Keywords: microscopic traffic modelling, flood impacts, traffic disruption, flood modelling, model integration

11.1 Introduction

In order to consider flood impacts on traffic, general aspects of flood impacts should be addressed. Floods can impact human activities in many ways and this is why it is common to categorise these impacts. The flood consequences can be grouped as direct or indirect, tangible or intangible, or a combinations of both [1].*Direct* damages occur if the asset of interest is physically exposed to flood waters (i.e., buildings, people or environment). *Indirect* damages are outside the flooded area and usually become apparent after a longer time [2]. A classic example of indirect losses is the interruption of production in a firm that might occur due to a supplier affected by flooding. Traffic disruption due to floods is another indirect flood impact, the importance of which has not been studied in detail. The main reasons are: 1) the complexity of integrating two highly dynamic and uncertain systems; 2) the need to assess flood impacts in monetary terms (for the purposes of cost-benefit ratio). Flood impacts on traffic are often intangible: loss of time, frustration, environmental degradation due to additional CO_2 emissions. However, they can also have monetary dimensions: additional

operating costs and fuel consumption have market prices, and loss of time could be monetized as well. Approaches to monetize the intangibles and the emerging importance of multi-criteria analysis for hazard impact assessments create the necessary conditions for the proper evaluation of flood impacts on traffic.

To date traffic disruption due to flooding has received little attention. Comprehensive flood impact guidelines recommend carrying out traffic disruption study only if the expected traffic losses are significant, because otherwise the cost of traffic disruption is negligible compared to direct or indirect tangible costs [3]. It should also be noted that the importance of impacts on traffic (relative to other flood impacts) varies – in some cities (e.g. in Beijing) it is a major problem; but in other cities it is not so significant. So far the flood impacts on traffic have been approached using simple mathematical models [3] or macroscopic traffic models [4], [5]. None of these methods consider the dynamics of the transportation system, rerouting whilst a street is closed, or the dynamics of the flooding event. These methods represent a static system, which uses homogeneous aggregated traffic flows. The reliability of such models is not high, especially when it comes to simulating decisions in complex urban traffic networks. Microsimulation represents traffic congestion situations and bottlenecks more realistically, mainly through its algorithms incorporating drivers' responses and intermodal transportation [6], [7].

To date micro-simulation has not been used for computing flood impacts on traffic congestion and this is one of the main goals of this research. Another primary objective of this research is to introduce monetizing techniques not only for lost hours in traffic congestion, but also for the cancelled trips. Thus, the importance of flood impacts on traffic will be emphasised. From the modelling part an evident gap in the current research is the fact, that traffic models are not based on the duration and propagation of the flood. The methods introduced in this paper will address this dynamic behaviour of the system, through timely changes of the status of the links (open, closed, or with certain speed limit in accordance to the changes in flood depth).

11.2 Methodology

The proposed conceptual framework for incorporating flooding conditions into a microscopic traffic model is outlined in Figure 11-1. The impact of extreme hydro-meteorological events on transportation is twofold. First, the extreme weather conditions lead to reduced maximum speed limits [8], [9]. This impact will be driven by the intensity and the duration of the rainfall event and it will result in reduced road capacity before the flood has even occurred. Thus the flood impacts will start evolving in a transportation system, which already has reduced capacity due to heavy rainfall intensities.

Different combinations of intensities of rainfall and storm surges are simulated to produce the time varying flood characteristics. The consequent flood intensities in terms of flood extent, depth and propagation determine whether a street in the road network is going to be closed for traffic. This closure will affect both the overall road capacities and the trip definition components of the flood model. The trips that have an origin or destination in the flooded area will be cancelled and the routes that pass through a flooded area will be rerouted to unfavourable routes. A micro-simulation technique facilitates a better and a more detailed

representation of the traffic processes, compared to macro-simulation. There are several reasons to adopt a micro-simulation technique for the assessment of flood impacts:

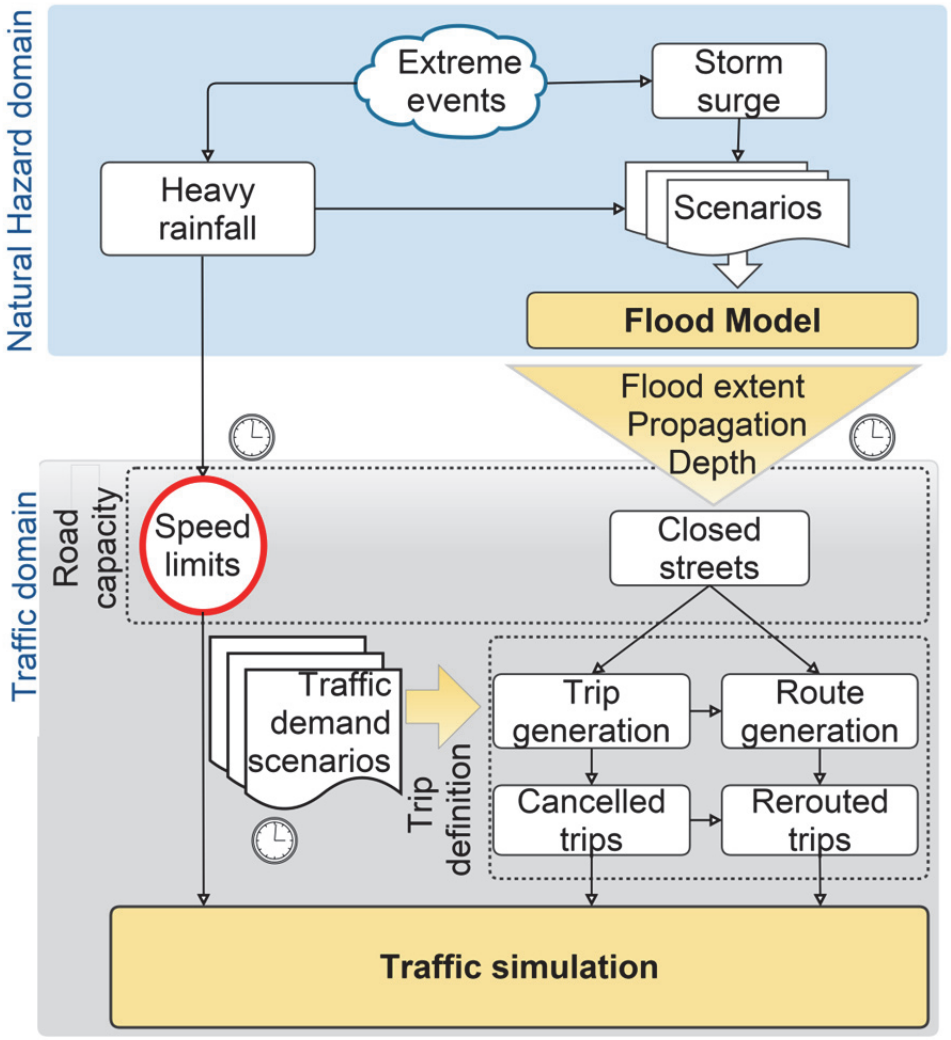


Figure 11-1: Flowchart of the proposed methodology

- When a street is closed due to flooding conditions, each vehicle will be rerouted individually, according to its destination. Hence, the rerouting algorithm ensures a detailed representation of the traffic condition during flooded conditions. This is particularly important if there are numerous flooded streets throughout the whole network;

- The micro-simulation technique is more reliable for the estimation of losses, related to the cancelled trips that will occur due to the flooding, because it contains a detailed description of each trip;

- The intermodal representation of different vehicle types is important for the overall consumption of fuel and greenhouse gas emissions. Different modes of transportation also indicate different cost of travel delays and will result in a more realistic representation of the flood impacts.

- Microscopic traffic models can simulate the dynamics of the flood propagation both in spatially and temporally. For instance, depending on the flood severity, it can allow closure of only one lane, whist keeping the traffic active in the other lanes;

At the end the results of the traffic simulations will be compared for scenarios with and without flooding. The whole procedure will be performed for different flooding scenarios, different times of the day (pick and off-pick times). As stated before the end results will be presented in absolute measures of lost business hours, additional fuel consumption, and additional CO2 emissions. The lost time and the additional fuel consumption will be also represented in monetary terms, so that they can be easily compared to the other type of flood losses and damages in the studied area. Ultimately, such an approach will allow for testing the effects of both flood risk management measures and of traffic improvement systems.

The model will be applied to a case study in a Caribbean island – St Maarten. This case study is considered appropriate for the research for two reasons: first, it has been a frequent victim of tropical storms and hurricanes and second, a closed road network system of an island helps assessing indirect impacts easier.

The following sections elaborate the hydraulic model, used to simulate the flooding conditions, the translation of flooding results into SUMO environment and the SUMO modelling setup.

11.2.1 Hydraulic model

The case study area of St Maarten is prone to tropical storms and hurricanes. Even small scale floods in the past posed a serious threat to traffic [10]. The hydraulic modelling is carried out on a catchment level for the most hazardous catchments in the island of St Maarten. The flood hazard characteristics (depth and velocity) are computed using DHI software MIKE FLOOD [11]. This software makes it possible to couple MIKE 11 (1D river model) and MIKE 21 (2D model, computing the flood plain and the coastal flooding). The results from the coastal flooding model are used as boundary conditions in the MIKE FLOOD simulation. This ensures integration between surface runoff and coastal conditions at each time step. The flooding conditions are simulated for different return periods of storm events, assuming independence of the rainfall and storm surge occurrence. The results of the hydraulic model provide maps for relevant flood depth over time, depending on the flood propagation at a particular site.

11.2.2 Translation of flood model results into SUMO model input

The time varying flood maps identify the streets that will be closed and the duration of the closure. This extraction is performed in a GIS environment by overlaying the flood map with a road network (Figure 11-2). The roadmap is a modified version of Open Street Map (OSM), which ensures all street types and speed limits are correct. The description of the roads in OSM does not allow a precise identification of the location of the flood because streets are represented with only one line. In order to avoid conversion discrepancies, a reliable translation of the link indices is desired. This is performed first by segregating major streets into edges in a GIS environment and second, by giving unique indices of the individual edges. The resultant shapefile is saved as an OSM network and then translated to SUMO, using the netconvert application. That way, no data will be lost in conversion, i.e. space varying speed limits, or number of lanes per street. This operation also established a linkage between the ArcGIS and SUMO environments, by using the same edge IDs. Thus a list of flooded streets is identified by their IDs in GIS environment and is readily available for rerouting in SUMO. To

provide consistency, the newly created road map is used for simulating traffic with and without flooding.

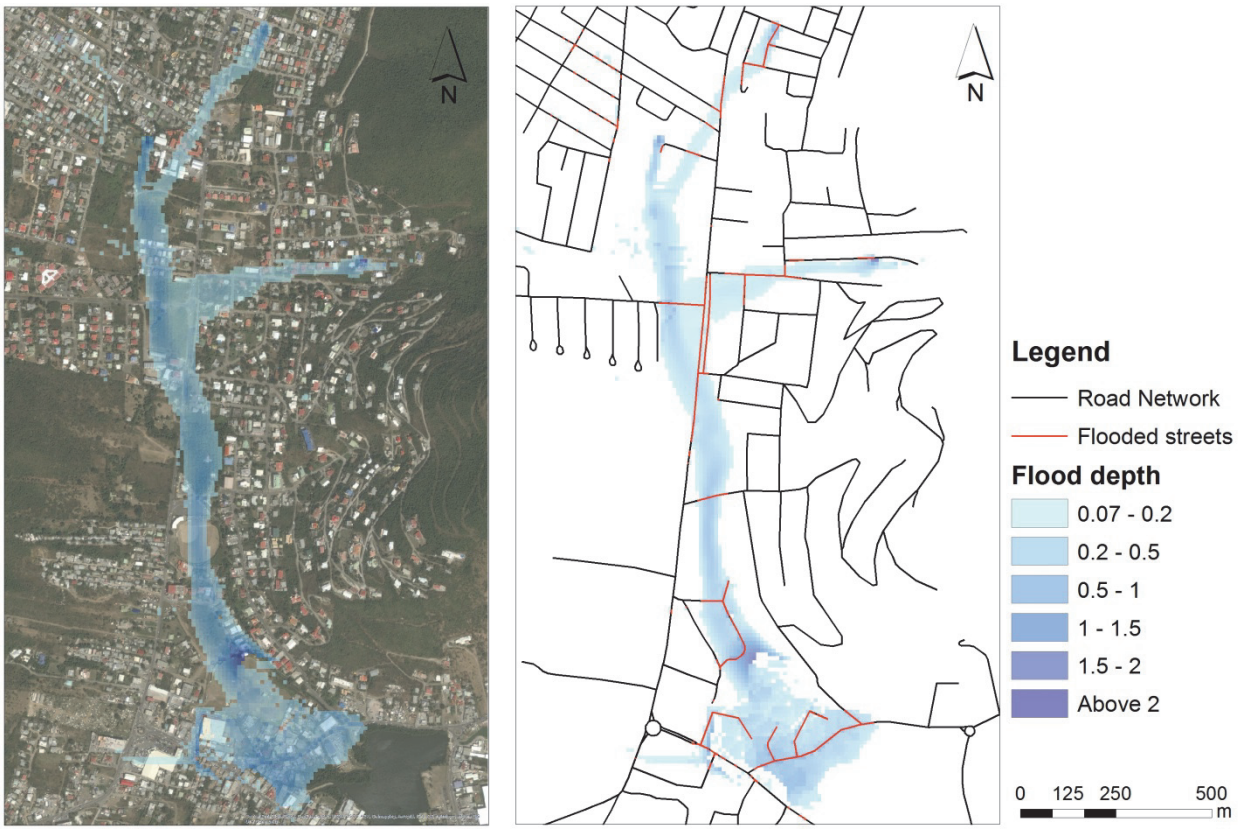


Figure 11-2: Flood map (left), and a road map overlaid with a flood map (right), showing in red the roads closed to traffic

11.2.3 SUMO parameters setting and traffic volume estimation

The SUMO software [12] has been used to create a basic model, so that the proposed methodology can be tested (Figure 11-3). The traffic model is limited by the reduced availability of transportation measurement data, but it is believed there is sufficient data with which to test the methodology. Currently the model uses the traffic network of the whole island of St Maarten, which is rather large for conventional microsimulation network (total area 34 km^2 and nearly 40 000 inhabitants).

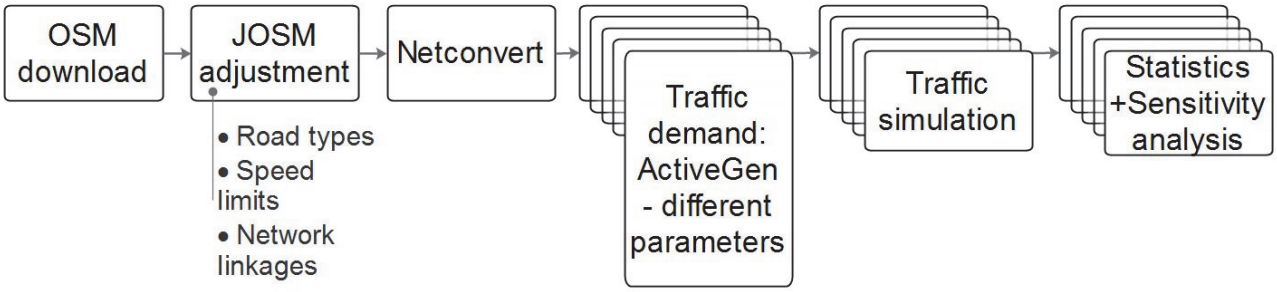


Figure 11-3: Flowchart of implemented traffic model

This network has been extracted from Open Street Map and later on modified for the purposes of SUMO, with special attention given to the different types of roads and their maximum speed limits. The traffic demand is based on a quasi-random route generation

(ActiveGen), based on statistics about settlements, population and the location of big employers. Different parameter combinations are used to run the model and obtain statistics for each scenario. The sensitivity analysis of the different scenario results can help improve the understanding of how the system functions. This strategy for computing traffic demand is hoped to help overcome the lack of data for calibration. The large number of quasi-randomly generated data approaches the traffic demand distribution from a probabilistic standpoint.

Another objective of the research is to investigate what the environmental impact of traffic congestion during floods can be. To achieve this, the SUMO model employed a simplified HBEFA classification of vehicles and their relevant CO2 emission levels for different engines. This model also provides a description of fuel consumption for individual trips in the simulation and thus can help monetize the impacts of floods on traffic congestion.

The actual flood impacts on traffic are estimated as a comparison between the results of a flooded and non-flooded situation for different scenarios of flooding, time of the day and traffic demand generation. The final result files provide statistics about each individual trip that has been computed. As stated before the main interest will be the difference in trip duration, fuel consumption and greenhouse gas emissions between the flooded and the baseline scenario.

11.2.4 Monetization of traffic delays

Previous studies in the field of flood impact to traffic congestion [4], [5] indicate that wasted time in traffic congestion will be the most significant flood impact to transportation. This imposes the need to monetize business hours lost in traffic, so that they can be compared to other tangible flood impacts as damage to built environment or business interruption. Value of time per individual person (driver or passenger) is defined by the purpose of the trip, mode of transportation and the type of vehicle [13]. The cost of the additional travel time can also depend on the duration of the delay. Interviews showed delays longer than 30 min have higher relative costs than shorter delays [14]. This research will employ a monetizing method which will consider a UK methodology [13] to estimate costs of travel times based on assumptions on average income.

11.3 Concluding remarks

This paper presents a novel methodology for assessing flood impacts on traffic. Micro-simulation traffic models have not been used yet to approach that problem, even though only a microsimulation model can capture the dynamics of both the natural and social-technological sphere.

The methodology presented in this paper is to be further confirmed by modelling results and supported by actual traffic measurements for calibration of the representation of traffic demand. The model for cost assessment of travel delays also needs to be adjusted to regional specifications of salaries in Saint Maarten.

The approach presented in this paper relates to off-line analysis of combined flood and traffic modelling. This methodology lends itself nicely for real-time modelling and decision making for coupled flood and traffic management systems.

11.4 References

[1] E. Penning-Rowsell, J. Chatterton, and E. P. Rowsell, "Assessing benefits the of flood alleviation and land drainage schemes," ICE Proc., vol. 69, no. 2, pp. 295–315, Jan. 1980.

[2] B. Merz, H. Kreibich, R. Schwarze, and A. Thieken, "Review article 'Assessment of economic flood damage,'" Nat Hazards Earth Syst Sci, vol. 10, no. 8, pp. 1697–1724, Aug. 2010.

[3] E. Penning-Rowsell, The benefits of flood and coastal risk management: a manual of assessment techniques / Edmund Penning-Rowsell ... [et al.]. London: : Middlesex University Press, 2010.

[4] H. Chang, M. Lafrenz, I.-W. Jung, M. Figliozzi, D. Platman, and C. Pederson, "Potential Impacts of Climate Change on Flood-Induced Travel Disruptions: A Case Study of Portland, Oregon, USA," Ann. Assoc. Am. Geogr., vol. 100, no. 4, pp. 938–952, Aug. 2010.

[5] P. Suarez, W. Anderson, V. Mahal, and T. R. Lakshmanan, "Impacts of flooding and climate change on urban transportation: A systemwide performance assessment of the Boston Metro Area," Transp. Res. Part Transp. Environ., vol. 10, no. 3, pp. 231–244, May 2005.

[6] D. Helbing, A. Hennecke, V. Shvetsov, and M. Treiber, "Micro- and macro-simulation of freeway traffic," Math. Comput. Model., vol. 35, no. 5–6, pp. 517–547, Mar. 2002.

[7] B. S. Kerner and S. L. Klenov, "Microscopic theory of spatial-temporal congested traffic patterns at highway bottlenecks," Phys. Rev. E, vol. 68, no. 3, p. 036130, Sep. 2003.

[8] D. Jaroszweski, L. Chapman, and J. Petts, "Assessing the potential impact of climate change on transportation: the need for an interdisciplinary approach," J. Transp. Geogr., vol. 18, no. 2, pp. 331–335, Mar. 2010.

[9] M. J. Koetse and P. Rietveld, "The impact of climate change and weather on transport: An overview of empirical findings," Transp. Res. Part Transp. Environ., vol. 14, no. 3, pp. 205–221, May 2009.

[10] UNDP, "Innovation and technology in risk mitigation and development planning in SIDS: Towards flood risk reduction in Sint Maarten. United Nations Development Programme, Barbados and the OECS," 2012.

[11] DHI, "MIKE FLOOD Modelling of Urban Flooding, A Step-by-step training guide." 2007.

[12] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO - Simulation of Urban MObility - an Overview," presented at the SIMUL 2011, The Third International Conference on Advances in System Simulation, 2011, pp. 55–60.

[13] R. Vickerman, "Evaluation methodologies for transport projects in the United Kingdom," Transp. Policy, vol. 7, no. 1, pp. 7–16, Jan. 2000.

[14] N. J. Douglas, L. J. Franzmann, and T. W. Frost, "The estimation of demand parameters for primary public transport service in Brisbane attributes," in Australasian Transport Research Forum (ATRF), 26th, 2003, Wellington, New Zealand, 2003.

12 Experiment of a common sense based-approach to improve coordination of Traffic Signal Timing System with SUMO

François Vaudrin and Laurence Capus
Laval University, Québec City, Canada

francois.vaurdrin.1@ulaval.ca, Laurence.Capus@ift.ulaval.ca

Abstract

Coordination of Traffic Signal Timing Systems has significant impacts on traffic congestion, waiting time, risks of accidents and unnecessary fuel consumption. But, despite more than 50 years of researches on Traffic Flow Theory, about half of Traffic Signal Timing systems in the U.S. are not updated on a regular basis as recommended by The Institute of Transportation Engineers (ITE). Also, the efficiency of traffic signals timing plans depends greatly on knowledge and judgment of circulation experts. Moreover this process still difficult to accomplish in real time.

Actually, we conduct a research that aims to change traffic lights programming without complex calculations at the right time (in real time). The mass of parameters to be considered and uncertainty caused by human behavior and the environment confirms our idea to opt for a quick and easy way to apply for better efficiency. But in order to prove this hypothesis, we must be able to simulate the most realistic traffic network. As part of our work, we use free software SUMO (Simulation of Urban MObility). This article describes the approach, the main problems and things that we have being learning so far.

Keywords: SUMO, simulation urban mobility, circulation management, Artificial Intelligence.

12.1 Introduction

A study conducted in 2004 by Tarnoff and al. [1] in more than 100 states, cities or agencies on behalf of The Institute of Transportation Engineers (ITE[9]) showed that more than half of traffic systems in the U.S. were badly synchronized or poorly maintained. ITE also recommends updating traffic signals every three to five years. However, 35% of systems are reprogrammed at a frequency of ten years or more. In the same study it was shown that the

[9] Founded in 1930, ITE is a community of transportation professionals including, but not limited to transportation engineers,

transportation planners, consultants, educators, and a network of nearly 17,000 members, working in more than 90

countries, http://www.ite.org/aboutite/ , accessed March 25, 2015.

average cost for the upgrade of traffic signals plans was 2 675$ or more per intersection. Assuming an average cost of $ 3 000 today, it would cost more than six million dollars to reprogram the system of a city like Montréal in Canada (2 000 static traffic signals).

Additionally, the process of programming static signals is complex. The ITE [2] recommends taking an average traffic volume during a given period and using these data to develop various programming plans until a satisfactory solution is obtained. Thereafter, it is implemented on site and revalidated by a traffic expert who makes the final adjustments required (Fine Tuning).

Also, this process is even more complicated to achieve and implement in real time. The proposed solutions are often costly and time consuming to implement, so little used. As part of our work, we aim rather to change a system of traffic lights at the right time and without complex calculation. The amount of parameters to be considered and the uncertainty caused by human behavior and the environment leads us to choose a workable and useful solution.

But in order to prove this hypothesis, we must be able to simulate a realistic traffic network. As part of our work, we use open free software SUMO[10]. This tool is developed by the Institute of Transport of the German Aerospace Center DLR[11]. SUMO is visual, well documented and free. Users can also rely on the support of the DLR Research Center and community of international researchers. We believe that this is a valid alternative of commercial software that seems generally used by public administrations.

The next sections describe the approach, the methodology, the main problems encounter and things that we have learned so far. As part of this conference, the focus is put on the process related to SUMO simulator.

12.2 Approach

Almost all of existing systems are static and are programmed accordingly to time of the day (ex.: peak hour morning or afternoon). But even during these periods, there are fluctuations in traffic flow and it is important that the system is adapted to these variations. If the system can react to pre-congestion and congestion situations, that can lead to improvements in travel time. Also, even small improvements can have substantial impacts in delay and fuel consumption; hence the importance of continuing research works.

Actually, we conduct a research that aims to change traffic lights programming without complex calculations and at the right time (in real time). We do not try to find optimal

[10] SUMO is a free and open traffic simulation suite which is available since 2001. SUMO allows modelling of intermodal

traffic systems including road vehicles, public transport and pedestrians.

http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/ , accessed March 25, 2015

[11] DLR is the national aeronautics and space research centre of the Federal Republic of Germany. Its extensive research and

development work in aeronautics, space, energy, transport and security is integrated into national and international

cooperative ventures. DLR has approximately 8000 employees.

http://www.dlr.de/dlr/en/desktopdefault.aspx/tabid-10002/#/DLR/Start/About, accessed March 25, 2015

solution every moment but to improve the situation the best as we can. Our assumption is that the summation of small improvements will have a significant effect over a reasonable period of time (few hours).

To get there we will guide us on the work of traffic officers (Figure 12-1). Miller [3] has shown that short-term retention of human brain is 7 ± 2 items. Despite these limitations, human being can manage traffic in case of system malfunction (or during special events). The human strategy is simple and is based on common sense. Traffic officers try to redirect traffic to free places and to be fair with road users. We aim to use a similar strategy to develop an intelligent system that would be able to propose solutions like a traffic officer.

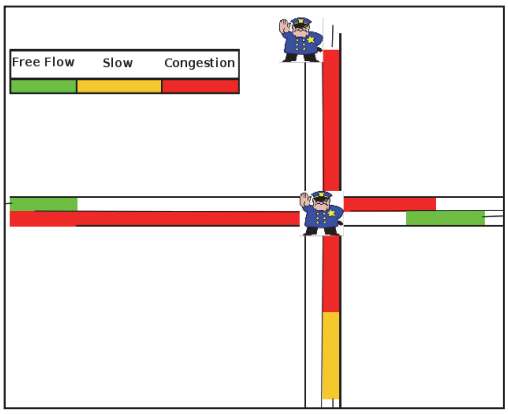


Figure 12-1. Traffic officers

There exist some studies that use intelligent systems but the context and the approach are different. To our knowledge, no other research uses the concept of human traffic officers to develop an intelligent traffic management system. We also seek to develop a system that can anticipate problems as does a human traffic officer. Our approach is based on common sense and we want to apply it on a larger number of intersections (10 or more). For example, Hossain and al. [4] propose a case-based system with libraries to identify recurring congestion situation on an artery with four intersections. Sadek and al. [5] developed a knowledge-based system for identifying incidents on a highway and divert traffic accordingly. Yang and al. [6] introduced a two-stage optimal combination fuzzy controller for traffic signals at isolated urban intersection. Wannige and al. [7] developed an adaptive neuro-fuzzy traffic signal control for multiple junctions. These works proposed intelligent solutions, but anyone does not try to implement the decisions done by traffic officers when they are on site.

However, the challenge is to translate this common sense behavior in a system. We propose to use the Density - Volume curve (Figure 12-2) and Speed - Volume curve (Figure 12-3) [8]. These two figures show that traffic must not exceed a critical threshold because the situation will deteriorate rapidly. If we look at Figure 12-2, we see that when the volume reaches the maximum critical point, the density is too high for the capacity of the road. And when it exceeds this point, the volume (or traffic flow) gradually decreases to the point of ultimate congestion (jam). This point is located at the right of the curve. This phenomenon is presented in Figure 12-3 by comparing the volume of flow and speed. When traffic is paralyzed (speed = 0), we have reached the point of extreme congestion. This point is located at the origin in Figure 12-3 and corresponds to the level of service E. The level of service is a standard of vehicle density of a road. The level of service A is a low vehicle density while E is the ultimate

density (the road is like a parking lot and the traffic is jam). There are five levels of service in figure 12-3 (A-B-C-D-E) and our approach is to avoid levels of service D and E.

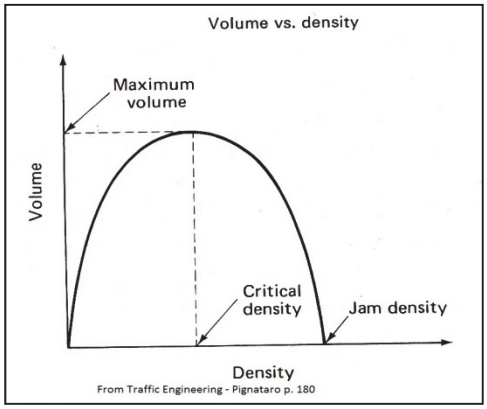


Figure 12-2. Volume vs. density

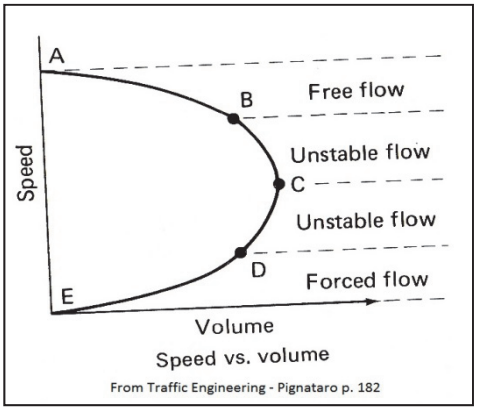


Figure 12-3. Speed vs. volume

In SUMO, a street network consists of nodes (junctions) and edges (streets connecting the junctions). Thus, if we want to create a network with two streets, subsequent to each other, we need three nodes and two edges [9]. Moreover, SUMO allows changing signal programming plan during the simulation with TraCI module that is part of the system. Also, it is possible to make a set of traffic lights program with it and to use them at the right time depending on the situation.

So, the aim is to calculate the density of each section (edge) and infer the level of service at every moment in each section of the network. This is the basic ingredient of our approach. Knowing the level of service of each section, we are able to decide what is the best decision as would do a traffic officer. This may result in an increase or a decrease in the duration of some traffic lights. In our approach, traffic lights replace the traffic officer.

Let's see an example. Suppose a set of sections (edges) and a set of service levels. For each edge, the system measures the level of service. By putting these service levels one after the other, one get a quick picture of the situation on the network. It is from this information that the decision to change the duration of traffic lights is taken. If one wants more accuracy, one just has to add dummy nodes.

Edges = $\{E_1, E_2, E_3 \ldots, E_n\}$

Level of services = {A, B, C, D, E}

Figure 12-4 shows a main road with different levels of services. In this situation, the obvious decision would be to reduce the arrival of vehicles to the edges that have already been reached the level of service D or E (the last ones). This situation is also described in figure 12-5 and a solution in this case could be to reduce the duration of green lights for the previous edges (AAAAA minus 20 seconds, BBBB minus 15 seconds, CCCCC minus 10 seconds). It will not solve congestion problem but will avoid extending the level of congestion in previous edges and it will reduce average waiting time for the secondary roads. This is one example and the idea is to apply similar reasoning for other situations. The goal is to keep it simple.

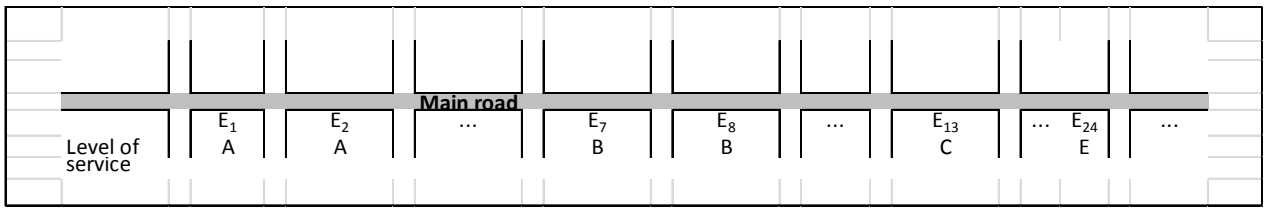


Figure 12-4. Edges and Level of service

E_1, E_2, E_3, E_4, E_5, E_6, E_7, E_8, E_9, E_{10}, E_{11}, E_{12}, E_{13}, E_{14}, E_{15}, E_{16}, E_{17}, E_{18}, E_{19}, E_{20}, E_{21}, E_{22}, E_{23}, E_{24}
A A A A A B B B B C C C C C D D D E E E E E E E

Figure 12-5. Edges and Level of service

12.3 Methodology

To test our approach, we use a network in Québec city in Canada. The network is located along the Sainte-Foy road (Figure 12-6). This is a busy area of Québec city with significant congestion problems. The studied network stretches over a distance of about 4 km. This network is made using free software OpenStreetMap (OSM[12]) and the Java application JOSM [13].

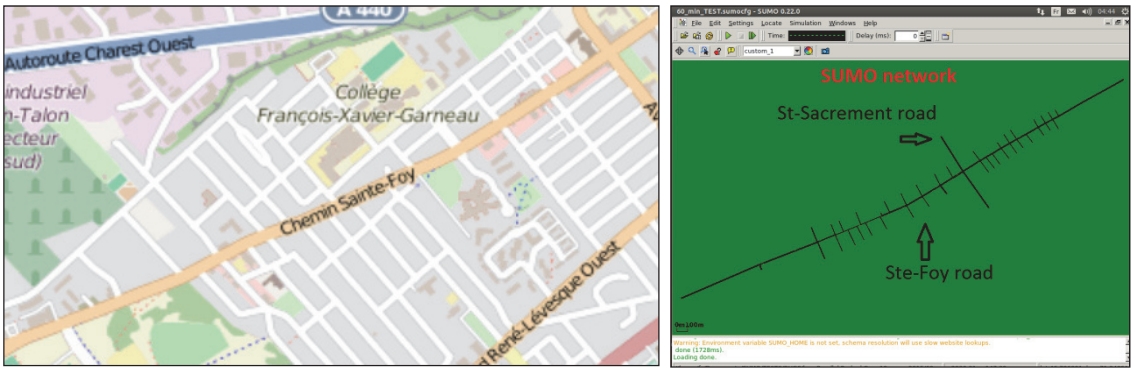


Figure 12-6. OSM and SUMO

The goal is to replicate as closely as possible the actual network and traffic light systems. The vehicles paths infer from the data collected in the field. We build a kind of Origin-Destination matrix from these data. Data were measured with the short count method from 5 to 30 November 2012. The short-count method evaluates the number of vehicles that pass each intersection for 15 minutes and then extrapolate the results for one hour. Each intersection was observed during the peak period of the afternoon between 16:00 and 18:30. In addition, the signaling systems in place were measured (duration of each cycle to each intersection). Geometric data were collected from the OSM map and validated in the field. Parallel parking

[12] OpenStreetMap (OSM) is a collaborative project to create a free editable map of the world, https://en.wikipedia.org/wiki/OpenStreetMap, accessed March 25, 2015.

[13] JOSM is an extensible editor for OpenStreetMap written in Java 7, https://josm.openstreetmap.de/, accessed March 25, 2015.

areas along the Sainte-Foy road (and time permit parking), speed limit and bus stops have been identified in the field. Schedules are those of the bus transport network of Québec city (lane 7). We have not considered pedestrians.

The first step was to develop the network from an OSM map and JOSM software. Thereafter, the map has been transferred and converted to be usable in SUMO. There are roughly 70 knots, 140 edges, 40 secondary roads and 17 signalized intersections in the network. Each node and each edge of the network is identified by an ID number (--21111#0, -21408#0, --25012, 1690707806 ....). Nevertheless, there is no logical order in the list of IDs and this can complicate the analysis. During simulation there is a continual back and forth between the screen and the xml files and it is better to easily find the information. So, to facilitate identification of nodes and edges, a Python program was carried out and IDs have been changed (Figure 12-7). This program is based on manipulation of array lists and regular expressions. We think it is important because it would speed up analysis. This approach also reduces the risk of errors. The nodes on the main road are now classified from West to East or left to right (100, 200, 300, 400, 500 .... 2600) and those on the secondary roads are incremented by 1000 according to the x coordinate of each terminal node (5000, 6000, 7000 ... 44000). A similar approach was used for the edges.

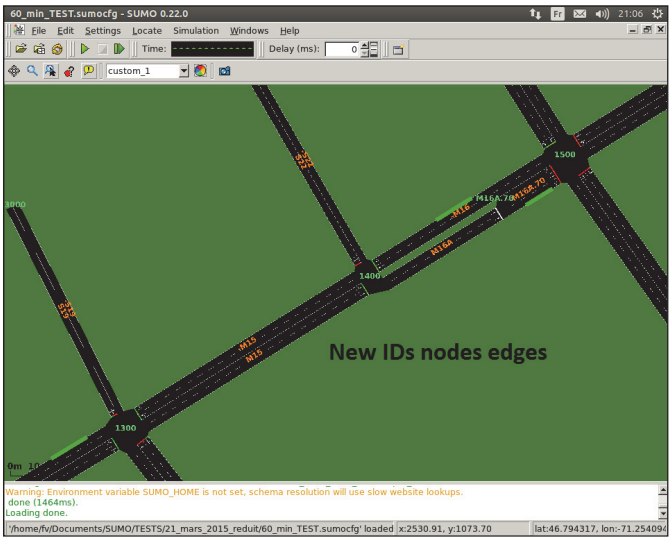


Figure 12-7. New designation IDs nodes edges

When the xml nodes and edges files have been completed, the files describing lanes and vehicles have been established. To simplify, we used flow paths rather than determining a route for every vehicle. In comparing data counted at each intersection, it was possible to achieve relatively realistic approximations of displacement. Since there is only one main road and perpendicular roads, it was easy to figure on the trips. The number of trucks was approximated to 5%. Parallel parking cars (Figure 12-8) along the main road were put at the beginning of each simulation in specific edges (and they do not move afterwards). However, there was a problem during the simulation caused by parked cars and a way to bypass this issue is to remove a lane on these edges. The duration of traffic signals generated by SUMO and OSM were modified to match traffic lights system in the field.

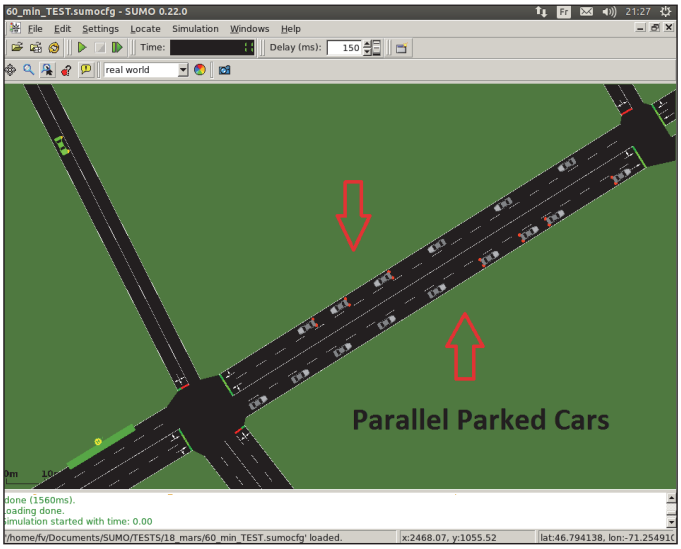


Figure 12-8. Parallel Parked Cars

12.4 Experimentation and results

Several tests were performed to validate the simulation. The main congestion problem is at the intersection of St-Sacrement road (Figure 12-9) and it reflects the observation in the field. We doubt that we can resolve major congestion problems like that but we hope to be able to prevent as much as possible such situations.

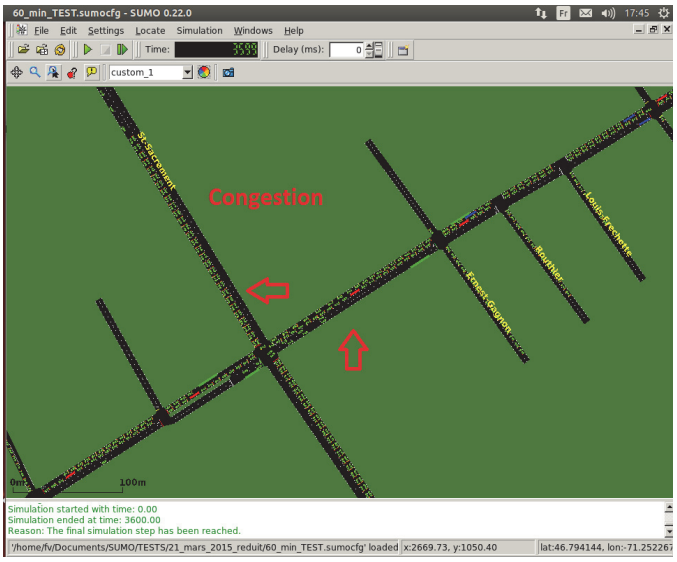


Figure 12-9. Congestion St-Sacrement road intersection

SUMO allowed us to make several manual tests quickly to validate partly our hypothesis. We recall that the strategy is to build on the work of a traffic officer. We found that cars queues on some secondary roads improperly stretched during the simulation. Tests were then carried out to check if it was possible to reduce the length of cars queues with varying cycle times at specific intersections. We found that when there was free space (or less density) on neighbor edges, it was possible to quickly improve the situation without getting worse the cross-road. For example by increasing the cycle of green light four seconds on a busy cross road it empty its queue every cycle and does not appear to cause serious problems on the main road

(Figure 12-10). If we can apply that concept for short periods elsewhere, we believe it is very likely that the overall result will be improved.

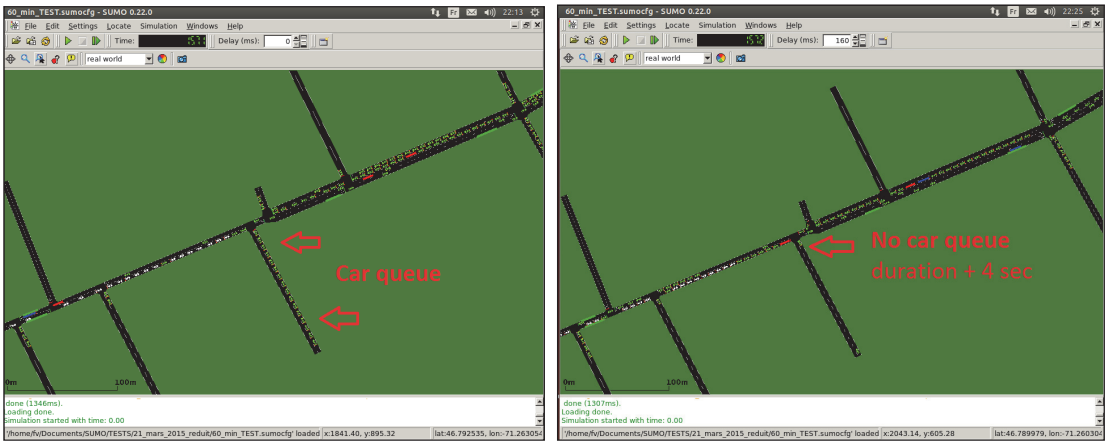


Figure 12-10. Alteration of duration of green light + 4 sec

So, the results look promising so far but we have not at this stage build decision trees to change the duration of the traffic lights on every signalized intersection depending the vehicle density on neighbor edges. This program in Python language must interact with the simulator SUMO via TraCI module which operates on the base of client-server system. So, the next step will be to develop this intelligent system that will interact at the right time with the simulator.

12.5 Conclusion

In this research project the first steps were to collect data, understand how the simulator work and develop a realistic network. As part of the work we had to learn SUMO and OSM software. We found that SUMO is relatively easy to learn even if it requires perseverance and training. We estimate a few months learning time before understanding the tutorials and be at ease with the simulator. However, once this stage is reached, the experiments are relatively easy to implement.

We believe SUMO best advantage is that it can be easily parameterize via xml files system. In addition, users can benefit support from the research center DLR and a community of researchers via the newsgroup sumo-users. Most questions are answered the same day which is certainly comparable if not better than services provided by commercial software vendors. Also it is free which is definitely a plus.

One advantage of using OSM maps is to develop a network quickly. However, we recommend beginners to enlist the support of people who know that open source project. This is what we have done by contacting the OSM project responsible in the Québec city area and it allowed us to save a lot of time with small ad hoc training. We also recommend taking time to simplify the nomenclature of nodes and edges when transferring data from OSM to SUMO. This may appear trivial but it simplifies the analysis and prevents errors. These adjustments also allow saving a lot of recurring waste of time if one plan to use the network for months or years.

The advantage of a program like SUMO is that it can quickly test a hypothesis. We can not at this stage certify that our approach is valuable but preliminary results lead us to believe that this track is promising. We are optimist that it will be possible to demonstrate this approach

during pre-congestion stage. It remains to build a decision tree and to link with the simulator via TraCI module. We also need to verify that the change in the duration of traffic lights in some places does not move the problem elsewhere on the network.

Finally, in traffic management each case is unique and quality of solution relies heavily on the experience of the expert. So one has to be careful and even if a solution seems attractive, field validation is essential.

12.6 References

[1] Philip J. Tarnoff, Javier Ordonez: Signal Timing Practices and Procedures, State of the Practice, University of Maryland, Center for Advanced Transportation Technology, Maryland (2004)

[2] Institute of Transportation Engineers, Traffic Signal Timing Manual, Washington (2009)

[3] George A. Miller: The Magical Number Seven Plus or Minus Two: Some Limits on our Capacity for processing Information, p. 81-97, Harvard University-Psychological Review, (1956)

[4] Shahadat Hossain, Lina Kattan, Ahmad Radmanesh: Responsive Signal Control for Non-recurrent Traffic Congestion on an Arterial, TRB 90th Annual Meeting, Washington (2011)

[5] Adel Sadek, Spencer Morse, Spencer Morse, Wael El–Dessouki: Case-Based Reasoning for Assessing Intelligent Transportation Systems Benefits, Computer-Aides Civil and Infrastructure Engineering, Blackwell Publishing, USA (2003)

[6] Wenchen Yang, Lun Zhang, Zhaocheng He, Yuchen Yang, Yungen Fang: Urban traffic signal two-stage combination fuzzy control and Paramics simulation, p. 771-775, IEEE explore Digital Library (2012)

[7] Wannige C.T., Sonnadara D.U.J., Adaptive neuro-fuzzy traffic signal control for multiple junctions, p. 262-267, International Conference on Industrial and Information Systems (ICIIS) (2009)

[8] Louis J, Pignataro., Traffic Engineering - Theory and Practice, p. 180-182, Prentice-Hall, New Jersey (1973)

[9] Tutorials/Hello Sumo, Hello Sumo – Introduction, http://sumo.dlr.de/wiki/Tutorials/Hello_Sumo (2015)

13 eNetEditor: Rapid prototyping urban traffic scenarios for SUMO and evaluating their energy consumption

Tamás Kurczveil, Pablo Álvarez López;
Institut für Verkehrssicherheit und Automatiosierungstechnik, Technische Universität Braunschweig, Hermann-Blenk-Straße 42, 38108 Braunschweig, Germany

kurczveil@iva.ing.tu-bs.de, p.alvarez-lopez@tu-braunschweig.de

Abstract

The increasing mobility and transport demand and the sinking global supply of fossil energy carriers will eventually cause a growing trend towards alternative drive concepts and the development of corresponding energy supply infrastructures. These emerging solutions and their interaction with the prevailing traffic will need to be evaluated for their optimal integration. SUMO is a preferred tool when it comes to evaluating measures in urban traffic behavior. When using SUMO, however, the creation of corresponding scenarios is accompanied by challenges in network creation and corrections as well as traffic demand generation and calibration. Motivated by the projects emil and InduktivLaden, both funded by the German Federal Ministry of Transport and digital Infrastructure, this paper presents the newly developed tool eNetEditor, which allows users the rapid prototyping of custom and calibrated traffic scenarios based on traffic counts and their evaluation in regard of energy consumption.

Keywords: network generation, traffic assignment and calibration, energy consumption

13.1 Introduction

Current traffic is mostly driven by fossil fuels. In 2014, the number of newly registered vehicles in Germany was 3.048.507 References

[1]. 8522 of these vehicles (2.8 ‰) were electric vehicles [2]. The same statistic indicates that the number of electric vehicles is expected to rise in the coming years. An exponential extrapolation of these numbers yields an approximate range of 14283 to 27041 newly registered electric vehicles for the year 2015.

One of the current challenges for the customers of electric vehicles is the multiplicity of charging systems and interfaces. Whereas 14622 gas stations in Germany supply conventional traffic with the adequate amount of fossil fuel [3], only 5050 publicly available electric charging points have been installed for electric vehicles in Germany until June 2014 [4]. Many manufacturers offer proprietary solutions, tieing customers to a limited amount of available charging stations, such as Tesla. In other cases, countries or regions have agreed on and effected the installation of a more consistent charging infrastructure, such as CHAdeMO in Japan and France or CCS in Germany. This heterogeneous situation has resulted in charging station infrastructures of several different manufacturers and providers with varying and

largely incompatible standards. In addition to this heterogeneous situation with the charging infrastructure, new developments indicate that battery-driven electric vehicles might not remain the only alternative solution for the coming years: Most of the leading automobile manufacturers have invested many efforts into the development of hydrogen fuel-cell vehicles over the past years. Toyota, for example, has introduced one of the first of these vehicles to be sold commercially starting in 2015. This further results in a comparably increasing diversity regarding drive train concepts.

However, even with these trends, the development of new compatible charging infrastructures is still at its beginning. This situation is a chance for the introduction of more homogeneity in the charging infrastructure and, as opposed to the location of current gas stations, its optimal operational integration into prevailing traffic situations. For this unification of the energy supply infrastructure, traffic planners will need tools in the future that take into account the energy consumption of vehicles along their routes allowing the optimal positioning of components for the corresponding energy supply infrastructure. These results can further be used for infrastructure operators to determine the amount of energy that electric vehicles are expected to gain at specific locations within the road network and how much power will be required for their sufficient supply.

This paper introduces eNetEditor: a tool with a graphical user interface for traffic planners that allows the rapid prototyping of arbitrary road traffic networks and calibrated scenarios based on flow measurements and the evaluation of the corresponding energy consumption.

13.2 Concept

For the simulation of vehicles within the road network and their interaction with infrastructure objects and other vehicles, the microscopic traffic simulation tool SUMO (Simulation of Urban Mobility) is used [5]. SUMO is "an open source, highly portable, microscopic and continuous road traffic simulation package designed to handle large road networks" [5]. The development of SUMO was initiated by the Institute of Transportation Systems of the German Aerospace Center (DLR), in 2001. It has evolved into a traffic simulation tool, high in features, functionality and interfaces. Even though instantiated vehicles follow a simplified behavior, traffic simulation tools like SUMO allow the realistic replication of prevailing traffic in arbitrary road networks.

Implementations were presented in [6] that newly introduced a simplified energy consumption model for vehicle objects and a corresponding declared charging station class as integral parts into SUMO. These implementations build the functional basis for the evaluation of the energy consumptions in traffic scenarios.

The application of these implementations subsequently require a traffic scenario consisting of a road network and traffic demand. Whereas a common procedure is to use open data to map transportation networks, this process is often accompanied by corrections that, if irregularities are found, are hard to carry out without the proper tools. In order to instantiate calibrated traffic in arbitrary road networks, eNetEditor (initially, a MATLAB-based tool) has been developed with a graphical user interface that allows the rapid and efficient generation of road networks. Network object data were structured such that they contain the properties required by SUMO's network generator netconvert.exe (i.e. edges, lanes, nodes, connections),

by SUMO itself (e.g. vehicle types, bus stops), and arbitrary custom parameters and their values (e.g. traffic counts or other traffic demand data) for user-defined functions. The data structure for the network creation is outlined in section 13.13.3.1. After the definition of vehicles, traffic can be instantiated and calibrated from within this tool as described later in sections 13.13.3.2 and 13.13.3.3, respectively.

Using traffic data and measurements from different sources, such as flows from induction loops, eNetEditor allows the generation of traffic demand. The goal is a simulation output in form of a structure that can be used to represent energy consumption of individual vehicles over time and vehicle position or along individual lanes of a road network over time. An example urban traffic scenario will be shown in chapter 13.4 that was generated by eNetEditor, accompanied by evaluations in regard of its constituents' energy consumptions.

13.3 Implementation

This chapter explains the structure of eNetEditor's implementations. The modules are split into three parts:

1. network definition and generation,
2. vehicle definitions and generation, and
3. traffic demand generation and calibration.

The following sections in this chapter give an overview on the implementation of each of the modules and the used data structures that broadly represent SUMO's object structures.

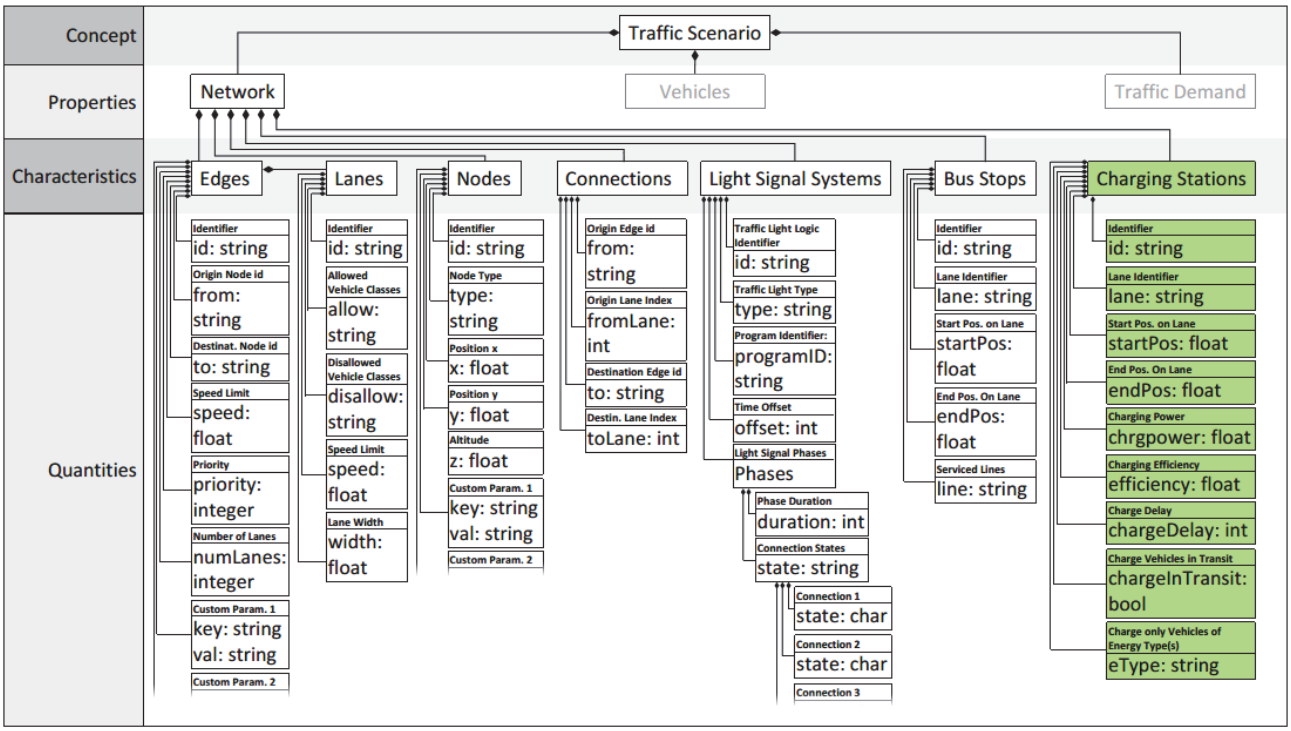


Figure 13-23: Constituents of a traffic scenario's network represented in form of a UML class daigram

13.3.1 Network definition and generation

A SUMO Network consists of edges, nodes, lanes, vehicle class restrictions on specified lanes, lane connections (and prohibitions), bus stops, and light signal systems. As a new infrastructure element, charging stations have been introduced. Figure 13-23 shows the structure of these constituents as definable in eNetEditor, axiomatizing the terms in form of a class diagram as introduced in [7] and [8]. The following sections will describe the data structure for each network constituent that can be represented in eNetEditor, the interface for their generation, and exemplified output data. Newly introduced classes and attributes are shown in green.

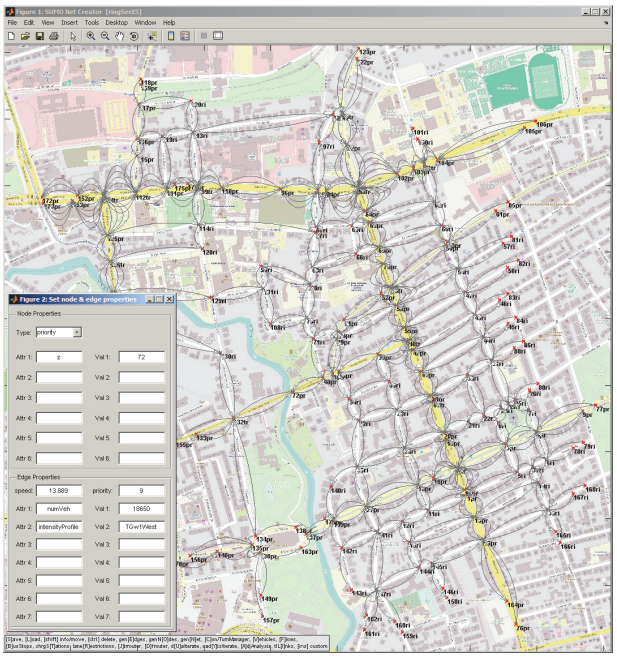


Figure 13-24: eNetEditor's graphical user interface

13.3.1.1 Nodes and edges

Nodes and edges constitute the essential components of a network's graph. These can be created directly in the GUI with a mouse click or by connecting two nodes with a click-and-drag. To allow efficient analyses and operations with the resulting multi-edged directed graph and its adjacency matrix, edges and their properties are stored in the 3-dimensional array E of the following structure

$$E^{n \times n \times (2m+1)} = (e_{i,j,k}), \text{ with } \ldots \quad (1)$$

$$\begin{aligned} \text{origin node index} \quad & i = 1 \ldots n, \\ \text{destination node index} \quad & j = 1 \ldots n, \\ \text{property index} \quad & k = 1 \ldots 2m + 1, \\ \text{number of nodes} \quad & n \text{ and} \\ \text{number of edge properties} \quad & m. \end{aligned}$$

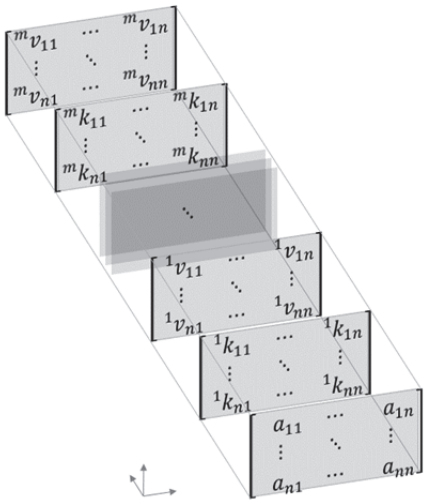
The entries $(e_{i,j,k=1})$ represent the network graph's adjacency matrix, where each element $e_{i,j,k=1}$ equals the number of lanes from node i to node j. Edge property and corresponding value pairs are stored in successive entries of $E_{i,j,k=2,4,6,\ldots}$ (property name) and $E_{i,j,k=3,5,7,\ldots}$ (property value) in form of strings. Whereas SUMO and its netconvert application will only process the values of specified property tags, eNetEditor allows the declaration of arbitrary properties and property values that can be used for eNetEditor-/MATLAB-based pre-/post-processing.

Nodes and their properties are stored in the 2-dimensional array O with the following structure

$$O^{n \times 18} = (o_{i,k}), \text{ with } \ldots \quad (2)$$

$$\begin{aligned} \text{node index} \quad & i = 1 \ldots n, \\ \text{property index} \quad & k = 1 \ldots 18 \text{ and} \\ \text{number of nodes} \quad & n. \end{aligned}$$

Next to mandatory property values in $o_{i=1\ldots n,\, j=1\ldots 6}$, the remaining entries in O ($o_{i=1\ldots n,\, j=7\ldots 18}$) can be used for the declaration of arbitrary node properties and their corresponding values. The structure of the edge and node variables E and O is depicted In **Fehler! Verweisquelle konnte nicht gefunden werden.**.



Node	position		id	[...]	type	[...]	6 custom property keys and values		
	x	y							
1	x_1	y_1	id_1	reserved for handlers	t_1	reserved for handlers	${}^1k_1, {}^1v_1$	⋯	${}^6k_1, {}^6v_1$
2	x_2	y_2	id_3		t_2		${}^1k_2, {}^1v_2$	⋯	${}^6k_2, {}^6v_2$
3	x_3	y_3	id_2		t_3		${}^1k_3, {}^1v_3$	⋯	${}^6k_3, {}^6v_3$
	⋮	⋮	⋮		⋮		⋮	⋱	⋮
n	x_n	y_n	id_n		t_n		${}^1k_n, {}^1v_n$	⋯	${}^6k_n, {}^6v_n$
	2 × float		string	float	string	float	2 × string	⋯	2 × string

Figure 13-25: Data structure of edge (left) and node (right) description arrays O

Figure 13-24 shows eNetEditor's user interface for the creation of nodes and edges with arbitrary property keys and property values. The output in the specified XML-format, as required by SUMO's netconvert.exe, is exemplified in Listing 1 and Listing 2. The corresponding node and edge xml-files are created by parsing the described data structures and can be initiated by a keyboard entry of the buttons o and e respectively.

Listing 1: Example edge definition file in the SUMO specified xml-format (typically .edg.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<edges>
  <edge from="1" id="1to5" intensityProfile="TGw3West" numLanes="1" numVeh="1050"
     priority="4" speed="8.333" to="5"/>
  <edge from="1" id="1to11" intensityProfile="TGw3West" numLanes="1" numVeh="1050"
     priority="4" speed="8.333" to="11"/>
  <edge from="1" id="1to13" intensityProfile="TGw1West" numLanes="2" numVeh="14900"
     priority="9" speed="13.889" to="13"/>
  ...
</edges>
```

Listing 2: Example node definition file in the SUMO specified xml-format (typically .nod.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<nodes>
  <node id="1" type="priority" x="1106.15" y="339.02" z="0"/>
  <node id="2" type="traffic_light" x="1074.55" y="419.47" z="0"/>
  <node id="3" type="priority" x="1221.08" y="485.55" z="0"/>
  <node id="4" type="right_before_left" x="1347.50" y="443.89" z="0"/>
  ...
</nodes>
```

13.3.1.2 Connections

Every edge in SUMO represents a road, on which vehicles can travel into one direction. A bidirectional street in SUMO would therefore be modeled by two edges. Each edge further consists of one or more lanes. Most nodes in a network have at least one incoming and one outgoing lane. When generating a network, netconvert will make assumptions about these connections. However, many complex junctions' structure will deviate from the assumptions made by netconvert's generalized heuristics. Figure 13-26 shows such an irregular junction with many specific lane-to-lane (red) connections. To allow these kinds of junction definitions and for their correct representation and functionality, connections between nodes' (junctions') incoming and outoing lanes often need to be specified explicitly.

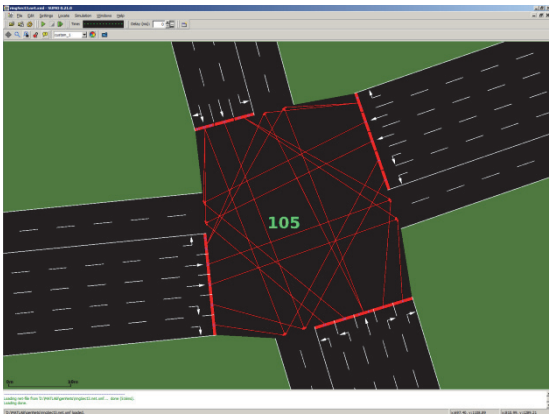


Figure 13-26: A complex junction structure in SUMO with irregular connections

These connections can be specified in SUMO by explicitly declaring them in a separate connection file in xml format, typically .con.xml. Since connections are lane-specific definitions, each connection possesses 6 degrees of freedom: the incoming lane's parameters, specified by (1) its origin and (2) destination node and (3) its lane index and the outgoing lane's parameters, specified by (4) its origin and (5) destination node and (6) its lane index. The incoming edge's destination and the outgoing edge's parameters (2) and (4) will most often be identical.

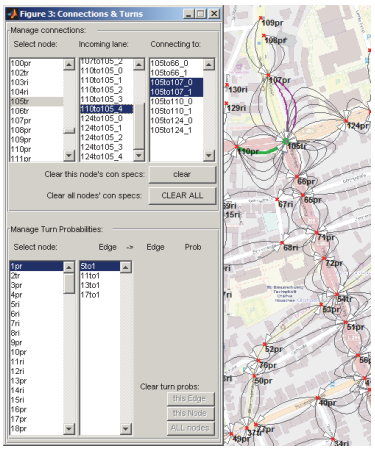


Figure 13-27: Connection input, origin node,

The chosen data format for the connection specification is a 3-dimensional array C, where each element $c_{i,j,k}$ represents an incoming lane and contains a list of connections to outgoing lanes. The connection variable C has the following structure:

$$C^{n \times n \times l} = (c_{i,j,k}), \text{ with } \ldots \tag{3}$$

incoming lane's origin node index $i = 1 \ldots n$,
incoming lane's destination node index $j = 1 \ldots n$,
incoming lane's index $k = 1 \ldots l$,
number of nodes n and
lane number of the edge with the most lanes l .

Whereas the indices i, j, and k of each element in $(c_{i,j,k})$ represent an origin lane, each element $c_{i,j,k}$ itself is a list of connecting destination lanes:

$$c_{i,j,k} = \begin{bmatrix} r_1 & s_1 & t_1 \\ r_2 & s_2 & t_2 \\ \vdots & \vdots & \vdots \\ r_x & s_x & t_x \end{bmatrix}, \text{ with } \ldots \tag{4}$$

outgoing lane's origin node index $r = 1 \ldots n$,
outgoing lane's destination node index $s = 1 \ldots n$,
outgoing lane's index $t = 1 \ldots l$,
number of nodes n,
lane number of the edge with the most lanes l and
number of incoming lane's connections x.

Connections will be guessed by netconvert if incoming and outgoing edges exist and no lane connections are specified for a node. Connections can also be specified in eNetEditor using the dialog box shown in Figure 13-27, which can be called by the keyboard entry of the button ***c***. The resulting xml file (typically .con.xml) is automatically generated after closing the dialog window. The example of a resulting xml is shown in Listing 3.

Listing 3: Example connection definition file in xml-format (typically .con.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<connections>
  <connection from="110to105" fromLane="0" to="105to66" toLane="0"/>
  <connection from="110to105" fromLane="1" to="105to66" toLane="1"/>
  <connection from="110to105" fromLane="2" to="105to124" toLane="0"/>
  ...
</connections>
```

13.3.1.3 Lanes

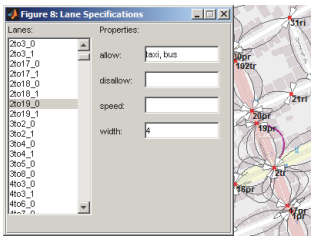


Figure 13-28: Lane-based specifications input dialog

Next to connections, lanes can be attributed with further specifications that serve two purposes: visualization and lane-based restrictions or allowances. eNetEditor allows the specification of these introduced parameters by calling the dialog window (shown in Figure 13-28) with the keyboard entry of button ***r***. The corresponding variable *R* consists of a list implemented as a 2-dimensional array

$R=(r_i)$ with … (5)
lane specification i r_i,
lane specification index $i = 1 \ldots r_{ln}$ and
total number of lane specifications ln,

where element r_i contains all lane-based parameters for the corresponding lane:

$r_i = \left[{}^{L}id_i \quad a_i \quad d_i \quad s_i \quad w_i\right]$, with … (6)
identifier of lane l_i ${}^{L}id_i$,
allowed vehicle classes on lane l_i a_i,
disallowed vehicle classes on lane l_i d_i,
speed limit on lane l_i s_i (in $^{m}/_{s}$) and
width of lane l_i w_i (in m).

The parameters *allow* and *disallow* refer to the abstract vehicle parameter *vehicle class*. For the intended functionality, vehicles need to be assigned the corresponding *vehicles class* parameter, as outlined later in section 13.13.3.2.

If specified, lane-based specifications require a subsequent generation of the edge descriptions. Further notes regarding on the order of generating individual components of the

network can be found later in subsection 13.13.3.1.7. An example output is shown in Listing 4 that contains lane-based specifications in the edge definition file (typically .edg.xml).

Listing 4: Example edge definition with lane-specific parameters in xml-format (typically .edg.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<edges>
  …
  <edge from="2" id="2to17" numLanes="2" numVeh="17050" priority="9" speed="13.889"
     to="17"/>
  <edge from="2" id="2to19" numLanes="2" numVeh="18600" priority="9" speed="13.889"
     to="19">
      <lane allow="bus, taxi" index="0" width="4"/>
  </edge>
  <edge from="3" id="3to5" numLanes="1" numVeh="975" priority="4" speed="8.333"
     to="5"/>
  ...
</edges>
```

13.3.1.4 Bus stops

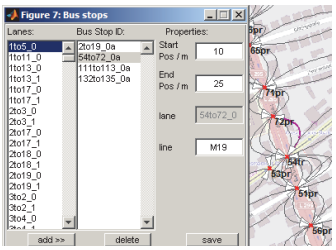


Figure 13-29: Bus stop input dialog

To model public transport behavior, recurring vehicle halts can be specified in SUMO using *bus stop* objects. eNetEditor allows the definition of bus stops using an input dialog that can be called by the keyboard entry ***b*** and is shown in Figure 13-29. The corresponding variable *B* consists of a list, implemented as a 2-dimensional array

$$B=(b_i), \text{ with } \ldots \tag{7}$$

specification of bus stop i b_i ,
bus stop index i = 1 … b_{st} and
total number of specified bus stops b_{st} ,

where each element b_i contains all parameters for the corresponding bus stop

$$b_i = \left[{}^{B}id_i \quad l_i \quad {}^{B}j \quad {}^{B}x_{s,i} \quad {}^{B}x_{e,i}\right], \text{ with } \ldots \tag{8}$$

identifier of bus stop b_i ${}^{B}id_i$,
lane on which bus stop b_i is located l_i ,
index of bus stop b_i on lane l_i ${}^{B}j$ = 1 … 26,
start position of bus stop b_i on lane l_i ${}^{B}x_{s,i}$ (in m) and
end position of bus stop b_i on lane l_i ${}^{B}x_{e,i}$ (in m).

The output is a SUMO compatible xml-file, which is exemplified in Listing 5.

Listing 5: Example bus stop definition file in the SUMO specified xml-format

```
<?xml version="1.0" encoding="utf-8"?>
<additional>
  <busStop id="bS_2to19_0a" lane="2to19_0" startPos="10" endPos="25"/>
  <busStop id="bS_54to72_0a" lane="54to72_0" startPos="10" endPos="25"/>
  <busStop id="bS_54to72_0b" lane="54to72_0" startPos="35" endPos="50"/>
  ...
</additional>
```

13.3.1.5 Charging stations

Charging stations constitute the infrastructural complement to the energy consumption of vehicles. In analogy to bus stops, eNetEditor also allows the definition of charging stations, where compatible vehicles can be supplied with a specified power, if operational conditions allow. The placement and definition of bus stops in the network is similar to that of bus stops. The dialog for the definition of charging stations can be called in eNetEditor by the keyboard entry ***t***. eNetEditor's dialog window is shown in Figure 13-30. Variable *T* contains all of the network's charging station information in form of a list, implemented as a 2-dimensional array

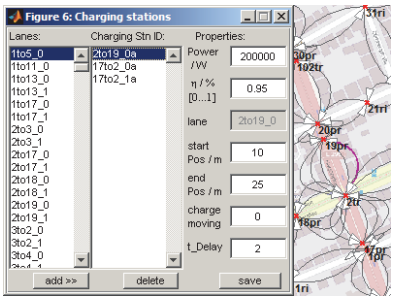


Figure 13-30: Charging station input dialog

$$T=(t_i), \text{ with } \ldots \quad (9)$$

specification of charging station i t_i ,

charging station index i = 1 … t_{cs} and

total number of specified charging stations t_{cs} ,

Each element t_i contains all parameters for the corresponding charging station:

$$t_i = \left[{}^{T}id_i \quad l_i \quad {}^{T}j \quad {}^{T}x_{s,i} \quad {}^{T}x_{e,i} \quad {}^{T}P_i \quad {}^{T}\eta_i \quad T_i \quad d_i \quad c_i\right], \text{ with } \ldots \quad (10)$$

identifier of charging station t_i ${}^{T}id_i$,

lane on which charging station t_i is located l_i ,

index of charging station t_i on lane l_i ${}^{T}j$ = 1 … 26,

start position of charging station t_i on lane l_i ${}^{T}x_{s,i}$ (in m),

end position of charging station t_i on lane l_i ${}^{T}x_{e,i}$ (in m),

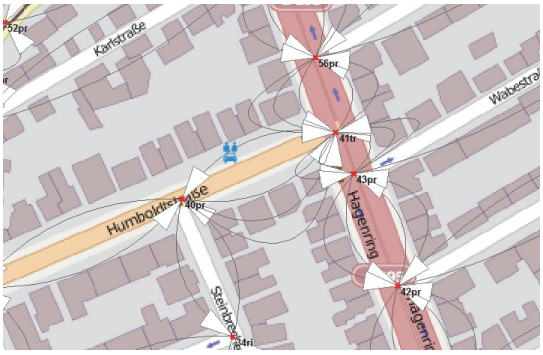
charging power of charging station t_i ${}^{T}P_i$ (in W),

charging efficiency of charging station t_i ${}^{T}\eta_i$,

delay between arrival and charging at charging station t_i T_i (in s),

(dynamic) charging of moving vehicles above charging station t_i d_i and

compatible vehicle eType(s) that charging station t_i can charge c_i



node	link_index	link_from	link_to
'41'	[1]	'43to41_0'	'41to56_0'
'41'	[2]	'43to41_1'	'41to56_1'
'41'	[3]	'43to41_2'	'41to40_0'
'41'	[4]	'40to41_0'	'41to43_0'
'41'	[5]	'40to41_1'	'41to43_1'
'41'	[6]	'40to41_1'	'41to56_0'
'41'	[7]	'40to41_2'	'41to56_1'
'41'	[8]	'56to41_0'	'41to40_0'
'41'	[9]	'56to41_1'	'41to43_0'
'41'	[10]	'56to41_2'	'41to43_1'

Figure 13-31: Layout of traffic light signal controlled junction '41' and output of its link indices for its program

The output is an xml-file that is compatible with SUMO and the implementations described in [6] for modeling the energy consumption of vehicles and their energy supply with the specified infrastructure elements. An output is exemplified in Listing 6.

Listing 6: Example charging station definition file in xml-format

```
<?xml version="1.0" encoding="utf-8"?>
<additional>
  <chargingStation id="cS_2to19_0a" lane="2to19_0" startPos="10" endPos="25"
     chrgpower="200000" efficiency="0.95" chargeDelay="2" chargeInTransit="0"
     eType="a"/>
  <chargingStation id="cS_17to2_0a" lane="17to2_0" startPos="0" endPos="62"
     chrgpower="220000" efficiency="0.9" chargeDelay="5" chargeInTransit="1"
     eType="b"/>
  <chargingStation id="cS_17to2_1a" lane="17to2_1" startPos="0" endPos="62"
     chrgpower="250000" efficiency="0.9" chargeDelay="5" chargeInTransit="1"
     eType="a"/>
  ...
</additional>
```

13.3.1.6 Light signal systems

The definition of traffic light signal system plans for nodes of type *traffic_light* takes place during the network creation by netconvert. In most cases, the automatically generated programs/plans of a junction's light signal system will deviate from its real behavior. SUMO allows multiple ways to interact with light signal systems, both during simulation definition and during runtime. Whereas traffic light signals systems can be declared as actuated for vehicle-flow based triggering or their program switched during simulation runtime using TraCI [9] or WAUT [10], it is also possible to declare static programs with phases of constant durations. Junction's programs can often be sufficiently approximated by a static program, if the period under observation is characterized by similar traffic intensities.

Each phase of a traffic light-signal system in SUMO consists of a duration and a state, where the state is an aggregation of all connection/link states. While the indexing of links is

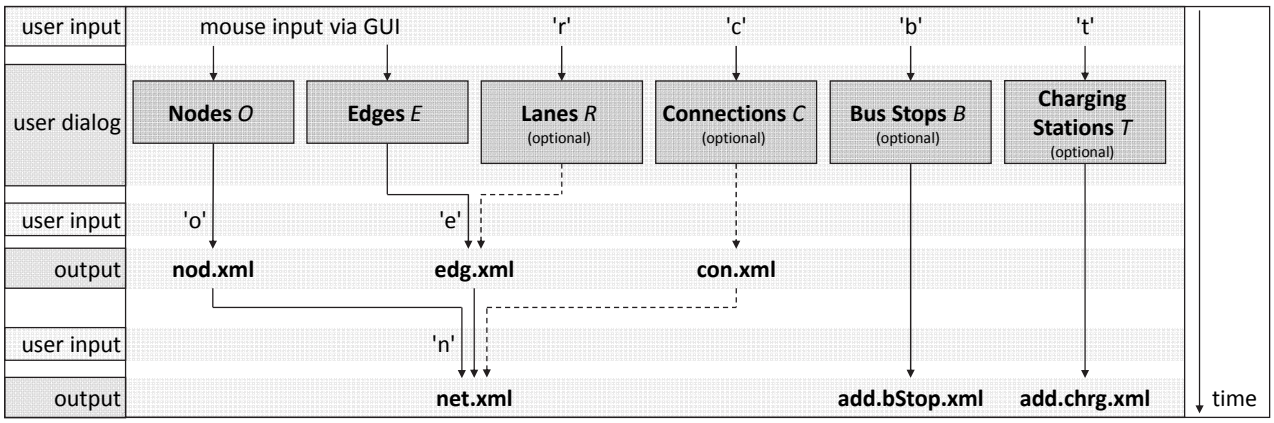


Figure 13-32: Sequence chart of the network creation process

(currently) carried out in clockwise order, one difficulty of defining programs beforehand is that link indices are defined by netconvert and are known only after a junction polygon within the net file has been created. Therefore, the chosen solution for eNetEditor is the manual definition of traffic light signal programs in the network file (.net.xml). After the network generation, however, eNetEditor parses the generated network file for all junctions controlled by a traffic light signal system and their link indices and outputs these to support the user with the definition of light signal programs. Figure 13-31 exemplifies a traffic light signal controlled junction ('41') along with its corresponding output.

13.3.1.7 Network creation

After all constituents of the simulation's road network have been specified in eNetEditor's graphical user interface and user dialogs, the network can be created with the specified parameters using the sequence chart shown in Figure 13-32.

It is a multi-stage creation process that ultimately results in a call to netconvert, passing it all required files, to generate the net file (.net.xml). Bus stop and charging station definitions are created separately (in .add.bStop.xml and .add.chrg.xml respectively) as *additional files* and can subsequently be used under the xml tag `<additional>` of a SUMO configuration. The generated command line that executes netconvert with the required arguments has the following syntax:

```
netconvert.exe -n genNets\Test.nod.xml -e genNets\Test.edg.xml -o genNets\Test.net.xml -x genNets\Test.con.xml.
```

13.3.2 Vehicle definitions and generation

The creation of a scenario by filling a defined network with life, i.e. vehicle objects with routes, initially requires defined vehicle types. SUMO allows the definition of various vehicle types along with their parameters that vehicle objects have access to. Most of these parameters are used by implemented vehicle models and include physical constraints and descriptions of the vehicle itself (e.g. maximum speed, vehicle length, color), driver specific parameters (e.g. minimum gap between vehicles, impatience, deviation from speed limit) or model specifications (e.g. the car following behavior, lane-change model, and other user-defined devices. Based on previous implementations described in [6], an energy device has been implemented in SUMO that allows the realistic recreation of the energy variation of vehicles' energy content during simulation runtime, based on newly introduced and defined vehicle parameters, such as vehicle mass, maximum battery capacity (i.e. energy content), drivetrain efficiencies, or various drag coefficients. To attract a wide user base, the goal of these implementations was simplicity by introducing basic physical and comprehensible vehicle parameters. After defining vehicle types, vehicle objects can be instantiated with *vehicle attributes* (initial values) for different model variables. These parameters are contained within each vehicle object itself, are not part of the vehicle type definition and therefore need to be set separately, e.g. manually or by a router described in 13.13.3.3. Most popular among these parameters are the vehicle's initial speed, its desired lane to depart from (enter the simulation), its departure (simulation entry) time, or its route. Figure 13-34 shows the class diagram of thevehicle attribute structure as definable in eNetEditor. The newly introduced energy device and its

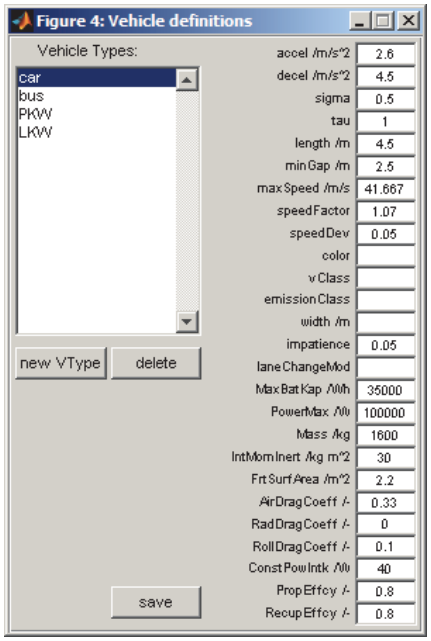


Figure 13-33: Vehicle type input dialog

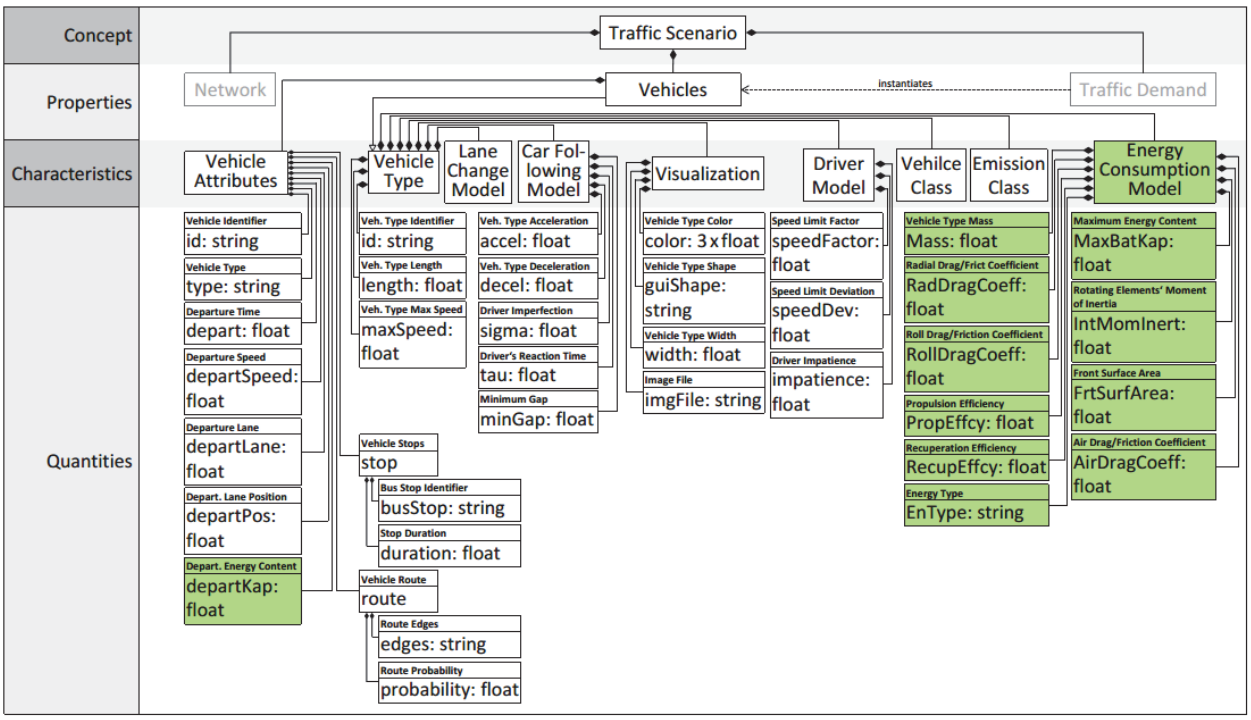


Figure 13-34: Structure of vehicle and vehicle type attributes as implemented in eNetEditor, represented in form of a UML class diagram

attributes are shown in green as well as the initial value of a vehicle object's energy content.

The configuration of vehicle attributes can be performed manually. However, in large simulation scenarios this task is carried out by the different routers during traffic demand generation and calibration that will be described in section 13.13.3.3. In the following, an overview will be given on eNetEditor's data structures and user interface in for definition and generation of vehicle types.

Vehicle types can be specified with all relevant parameters in eNetEditor using the user interface shown in Figure 13-33. Vehicle type properties are stored in the 2-dimensional array V of the following structure

$$V^{n_V \times 27} = (v_i), \text{ with } \ldots \tag{11}$$

$$\text{vehicle type index} \quad i = 1 \ldots n_V \text{ and}$$

$$\text{total number of vehicle type} \quad n_V \ .$$

In the same list-type format as introduced in section 13.13.3.1 for lanes, bus stops, and charging stations, each vehicle type description v_i contains 28 parameters that are exemplified in Listing 7. It also shows the xml format required by SUMO and the implemented energy device [6] for the correct definition of vehicle types.

Listing 7: Example vehicle type definition file in the SUMO specified xml-format with additional parameters used by the newly implemented energy device [6]

```
<?xml version="1.0" encoding="utf-8"?>
<vType id="car" accel="2" decel="5" maxSpeed="42" length="4.5" minGap="2.5" sigma="0.5"
     speedDev="0.05" speedFactor="1.1" tau="1" impatience="0.1">
  <param key="MaxBatKap" value="20000"/>
  <param key="PowerMax" value="80000"/>
  <param key="Mass" value="2000"/>
  <param key="InternalMomentOfInertia" value="50"/>
  <param key="FrontSurfaceArea" value="2.0"/>
  <param key="AirDragCoefficient" value="0.4"/>
  <param key="RadialDragCoefficient" value="0"/>
```

```
  <param key="RollDragCoefficient" value="0.15"/>
  <param key="ConstantPowerIntake" value="40"/>
  <param key="PropulsionEfficiency" value="0.8"/>
  <param key="RecuperationEfficiency" value="0.5"/>
  <param key="eType" value="a"/>
</vType>
```

If left unspecified, default values are assigned to the vehicle energy model parameters, which are explained in more detail in [6].

13.3.3 Traffic demand generation and calibration

After the definition of the network and declaration of vehicle types, the target traffic scenario can be finalized by a subsequent traffic demand generation and, if desired, its calibration. Initial and most important part of vehicle instantiation is the definition of vehicle routes. For this purpose, SUMO comes with a variety of routers. Secondly, initial parameters of the routed vehicle objects need to be set. SUMO's mostly applied routers for the creation of traffic demand are

MAROUTER: macroscopic data from origin/destination-matrices and traffic assignment zones or districts is used to assign routes to vehicles within a given network,
DFROUTER: detected vehicle flow data is used to calculate the flow proportion at junctions to build vehicle routes [11], and
JTRROUTER: definitions of traffic flows on edges and turn ratios at junctions are used to compute vehicle routes within a given network.

For the creation of traffic scenarios using MAROUTER, *SUMO Traffic Modeler* has been developed [12], which is a very convenient and functional third-party tool that supports the generation traffic demand within an existing network. It aims at supplementing a network with demographic data to create time-varying origin/destination-data, which can be used by SUMO's MAROUTER for the generation of vehicle routes. Therefore, eNetEditor focuses on utilizing SUMO's JTRROUTER and (more importantly) DFROUTER for the rapid generation of traffic demand, which is described in more detail in section 13.13.3.3.1.

After an initial set of desired routes (edge-by-edge declarations) or trips (origin-destination declarations) has been generated by a router that most often does not take into account the resulting interaction between vehicles, the route choices for individual vehicles need to be calibrated to create realistic traffic scenarios by distributing vehicles iteratively among alternative routes within the network. Two of the most widely utilized applications for traffic calibration are

DUAROUTER: based on [13], this user assignment algorithm aims at finding a dynamic user equilibrium for given vehicle trips by assigning routes iteratively until no vehicle can reduce its travel time by an alternative route choice and
cadyts: based on [14], this method further takes into account a supply model of the network to iteratively find a stationary condition, which is consistent with existing traffic counts.

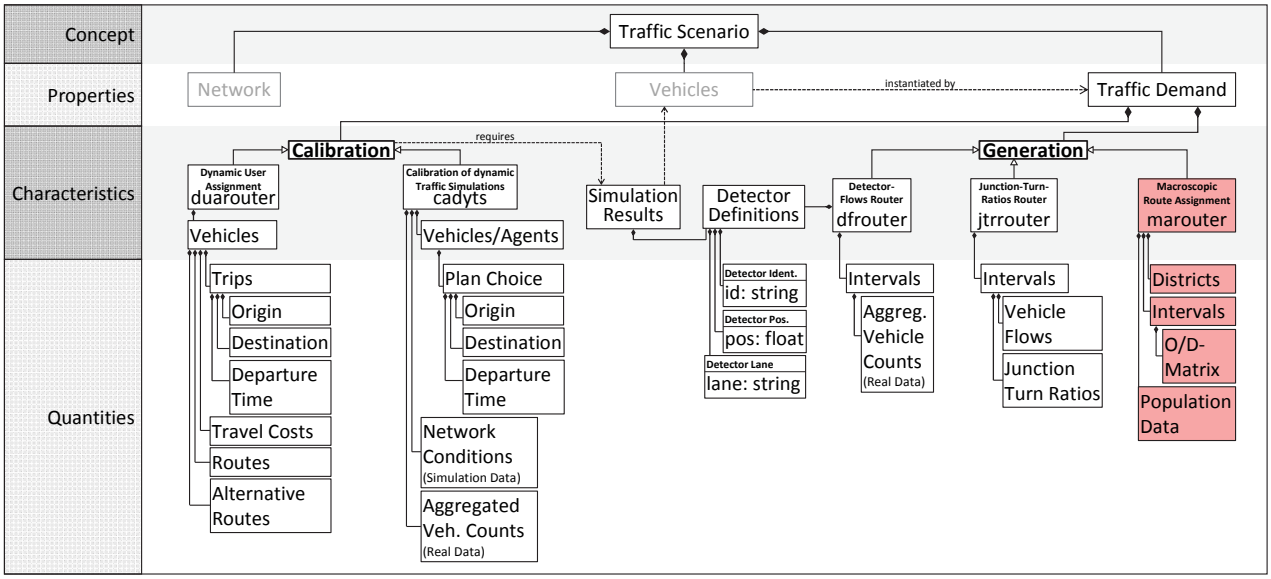


Figure 13-35: Class diagram of traffic demand generation and calibration structure in SUMO and eNetEditor

Figure 13-35 shows the implemented data structure for generating and calibrating traffic demand that the following sections will give more details on. The necessary steps in eNetEditor will further be outlined to build calibrated traffic scenarios. The components shown in red are not implemented in eNetEditor. For creating routes based on existing origin/destination matrices with MAROUTER, the application of *SUMO Traffic Generator* is recommended instead.

13.3.3.1 Traffic demand generation

For the generation of vehicle instances and routes, eNetEditor offers two possibilities: JTRROUTER and DFROUTER.

13.3.3.1.1 JTRROUTER

For demand generation with JTRROUTER, eNetEditor allows the definition of vehicle flows and junction turn ratios.

Flow definitions are stored in the 2-dimensional array F with the following structure:

$$F^{n \times 8} = (f_{i,k}), \text{ with } \ldots \quad (12)$$

$$\text{flow index } i = 1 \ldots n,$$

$$\text{property index } k = 1 \ldots 8 \text{ and}$$

$$\text{total number of flow definitions } n.$$

Each element f_i contains all required parameters of a flow definition:

$$f_i = \left[{}^{F}id_i \quad {}^{F}t_{s,i} \quad {}^{F}t_{e,i} \quad {}^{F}e_i \quad {}^{F}V_i \quad {}^{F}T_i \quad {}^{F}l_i \quad {}^{F}v_i \right], \text{ with } \ldots \quad (13)$$

$$\text{identifier of flow definition } f_i \quad {}^{F}id_i,$$

$$\text{start time of flow definition } f_i \quad {}^{F}t_{s,i},$$

$$\text{end time of flow definition } f_i \quad {}^{F}t_{e,i},$$

$$\text{origin edge of flow definition } f_i \quad {}^{F}e_i,$$

$$\text{number of vehicles in flow definition } f_i \quad {}^{F}V_i,$$

$$\text{vehicle type of flow definition } f_i \quad {}^{F}T_i,$$

$$\text{departure (initial) lane of flow definition } f_i\text{'s vehicles} \quad {}^{F}l_i \text{ and}$$

departure (initial) speed of flow definition f_i's vehicles ${}^{F}V_i$.

The resulting xml-output is shown in Listing 8.

Listing 8: Example vehicle flow definition file in the SUMO specified xml-format

```
<?xml version="1.0" encoding="utf-8"?>
<flows>
  <flow id="86to11flow1" begin="0" end="8640" from="86to11" number="88" type="car"
     departLane="random" departSpeed="random"/>
  <flow id="87to13flow1" begin="0" end="8640" from="87to13" number="90" type="car"
     departLane="random" departSpeed="random"/>
  <flow id="88to13flow2" begin="0" end="8640" from="88to13" number="15" type="bus"
     departLane="random" departSpeed="random"/>
  ...
</flows>
```

The typical use-case of JTRROUTER is that incoming flows into a network are defined at the network boundaries, along with turn ratio definitions at junctions to fill the network with vehicle interaction. When the vehicle flow input dialog is opened for the first time, one vehicle flow is initialized for each incoming edge into the regarded network, whose parameters have to be specified. An edge $e_{i,j,1}$ is identified as an incoming edge if its origin node is not a destination node of *any* other edge (not even $e_{j,i,1}$). Equation 14 formally defines the set of incoming edges E_I as

$$E_I = \{e_{i,j,1} | \forall j \in [0,n] : e_{i,j,1} = 0\}. \tag{14}$$

In addition to vehicle flows, turning ratios need to be defined at junctions to feed JTRROUTER with all required information. The concept behind the structure for the turn ratio definitions at junctions is similar to that of connections described in section 13.13.3.1.2. The main difference between the two declarations is that connections are lane-based definitions (each connection is described by an incoming and outgoing *lane*), where turning ratios are edge based (each turn ratio is described by an incoming and outgoing *edge*). Consequently, the complexity for turn ratio definitions at junctions is lower than that of connections and can be reduced to three degrees of freedom: the incoming and outgoing edge, each described by their origin and destination nodes, where the destination node of the incoming edge is identical to the origin node of the outgoing edge. Each resulting turn ratio is a sequence of three nodes, attributed with a probability. The chosen data structure for turn ratio definitions is a three-dimensional matrix, implemented as

$$J^{n \times n \times n} = (j_{k,l,m}), \text{ with } \ldots \tag{15}$$

$$\begin{aligned} \text{origin node index of incoming edge} \quad & k = 1 \ldots n, \\ \text{destination/origin node index of incoming/outgoing} \quad & l = 1 \ldots n, \\ \text{destination node index of outgoing edge} \quad & m = 1 \ldots n, \\ \text{total number of nodes} \quad & n \quad \text{and} \\ \text{turn ratio (probability)} \quad & j_{k,l,m} = [0,1]. \end{aligned}$$

Even if J grows exponentially with the amount of nodes, the resulting variable will most often be a sparse matrix, whose required amount of memory can correspondingly be reduced by explicitly declaring it as one in MATLAB (command: *sparse*). Since junction turn ratio definitions are treated as probabilities when JTRROUTER creates vehicle routes, the definitions have to meet requirements to be valid. For the conservation of vehicle flows at junctions, equation 16 defines that the turning ratios

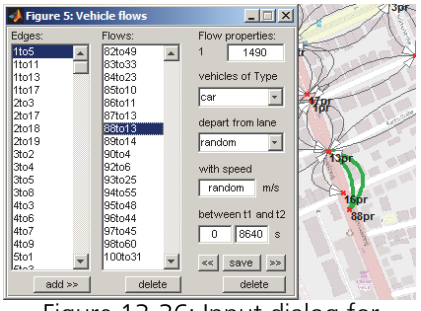


Figure 13-36: Input dialog for vehicle flow definitions

from an incoming edge, taken from the adjacency matrix $E = (e)$, must add up to 1. The resulting variable J is automatically validated accordingly when entering turning ratios. Listing 9 exemplifies the xml-output for the definition of junction turn ratios.

$$\sum_m j_{k,l,m} = \begin{cases} 1 & \text{if } e_{k,l,1} \neq 0 \\ 0 & \text{else.} \end{cases} \tag{16}$$

Listing 9: Example junction turn ratio definition file in the SUMO specified xml-format

```
<?xml version="1.0" encoding="utf-8"?>
<turns>
  <fromEdge id="23to29">
    <toEdge id="29to30" probability="0.2"/>
    <toEdge id="29to33" probability="0.3"/>
    <toEdge id="29to34" probability="0.5"/>
  </fromEdge>
  ...
</turns>
```

The vehicle flow and junction turn ratio input dialogs are shown in Figure 13-36 and Figure 13-37 and can be called by the keyboard entries of ***f*** and ***c*** respectively (due to their similarity, junction turn ratios can be defined along with connections in the lower half of the same input window as shown in Figure 13-27).

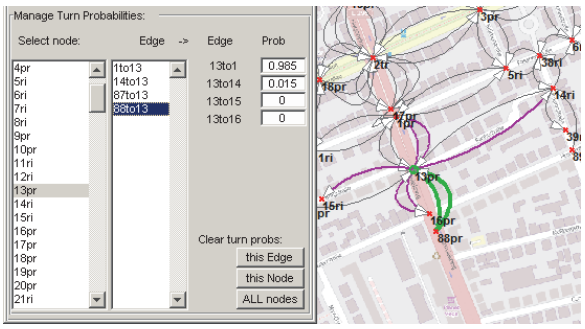


Figure 13-37: Input dialog for junction turn ratio definitions

If the vehicle flow and junction turn ratio definitions have been created, the specified scenario can be built with a keyboard entry ***j***. This (1) calls JTRROUTER with all required options, which returns vehicle routes that (2) are used to build the resulting SUMO configuration (scenario) file, which is saved in the working directory. The following syntax exemplifies the created command line call to of JTRROUTER with all required options:

```
jtrrouter --random 1 -f Test.flows.xml -t Test.turns.xml -n Test.net.xml -o Test.rou.xml.
```

13.3.3.1.2 DFROUTER

If the regarded road network is equipped with induction loops or other sensors to measure vehicle traverses, the measurement data can be used to build vehicle routes using DFROUTER, which performs a stochastic route estimation for vehicles. eNetEditor offers an interface to DFROUTER that will be described in this section.

For the definition of induction loops (detectors), the keyword numVehs has been reserved as a property name for edges described in section 13.13.3.1.1. If vehicle counts exist for an edge, their values can be assigned to the edge attribute *numVehs* for subsequent processing to build a command line argument for DFROUTER, including detector and vehicle count definitions. DFROUTER declares routes for two vehicle types: *PKW* and *LKW*, representing the average passenger and commercial/freight vehicle in the simulation scenario respectively. PKW and LKW need to be declared as vehicle types in eNetEditor before building the scenario using DFROUTER's routes. The following descriptions are further required:

Detector Definitions: Detector definitions are built according to the declaration of the attribute *numVehs* on the corresponding edge, as exemplified in Listing 10 and

Vehicle Counts: Synthetic vehicle counts are generated according to the specified value of the *numVehs* attribute. Their timestamps are distributed according to the daily traffic curve *TGw2West* from [15] (Figure 13-4) within a user-specified time (e.g. 15:00--18:00) and aggregation periods (e.g. 20 minutes). A constant distribution of 95% passenger and 5% commercial/freight vehicles is currently assumed. The vehicle flow measurement file is exemplified in Listing 11.

Listing 10: Example induction loop (detector) definition as input for the DFROUTER command line

```
<?xml version="1.0" encoding="utf-8"?>
<detectors>
  <detectorDefinition id="det0_1to5_0" lane="1to5_0" pos="0"/>
  <detectorDefinition id="det1_1to11_0" lane="1to11_0" pos="0"/>
  <detectorDefinition id="det2_1to13_0" lane="1to13_0" pos="0"/>
  ...
</detectors>
```

Listing 11: Example of the synthetically generated vehicle flow measurement file for DFROUTER command line

```
Detector;Time;qPKW;vPKW;qLKW;vLKW
det0_1to5_0;0;4.2989;7;0.22626;5
det0_1to5_0;20;8.334;7;0.43863;5
det0_1to5_0;40;10.5241;7;0.5539;5
det1_1to11_0;0;4.4204;7;0.23265;5
  ...
```

If all required definitions have been made, the generation of a scenario with vehicle routes from DFROUTER can be initiated by the keyboard entry d. Much like the process defined in 13.13.3.3.1.1 for JTRROUTER, this (1) calls DFROUTER with all required options. DFROUTER then returns two files: a route file containing a list of all generated routes and a vehicle instantiation file (with reference to the declared routes). These files (2) are subsequently used to build the resulting SUMO configuration. The following command exemplifies the DFROUTER execution to build the described routes:

```
dfrouter --random 1 --net-file Test.net.xml --detector-files Test.det.xml --measure-files
Test.flowMeas.xml --routes-output Test.rou.xml --emitters-output Test.veh.xml --time-step 1200
--departlane random --departspeed random --vtype true
```

The sequence chart for traffic demand generation with is shown in Figure 13-38.

13.3.3.2 Traffic demand calibration

The problem of the routers described in section 13.13.3.3.1 is that vehicle instances are assigned a route under static conditions, e.g. by searching for the shortest path through the network. Prevailing traffic conditions are not regarded. This task can be compared to that of trying to find the shortest way through a traffic-intense urban area, only using a map, in hope that the chosen route will be the fastest or the one with the least amount of energy required. It is obvious that prevailing traffic conditions can have a high impact on these optimization criteria. In real traffic, participants without (also often with) assistance usually need a few tries, where iterative variations of departure time and route choice ultimately yields the optimal perceived route for a recurring trip.

A variety of methods exist for traffic simulations that aim to minimize a cost function to find an optimum route distribution among participants (vehicles) in a similar manner as described above. In the scope of eNetEditor, demand calibration refers to the adaption of route assignments. A comparison between different traffic demand assignment methods can be found in [16]. eNetEditors provides an interface to two established implementations for SUMO that aim to optimize the route assignment for vehicle instances: DUAROUTER and cadyts. The basis for both is a validated set of vehicle trips, each consisting of origin, destination, and departure time. These parameters are taken from the route definitions that

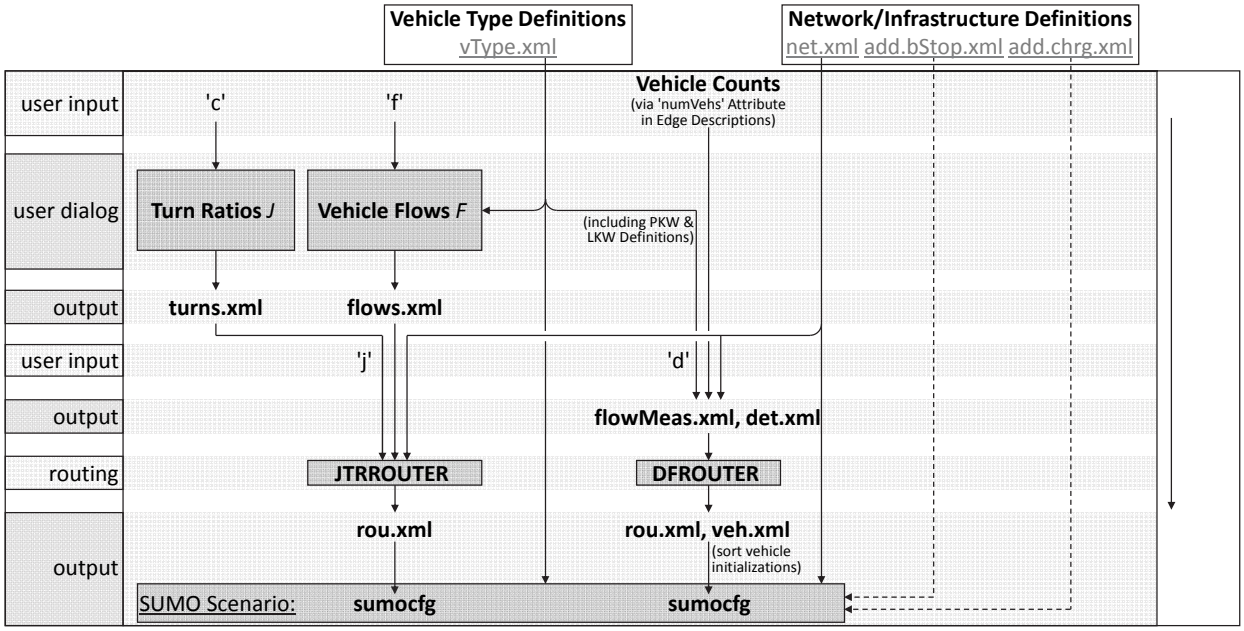


Figure 13-38: Sequence chart of the traffic demand creation process using JTRROUTER and/or DFROUTER

were stochastically determined by DFROUTER. The calibration of traffic demand in eNetEditor is implemented as a 2-phase process:

DUAROUTER: Vehicle routes are distributed by creating alternative routes and evaluating the resulting edge-based weights for specified vehicle trips, until no vehicle can modify its route choice without increasing its travel cost. One of the outputs is a vehicle route file for vehicles containing alternative route probabilities and corresponding travel costs.

cadyts: cadyts itself performs no routing. The alternative routes created by DUAROUTER are evaluated and vehicle route choices adapted, such that the chosen routes comply with detector measurements passed as an input to cadyts.

If traffic scenarios are to be calibrated with cadyts, a previous calibration with DUAROUTER is required in eNetEditor to obtain a required alternative route definition for cadyts.

13.3.3.2.1 DUAROUTER

While DUAROUTER itself has been implemented in C++, the iterative process of finding alternative routes and executing the simulation has been implemented in form of a Python script, dualterate.py. eNetEditor automates the creation and execution of the required command-line argument by the keyboard entry of the ***u***, if previous vehicle routes from DFROUTER (section 13.13.3.3.1.2) are present. These routes are stripped down to each of

their trip definitions using the existing Python script route2trips.py: the route's origin edge, destination edge, and departure time. A trip definition is exemplified in Listing 12.

Listing 12: Example trip definition as a result of route definitions from DFROUTER used as input for DUAROUTER

```
<?xml version="1.0" encoding="utf-8"?>
<trips>
  <trip depart="0.00" departLane="random" departPos="0" departSpeed="random"
     from="82to49" id="emitter_det211_82to49_0_0" to="13to16" type="PKW"/>
  <trip depart="0.00" departLane="random" departPos="0" departSpeed="random"
     from="83to33" id="emitter_det212_83to33_0_0" to="10to27" type="LKW"/>
  <trip depart="0.00" departLane="random" departPos="0" departSpeed="random"
     from="84to23" id="emitter_det213_84to23_0_0" to="23to28" type="PKW"/>
  ...
</trips>
```

The resulting output in eNetEditor is a command-line argument that executes the iterative DUAROUTER Python script with the specified options as exemplified in the following command:

```
duaIterate.py -n Test.net.xml -t Test.trips.xml -x detailed -+ "Test.vtypes.xml,
Test.add.chrg.xml,Test.add.bStp.xml" -l 20
```

13.3.3.2.2 cadyts

If DUAROUTER has finished calibration, a subsequent call to the Python script *cadytsIterate.py* can be made from eNetEditor by keyboard entry of the button ***y***. cadyts requires traffic flow measurements in the xml-format exemplified in Listing 13. In regard of the contained information, this measurement file is nearly identical to that required by DFROUTER, with an additional standard deviation for each measurement, only in a different format. Currently, only one percentage value will be queried by eNetEditor, which derives the standard deviation as a fixed percentage of the corresponding flow for all measurement values. The measurement file is constructed from the values of the user-specified edge parameter *numVehs* in the same manner as described in section 13.13.3.3.1.2 for DUAROUTER.

Listing 13: Example flow measurement definition created for subsequent call to cadyts

```
<?xml version="1.0" encoding="utf-8"?>
<measurements>
  <onlink start="0"    end="1199" link="1to5"  value="4.280"  stddev="0.428"
     type="COUNT_VEH"/>
  <onlink start="1200" end="2399" link="1to5"  value="8.158"  stddev="0.816"
     type="COUNT_VEH"/>
  <onlink start="2400" end="3599" link="1to5"  value="12.036" stddev="1.204"
     type="COUNT_VEH"/>
  <onlink start="0"    end="1199" link="1to11" value="4.280"  stddev="0.428"
     type="COUNT_VEH"/>
  ...
</measurements>
```

The resulting output is a command-line argument that executes cadyts with the specified options to iteratively adapt the route choice of vehicles. The following exemplifies a command line call to cadyts with all required input files and arguments:

```
cadytsIterate.py -b 0 -n Test.net.xml -d Test.calibFlowMeas.xml -+ "Test.vtypes.xml, Test.add.
chrg.xml, Test.add.bStp.xml" -r Test.rou.alt.xml -W Test.flowsEvaluation.txt -l 20 -a 1200
```

13.3.3.3 Manual initial vehicle energy content adjustments

Since the current implementation of the described routers are not yet compatible with the newly developed energy device [6], there are currently two possibilities for setting vehicles' initial energy content at their departure (in analogy to the parameters *departSpeed*, *departLane*, etc.): (1) if left unspecified, vehicles' initial energy content is set to *half of the vehicle type's maximum energy content* by default or (2) vehicle instantiations need to be modified manually within the vehicle route file as exemplified by the syntax in Listing 14.

Listing 14: SUMO configuration of a scenario calibrated with cadyts

```
<?xml version="1.0" encoding="utf-8"?>
...
<vehicle depart="0" id="veh1" route="route02" type="ElectricCar" departSpeed="max">
    <param key="ActBatKap" value="1000"/>
</vehicle>
...
```

13.3.4 User interaction and interface

Beyond the described user dialogs, user interaction is implemented on the basis of hotkeys. eNetEditor can be executed with the following command-line from MATLAB.

```
netBuild('quadraticBackgroundImage.png',widthInMeters,'projectName')
```

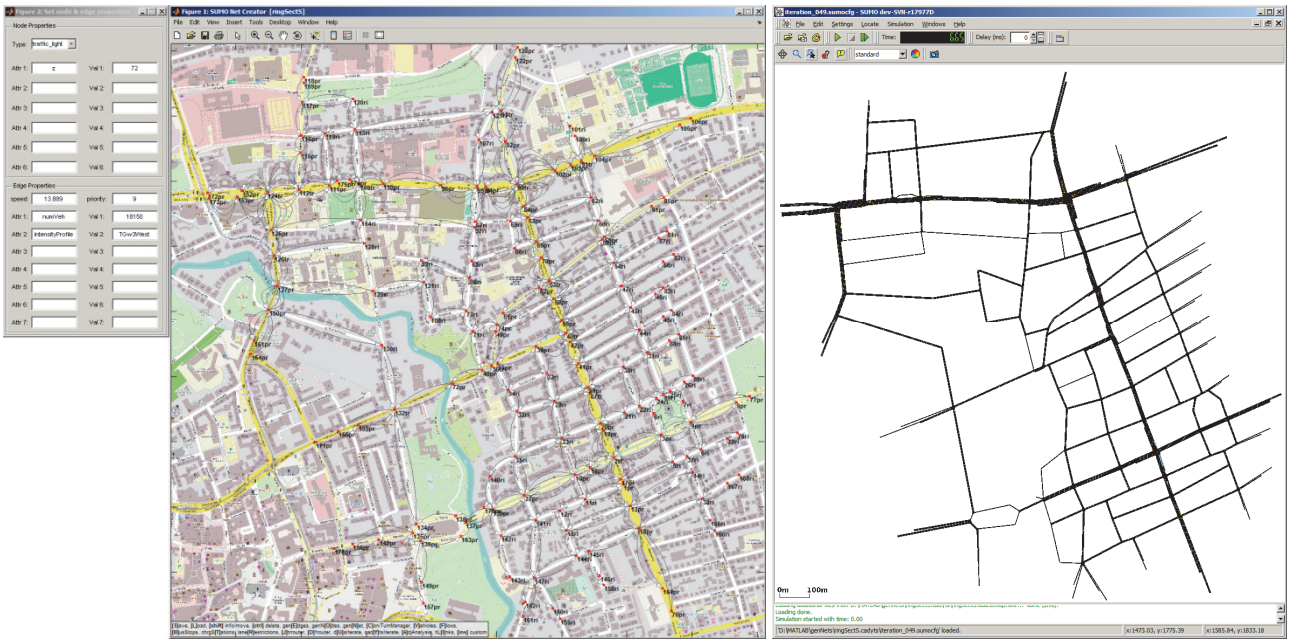


Figure 13-39: eNetEditor's graphical user interface showing a user-specified background image with a user-created network's digraph (left) and a snapshot of the resulting scenario in the SUMO GUI (right)

The first command argument specifies the filename of a quadratic background image for visualization. The second argument specifies the width (= height) in meters as an integer of the regarded network to be modeled. The third argument specifies the project name as a string.

Projets can be saved in the main window at any time with the keyboard entry ***s***. If an existing project shall be opened, the above command-line argument must be executed in MATLAB with its project name specified in the third argument. After all windows have finished loading, the project can be loaded with the keyboard entry ***l***. An overview of all hotkeys can be found at the bottom of the main window.

13.4 Example scenario and results

An example scenario was created that represents a section of Braunschweig's road traffic. Traffic measurements were taken from the city's traffic intensity map [17] and used for calibration as described in section 13.13.3.3. Figure 13-39 depicts eNetEditor's user interface along with a created scenario (left) next to its visualization in SUMO's graphical user interface. The SUMO corresponding configuration file, using the vehicle energy device and charging stations, after calibration with cadyts is shown in Listing 15.

Listing 15: SUMO configuration of a scenario calibrated with cadyts

```
<?xml version="1.0" encoding="utf-8"?>
<configuration ...>
  <input>
    <net-file value="ringSect5.net.xml"/>
    <route-files value="ringSect5_049.cal.xml"/>
    <additional-files value="dua_dump_049.add.xml,ringSect5.DFvtypes.xml,
        ringSect5.add.chrg.xml,ringSect5.add.bStop.xml"/>
  </input>
  <output>
    <battery-output value="iteration_049.battery.out.xml"/>
    <battery-output.precision value="4"/>
    ...
  </output>
  ...
</configuration>
```

The resulting output file (in this case `iteration_049.battery.out.xml`) can be used by subsequent analyses and optimization algorithms (subject of future work) to determine (1) where in the network energy is consumed and (2) which locations are suitable for the placement of charging stations to supply vehicles with adequate energy *on the fly*, i.e. during their operation. Figure 13-40 shows first results that aim to supplement these future analyses.

Whereas the left image shows the cumulated energy that was consumed by all vehicles in the simulation, the center and right image focus on the energy consumption's complement: the time, which can be used for transferring energy into vehicles. If vehicles can be charged while moving at arbitrary speeds (e.g. catenary/trolley systems), any location within the road

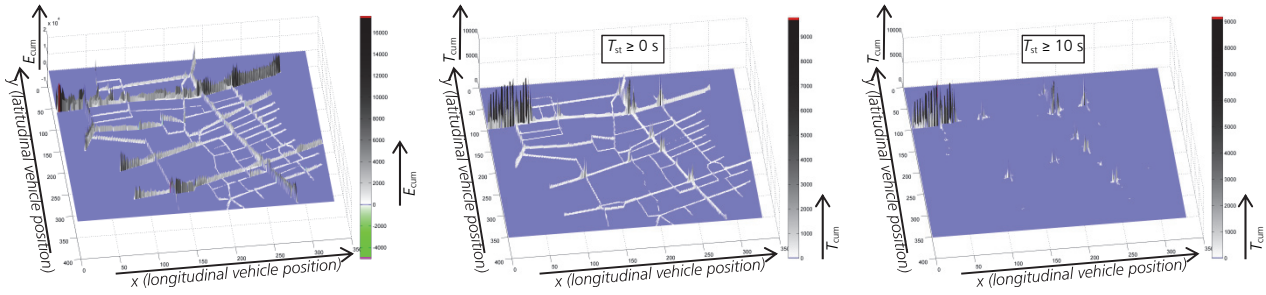


Figure 13-40: Cumulated energy consumed (left) and cumulated vehicle standstill duration T_{st}, with $T_{st} \geq 0$ s (left) and $T_{st} \geq 10$ s (right), of all vehicles in the simulation over their position

network can potentially be used for transferring energy into vehicles. The resulting cumulated time of all vehicles in the example scenario as a function of vehicle position is shown in the center image ($T_{st} \geq 0$ s). However, most charging stations (e.g. gas stations) require vehicles to be standing still for a certain duration. If only halts can be used for charging that have a minimum duration, the amount of energy that can be charged reduces drastically. The image

to the right shows the cumulated duration of vehicles halts, of which each exceeded 10 seconds in the scenario as a function over vehicle position.

13.5 Summary and outlook

The development of eNetEditor was motivated by the projects *emil* and *InduktivLaden*, both funded by the German Federal Ministry of Transport and digital Infrastructure. An inductive charging infrastructure and a compatible fully electric prototype bus fleet was successfully integrated in Braunschweig's traffic infrastructure and public transport. With the application of Bombardier's inductive charging system PRIMOVE, vehicles can be charged with powers beyond 200 kW. A crucial factor for the reliable operation of vehicles with limited range is the placement of charging stations. In the project *emil* infrastructure placement was optimized on the basis of measured public transport vehicle trajectories [18] using the FMS interface [19] and GPS loggers.

eNetEditor was developed for city and traffic planners to evaluate the application of alternative energy supply systems in *early* idea and concept phases, where such data is not readily available and only general traffic measurements exists. The ultimate goal is to adapt and implement *flow capture* and *flow refueling* optimization algorithms from [20], [21], and [22] to determine optimal charging infrastructure locations and thereby help in initial estimations of installation costs.

Regarding the validity of the created scenario, calibrated vehicle routes will be checked next against the measured public transport vehicle trajectories. Future implementations in eNetEditor itself include the assistance in modelling public transport by the creation of recurring vehicle routes and their integration in calibrated scenarios. For a more realistic integration and application of DFROUTER's and cadyts' functionalities, user-specifiable daily traffic intensity curves as well as lane-based measurement variations and distributions between passenger and commercial/freight vehicles (*PKW* and *LKW*) will be implemented. Regarding the application of the newly developed energy device [6], routers are currently not compatible with the definition of vehicles' initial energy content at their departure (in analogy to *departSpeed*). It is planned to adapt the routers correspondingly or to implement a parser of vehicle route declarations to allow for custom initial vehicle energy content distributions. In the long term, an import function for .net.xml-files is planned as well as the migration to a platform-independent implementation.

13.6 References

[1] statista. Anzahl der Neuzulassungen von Personenkraftwagen in Deutschland im Jahr 2014 nach Marken. http://de.statista.com/statistik/daten/studie/167008/umfrage/neuzulas sungen-von-pkw-nach-marken-in-deutschland/, (rev. 02. February 2015)

[2] statista. Anzahl der Neuzulassungen von Elektroautos in Deutschland in den Jahren 2003 bis 2014. http://de.statista.com/statistik/daten/studie/244000/umfrage/neuzulassungen-von-elektroautos-in-deutschland/, (rev. 02. February 2015)

[3] statista. Anzahl der Tankstellen in Deutschland von 1950 bis 2014. http://de.statista.com/statistik/daten/studie/2621/umfrage/anzahl-der-tankstellen-in-deutschland-zeitreihe /, (rev. 02. February 2015)

[4] BDEW: BDEW-Erhebung Elektromobilität: Zuwachs bei öffentlichen Lademöglichkeiten. https://www.bdew.de/internet.nsf/id/bdew-erhebung-elektromobilitaet-zuwachs-bei-oef fentlichen-lademoeglichkeiten-de (rev. 02. February 2015)

[5] Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.: Recent Development and Applications of SUMO - Simulation of Urban MObility. International Journal on Advances in Systems and Measurements, 5 (3&4): 128--138 (2012)

[6] Kurczveil, T., López, P.A., Schnieder, E.: Implementation of an Energy Model and a Charging Infrastructure in SUMO. In: Behrisch, M., Krajzewicz, D., Weber, M. (eds.) Simulation of Urban Mobility. Lecture Notes in Computer Science, vol. 8594 , pp. 33--43. Springer, Heidelberg (2014)

[7] Schnieder, L., Stein, C. Schielke, A.G., Pfundmayr, M.: Effektives Terminologiemanagement als Grundlage methodischer Entwicklung automatisierungstechnischer Systeme. at - automatisierungstechnik, 59 (1): pp. 62--70 (2011)

[8] Schnieder, L.: Formalisierte Terminologien technischer Systeme und ihrer Zuverlässigkeit. Dissertation, Technische Universität Braunschweig (2010)

[9] Wegener, A., Piorkowski, M., Raya, M., Hellbrück, H., Fischer, S., Hubaux, J.-P.: TraCI: An Interface for Coupling Road Traffic and Network Simulators. CNS '08 - 11th Communications and Networking Simulation Symposium, Ottawa (2008)

[10] German Aeroscace Center. Simulation/Traffic Lights. http://sumo.dlr.de/wiki/Simulation/Traffic_Lights (rev. 30. December 2014)

[11] Nguyen, T., Fullerton, M., Krajzewicz, D., Mai, S.T.: DFROUTER - Route estimate method based on detector data. SUMO2014 - Modeling Mobility with Open Data, Berlin (2014)

[12] Papaleontiou, L.G.: High-Level Traffic Modelling and Generation. Master Thesis, University of Cyprus, Nikosia (2008)

[13] Gawron, C.: Simulation-based traffic assignment - computing user equilibria in large street networks. Ph.D. Dissertation, University of Köln, Cologne (1998)

[14] Flötteröd, G., Bierlaire, M., Nagel, K.: Bayesian demand calibration for dynamic traffic simulations. Transportation Science, 45(4): pp. 541--561 (2011)

[15] Forschungsgesellschaft für Straßen- und Verkehrswesen e.V.: Handbuch für die Bemessung von Straßenverkehrsanlagen (HBS). FGSV Verlag GmbH, Cologne (2009)

[16] Behrisch, M., Krajzewicz, D., Wang Y.-P.: Comparing performance and quality of traffic assignment techniques for microscopic road traffic simulations. DTA2008 - International Symposium on Dynamic Traffic Assignment, Leuven (2008)

[17] WVI GmbH: Verkehrsmengen im Werktagsverkehr Mo - Fr in [Kfz/24h] - Querschnittswerte. Stadt Braunschweig, Braunschweig (2009)

[18] Kurczveil, T., Schnieder, E.: Messdatenauswertung für die Auslegung einer induktiven Ladeinfrastruktur für den öffentlichen Personennahverkehr in Braunschweig. AAET 2014, Braunschweig (2014)

[19] HDEI/BCEI Working Group: Fleet Management System Standard Description.

[20] Hodgson, M.J.: A flow capturing location-allocation model. Geographical Analysis, 22(3): pp. 270--279 (1990)

[21] Berman, O., Larson, R.C., Fouska, N.: Optimal location of discretionary service facilities. Transportation Science, 26: pp. 201--211 (1992)

[22] Kuby, M., Lim, S.: The flow-refueling location problem for alternative-fuel vehicle. Socio-Economic Planning Sciences, 39(2): pp. 125--145 (2005)

Authors Index