SUMO Conference Proceedings
2 | 2021

TIB OPEN PUBLISHING

SUMO User Conference

13 – 15 September 2021
Virtual Event

SUMO
User Conference
September 13-15, **2021**

Editors:
Pablo Alvarez Lopez
Olaf Angelo Banse Bueno
Maria Giuliana Armellini
Michael Behrisch
Laura Bieker-Walz
Jakob Erdmann
Yun-Pang Flötteröd
Robert Hilbrich
Ronald Nippold
Johannes Rummel
Matthias Schwamborn
Peter Wagner
Melanie Weber

DLR Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

SUMO Conference Proceedings

SUMO Conference Proceedings (SCP) is dedicated to publish the proceedings of the SUMO user conference.

Traffic simulations are of immense importance for researchers as well as practitioners in the field of transportation. SUMO has been available since 2001 and provides a wide range of traffic planning and simulation applications. SUMO consists of a suite of tools covering road network imports and enrichment, demand generation and assignment, and a state-of-the-art microscopic traffic simulator capable of simulating private and public transport modes, as well as person-based trip chains. Being open source, SUMO is easily extensible by new behavioral models and can be dynamically controlled via a well-defined programming interface. These and other features make SUMO one of the most popular open source traffic simulators with a large and international user community.

The SUMO Conference aims in bringing SUMO users and people interested in traffic simulation and modelling together to exchange their research results, used models and tools and discuss their findings. The papers of this conference can be found here publicly available.

ISSN (online): 2750-4425

SUMO Conference Proceedings (SCP) are published by TIB Open Publishing (Technische Informationsbibliothek, Welfengarten 1 B, 30167 Hannover) on behalf of DLR Institute of Transportation Systems.

Volume 2
SUMO User Conference 2021

13 – 15 September 2021, Virtual Event

Conference papers

https://doi.org/10.52825/scp.v2i

Editors

Pablo Alvarez Lopez, German Aerospace Center, Institute of Transportation Systems

Olaf Angelo Banse Bueno, German Aerospace Center, Institute of Transportation Systems

Maria Giuliana Armellini, German Aerospace Center, Institute of Transportation Systems

Michael Behrisch, German Aerospace Center, Institute of Transportation Systems

Laura Bieker-Walz, German Aerospace Center, Institute of Transportation Systems

Jakob Erdmann, German Aerospace Center, Institute of Transportation Systems

Yun-Pang Flötteröd, German Aerospace Center, Institute of Transportation Systems

Robert Hilbrich, German Aerospace Center, Institute of Transportation Systems

Ronald Nippold,German Aerospace Center, Institute of Transportation Systems

Johannes Rummel, German Aerospace Center, Institute of Transportation Systems

Matthias Schwamborn, German Aerospace Center, Institute of Transportation Systems

Peter Wagner, German Aerospace Center, Institute of Transportation Systems

Melanie Weber, German Aerospace Center, Institute of Transportation Systems

Review process

The papers are reviewed by at least two independent reviewers. All papers which are submitted by authors from DLR are only reviewed by external steering committee members to avoid conflicting interests. After the review, the author receives the decision whether the paper was accepted or has been rejected. Additionally, they are receiving remarks, questions and hints how to improve the quality of the papers for the final version of the publication. The authors have at least two weeks to rework and submit the final version of their paper.

Financing

The organization of the conference and publication of the papers are financed by the conference fee of the participants of the SUMO conference.

https://doi.org/10.52825/scp.v2i

Editors

Pablo Alvarez Lopez, German Aerospace Center, Institute of Transportation Systems

Olaf Angelo Banse Bueno, German Aerospace Center, Institute of Transportation Systems

Maria Giuliana Armellini, German Aerospace Center, Institute of Transportation Systems

Michael Behrisch, German Aerospace Center, Institute of Transportation Systems

Laura Bieker-Walz, German Aerospace Center, Institute of Transportation Systems

Jakob Erdmann, German Aerospace Center, Institute of Transportation Systems

Yun-Pang Flötteröd, German Aerospace Center, Institute of Transportation Systems

Robert Hilbrich, German Aerospace Center, Institute of Transportation Systems

Ronald Nippold,German Aerospace Center, Institute of Transportation Systems

Johannes Rummel, German Aerospace Center, Institute of Transportation Systems

Matthias Schwamborn, German Aerospace Center, Institute of Transportation Systems

Peter Wagner, German Aerospace Center, Institute of Transportation Systems

Melanie Weber, German Aerospace Center, Institute of Transportation Systems

Review process

The papers are reviewed by at least two independent reviewers. All papers which are submitted by authors from DLR are only reviewed by external steering committee members to avoid conflicting interests. After the review, the author receives the decision whether the paper was accepted or has been rejected. Additionally, they are receiving remarks, questions and hints how to improve the quality of the papers for the final version of the publication. The authors have at least two weeks to rework and submit the final version of their paper.

Financing

The organization of the conference and publication of the papers are financed by the conference fee of the participants of the SUMO conference.

SUMO User Conference 2021
https://doi.org/10.52825/scp.v2i.88

Published: 29 Jun. 2022

Online calibration with SUMO for network-wide traffic and emission monitoring – Case study ITS Huainan

Yun-Pang Flötteröd[1] and Michael Behrisch[1]
[1] Institute of Transportation Systems, German Aerospace Center
yun-pang.floetteroed@dlr.de, michael.behrisch@dlr.de

Abstract

Currently, the city of Huainan, China, is constructing its intelligent transportation system, and traffic and environmental monitoring system for efficiently and effectively monitoring and managing city traffic. DLR's traffic information platform KeepMoving is adopted in the aforementioned system, where the existing online calibration module with SUMO has been extended in order to provide current and predicted traffic states and the resultant emission information in real-time. Accordingly, a comprehensive city-wide traffic situation can be captured and shown at the KeepMoving portal, as a decision support tool for traffic management personnel. The comparison between the real and simulated data shows a promising calibration result.

1 Introduction

Rapid economic growth and technology development facilitate people's mobilities and expand their life and social cycles. Accordingly, the usage of private vehicles has continually increased since years and resulted in traffic congestions especially in dense urban areas. Different traffic management strategies have been applied with use of real-time data collected by different kinds of sensors, such as loop-based, microwave-based or camera-based detectors, drones and floating car data (FCD). With regard to high investment and maintenance costs the aforementioned sensors are often only located at limited places with heavy traffic. Therefore, real-time traffic states can be monitored locally, but not networkwide. To deal with this issue, traffic simulation tools have been applied as a part of traffic monitoring and management system in some cities.

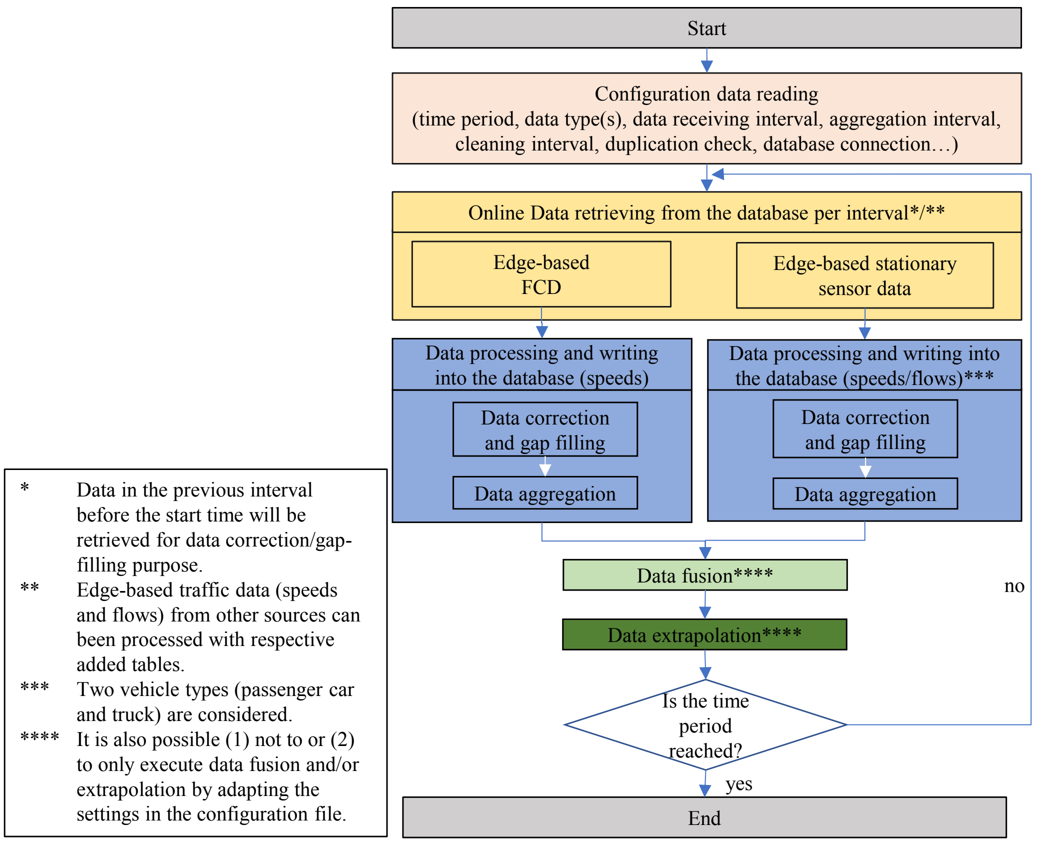
As known, good data, such as traffic demand, road network, signal control and traffic management strategies, is the basic prerequisite for setting up a representative traffic simulation. Normally, it takes lots of time to generate/update general traffic demand data when using household survey methods. Moreover, traffic demand varies from time to time due to travelers' decisions on departure time, transport mode, destination and their trip purposes when observing it at a finer level. Once there are changes in urban planning, area development and network structure, traffic demand and its distribution will also be significantly affected. Such situations occur quite often in many cities in China currently. In this case, it is difficult to solely use traffic simulation as complement for network-wide traffic monitoring. Online simulation and calibration with real-time data is thus a great extension to the traffic managers toolbox. Following several research works executed within the DLR fundamental projects, such as DELPHI, VABENE and VABENE++ ((Behrisch, et al., 2010) (Erdmann, 2012) (Bieker, Behrisch, & Ruppe, 2012)) and the on-going DLR project D.Move, SUMO (Alvarez Lopez, et al., 2018) can be integrated into a real-time traffic information platform and provide both real-time simulated and predicted traffic data. Moreover, a module to online calibrate flows and speeds with use of sensor and FCD data is implemented together with data processing.

This paper describes the simulation part of the traffic and environmental monitoring system, which efficiently and effectively monitors and prospectively manages the city traffic of Huainan, China. Moreover, the data processing part is also briefly explained, since its resultant data is used for calibration and prediction. DLR's traffic information platform KeepMoving (Brockfeld, 2014) is adopted in the aforementioned system, where SUMO should provide calibrated traffic data and the resultant emission information for real-time and predicted traffic state especially at places where no sensors and FCD are available. In order to achieve the goal, the online calibration module is modified for fitting the information system and the used Oracle database is extended for generating and delivering emission information. In the following, the framework and the concept used for data processing and online calibration is firstly introduced. The data preparation for simulating the traffic in Huainan is then explained. After that, the calibration results are presented, and the conclusions and remarks are then made at the end.

2 Framework and methodology

The applied online process consists of two parts, i.e. data processing and simulation with calibration and prediction. Their workflows are illustrated in Figure 1 and Figure 2 respectively. These two components are executed by the same script, but with different option settings. Besides, they can share the same configuration file, in which many settings can be defined, such as time period, data type(s), data receiving interval and aggregation interval, data duplication check, simulation inputs, calibration data types, data quality threshold, interval, simulation type (micro/meso), data cleaning interval, prediction, database connection. A database, e.g. PostgreSQL or Oracle, is required for data processing, exchange and output saving. The whole process chain is implemented in Python, and has been customized and enhanced for the Huainan ITS system.

The data process part include data correction, data aggregation, data fusion and data extrapolation. Currently, edge-based FCD and traffic measurements (speed and flow) either from stationary sensors or other data providers can be considered in the data processing, where flow types include passenger cars and trucks. As mentioned before, data processing and online calibration and prediction needs to be started separately. In the former case, speed average at each detector group is weighted with respect to the number of flows, and the interval-based data quality indicator is calculated according to the number of available data records in a pre-defined interval. If two or more detector groups locate at the same edge, average values will be derived and used. In the latter case, required data and data format for calibration needs to be in the pre-defined tables of the respective database. Moreover, route-based daily traffic demand is required as base. The related edge-based route sets/distributions and traffic demand per pre-defined interval for calibration are then generated accordingly and used together with the given network as inputs in simulation. In addition, detector group locations and the corresponding location-matching in the investigated sumo network are also required so that SUMO knows exactly where speeds and flows should be calibrated.


Start
Configuration data reading
(time period, data type(s), data receiving interval, aggregation interval, cleaning interval, duplication check, database connection…)
Online Data retrieving from the database per interval*/**
Edge-based FCD
Edge-based stationary sensor data
Data processing and writing into the database (speeds)
Data processing and writing into the database (speeds/flows)***
Data correction and gap filling
Data aggregation
Data correction and gap filling
Data aggregation
Data fusion****
Data extrapolation****
Is the time period reached?
no
yes
End

*	Data in the previous interval before the start time will be retrieved for data correction/gap-filling purpose.
**	Edge-based traffic data (speeds and flows) from other sources can been processed with respective added tables.
***	Two vehicle types (passenger car and truck) are considered.
****	It is also possible (1) not to or (2) to only execute data fusion and/or extrapolation by adapting the settings in the configuration file.

Figure 1: Overview of the online data correction and gap-filling process

When the whole process is started, data availability is checked during data processing. If data does not arrive in the database in time, the data process will be hold until the pre-defined time threshold is reached. Once data is prepared and filtered with the given quality indicator, the simulation will use the simulated traffic state from the previous interval as reference and try to get flow and speed data per interval in real-time for calibration, where it is possible to choose to use aggregated or fused data. For traffic prediction, data extrapolated from the fused data is used. If no data is available, simulation will still continue to run without calibration. With these data the on-line calibration will be executed for each pre-defined time interval, e.g. 10 minutes. Based on that, traffic predication will be made for the pre-defined duration, e.g. 30 minutes. When travelling speeds in the simulation differ from the respective measured edge speed averages, the related maximal speeds will be adjusted in the corresponding intervals accordingly. When simulated flows are greater than the related measured flows, route sampling actions will be executed in the respective edge-based route sets. Vehicles with the sampled routes and the current edge travelling speeds will then be inserted to the corresponding edges in the simulation for eliminating the differences between the simulated and measured flows. If the flow situation is the other way around, vehicles will be sampled and then directly removed from the network. SUMO tries to delay the insertion process of new vehicles as long as possible in order not to force the later removal of vehicles coming from the "real" demand (are not a calibration result). In addition, failure on vehicle insertion could happen when no space on the respective edges is available.

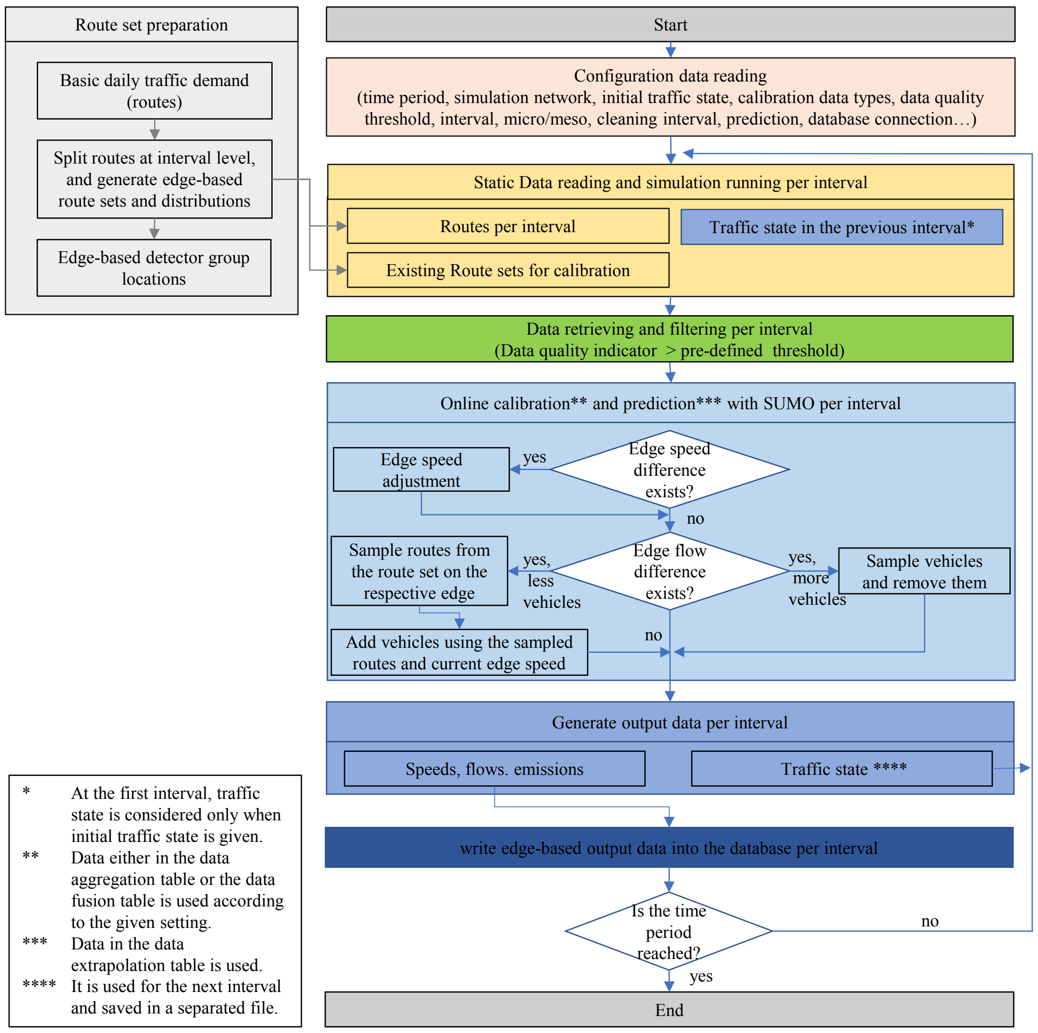

Route set preparation
Basic daily traffic demand (routes)
Split routes at interval level, and generate edge-based route sets and distributions
Edge-based detector group locations
Start
Configuration data reading
(time period, simulation network, initial traffic state, calibration data types, data quality threshold, interval, micro/meso, cleaning interval, prediction, database connection…)
Static Data reading and simulation running per interval
Routes per interval
Traffic state in the previous interval*
Existing Route sets for calibration
Data retrieving and filtering per interval
(Data quality indicator > pre-defined threshold)
Online calibration** and prediction*** with SUMO per interval
Edge speed adjustment
yes
Edge speed difference exists?
no
Sample routes from the route set on the respective edge
yes, less vehicles
Edge flow difference exists?
yes, more vehicles
Sample vehicles and remove them
Add vehicles using the sampled routes and current edge speed
no
Generate output data per interval
Speeds, flows. emissions
Traffic state ****
write edge-based output data into the database per interval
Is the time period reached?
no
yes
End
* At the first interval, traffic state is considered only when initial traffic state is given.
** Data either in the data aggregation table or the data fusion table is used according to the given setting.
*** Data in the data extrapolation table is used.
**** It is used for the next interval and saved in a separated file.

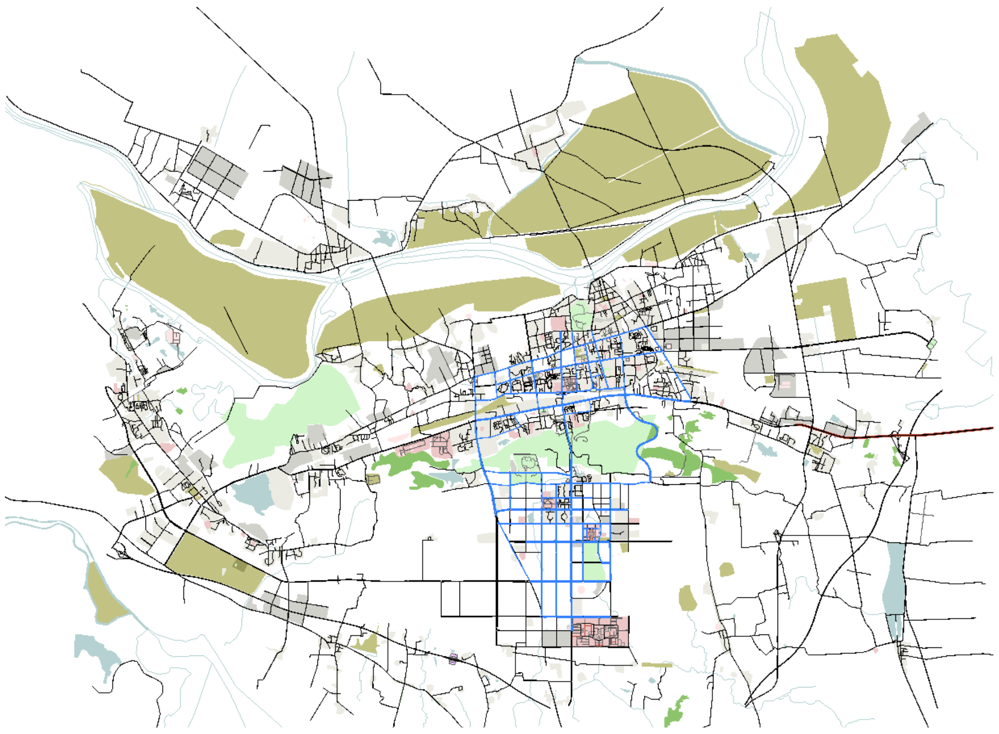
Figure 2: Overview of the online simulation and calibration process

As output, the simulated and calibrated data about traffic efficiency (flows and speeds) and emissions (CO_2, CO, NOx, PMx, HC) will be directly written into the database every interval according to the customized setting. Traffic state of each interval will be saved on the used server as well and used as reference for the simulation in the coming interval. The whole process will be iteratively executed until the pre-defined time period is reached. The simulated and predicted data can be used as basis for complementing real-time traffic state in the city and visualized in a selected platform, such as the KeepMoving Portal. In order to keep the whole processing chain continually running, a mechanism is built so that the processing chain will be automatically started if the server is restarted for any reason.

3 Data preparation

3.1 Simulation network

The OpenStreetMap (OSM) database, released in the 3rd quarter, 2019, was used as base for setting up the simulation network with SUMO's netconvert. Due to the limited road geometry and signal information in OSM the network has been further manually adjusted according to satellite pictures from Google Maps and images from Baidu Maps. Moreover, detailed geometry information for the roads in the old town and new central areas, obtained from the project partner, has been used for refining the network. The overview of the simulation network and the detailed road geometry example are shown in Figure 3 and Figure 4**Fehler! Verweisquelle konnte nicht gefunden werden.** respectively. The total network contains about 10000 edges and 5000 nodes with a total length of about 2600 km.



* the available detailed road geometry information is marked in blue

Figure 3: Simulation network layout

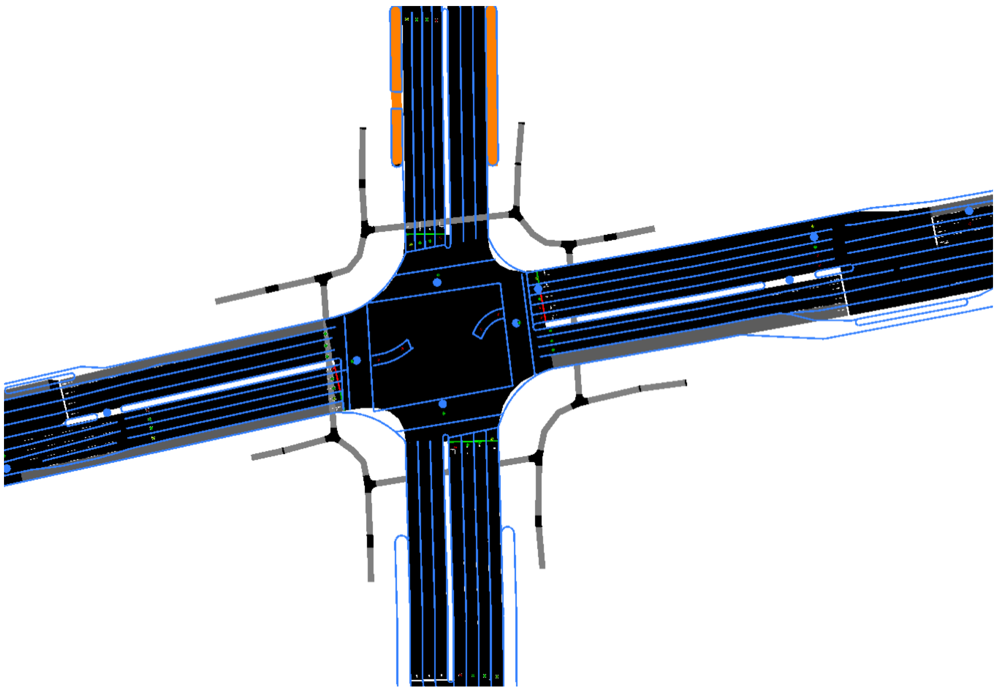


Figure 4: Illustration example of an intersection with detailed geometry information

3.2 Traffic demand and zone connections

The OD-Matrix data for the Huainan City is available and is generated by the transportation planning work, conducted in 2009 (Huainan City Planning Bureau; School of Transportation of the Southeast University, 2010). 81 traffic analysis zones (TAZ) were defined, as shown in Figure 5. An OD matrix prediction for 2020 is also available. However, this OD matrix is based on 26 TAZ, aggregated from the aforementioned 81 TAZ and for one peak hour period. Based on the estimated peak-hour traffic demand for 2020 and the populations of the small traffic analysis zones, used in the transportation planning from 2009, the peak-hour traffic demand was disaggregated into 81 traffic analysis zones. With the time-series data from the detectors 5-min flow distributions both weekdays and weekends were generated. Together with these flow distributions along a whole day and the relevant survey result, made in 2009, the daily traffic demand was derived and the departure times of the trips in each zone were then refined accordingly. Totally, the number of estimated daily trips is around 1.34 million.

Moreover, traffic analysis zone data is also available and converted from the transportation planning software TransCAD (Caliper Mapping & Transportation Software Solutions, 2021) format to the shapefile format. When further converting the shape file into SUMO format, the map is distorted to a certain degree and the respective correction work had been carried out manually. Together with the prepared network and the TAZ polygons the roads with a maximum travelling speed 50 km/h in each zone are selected with SUMO's tool edgesInDistrict.py and used as zone connections for traffic assignment. Some connections in TAZ 63, 64, 65 and 66 were manually adjusted due to the restricted road accesses.

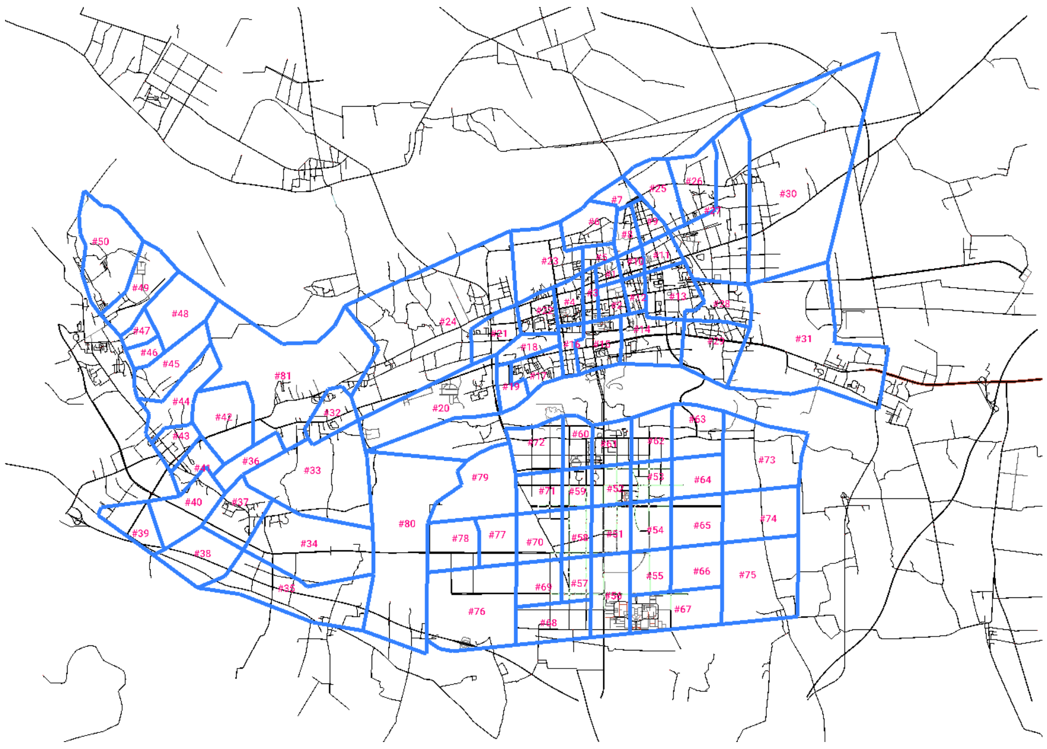
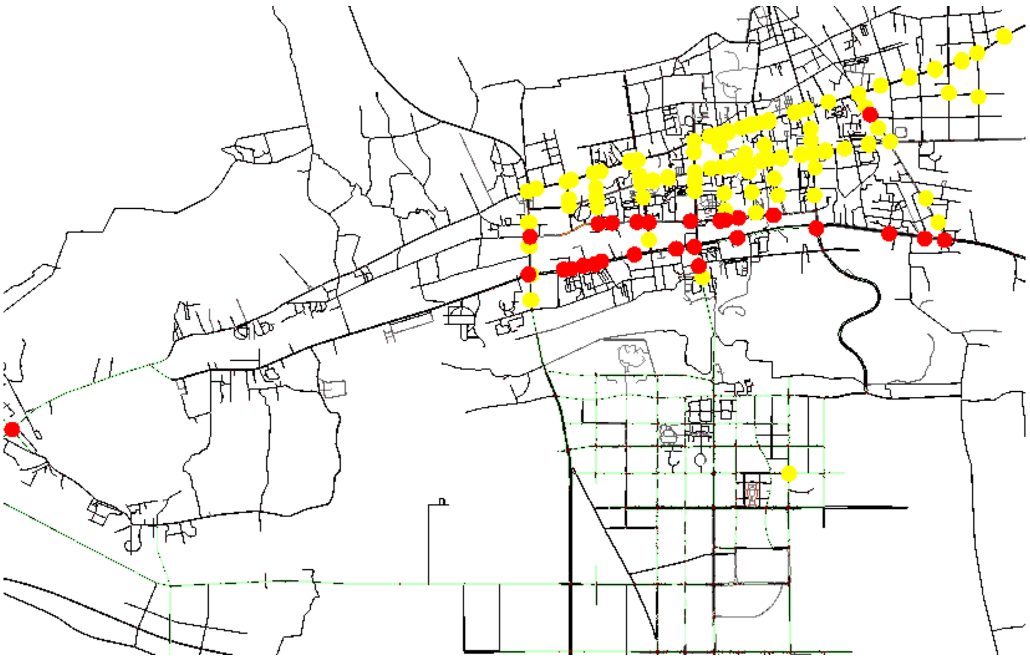


Figure 5: Distribution of the 81 traffic analysis zones in the City Huainan

3.3 Stationary sensor data

In order to enable the simulation to consider sensor data for calibration it is necessary to match the sensors to the respective lanes in the simulation network. Currently, 709 sensors are connected to the database. 509 of them are microwave-based sensors and the rest of the sensors are video-camera based sensors. Each detector is either deployed on or connected to a lane and detected data is transmitted back to the database every minute. Detectors on the same edge and closed to each other are grouped as a detector group. Figure 6 gives an overview of the detector groups' locations and indicates that most of the detector groups are in the old down town area. Only one detector group locates in the southern and western area and the new central area respectively. It is expected to have some more sensors installed in the new central area when the respective development and construction works are finished. It implies that the simulation and calibration performance with respect to the reality will be severely limited outside the old town area.



*: red ones are video-camera based sensors, while yellow ones are microwave-based sensors.

Figure 6: Locations of the stationary sensor groups

4 Simulation setting and execution

With the concern of the real-time operational readiness mesoscopic simulation is applied (Eissfeldt, 2004). In order to get route-based daily traffic demand a traffic assignment with the trip-based traffic demand, TAZ information, zone connections, described in Section 3.2, and DUAROUTER was executed. The resultant route file was then further used as described in Section 2. Based on the pre-defined interval, the given daily route file will be split into several files, i.e. 900 sec for the case study ITS Huainan. Only the temporally corresponding route files will be used in each simulation interval. Moreover, the routes and the route distribution on each calibrated edge will be generated as well.

When executing the simulation, a python script, as main script, is called to start the whole calibration process, where the definition of database tables and the general functions to get schemas, insert data into the database, update data and compare data can be found there. A configure file will then be read for mainly getting the following information:

1. start and end simulation/calibration times,
2. input files and their locations,
3. the intervals for data processing, calibration and prediction,
4. maximum delay for receiving sensor data,
5. data source for calibration, i.e. data either from FCD, sensors or data fusion result,

6. calibration parameter, i.e. only flow or both flow and speed,
7. action, i.e. either data processing or simulation with calibration and prediction,
8. data quality threshold,
9. look-back time for prediction,
10. prediction action and period
11. database connection information, as well as
12. durations to keep the data in the database and the output files on the used server.

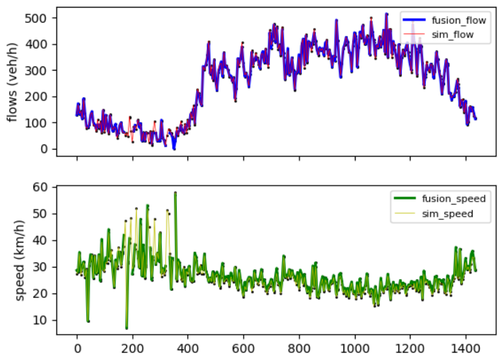
For the data processing and simulation in the Huainan project, the intervals in Point 3 are set to 5 minutes, which corresponds to the interval used at the KeepMoving platform. 2 minutes is used as data transmission delay, so that the data processing will be hold at most for 2 minutes if corresponding data does not arrive in the database. Moreover, both fused speed and flow data is selected as data source for calibration. When running the simulation, the traffic state for each interval is saved and used as input for the next simulation interval in order to properly reflect the respective traffic situation in the network. Regarding traffic prediction, the look-back time for prediction and the prediction period are 10 and 30 minutes respectively. It means that traffic prediction at 5-min interval for 30 minutes will be made according to every past 10-min simulation result. With the concern of the limited reliability of basic daily traffic demand, due to on-going road constructions and urban development in Huainan, only data with a quality indicator larger than 0 is considered in the online calibration which will exclude all completely disabled detectors.

Currently, the maximum experienced simulation duration for each 5- min interval is around 130 s, whilst the common simulation duration is between 70 and 90 sec. The total required time from data processing to the output, writing to the database, is less than 5 minutes, corresponding to the expected performance.

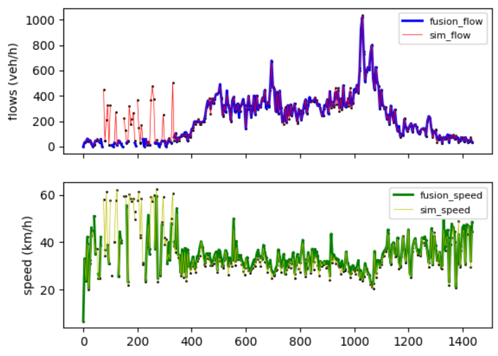
5 Preliminary results

The establishment of the whole ITS system and environmental monitoring system is newly finished. Currently, it is under the testing and maintenance phase. 2 weeks data from 2021.01.14 to 2021.01.31 has been used to examine the calibration performance. It is noticed that online 5-min time-series data is not yet entirely available for most of detectors currently. According to the used data set, it is quite common that data gaps exist for some short periods. Sometimes, data gaps also happen for longer periods. When looking at the calibration result at 5-min interval, around 80% of the calibrated flows and more than 95% of the calibrated speeds have an absolute relative error (ARE) less than or equal to 15%. At the hourly flow level, the GEH statistic is used to evaluate the hourly flows, aggregated from the 5-min flows. The result shows that around 92% of the calibrated flows have a GEH-value less than or equal to 5, which indicates a good fit.

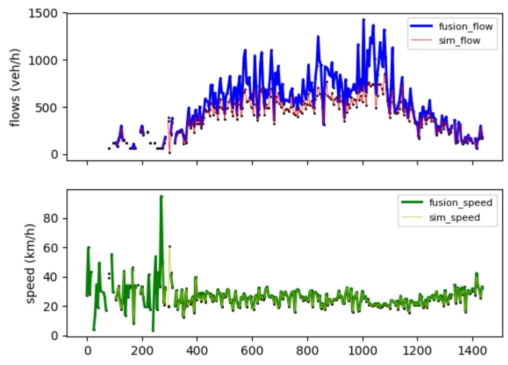
Moreover, four edges are selected as examples for further investigating the calibration performance along a whole day. In Figure 7 (a) and (b), it shows that, as expected, the simulation and calibration result is quite promising for both flow and speed, even when speed fluctuates largely. When traffic congestion raises in the simulation, the calibrator cannot insert more vehicles into the network for reflecting the flows in the reality. The simulated flows are then lower than the measured ones, such as the situation indicated in Figure 7 (c). Such case may sometimes occur due to other reasons, such as that travel speeds are set too low due to the low accuracy of detected speeds and lower quality threshold. More investigation on it is then needed. Figure 7 (d) shows that the sensor data is only partly available (until the early afternoon). With the FCD supplement, the related edge speeds in the late afternoon can still be properly calibrated.



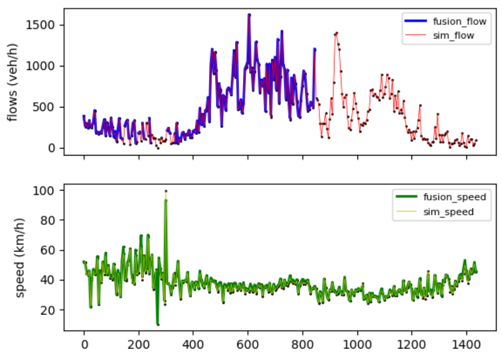
(a) at Edge with ID 1467 (2021-01-13)



(b) at Edge with ID 2402 (2021-01-13)



(c) at Edge with ID 2478 (2021-01-16)



(d) at Edge with ID 7760 (2021-01-23)

Figure 7: Comparison of real and simulated data for the selected edges

6 Conclusion and remarks

An online continuing calibration and prediction process with SUMO and its enhancement has been described in this paper. Generally speaking, the simulation keeps a defensive manner when conducting online calibration, i.e. the simulated state (flow and speed) is preferred to be untouched until the end of each interval is approached. This mechanism may result in failed vehicle insertion and affect the calculation of the number of vehicles in the respective intervals sometimes. For example, two vehicles need to be inserted at Interval x. To be sure that no other vehicles will be going to enter the observed edge in Interval x, a vehicle insertion action will not be done until the end of Interval x. Sometimes, such action cannot be carried out due to lack of space on the related edge, and affects calibration performance. Under this circumstance, the current analysis result indicates that the calibration and the overall computation performances are principally still quite well. Some larger fluctuation in speed occurs, especially at late night or very early morning. More related investigation is needed.

City Huainan is continuously realizing its urban planning and development as well as completing its road network system. Several road construction projects with ITS based infrastructure are on-going. It implies that the related parts in the simulation network need to be continuously updated accordingly for reflecting actual traffic state. Moreover, more real-time traffic information should be then available. It is expected that the simulation result can be further improved later on. Due to the existing and coming changes in land use and road network it is indeed necessary to update the daily traffic demand data and, if necessary, the respective TAZ definition either with the conventional demand modelling method or/and with other data sources, such as data from OSM, Wikidata, mobile phones, social media and navigation systems. The overall online calibration and prediction performance will also be benefited by the updated daily traffic demand.

References

Alvarez Lopez, P., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., . . . Wießner, E. (2018). Microscopic Traffic Simulation using SUMO. *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, (S. 2575-2582). Maui, USA.

Behrisch, M., Hohloch, M., Junghans, M., Kuhns, G., Krajzewicz, D., & Wang, Y.-P. (2010). Traffic management decision support based on on-line data. *POLIS CONFERENCE - Innovation in transport for sustainable cities and regions.*

Bieker, L., Behrisch, M., & Ruppe, S. (2012). EmerT – a web based decision support tool for Traffic Management. *19th ITS World Congress 2012, 22-26 October 2012.* Vienna, Austria.

Brockfeld, E. (2014). Der Reiseassistent KeepMoving. *5. Tagung Mobilitätsmanagement von Morgen "Auf dem Weg zur emissionsarmen Mobilität".* Berlin, Germany.

Caliper Mapping & Transportation Software Solutions. (2021). *TransCAD Transportation Planning Software*. Von https://www.caliper.com/tcovu.htm abgerufen

Eissfeldt, N. (2004). *Vehicle-based modelling of traffic . Theory and application to environmental impact modelling.* Dissertation, Universität zu Köln.

Erdmann, J. (2012). Online-Kalibrierung einer Mikroskopischen Verkehrssimulation. *VIMOS-Kolloquium 2012.* Dresden, Germany.

Huainan City Planning Bureau; School of Transportation of the Southeast University. (2010). *Urban Comprehensive Transportation Planning of Huainan City (2009～2020).* Huainan, China.

SUMO User Conference 2021
https://doi.org/10.52825/scp.v2i.89

Published: 29 Jun. 2022

Using Deep Reinforcement Learning to Coordinate Multi-Modal Journey Planning with Limited Transportation Capacity

Lara CODECÁ and Vinny CAHILL
Trinity College Dublin, School of Computer Science and Statistics, Dublin, Dublin 2, Ireland.
Lara.Codeca@tcd.ie, Vinny.Cahill@tcd.ie

Abstract

Multi-modal journey planning for large numbers of simultaneous travellers is a challenging problem, particularly in the presence of limited transportation capacity. Fundamental trade-offs exist between balancing the goals and preferences of each traveller and the optimization of the use of available capacity. Addressing these trade-offs requires careful coordination of travellers' individual plans. This paper assesses the viability of Deep Reinforcement Learning (DRL) applied to simulated mobility as a means of learning coordinated plans. Specifically, the paper addresses the problem of travel to large-scale events, such as concerts and sports events, where all attendees have as their goal to arrive on time. Multi-agent DRL is used to learn coordinated plans aimed at maximizing just-in-time arrival while taking into account the limited capacity of the infrastructure. Generated plans take account of different transportation modes' availability and requirements (e.g., parking) as well as constraints such as attendees' ownership of vehicles. The results are compared with those of a naive decision-making algorithm based on estimated travel time. The results show that the learned plans make intuitive use of the available modes and improve average travel time and lateness, supporting the use of DRL in association with a microscopic mobility simulator for journey planning.

1 Introduction

Transportation capacity in cities is a limited resource whose use needs to be optimized to achieve sustainable mobility. One approach to such optimization might be through coordination of travellers' itineraries achieved by providing them with journey plans that respect their individual goals and preferences while making the best use of available capacity. One scenario in which this might be possible, and the use case considered in this paper, is where event-goers are travelling to a large-scale event, such as a concert, conference, or sports event, each with the goal of arriving as close as possible to the event's start time. The lack of coordination of travel to such events coupled with the fact that capacity may not be sufficient to satisfy such a spike in demand is often reflected in traffic congestion present before, sometimes during, and after these events [1]. We hypothesize that the venue provides an application to the event-goers that enables them to request a journey plan to reach the event on time, considering their access to different transportation modes, and in the context of other background traffic (e.g., the 'usual' traffic in the vicinity of the venue) and the plans already offered to other event-goers. The event-goers need to specify their requirements (if any), request a trip plan, and execute it.

Among the issues to be addressed to make such a model viable, including incentivizing its uptake by event-goers and their compliance with the plans provided, designing coordinated individual plans is particularly challenging as a fundamental trade-off exists between balancing the goals and requirements of each event-goer and the optimization of the use of available capacity. In this paper, we assess the viability of using Multi-Agent Deep Reinforcement Learning (MADRL) to learn coordinated multi-modal trip plans aimed at maximizing just-in-time arrival while considering the limited capacity of the infrastructure. Each event-goer is provided with a plan to arrive at the venue on time while also minimizing waiting time at the venue before the event starts. Plans are derived from a policy learnt using Reinforcement Learning (RL) [2], in which agents (representing event-goers) learn how to achieve their goal (i.e., just-in-time arrival) through trial and error while interacting with an environment by taking actions (e.g., wait or start a journey using a particular transportation mode) and receiving a reward (or punishment) depending on the expected future benefit of performing that action with respect to achieving the goal. Deep Reinforcement Learning (DRL) [3] incorporates the use of artificial neural networks to solve tasks that require a high-dimensional feature space representation. In our case, because of the need for agents to observe the effects of their actions and those of other agents on the environment, training uses a microscopic mobility simulator in which the effects of the actions performed can be assessed.

For our experiments, we made use of the Proximal Policy Optimization (PPO) [4] algorithm provided by Reinforcement Learning library (RLlib) [5], and we implemented multiple transportation environments (based on a hand-crafted synthetic mobility scenario) offering different reward models, action spaces and state space features. We then evaluated the achievement of the agents' goals under different coordination strategies and with different combinations of available transportation modes (especially those having different observed travel times). For public transportation, we varied the capacity and reliability of services and assessed the impact of parking requirements on specific vehicle types. Finally, we explored the impact of specific hard constraints, the ownership of vehicles, on achievement, leaving the study of soft constraints (e.g., user preferences) as future work.

Of particular relevance to this scenario is how the required coordination of agents' actions is achieved. Following [6], active coordination can be achieved by agents learning to explicitly exchange relevant information and make joint decisions, incurring an overhead during learning [7]. Passive coordination (whether implicit or explicit) is usually driven by the problem to be solved and the environment at hand. In our case, the transportation capacity is limited and shared. Here, implicit coordination is learnt in response to agents' observations of the effects of other agents' actions, while explicit coordination can be achieved by observing residual demand and tuning the reward model to penalize each agent when other agents' goals are not achieved (see Section 4.2 for further details). We compare both implicit and explicit passive coordination strategies for all the experimental settings.

To combine microscopic mobility simulation using Simulation of Urban MObiltiy (SUMO) [8] and the distributed DRL infrastructure provided by RLlib [5], we implemented a general-purpose framework that enables the study of a wide variety of transportation problems. The connector between the mobility simulator and the RL infrastructure has been released to the community on GitHub[1], and it is available in the official RLlib release since v1.1.0

The results show that our agents can learn coordinated multi-modal trip plans by exploring the transportation environment, figuring out how long they need to wait before leaving for the event, and which transportation mode is more appropriate to reach the destination on time. We compared our results with a naive trip plan based solely on mode preference and estimated travel time, showing that by following the learnt trip plans, there is an improvement in the travel time and lateness, and that the impact of explicit passive coordination on the waiting time before the event varies depending on the experiment characteristics.

The main contributions of this paper are the implementation and evaluation of a MADRL framework

[1]RLlib SUMO Utils: https://github.com/lcodeca/rllibsumoutils Last access: February, 2022

to study coordinated multi-modal trip planning with capacity constraints, and the demonstration that the use of DRL with a mobility simulator is a viable option to study this class of problems. Additionally, RLlib SUMO Utils (the connector for the RLlib framework and SUMO), the mobility simulations, and the MADRL framework are available on GitHub[2].

2 Related Work

Historically, a significant amount of work has been done on the use of Artificial Intelligence (AI) and Machine Learning (ML) techniques in the context of Intelligent Transportation Systems (ITS). More recently, motivated by the large quantity of transportation-related data that is becoming available as well as their success in other application domains, the use of deep learning and now DRL to build mobility models and to optimize mobility is being studied extensively. However, to date, we are not aware of any work applying DRL to the coordinated journey planning problem with constrained capacity that we consider.

2.1 Deep Reinforcement Learning in Transportation

Two extensive surveys of ML techniques, and more precisely deep learning, applied to ITS are available in [9] and [10]. Currently, even with a large volume of traffic data, it remains challenging to build reliable (predictive) models to be used to address issues such as traffic flow forecasting, travel demand prediction, traffic signal control, transportation network representation, and autonomous driving using these techniques. Moreover, the use of deep learning is constrained due to intrinsic limitations, scalability and portability.

In our case, where we are dealing with infrequent traffic patterns (i.e., due to sporadic large-scale events), it is hard to acquire the amount of data necessary to apply the usual Deep Learning (DL) methods. We address the problem by using mobility simulation as the learning environment. A simulation of the usual mobility can be built based on data on the transportation infrastructure (e.g., OpenStreetMaps [11]), mobility data, and statistical information on the population usually provided by the government (e.g., [12, 13, 14, 15, 16]). Once realistic background traffic is available, the model can be adjusted to simulate specific requirements such as capacity constraints. Moreover, in the case of RL, the effects of agents' actions on the environment can be easily modelled. Additionally, most simulators provide heuristics for travel time estimation and routing. The use of a simulated environment facilitates the development of RL methods.

Emerging applications of DRL in transportation are mostly limited to optimization of fleet dispatching, autonomous driving and Traffic Signal Control (TSC). For example, DeepPool [17] is a distributed model-free application for ride-sharing. It uses Deep Q-Network (DQN) to learn optimal dispatching of the fleet using a deep neural network trained with travel demand data. It has been tested using a real-world database of taxi trip records from New York. Another example of a distributed model-free approach to dispatching using DQN is FlexPool [18], where the optimization addresses transportation of both passengers and goods. A more generic study of large-scale fleet management is presented in [19], where the authors propose a contextual MADRL framework that uses both DQN and actor-critic algorithms to optimize dispatching and routing. Both approaches are extensively tested through empirical studies. Most of the tasks that comprise autonomous driving are surveyed in [20], where a taxonomy of the tasks is presented and the challenges of applying DRL are discussed. Moreover, the survey explicitly discusses the role of traffic simulators for learning and validation purposes. Finally, an extensive survey on the use of DRL applied to TSC is available in [21]. Here the authors discuss the metrics

[2]Persuasive: https://lcodeca.github.io/persuasive/ Last access: February, 2022

usually associated with intersection control, present the deep learning methods deployed and the traffic simulators used to test and evaluate them, discussing their limitations.

All the previous examples are a demonstration that DRL can be used to study ITS problems, and provide a step forward from DL. Additionally DRL has previously been used in similar planning problems, such as path planning for robots [22], unmanned aerial vehicles [23], and unmanned ships [24]. However, to the best of our knowledge, it is not usually applied to multi-modal trip planning, where the optimization needs to take into account the presence and behaviour of different transportation modes, and the limitations arising from constrained transportation capacity.

2.2 Multi-Modal Trip Planning Solutions

Although the problem implied by the need for coordination to address capacity constraints is not investigated, much work has been done on multi-modal trip planning using a multitude of methodologies.

The first issue to be addressed is the data aggregation required to build a layered interconnected graph representing all available modes of transport, the relevant transportation infrastructure, and possibly user preferences. An example of multi-modal transportation graph construction is presented in Trans2Vec [25], an analytical framework for multi-modal transportation recommendation. It uses joint representation learning with anchor embedding to capture the preferences of the users and the traffic demand, and its effectiveness has been evaluated using real-world data. Similarly, [26] presents a co-modal framework for optimal trip planning that uses multi-objective optimization based on genetic algorithms to provide a solution in terms of user (with preferences), mode of transport, and route. This system is tested with a real case study of Lille (Nord Pas de Calais, France). An example of exploration of the multi-modal transportation network is presented in [27], where graph embedding techniques are implemented to provide multi-modal trip plans to users. Additionally, the authors present a post-processing technique to filter out the inconsistency between the objective function and evaluation metric, and evaluate it using real-data provided by Baidu-Maps.

Another problem to be taken into account is the reliability of the estimation used to decide which plan is 'better'. Another graph-based approach is presented in [28], where the authors present a network search algorithm applied to a graph based on both scheduled and unregulated modes, aimed at improving the reliability of the itineraries built with multiple hops and multiple modes (planes included). Additionally, they investigate the amount and quality of data required to build a reliable model. Another project that takes reliability into account is [29], where delays are directly modelled and the problem is formalized as stochastic shortest path optimization. The proposed solution is meant to optimize just-in-time arrival using Q-Learning, and the validity of this approach is measured with experimental results based on the cities of Beijing, Munich and Singapore.

One additional issue arising is the complexity of generating these models and computing plans. [30] presents a solution based on dynamic transfer patterns, where the public transportation schedule is merged with the real-time passenger information and additional links are generated in the knowledge-graph to take delays into account. This methodology is among those implemented in OpenTripPlanner[3]. Additionally, the paper presents a discussion on the complexity of generating these plans, the amount of pre-computation required, and possible solutions to speed it up. Finally, two personalized multi-modal trip recommenders are presented in MTRecS-DLT [31] and RouteMe [32]. MTRecS-DLT combines convolutional neural networks and gradient-boosted decision trees to extract context features from the training data and then learns how to build trip plans that satisfy user needs while respecting their various preferences. The methodology has been tested using the dataset provided by the Context-Aware Multi-Modal Transportation Recommendation challenge promoted by Baidu. Similarly, RouteMe is a mobile recommender system that provides multi-modal plans to the users. Interestingly, it introduces

[3]OpenTripPlanner: https://www.opentripplanner.org/ Last access: February, 2022

the concept of collaboration, where through the ranking and evaluation of the plans provided by the users, collaborative filtering of the routes is applied to the knowledge-base, improving it over time. The methodology has been evaluated through a small user study (17 people) where a prototype application has been implemented and the users provided standardized feedback.

All the examples mentioned provide some limited solutions to the multi-modal trip planning problem by focusing on specific optimization requirements.

2.3 Coordination in Planning

Considering explicitly the issue of coordinated trip planning, COPTER [33] incorporates state-of-the-art solvers to deal with energy consumption reduction in multi-modal trip planning. In this case, both single user planning and cooperative decision making need to take into account the trade-offs between user acceptance and system-optimized routes, with the focus on energy savings. The evaluation is done through a simulation study based on Los Angeles, and a planned pilot deployment is discussed.

The specific problem we are studying is multi-modal trip planning with limited transportation capacity. More precisely, our problem of limited capacity cannot be addressed with construction works and long-term solutions. The transportation infrastructure is not designed to sustain the demand spike and consequent overload arising from a large-scale event. The usual solution comprises a mix between diverting traffic from the vicinity of the venue and enhancing public transportation capacity by increasing its frequency and adding shuttles [34].

To the best of our knowledge, we are the first to explore the use of microscopic mobility simulation and MADRL to deal with journey planning with passive cooperation, where the plans are discovered while exploring the transportation environment.

3 Reinforcement Learning with SUMO

RL is an ML approach to overcoming the limitations of manually-designed algorithms and policies for control and optimization. In Multi-Agent Reinforcement Learning (MARL), a set of agents interact with and explore an environment, based on which they optimize their behavior to achieve a goal. Many real-world problems offer high-dimensional state spaces so that finding an optimal solution is a hard problem. Often referred to as the curse of dimensionality, the complexity of a problem increases exponentially with the number of state dimensions. Deep learning is a branch of Machine Learning (ML) that uses artificial neural networks as function approximators to deal with such high dimensional state spaces. When deep learning is used with Reinforcement Learning (RL), we refer to Deep Reinforcement Learning (DRL) and Multi-Agent Deep Reinforcement Learning (MADRL).

MDP. RL problems are typically formalized as Markov Decision Processes (MDP). An MDP [2] is described by a tuple consisting of: a set of states S, a set of actions A, a state transition probability function $T(s_{t+1}|s_t, a_t)$ that defines the probability distribution of the next state s_{t+1} given the current state s_t and action a_t, a reward function $R(s_t, a_t)$ that determines the reward r_t (a scalar value) based on the action a_t performed in state s_t, a discount factor $\gamma \in [0, 1]$ representing the impact of future rewards on the decision-making. The Markov property is a basic requirement for an MDP, and it assumes that the current state s_t contains all relevant information from the past states, implying that the state s_{t+1} depends only on the current state s_t and action a_t. Additionally, MDP models can be used to represent both continuous/infinite tasks or episodic tasks where a terminal state $s_{terminal}$ is defined.

Learning. In RL, the interaction with the environment is discretized in steps t. For every step, each agent receives information about the current state of the environment s_t. Based on s_t, each agent selects an action a_t to perform in the environment based on a policy that is learnt over time. The current policy π determines the behavior and the decisions made by the agents. In our case, the policy is a stochastic

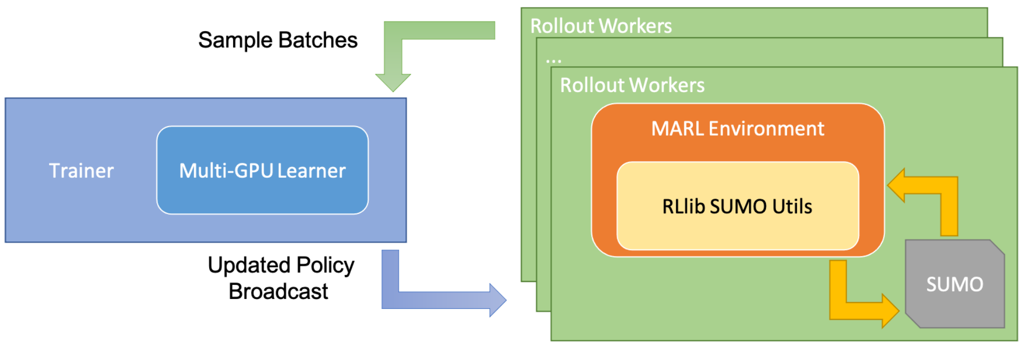


Figure 1: RLlib and SUMO Overview.

function that maps states to actions $\pi(a_t|s_t)$. The agents learn this mapping through the exploration of the environment. The learning mechanism is based on the reward r_t received from the environment after performing the action a_t. Each agent's goal is to maximize its cumulative reward in the long run.

Exploration vs Exploitation. The learning process introduces a trade-off between exploring the environment and exploiting the knowledge gathered previously. During exploration, the agent may perform random actions, trying something different from the last time a state was visited. During exploitation, the agent taps into the knowledge learnt, and performs actions that served it well in the past. The exploration vs exploitation trade-off brings up the additional issue of sampling, in which the state-action pairs need to be gathered from a sampling distribution, a dataset containing actual data, or in our case, a simulator. Deep learning typically requires a large number of samples to converge to a stable policy.

3.1 Distributed RL and Simulation of Urban MObility (SUMO)

Due to the complexity of the journey planning problem, scalability is an essential requirement of our architecture. For this reason, we make use of RAY [35], an open-source distributed execution framework used to scale ML applications. RAY provides a universal API for building distributed applications, and based on it, several libraries for solving problems in ML have been implemented. We use the Reinforcement Learning library (RLlib) [5], an open-source DRL library built on top of RAY. RLlib offers scalability and natively supports TensorFlow[4] and PyTorch[5], and is compatible with OpenAI Gym[6], a toolkit for developing and comparing RL algorithms. It provides a well-defined API to build and extend learning environments.

Simulation of Urban MObiltiy (SUMO) [8] is an open-source microscopic traffic simulator designed to handle large networks, that allows multi-modal mobility simulation, including pedestrians. SUMO implements the Traffic Control Interface (TraCI) API[7] providing interactive control of the mobility simulation. Using TraCI, we implemented RLlib SUMO Utils[8], an open-source general-purpose connector that extends the Gym environment provided by RLlib, and integrates the mobility simulation in the learning environment.

Figure 1 provides an overview of the training and evaluation infrastructure. The overall architecture

[4]TensorFlow: https://www.tensorflow.org/ Last access: February, 2022
[5]PyTorch: https://pytorch.org/ Last access: February, 2022
[6]OpenAI Gym https://gym.openai.com Last access: February, 2022
[7]SUMO Wiki: TraCI API https://sumo.dlr.de/docs/TraCI.html Last access: February, 2022 [8]RLlib SUMO Utils: https://github.com/lcodeca/rllibsumoutils Last access: February, 2022

is implemented by RLlib. The trainer (in blue) and the rollout workers (in green) can be deployed locally or on a distributed cluster thanks to the RAY orchestration. RLlib provides a high-level trainer class that handles all the interactions required to update a policy and evaluate it. Through the general-purpose trainer interface, one or more policies can be trained, checkpoints can be created and restored, and the policy can be evaluated. The rollout workers are an abstraction that implements the interactions with the learning environment, where based on a locally-stored policy, actions for the agents are selected and actuated in the environment, and samples (composed by s_t, a_t, r_t, and s_{t+1}) are collected in batches, and eventually sent to the trainer. Everything from the size of the batches, to the sampling policy is configured based on the learning algorithm. Additionally, a rollout worker can be deployed for training or for evaluation, depending on the trainer requirements. Based on the specific learning algorithm implementation[9], the number of rollout workers and multi-GPU learners may vary, and both sampling and policy updates can be asynchronous or synchronous. In general, each rollout worker can deploy more than one MARL environment to speed up sampling. However, due to constraints imposed by TraCI, only one RLlib SUMO Utils environment and the associated SUMO instance can be deployed for each rollout worker. The parallelization required to speed up sample gathering needs to be achieved by deploying multiple rollout workers at the same time. The MARL environment (in orange) implements the learning environment and it extends the general-purpose RLlib SUMO Utils Gym-like environment (in yellow). The MARL environment implements all of the APIs required by the trainer to understand both the state and action spaces, it implements the meaning of the actions and the reward model. The RLlib SUMO Utils environment takes care of setting up and moving forward the SUMO simulation, gathers the mobility information from TraCI and provides an interface to interact with the entities in the simulation.

3.2 Actor-Critic and Proximal Policy Optimization

For our purposes, we configured the implementation of a state-of-the-art actor-critic RL algorithm [2], Proximal Policy Optimization (PPO) [4], provided by RLlib.

The PPO algorithm collects a small batch of experiences while interacting with the environment, and it uses that batch to update the policy. After every policy update, the samples are discarded, and a new batch is collected with the newly updated policy. The main strength of PPO is that the changes between the old and the newly learnt policy are minimal by design. This reduces the variance in training at the cost of some bias, but improves the stability of learning, leading to faster training in costly simulated environments.

RLlib's implementation of PPO supports both TensorFlow and PyTorch multi-GPU optimizations, it allows both discrete and continuous actions (although we need only discrete actions), and it supports multiple deep neural network models (with attention and auto-regressive options). The PPO implementation we configured is synchronous, where for each training step, the learnt policy is sent to all the rollout workers. The policy is then used unchanged to gather all the samples required to fill the batches, and finally all the batches are sent back to the trainer, where the policy is updated.

In order to speed up the learning process, every agent in the system contributes to a single policy, sharing the knowledge acquired during the exploration. The stochastic policy represents a set of coordinated trip plans for the given number of agents operating in the same environment. To obtain its journey plan, an individual agent consults the policy (repeatedly) to choose actions based on its own circumstances and the state of the system. Based on the features representing the agent and the state of the system, an action (waiting or transportation mode selection) is picked and the trip plan is progressed. The metrics we use to compare the plans and evaluate the goodness of a policy are the observed travel

[9]RAY documentation: RLlib Learning Algorithms https://docs.ray.io/en/master/rllib-algorithms.html#rllib-algorithms Last access: February, 2022

time, the waiting time before the event, and the late arrival. Passive implicit cooperation is achieved through the transportation environment. Due to its limited capacity, the (selfish) decisions taken by each agent have a direct impact on the others, with increasing travel times and possible delays, to which other agents learn to react. Passive explicit cooperation is implemented by modifying the reward model (see Section 4.2), and directly penalizing each agent based on the number of other agents that failed in the task.

4 Experimental Setup

As explained, the experimental infrastructure is mainly composed of three parts, (i) the distributed orchestration provided by RAY [35], (ii) the RL infrastructure provided by RLlib [5], and (iii) the mobility simulation provided by SUMO [8]. In our case, we configured the PPO algorithm provided by RLlib, implemented an Open AI Gym-compatible learning environment integrating SUMO using RLlib SUMO Utils, and crafted a representative mobility scenario that provides the required transportation capacity limitations.

4.1 Learning Environment

The learning environment provides the mobility and transportation infrastructure. Agents represent event-goers, and their task is to explore trip plans to reach their destinations on time while their passive coordination is underpinned by the shared transportation infrastructure. The available actions depend on the experiment in question but are of two kinds: (i) *wait* (always available to each agent) and (ii) *select a transportation mode* and start the trip. Based on the experiment, the modes available are different (e.g., walk, bicycle, public transport, car, and/or powered two wheelers (PTW)), and the meaning of an action may vary (e.g., number or modes used together, requiring parking or not, the composition of public transport). During learning, all the agents independently explore the environment, but they feed all their observations to a common policy.

The features available from the environment include the *origin* and *destination* of each agent as latitude/longitude pairs, the *time to event* counting down the passage of time, discretized with a configurable parameter to 3 minutes in all experiments, and the current traffic situation computed using the *Estimated Travel Time (ETT) for each mode* provided by SUMO[10]. The ETT is discretized with the same parameter used for the time to the event. The *mode usage* is provided for event-goers that have already chosen a mode but have not reached their destinations. The *future demand* is defined by how many event-goers still need to make a mode choice. The *ownership by mode* is defined as a boolean flag, available only in specific experiments.

4.2 Reward Models

We implemented and evaluated three different reward models aimed at having all the event-goers arrive within a 15-minute window before the event, without being too early or too late. In all models, the agents receive a non-zero reward only at the end of their journey.

Simple In this model, used as a baseline, the rewards are 1 for achieving the goal of just-in-time arrival, -1 for arriving too early, -2 for being too late or when the agent does not arrive at the destination. Non-arrival may happen if the agent waits for too long, past the event time, or if a mode that is not usable is selected (e.g., a mode not available in the area, the means of transport not available to the

[10]SUMO Wiki: Estimated travel time computation https://sumo.dlr.de/docs/TraCI/Simulation_Value_Retrieval.html#command_0x87_find_intermodal_route Last access: February, 2022

agent, or the route associated to the mode is undefined). The simple model does not reward lower travel time experienced by agents or active coordination between them.

OTT The second model introduces time into the reward. The *observed time travelled*, the *waiting time* before the event, and the *late arrival* time are discretized and added to the reward as a penalty. The discretization is configurable, and in our experiments, it is set to 5 minutes. The reward function R_{OTT} for each agent is defined as

$$R_{OTT} = \begin{cases} 1 - ott & \text{if on time} \\ 0 - (ott + t_{wait}) * w_{wait} & \text{if too early} \\ 0 - (ott + t_{late}) * w_{late} & \text{if too late} \\ 0 - (ett_{max} * 2) * w_{late} & \text{if never arrived} \end{cases}$$

If the agent arrives on time (i.e., within the 15-minute window before the event), the only penalty is the observed travel time ott measured as the elapsed time from departure to arrival at the destination. In case the agent arrives too early, the waiting time t_{wait} (computed from the arrival at destination to the time of the event) is added to the ott and then multiplied by the waiting time modifier w_{wait}. In all our experiments $w_{wait} = 1.0$ for all agents. If the agent arrives too late, its lateness t_{late} is measured from the event time to the arrival at the destination. Similarly to the early arrival, t_{late} is added to the ott and then multiplied by the lateness modifier w_{late}. In all our experiments $w_{late} = 2.0$ for all the agents. In case an agent never arrives at the destination, the penalty computed is based on the assumption that the agent chose the worst transportation mode at the time of the event and arrived too late. In this case, the maximum estimated travel time ett_{max} is obtained by computing the ETT from origin to destination for each mode using SUMO, and then selecting the maximum. The penalty is then computed by doubling the ett_{max} and multiplying it with the lateness modifier w_{late}.

OTTCoord The last reward model is based on OTT but introducing an additional penalty based on how many of the other event-goers arrived too late. $R_{OTTCoord}$ is defined as

$$R_{OTTCoord} = R_{OTT} * w_{coord}$$

and is initially computed using R_{OTT}, and then multiplied by the penalty $w_{coord} \in]0, 1]$ defined as

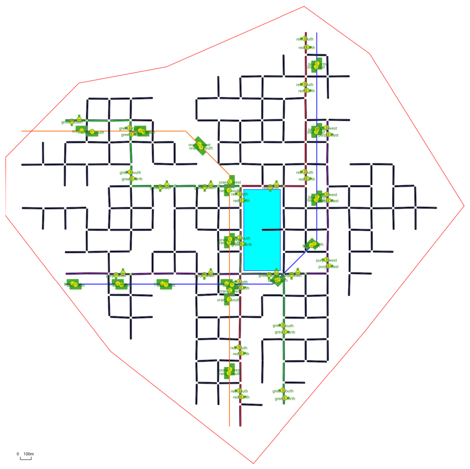
$$w_{coord} = \begin{cases} \frac{1}{\frac{1}{N_{agents}}} & if \sum A_{early} + \sum A_{ontime} = 0 \\ \frac{1}{\frac{\sum A_{early} + \sum A_{ontime}}{N_{agents}}} & otherwise \end{cases} \tag{1}$$

and representing all the agents that arrived late (or never). In the equation, N_{agents} is the number of agents, A_{early} is the number of agents that arrived early, and A_{ontime} is the number of agents that arrived on time. If every agent has a viable plan, and there are no late agents, $w_{coord} = 1$ and there is no penalty.

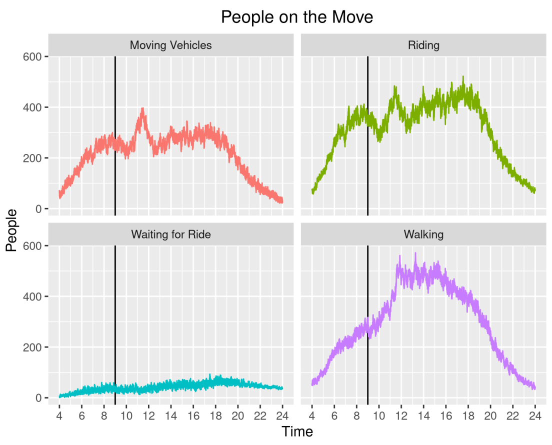
4.3 Mobility Scenario

The mobility models provided by SUMO allow us to configure different behaviours based on the vehicle type in use. With SUMO's sub-lane model, [36] it is possible to have smaller vehicles (e.g., bicycles and motorbikes) seeping through the rest of the traffic. For pedestrians, we use SUMO's striping model[11], where people walk on sidewalks and pedestrian crossings, and are capable of interacting with vehicles, increasing congestion, and delaying each other.

[11]SUMO Wiki: Striping Model definition https://sumo.dlr.de/docs/Simulation/Pedestrians.html#model_striping Last access: February, 2022



(a) SUMO Network



(b) Mobility plans generated with SAGA.

Figure 2: SUMO Random Grid scenario

We configured a mobility scenario based on a random grid topology that would not serve the 1000 agents we use in training without optimization of the load among the available transport modes and a staggered departure time for each agent. In Figure 2a, the roads are mapped in black, and the blue box in the middle represents the venue, with its main entrance in the middle. The yellow/green boxes are bus and metro stops. We introduced two metro lines and three bus lanes. The coverage is uneven by design, and the buses share the road with the other vehicles. We have the capability of introducing variations to the public transportation infrastructure depending on the requirements of different experiments. Figure 2b presents the background mobility generated with SAGA [37], an activity-based multi-modal traffic generator for SUMO. Based on the default activity chains provided by the tool, we generated 24-hours of background mobility for 20K people. In red, the moving vehicles are composed of bicycles, cars, taxis, PTW, and public transport. The green line shows all the people riding in or driving a vehicle (taxi drivers and public transport drivers are not included). All the people waiting for a ride are the bottom line in blue. Finally, all the people that are walking are shown in purple. As a reference, the black vertical line is at 9 AM, the start of the event. We increased or reduced the number of parking spaces depending on the experiments' requirements. Given that the cars and PTW may need to park, delays due to cruising for parking and then walking to the destination are introduced.

4.4 MARL Environment and Agent Initialization

The MARL environment is initialized for each training run, and the agents are re-initialized for each episode. Every agent has the same destination (the venue's location) and expected arrival time (the event's start time). Their origins are randomized and uniformly distributed over the map. The possible actions (including the transport modes) are the same for all the agents; however, the usable modes may differ based on their origin, and the agents need to discover which of the possible modes are viable by exploring the environment.

Given that the agents need to learn how long they should wait before departure, we evaluated three different Pareto start-time distributions. The three distributions are tuned to vary the *minimum possible competition* (hence, coordination) required to achieve the goal.

- $30min_{\alpha=3}$: every agent has a minimum 30-minute time window before the expected arrival time. With this distribution, it may not be possible to have everyone arrive on time, and the competition for transportation resources is at its highest.
- $45min_{\alpha=4}$: everyone has a minimum time window of 45-minutes to reach the event.
- $60min_{\alpha=5}$: the minimum time window is 60 minutes. With a longer time before the event, the exploration takes longer, but there is the least possible competition, and the agents have a higher chance to spread their departures over time.

5 Evaluation and Results

This section shows how agents can learn coordinated multi-modal trip plans by exploring the transportation environment. The metrics we used to compare the different experiments are *travel time*, *waiting time at destination*, *lateness*, and the *the distribution of transportation mode usage*. In the following sections we present a representative subset of the experiments we ran during the study. The complete set of graphs is available on GitHub[12]. Due to the use of SUMO, the environment reflects the differing characteristics typically observed for different transportation modes (e.g., relative speed/immunity to congestion). Intuitively, the plans discovered by the agents should respect these differences facilitating the interpretation of the results.

DRL is notoriously unstable during training [38]. Once learning is finished and the reward has converged to a stable outcome, we select the *best* policy from the last 25 training runs to collect the metrics used for comparison. Each selected policy is then evaluated by averaging its outcome, defined as the sum of the rewards collected by each agent in a single episode, over ten episodes. The *best* policy is the one with the highest average total reward.

5.1 Preliminary Experiments

We conducted extensive initial experiments to gather the insights used to direct the study and to produce the baselines required for comparison purposes.

Naive solution We implemented an algorithm for naive trip planning based on people's common behavior [39]. Depending on the origin and destination, the estimated travel time for each of the (preferred) modes is computed using SUMO. The estimation is used to decide the departure time, taking into account a possible delay due to unforeseen congestion. The naive solution baseline is computed averaging over 100 simulations with a population of 1000 people, each with a random origin and the same destination, the venue. For each simulation, the preferred transportation mode distribution is selected randomly from a pool of six. One distribution presents balanced use of all the available modes, and the other distributions are skewed towards a preference for one specific mode. The actual weight associated with the preference is initialized each time, drawing from a Gaussian distribution. Additionally, each person has a personal buffer time estimation used to allow for unforeseen delays. This buffer time is randomly initialized and uniformly distributed between 5 and 15 minutes. For each simulation, we collect the same metrics we use in training and evaluation: travel time, waiting time, and lateness. These metrics are then averaged and used in all the relevant figures as the naive baseline for comparison.

DRL Baselines We initially evaluated the three reward models (Simple, OTT, and OTTCoord), with all of the actions (wait, walk, bicycle, public transport, car, and PTW) being available to every agent. Each of the above-mentioned start-time distributions ($30min_{\alpha=3}$, $45min_{\alpha=4}$, and $60min_{\alpha=5}$) are compared, and the resulting transportation mode distributions are shown in Figure 3. The Simple reward

[12]Results: https://github.com/lcodeca/results/Persuasive-training Last access: February, 2022

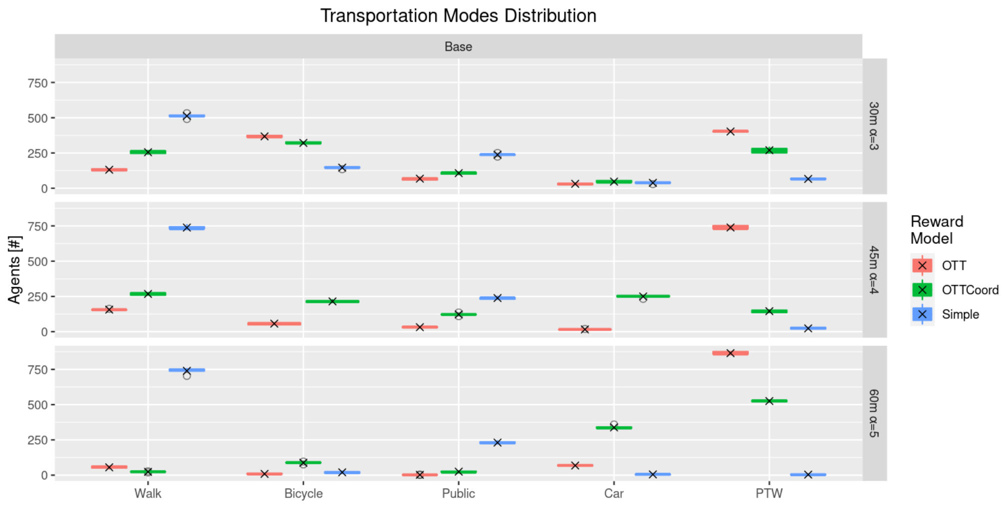


Figure 3: Mode Distribution for the base experiments.

model, in blue, does not take into account observed travel times, and the outcome is that it rewards the reliability provided by walking and using public transport (in a world practically devoid of other means of transportation generating congestion and delays). Once we introduce the observed travel time penalty (OTT, in red, and OTTCoord, in green), we can see that walking and public transport are not the most suitable choice anymore, and smaller and faster vehicles such as bicycles and PTW become more appealing. PTW are as fast as cars, and can filter through traffic congestion. Bicycles, although slower, have similar filtering capabilities in traffic, and being smaller, they seep through other vehicles more efficiently than PTW. Additionally, the start time distribution (thus, the competition among the agents) has a major impact on the mode choice. With $30min_{\alpha=3}$, where the competition is higher and the possibility of distributing departure over time is very limited, bicycles and PTW are the clear winners. In $60min_{\alpha=5}$, where the departure time can easily be spread, the clear winners are the faster modes: cars and PTW. Nonetheless, the presence of explicit coordination seems to favour the car and balance between the modes. The balanced mode usage is more explicit in $45min_{\alpha=4}$, where the mode choice is spread with explicit coordination (OTTCoord, in green), but the mode choice is completely skewed in favour of PTW where the only coordination is implicit and due to the limited resources (OTT, in red).

Background traffic For completeness, we run some experiments with the background mobility generated using SAGA and tuned to present a peak in demand around 8:30 AM, with the event starting at 9 AM. However, due to the complexity of the scenarios, we ran all other variations without background traffic in order to isolate the specific behaviors that we wanted to study. A selection of those run with background traffic is shown in Figure 4. Here we compare the OTTCoord model implementation used in the DRL Baseline, with (in blue) and without (in red) background traffic, to investigate if the results are significantly different. For these scenarios the start time distribution changed over the course of learning, starting with a 30-minute window and moving to the 60-minute one. The left plot in Figure 4 shows that the lateness and the travel time distributions with (in blue) and without (in red) traffic are comparable. The main difference is in the waiting time before the event. Due to the rush-hour traffic, the agents needed to depart in advance to arrive on time, resulting in increased waiting time before the event. The right plot in Figure 4 shows the transportation mode distribution. Although the trends are comparable, the few differences are intuitive. The public transportation capacity is more limited due to

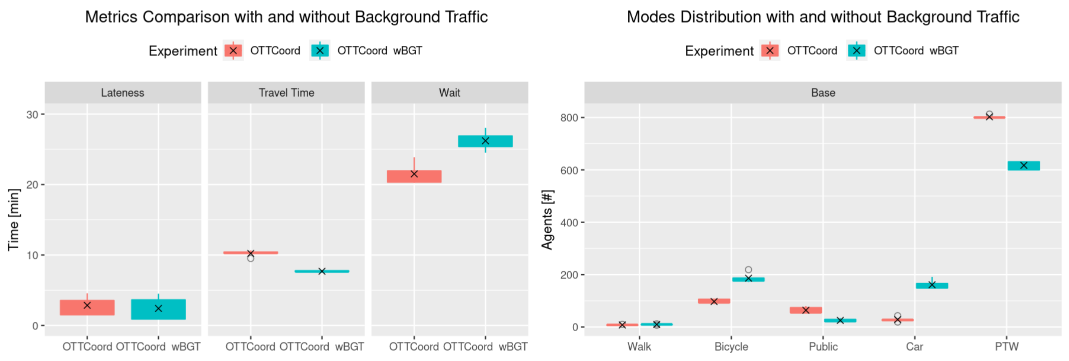


Figure 4: Impact of background traffic on the DRL Baseline experiment based on OTTCoord reward model.

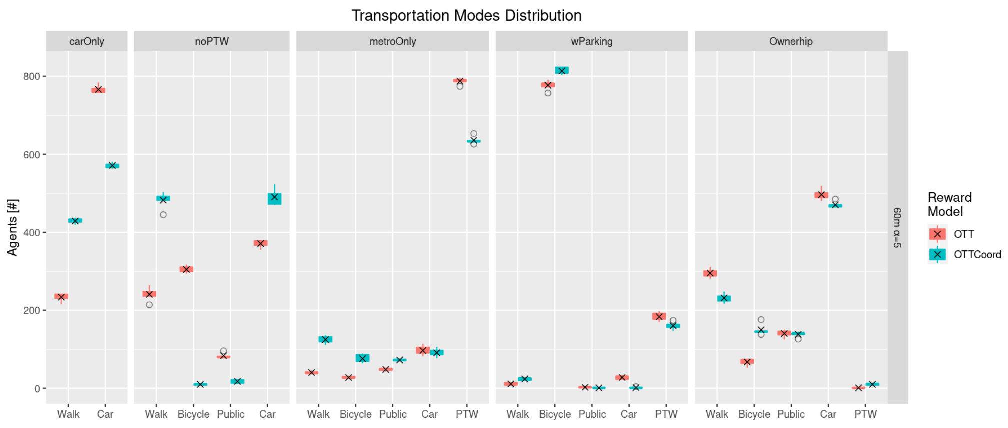


Figure 5: Mode Distribution for the starting distribution of 60m $\alpha = 5$.

the background traffic, and the additional vehicles on the road are slowing the buses. PTW filtering is impacted by the additional congestion, making it less appealing and spreading the load to other modes such as cars and bicycles.

5.2 Ablation Study

To study the capabilities and limitations of the DRL approach, we performed an ablation study where starting from the insights provided by the DRL baselines, we systematically removed or changed modes of transport (actions) to assess whether the resulting policies matched our intuitions. Additionally, we evaluated the impact of uncertainty over the learned policy by forcing cars and PTW to find a parking spot, and we added hard constraints based on vehicle ownership. The impact of all the above-mentioned experiments on the transportation mode distributions is shown in Figure 5.

The following three experiments are focused on removing specific modes or changing the meaning of an action to verify that the resulting policy reflects the intended change. We evaluated all combinations of the reward models and the start time distributions. In the following paragraphs, we focus on the OTT and OTTCoord models, without background traffic, and the $60min_{\alpha=5}$ start time distribution.

carOnly The first experiment removed access to all vehicles except for cars, limiting the available actions to wait, walk and car. The *carOnly* column in Figure 5 shows the comparison between the transportation mode distributions with implicit (in red) and explicit (in blue) coordination. The expectation is to see a preference for the car (low travel time) with the departure time spread over time. This actually happens with the OTT reward model, but the presence of explicit coordination in OTTCoord balanced out the mode usage. This effect is due to the penalty paid by everyone if *other* agents are late, combined with the reliability of walking.

noPTW In the second experiment, we removed the PTW from the transportation modes by removing the associated action. In this case, the expected outcome is an increase in the usage of the car (for their speed) and bicycles (due to filtering capabilities). The *noPTW* column in Figure 5 shows the results. With the OTT reward model (in red), the expectation is met, with more agents choosing to walk (similarly to carOnly), a balanced spread between car and bicycles, and few public transportation options. However, the explicit coordination implemented in the OTTCoord reward model (in blue) again emphasised the travel time and reliability requirements as seen in the previous experiment, reducing bicycles and public transportation to a negligible use.

metroOnly In the third experiment, we changed the public transportation composition, removing the buses and leaving only the metro. Given that the buses are sharing the road with the rest of the vehicles, removing them from the public transportation infrastructure leaves only the metro, improving its reliability. In this experiment, all the transportation modes are available to the agents. The expectation is that the additional reliability introduced to public transportation would make it a more favourable option. The resulting transportation mode distribution is shown in the *metroOnly* column of Figure 5. In this case, the expectations are not met. Leaving the option of using the PTW, in conjunction with the increased road capacity, skewed the mode selection in favour of the PTW once more. Although in the presence of explicit coordination (in blue), the rest of the modes are comparatively more used than in OTT (in red), public transportation is still not a favourable option. As future work, we intend to investigate additional changes aimed at increasing public transports reliability and usage.

5.3 Increasing Uncertainty

In reality, private vehicles need to be parked, introducing delays and decreasing the effectiveness and reliability of the mode. For this experiment, we changed the meaning of the action associated with the selection of cars and PTW. Once the mode is chosen, the target destination changes from the event location to the parking area closest to the final destination, and the 'last mile' is walked. If the closest parking area is already full, the agent will cruise for parking (based on the SUMO parking area reroute mechanism[13]). Building upon the knowledge obtained from the previous experiment, we expected to see a decrease in the use of cars and PTW in favor of other more reliable modes. More precisely, we expect to see a shift from the car and PTW to bicycles due to their capability of filtering through congested vehicles and their reliability. The results of this experiment are shown in the *wParking* column of Figure 5. In this case, the expectation is met for both OTT (in red), and OTTCoord (in blue), and the impact of implicit or explicit coordination is negligible.

5.4 Constraints

In reality, not all people have access to every transportation mode. For this reason, we introduced hard constraints regarding the ownership of a vehicle modelled as the probability of each agent owning one.

[13]Wiki SUMO: Parking Area Rerouters definition https://sumo.dlr.de/docs/Simulation/Rerouter.html#rerouting_to_an_alternative_parking_area Last access: February, 2022

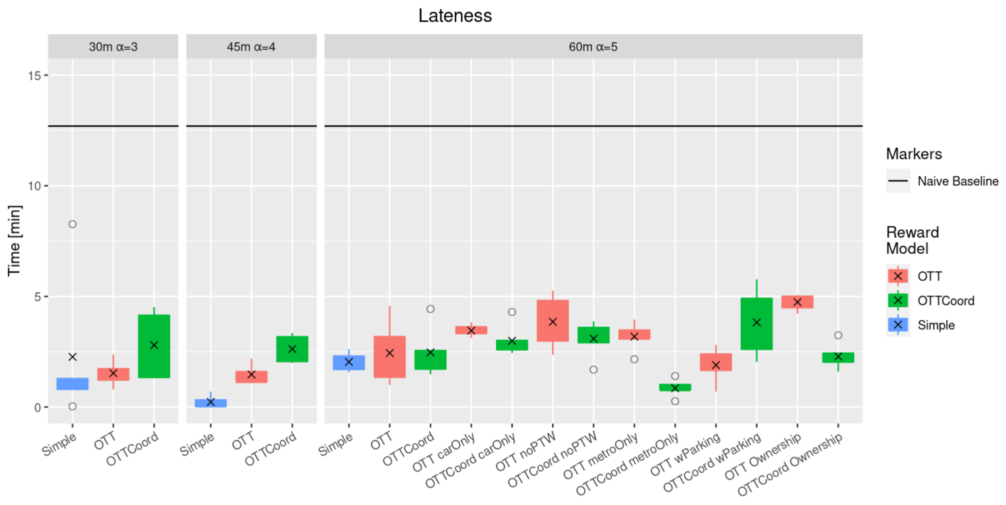


Figure 6: Lateness: late arrival distribution.

Following the National Household Travel Survey 2017 (reported on December 2018)[14] conducted by the Irish National Transport Authority (NTA), we configured the agent with the following probabilities: 89% of agents own a car, 35% own a bicycle, and only 10% own a PTW. In this experiment, all the actions are available to the agents, but they need to learn if they can use (own) a vehicle or not. With this experiment, we wanted to make sure that it was possible for the agents to learn a policy based on variable hard constraints. The expectation is that with the enforced limit over the PTW and the congestion generated by the presence of too many cars, the agents are able to learn how to distribute the load over the other modes. The comparison between the resulting mode distributions is shown in the *Ownership* column of Figure 5. In this experiment, the expectation is met with both reward models, and the hard constraints are respected. The cars are capped not by the defined ownership but by capacity constraints, and the other modes such as walking, bicycles and public transportation are used to spread the load.

5.5 Metrics

To measure the goodness of the stochastic policy the agents learnt over time, we evaluated the distribution of late arrivals, waiting time at destination, and observed travel time. Although departure time is not meaningful on its own due to its dependency on the resulting transportation mode distribution, it is useful to understand the other metrics.

Given the capacity constraints imposed by the problem at hand, finding the optimal set of trip plans able to maximize just-in-time arrival while minimizing late arrival may not be possible. Based on the reward models we implemented, the penalty for late arrival is higher than for waiting at the destination, increasing the chance that in case of unstable traffic congestion closer to the event's start time, the preferred option may incur higher waiting times.

Lateness The first metric we discuss is lateness, measuring the late arrival distribution of the agents. The late arrival is computed as the elapsed time between the event's start time and the agent's arrival

[14]National Household Travel Survey `https://www.nationaltransport.ie/wp-content/uploads/2019/01/National_Household_Travel_Survey_2017_Report_-_December_2018.pdf` Last access: February, 2022

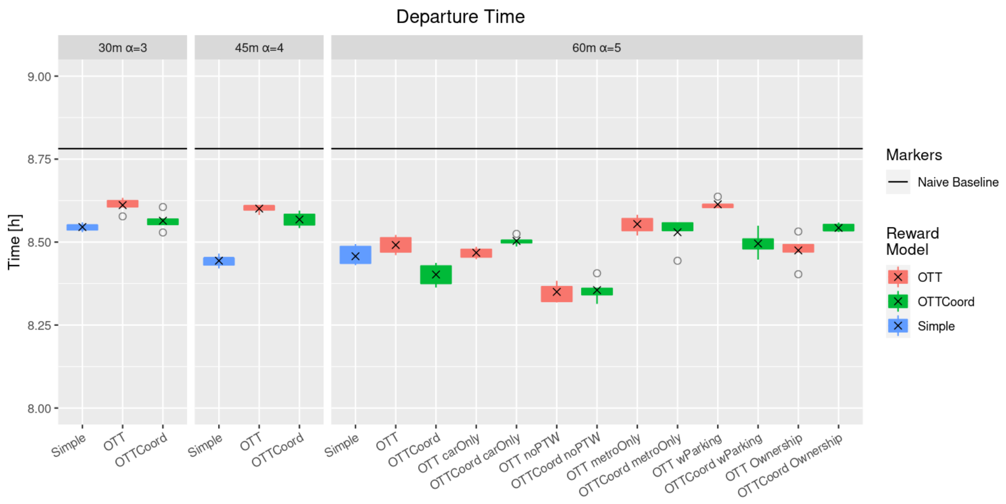


Figure 7: Departure time distribution.

at the destination. The box plots in Figure 6 show the late arrival distribution for all the experiments previously discussed. In the graph, the x in black shows the mean, and the empty black dots represent outliers. The black line is the naive solution's baseline (around 12 minutes), and it shows that every stochastic policy we learnt improves on it significantly. Nonetheless, there are no visible trends emerging from the difference between the OTT and OTTCoord reward models.

Departure The departure time metric on its own is not representative of the goodness of the policy. Nonetheless, it allows us to make some observations in conjunction with other metrics. Figure 7 depicts the departure time distributions for all the experiments discussed. The black line marks the naive solution baseline (around 8:45 AM). The figure shows that the agents learnt a more conservative strategy, expecting last-minute delays, hence leaving earlier. This is an additional reason why the lateness metrics are lower than in the naive baseline.

Waiting The higher penalty for the late arrival has a direct impact on the distribution of the waiting time at the destination. The waiting time is measured from the time of arrival at the venue until the event starts. Figure 8 shows the box plots associated with the waiting times for all the experiments. The naive baseline is marked by the black dashed line (about 7 minutes), and it shows that all our solutions require more waiting at the destination. Nonetheless, the solid black line represents the 15-minutes mark that we use to establish if the agents achieved their goal. Here we can see that all the DRL Baselines experiments performed for the initial start time distributions of $30m_{\alpha=3}$ and $45m_{\alpha=4}$ achieved the goal; for $60m_{\alpha=5}$, experiments such as Simple, OTTCoord carOnly, OTT wParking, and OTTCoord wOwnership achieved it too. Although the presence of coordination does not provide a clear trend in the results, it interesting to notice how its presence drastically decreases the waiting times in the carOnly experiment, while the departure time distributions are not significantly different. The same can be said for the noPTW and wOwnership experiments, and to a lesser extent, to metroOnly too.

Travel Time The final metric we consider is the observed travel time, measured between the departure time from the origin and the arrival time at the destination. Figure 9 shows the travel time distributions for each experiment. The solid black line marks the naive solution baseline (around 18 minutes). Apart from the Simple reward model (that does not take into consideration the travel time), in all the other experiments, we see an improvement. Similarly to the departure time, the actual travel

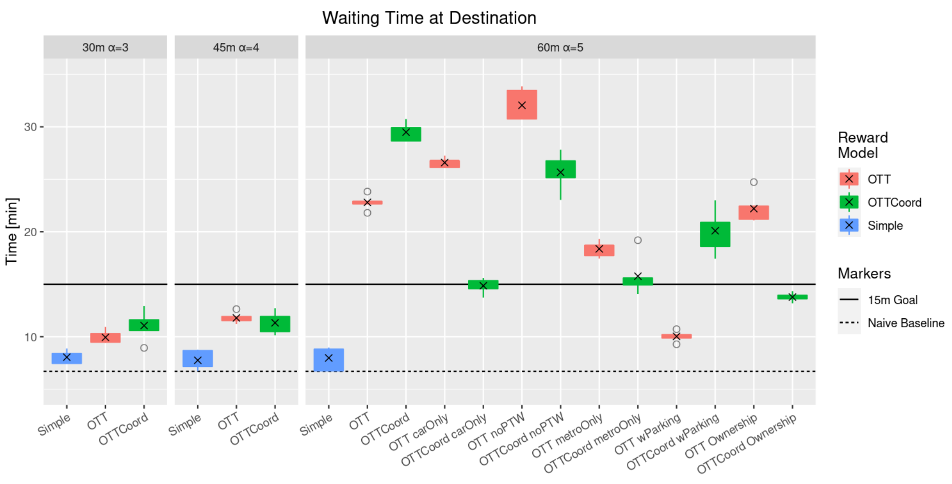


Figure 8: Waiting time at destination distribution.

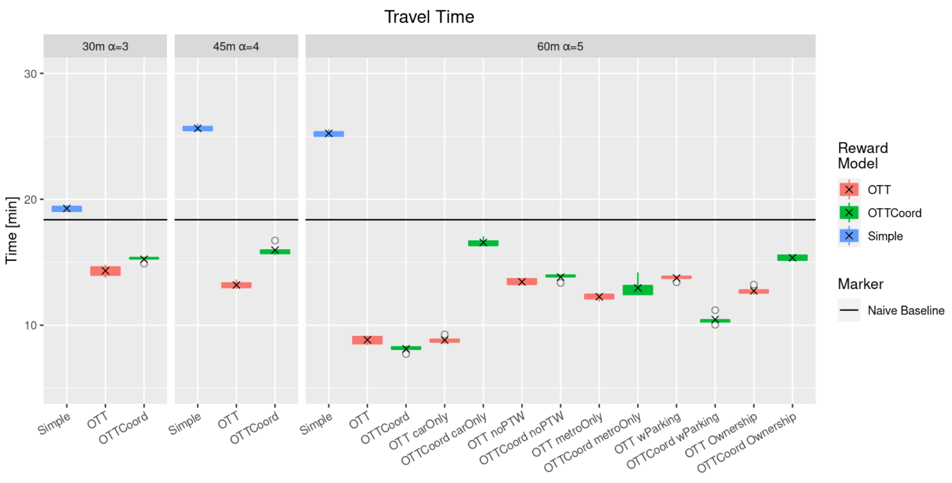


Figure 9: Travel Time distribution.

time is not a significant metric on its own due to its strong correlation with both the transportation mode distribution and the congestion levels. Like the previously discussed metrics, it is not possible to find a significant trend concerning the impact of coordination, leaving it on an experiment by experiment basis.

In summary, with the extensive evaluation presented in this section, we show that it is possible to use DRL to tackle the problem of coordinated multi-modal trip planning with limited transportation capacity. Through the exploration of a simulated transportation environment, it is possible to discover its features and learn viable plans. In all the variations discussed, the stochastic policy learnt is able to reduce late arrivals and travel times while keeping reasonable waiting times at destination for most of them. Finally, it is possible to learn a policy that takes into account hard constraints while providing reasonable trip plans.

6 Conclusions and Future Work

In this paper, we presented the implementation and evaluation of a MADRL framework to study the problem of coordinated multi-modal trip planning with constrained transportation capacity using a microscopic mobility simulator. The proposed solution leverages the distributed orchestration provided by RAY, the state-of-the-art ML algorithms implemented by RLlib, and RLlib SUMO Utils, a general-purpose connector to SUMO that enables the implementation of OpenAI Gym-like learning environments. All the software and infrastructure previously discussed is open-source and available on GitHub.

The use case presented is a large-scale social event, where the event-goers are assumed to have at their disposal an application provided by the venue that generates coordinated trip plans for just-in-time arrival at the event based on user requirements (if any). The DRL algorithm that we configured is the state-of-the-art implementation of PPO provided by Reinforcement Learning library (RLlib). Enabled by the general-purposes APIs definition, the learning environment we implemented for the case study is independent of the learning algorithm.

In this paper, we evaluated multiple learning environments with a combination of reward models (tied to travel time and coordination), availability of transportation modes (hard constraints such as ownership of a vehicle), and environmental factors (such as parking requirements).

The results showed that it is possible to use Deep Reinforcement Learning (DRL) and microscopic mobility simulations to study the problem of coordinated multi-modal trip planning with constrained transportation capacity. The stochastic policy learnt is able to reduce the late arrival and minimize the travel time. Additionally, the trip plans discovered by trial and error through the exploration of an unknown transportation environment are intuitive and easy to motivate, increasing the trust in DRL with microscopic mobility simulations as a viable option.

As future work, we intend to study the impact of soft constraints (i.e., user preferences) on coordinated trip plans. With the need to take into account user preferences concerning the transportation mode to use, the solution space increases drastically, and based on the distribution of the preferences, a solution may not exist. Finally, we intend to scale up the training from the 1000 agents in a controlled simulated environment to real numbers of event-goers in a realistic mobility scenario. Initial studies have been started using the Monaco SUMO Traffic (MoST) Scenario [14] and the Monaco stadium as the target venue.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 713567. ENABLE is funded under Science Foundation Ireland (16/SP/3804) and is co-funded under the European Regional Development Fund.

References

[1] Zhongyu Wang, Hang Yang, and Bing Wu. Traffic flow characteristics and congestion evolution rules for urban road networks during special events. In CICTP 2017: Transportation Reform and Change—Equity, Inclusiveness, Sharing, and Innovation, pages 2368–2380. American Society of Civil Engineers Reston, VA, 2018.

[2] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018.

[3] Hao Dong, Hao Dong, Zihan Ding, Shanghang Zhang, and Chang. Deep Reinforcement Learning. Springer, 2020.

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

[5] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. In International Conference on Machine Learning, pages 3053–3062. PMLR, 2018.

[6] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. PloS one, 12(4):e0172395, 2017.

[7] Alejandro Torreno, Eva Onaindia, Antonín Komenda, and Michal Štolba. Cooperative multi-agent planning: a survey. ACM Computing Surveys (CSUR), 50(6):1–32, 2017.

[8] Pablo Alvarez Lopez and Michael Behrisch and Laura Bieker-Walz and Jakob Erdmann and Yun-Pang Flötteröd and Robert Hilbrich and Leonhard Lücken and Johannes Rummel and Peter Wagner and Evamarie Wießner. Microscopic Traffic Simulation using SUMO. In The 21st IEEE International Conference on Intelligent Transportation Systems. IEEE, 2018.

[9] Hoang Nguyen, Le-Minh Kieu, Tao Wen, and Chen Cai. Deep learning methods in transportation domain: a review. IET Intelligent Transport Systems, 12(9):998–1004, 2018.

[10] Zahra Karami and Rasha Kashef. Smart transportation planning: Data, models, and algorithms. Transportation Engineering, 2:100013, 2020.

[11] Mordechai Haklay and Patrick Weber. OpenStreetMap: User-generated street maps. Pervasive Computing, IEEE, 2008.

[12] Sandesh Uppoor and Marco Fiore. Large-scale urban vehicular mobility for networking research. In Vehicular Networking Conference (VNC), 2011 IEEE, pages 62–69. IEEE, 2011.

[13] Luca Bedogni, Marco Gramaglia, Andrea Vesco, Marco Fiore, Jerome Harri, and Francesco Ferrero. The Bologna Ringway dataset: improving road network conversion in SUMO and validating urban mobility via navigation services. Vehicular Technology, IEEE Transactions on, 64(12):5464–5476, 2015.

[14] Lara Codeca and Jérôme Härri. Monaco SUMO Traffic (MoST) Scenario: A 3D Mobility Scenario for Cooperative ITS. In SUMO 2018, SUMO User Conference, Simulating Autonomous and Intermodal Transport Systems, Berlin, GERMANY, 05 2018.

[15] Marco Rapelli, Claudio Casetti, and Giandomenico Gagliardi. Tust: from raw data to vehicular traffic simulation in turin. In 2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications (DS-RT), pages 1–8. IEEE, 2019.

[16] Maxime Gueriau and Ivana Dusparic. Quantifying the impact of connected and autonomous vehicles on traffic efficiency and safety in mixed traffic. In 23rd IEEE International Conference on Intelligent Transportation Systems, 2020.

[17] Abubakr O. Al-Abbasi, Arnob Ghosh, and Vaneet Aggarwal. Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning. IEEE Transactions on Intelligent Transportation Systems, 20(12):4714–4727, 2019.

[18] Kaushik Manchella, Abhishek K. Umrawal, and Vaneet Aggarwal. Flexpool: A distributed model-free deep reinforcement learning algorithm for joint passengers and goods transportation. IEEE Transactions on Intelligent Transportation Systems, 22(4):2035–2047, 2021.

[19] Kaixiang Lin, Renyu Zhao, Zhe Xu, and Jiayu Zhou. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 1774–1783, 2018.

[20] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems, pages 1–18, 2021.

[21] Ammar Haydari and Yasin Yilmaz. Deep reinforcement learning for intelligent transportation systems: A survey. IEEE Transactions on Intelligent Transportation Systems, pages 1–22, 2020.

[22] Jing Xin, Huan Zhao, Ding Liu, and Minqi Li. Application of deep reinforcement learning in mobile robot path planning. In 2017 Chinese Automation Congress (CAC), pages 7112–7116, 2017.

[23] Ursula Challita, Walid Saad, and Christian Bettstetter. Deep reinforcement learning for interference-aware

path planning of cellular-connected uavs. In 2018 IEEE International Conference on Communications (ICC), pages 1–7, 2018.

[24] Siyu Guo, Xiuguo Zhang, Yisong Zheng, and Yiquan Du. An autonomous path planning model for unmanned ships based on deep reinforcement learning. Sensors, 20(2):426, 2020.

[25] Hao Liu, Ting Li, Renjun Hu, Yanjie Fu, Jingjing Gu, and Hui Xiong. Joint representation learning for multi-modal transportation recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 1036–1043, 2019.

[26] Mariagrazia Dotoli, Hayfa Zgaya, Carmine Russo, and Slim Hammadi. A multi-agent advanced traveler information system for optimal trip planning in a co-modal framework. IEEE Transactions on Intelligent Transportation Systems, 18(9):2397–2412, 2017.

[27] Yang Liu, Cheng Lyu, Zhiyuan Liu, and Jinde Cao. Exploring a large-scale multi-modal transportation recommendation system. Transportation Research Part C: Emerging Technologies, 126:103070, 2021.

[28] Michael Redmond, Ann Melissa Campbell, and Jan Fabian Ehmke. Data-driven planning of reliable itineraries in multi-modal transit networks. Public Transport, 12(1):171–205, 2020.

[29] Zhiguang Cao, Hongliang Guo, Wen Song, Kaizhou Gao, Zhenghua Chen, Le Zhang, and Xuexi Zhang. Using reinforcement learning to minimize the probability of delay occurrence in transportation. IEEE Transactions on Vehicular Technology, 69(3):2424–2436, 2020.

[30] Thomas Liebig, Sebastian Peter, Maciej Grzenda, and Konstanty Junosza-Szaniawski. Dynamic transfer patterns for fast multi-modal route planning. In The Annual International Conference on Geographic Information Science, pages 223–236. Springer, 2017.

[31] Ayat Abedalla, Ali Fadel, Ibraheem Tuffaha, Hani Al-Omari, Mohammad Omari, Malak Abdullah, and Mahmoud Al-Ayyoub. Mtrecs-dlt: Multi-modal transport recommender system using deep learning and tree models. In 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS), pages 274–278, 2019.

[32] Daniel Herzog, Hesham Massoud, and Wolfgang Wörndl. Routeme: A mobile recommender system for personalized, multi-modal route planning. In Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, pages 67–75, 2017.

[33] Filip Dvorak, Shiwali Mohan, Victoria Bellotti, and Matthew Klenk. Collaborative optimization and planning for transportation energy reduction. In ICAPS Proceedings of the 6th Workshop on Distributed and Multi-Agent Planning (DMAP, 2018.

[34] Tanzina Afrin and Nita Yodo. A survey of road traffic congestion measures towards a sustainable and resilient transportation system. Sustainability, 12(11):4660, 2020.

[35] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. Ray: A distributed framework for emerging AI applications. In 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), pages 561–577, 2018.

[36] Erdmann, Jakob. Lane-changing model in SUMO. Proceedings of the SUMO User Conference 2014: Modeling mobility with open data, 2014.

[37] Lara CODECA, Jakob ERDMANN, Vinny CAHILL, and Jérôme HARRI. Saga: An activity-based multi-modal mobility scenario generator for sumo. SUMO User Conference 2020 - From Traffic Flow to Mobility Modeling, 2020.

[38] Evgenii Nikishin, Pavel Izmailov, Ben Athiwaratkun, Dmitrii Podoprikhin, Timur Garipov, Pavel Shvechikov, Dmitry Vetrov, and Andrew Gordon Wilson. Improving stability in deep reinforcement learning with weight averaging. In Uncertainty in artificial intelligence workshop on uncertainty in Deep learning, 2018.

[39] Anne Durand, Lucas Harms, Sascha Hoogendoorn-Lanser, and Toon Zijlstra. Mobility-as-a-Service and changes in travel preferences and travel behaviour: a literature review. KiM— Netherlands Institute for Transport Policy Analysis, 2018.

SUMO User Conference 2021
https://doi.org/10.52825/scp.v2i.90

Published: 29 Jun. 2022

The Development of Pedestrian Gap Acceptance and Midblock Pedestrian Road Crossing Behavior Utilizing SUMO

Peter J. Lawrence [1*], Veronica Pellacini[1†], Darren Blackshields[1‡] and Lazaros Filippidis[1§]
[1] University Greenwich, London, UK.
P.J.Lawrence@greenwich.ac.uk, veronicapellacini@yahoo.it,
D.Blackshields@greenwich.ac.uk, L.Filippidis@greenwich.ac.uk

Abstract

While there are several published studies for modelling pedestrian behavior at signalized crossings in SUMO, the behavior of pedestrians crossing a road at a location other than a designated crossing, has not been considered to date. This work looks at how to represent pedestrian agents selecting to cross a road at arbitrary locations along the length of the road. The pedestrian agents utilize a gap acceptance model that represents how a pedestrian decides when to cross a road, based on the frequency and speed of approaching vehicles, while considering the spacing between them. Furthermore, the gap acceptance model allows the pedestrians to choose to cross all lanes in one go, when safe to do so, known as *Double Gap* or one stage crossing. Alternatively, if an agent is identified as a risk-taker, they may choose to cross lane by lane, sometimes waiting in the middle of the road, known as *Rolling Gap* or risk-taker crossing behavior. The inclusion of these two crossing behaviors allows for situations where urgency plays an important role in behavioral decision making, such as in emergencies, rush hour or in crowd management events. The outlined pedestrian crossing model is attained by integrating the pedestrian model EXODUS with SUMO, via the TraCI API.

1 Introduction

The interaction between pedestrians and vehicles in urban environments is a common occurrence, as such numerous studies exist on pedestrian safety issues in urban transportation, mainly focusing on pedestrian behavior at designated pedestrian crossings (Brosseau et al., 2017, Xin et al., 2014). However, a person may not be willing to walk longer distances to a designated crossing location if crossing the road from their current location will give them convenient and direct access to their destination (Islam et al, 2014, Zhao et al., 2017, Akyol et al., 2019).

Computer models such as LEGION (LEGION, 2017), VISSIM (Han et al., 2005) and MATSIM (Lämmel et al., 2009) can represent pedestrian interaction with vehicles. However, they consider pedestrians crossing only at designated locations, such as unsignalized or signalized pedestrian crossings. These models do not consider the possibility of people crossing at non-designated locations, as suggested by Wang (Wang, 2012). SUMO has supported an intermodal pedestrian-based simulation since 2010. This was extended by Erdmann and Krajzewicz (Erdmann and Krajzewicz, 2015) to simulate pedestrians in more detail, however, this still only allowed pedestrians to cross at designated

[*] Lead author, researcher and software developer.
[†] Research into crossing behavior, specification of pedestrian crossing model and contributing author.
[‡] Coupling of SUMO with pedestrian model EXODUS and contributing author.
[§] Research on human behavior, analysis of research and contributing author.

locations. Other researchers have looked at modelling pedestrians at traffic lights (Akyol et al., 2019) and the coupling of SUMO to external pedestrian models (Flötteröd et al., 2019) but neither these approaches allow for pedestrians to cross a road on non-designated locations.

Here a method of coupling a pedestrian simulation tool, namely EXODUS (Galea et al. 1996), with SUMO via the TraCI (Wegener et al. 2008) interface is outlined. The proposed model can represent midblock crossing behavior of pedestrians at arbitrary locations along the street. The work builds on research carried out by Wang (Wang, 2012) by implementing both one-stage and two-stage pedestrian crossing behaviors, to cater for two different pedestrian crossing traits.

One-stage crossing behavior is used when a pedestrian may choose to cross all vehicle lanes in one go, when safe to do so. This behavior is selected by non-risk-taking pedestrians. The two-stage crossing, or double gap behavior, is chosen when a pedestrian chooses to cross lane by lane, sometimes waiting in the middle of the road, if necessary. This behavior is also known as rolling gap (Brewer et al., 2006) or risk-taker crossing behavior (Song et al., 2003). These crossing behaviors allow for urgency, a psychological factor, to be represented, where a pedestrian in a hurry may be more willing to take risks while crossing a road, for example, during rush hour, or maybe in an attempt to catch a bus which is about to depart. Also, urgency is a key factor when considering emergency situations where people might be evacuating from a large building, such as a transportation hub, shopping center or tower block and inevitably interacting with the surrounding traffic.

This paper tries to address the research question –

Is it possible to represent pedestrian agents crossing vehicle lanes in SUMO at arbitrary midblock locations, taking into consideration traffic levels and different crossing behaviors?

This paper will outline the key behavioral developments required to model pedestrian crossing behavior, together with the technique utilized to interface EXODUS (Galea et al., 1996 and 2017) to SUMO (Lopez et al., 2018). It will then demonstrate the pedestrian crossing behavior and vehicle interaction, by modelling a crowd of people exiting from a large building to reach a point of assembly, positioned across a busy road. This example is chosen to examine how different types of crossing behaviors can impact the time for people to reach a destination and their impact on local traffic. The demonstration scenario could be considered as either representing a precautionary emergency evacuation or simply people exiting from the building heading towards a major event.

2 Methodology and Road Network

SUMO is a well-established open source, highly portable, microscopic and continuous road traffic simulation package designed to handle large road networks (Lopez et al., 2018). SUMO also includes its own pedestrian movement model, however this model is based around lanes and areas (Erdmann and Krajzewic, 2015) and it cannot represent people in buildings or exiting from them. Also, the lane-based nature of the model restricts pedestrians to just two directions of travel which limits its application in representing urban scale pedestrian movement. A possible solution to overcome these limitations and to include more complex and comprehensive pedestrian behaviors is to link SUMO via the TraCI (Wegener et al, 2008) interface to an external pedestrian model. One such model is EXODUS (Galea et al., 2017), which has been continuously developed by the University of Greenwich, since 1994.

EXODUS is an agent-based model capable of representing pedestrian movement for a variety of modelling domains such as building, maritime, aviation and rail (FSEG, 2021). EXODUS utilizes a hybrid spatial approach. This approach allows pedestrian movement to be represented at a macroscopic level, utilising coarse regions (i.e., a flow-based model), and at the microscopic level utilizing either a fine node grid, where pedestrian agents move on a discrete grid of nodes, or a continuous spatial representation (Chooramun et al., 2017). These different spatial representations are utilized depending on what data is available regarding pedestrian behavior, the type of expected interactions or travel speeds over a given terrain, or whether optimization is required due to the scale of the model. For example, at the urban scale, EXODUS would typically utilize coarse regions and/or a fine node grid to cater for large-scale models.

Within EXODUS the simulated pedestrians are characterized by various agent attributes grouped into four main categories (Galea et al., 2017). The first category comprises all physical attributes such as age, gender, mobility and unimpeded walking speed. Further, EXODUS simulates psychological attributes such as response time and patience. Their individual sojourn time in the environment, the total travelled distance, or the cumulative time that the agent has spent waiting in a queue or congestion are categorized as the agent's experiential parameters. Finally, pedestrian agents' hazard parameters measure the physical impact of fire hazards and the accompanying toxic gases on the condition of each individual.

To ensure that EXODUS and SUMO utilize the same road information, EXODUS has been adapted to import the SUMO road network file directly. In this way the naming and locations of all road edges, lanes and junctions are consistent. If desired by the user, EXODUS can automatically add sidewalks of a user definable width around the imported SUMO road network., see Figure 1.

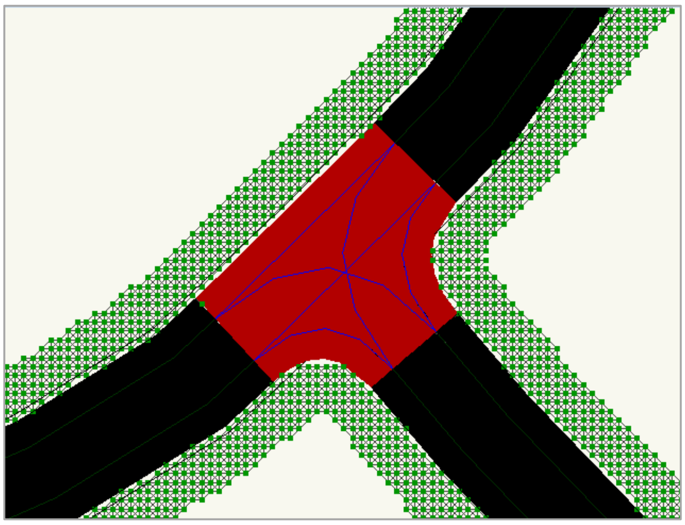


Figure 1: The EXODUS *Fine Node Mesh* representing pedestrian space around a road and junction network. Green squares represent EXODUS nodes. The black lines between nodes represent connected arcs, the large black areas represent roads, the red section is a junction and blue lines indicate traffic lanes**.**

The sidewalks are modelled in EXODUS utilizing a fine nodal mesh (Galea et al., 1996) consisting of an interconnected grid of nodes, with a typical spacing of 0.5 meters. Adjacent nodes, which can be travelled between, are connected by arcs representing the travel distance, see Figure 1. Only one pedestrian agent can occupy a given node at any given time in the simulation.

While pedestrian agents move using the EXODUS fine node model on the sidewalks, when crossing the road, they move using a continuous spatial movement algorithm (Chooranum et al., 2017). A continuous spatial pedestrian movement algorithm was selected for the road area as it consists of a well-defined region, providing greater flexibility for crossing at arbitrary locations and angles and allows for more accurate interaction with the vehicles. As the pedestrian agents need to be aware of approaching vehicles and other pedestrians on the road, a traffic lane-based collision avoidance algorithm, tied to the road network, is also utilised. In contrast, the fine node model is optimised for checking for pedestrian collisions (Galea et al., 1996) only within the locality of adjacent nodes, which is around 0.5 to 0.7 metres and thus its use on sidewalks. Furthermore, a typical vehicle can travel the distance of several fine nodes during a single EXODUS time step when travelling at a speed higher than 6m/s (21.6km/h), which prevents the utilisation of the node mesh for the road network without an increase to the EXODUS clock rate. This would in turn require recalibration to the fine node conflict resolution algorithm (Galea et al., 1996). However, the continuous spatial approach is not limited by these factors and was thus chosen.

3 Pedestrian Crossing Behavior

When a pedestrian agent encounters a road on their way to their destination and when their next move could take them onto the road surface, a probabilistic gap acceptance model is used to determine if and when the agent will attempt to cross the road. The gap acceptance model utilizes data from a number of previous studies (Wang, 2012, Schroeder, 2008, Yannis et al., 2010), see Section 3.1. The gap acceptance model allows the pedestrian agent to choose to cross all lanes, when safe to do so, in one go, known as *Double Gap* (Song et al., 2003) or one *Stage Crossing* (Paul et al., 2014). Alternatively, if the agent is identified as a risk-taker, they may choose to cross lane by lane, rather than crossing all lanes in one go, sometimes waiting in the middle of the road, known as *Rolling Gap* (Brewer et al., 2006) or risk-taker crossing behavior (Song et al., 2003). The risk-taker attribute is user defined in EXODUS, which can be either assigned at the individual pedestrian level or specified as a global percentage of the current pedestrian population.

At the start of the simulation EXODUS automatically divides the sidewalk into a number of equal sized regions as close as possible to 10 meters in width, called *Pavement Cells,* see Figure 2. These *Pavement Cells* are used to decide where a pedestrian will reattempt to cross the road, after a previous failed attempt. If the agent decides not to cross the road, they will continue along the sidewalk in a direction which takes them nearer to their target destination, i.e., *D* in Figure 2. They will then reattempt to cross if they enter the next *Pavement Cell* or if they cannot continue further along the sidewalk, for example, by being blocked by other pedestrians or have reached the end of the sidewalk. The gap acceptance model will then "fire" again to evaluate if it is now possible for them to cross the road.

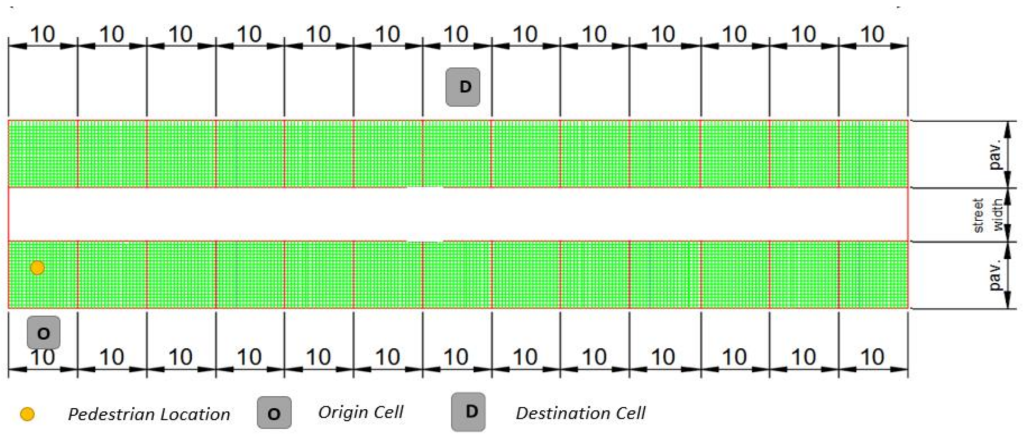


Figure 2: The Sidewalk is divided up into approximately 10-meter-wide *Pavement Cells*.

The approximate width of 10m for the *Pavement Cells*, was chosen based on research by Wang (Wang, 2012), who modelled where pedestrians chose to cross a 100m section of road with a zebra crossing. It was also identified by Song (Song et al. 1993) as representing a distance of not too far from a pedestrian crossing where a person may decide to either cross at a designated crossing or not.

3.1 Gap Acceptance (When to Cross)

The gap acceptance model deals with the behavior of how a pedestrian selects a gap in the traffic when crossing a road. Since the pedestrians have a choice to accept or reject a gap in the traffic the implementation is based on a *Cumulative Logistic Distribution Function* (Balakrishnan, 1992).

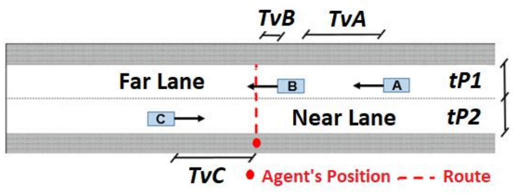


Figure 3: Gap acceptance parameters (two-way)

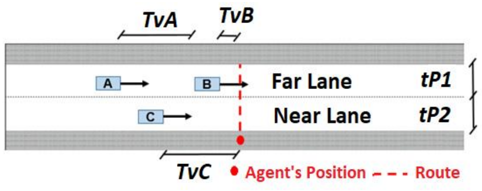


Figure 4: Gap acceptance parameters (one-way)

When an agent decides to cross the road, they consider vehicles in the near and far lane. In this model, only the nearest vehicle in the near lane and the first two vehicles in the far lane are considered. The key parameters for perpendicular road crossing are shown in Figure 3, for two-way traffic, and in Figure 4, for one-way traffic. The *TvA* parameter is the time gap, measured in seconds, between vehicles *A* and *B* in the far lane, called the *second far gap*. The parameters *TvC* and *TvB* are the times in seconds for vehicles *C* and *B* to reach the agent's crossing location, called the *first near* and *first far* gaps, respectively.

The pedestrians will use the *Double Gap Model* if they are identified as risk-averse; hence, will try to cross only when safe to cross both lanes in one go. This is when both the times to cross the first lane (*tP2*) and the second lane (*tP1*) are less than the vehicle gap in the near lane (*TvC*) and in the far lane (*TvB*). If this condition is met plus a safety margin *(S)* of *1*.5 seconds per a lane (Serag, 2014), a probability to cross is calculated, discussed later, as to whether the pedestrian accepts it and attempts to cross.

The *Rolling Gap Model* is used when the agent is considered a risk-taker. The agent will access the time to cross the first and second lanes considering *TvC* and *TvB*. Therefore, they will accept the gap on the first lane if *tP2 < TvC + S* and the time to cross the second lane if *tP1 < TvB -tP2 + S*. However, if this gap is not accepted, the agent will consider the *second gap far* (*TvA*). They will then accept the gap if *tP1 < TvA-tP2 + S and tP2 < TvC + S*. If either of these gaps are accepted, the agent will cross the first lane, then when they reach the middle of the lanes, the gap in the far lane is reassessed. Hence the behavior being modelled, is where the agent will cross the first lane, and then optionally, wait in the middle before continuing if in the second instance the far gap is not accepted.

If a pedestrian agent, waiting on the sidewalk to cross the road, fails to cross within the time specified by their patience attribute (uniformly randomly assigned between 1-30 seconds) (York et al., 2011) they will reassess where to cross and may choose to move further along the sidewalk and attempt to cross at a different location (i.e., choosing a different *Pavement Cell*).

Both models, *Rolling* and *Double Gap*, utilize a probabilistic model to assess if the pedestrian agent will accept a given viable gap in the traffic, which is a *Cumulative Logistic Distribution Function (CDF)* (Balakrishnan, 1992), as shown in Equation 1.

$$CDF = \frac{1}{1 + e^{-\frac{x-\mu}{s}}}$$

Equation 1

In Equation 1 *x* is a time gap parameter. For the *Double Gap* model *x* is equal to *TvC +TvB*, with *TvC* the near gap and *TvB* the far gap at time *t+Δt*, where *Δt = tP2*. The parameter *tP2* is the time to cross the first lane, see Figure 3. For the *Rolling Gap*, *x* is simply equal to *TvC,* the near gap. The *s,* parameter in Equation 1, is the scale parameter and is a function of the standard deviation in *x*. During the simulation, each time the CDF function is evaluated, *s* is assigned a random value between 0.992 and 1.221. This range is based on data collated by a number of researchers (Brewer et al., 2006, Schroeder, 2008, Yannis et al., 2010, Wang, 2012, Kadali, 2013, Serag, 2014, Pawar, 2014). The critical time factor μ, for the *Double Gap* model is calculated as *tP1 + tP2,* where for the *Rolling Gap model* it is simply *tP1,* the time to cross the near lane. A safety margin of 3 s, and 1.5 s is included in the *Double,* and *Rolling Gap* calculation, respectively, which are added to the μ value. These safety factor values represent the Safety Margin as defined by (Song et al. 1993).

Currently, the crossing behavior has only been developed for up to two lanes. For single lane roads the risk-takers and risk-averse pedestrian agents use the same one-stage crossing behavior, based on the *Double Gap Model*, without the consideration of a second lane *TvB* value. Crossing behavior for three or more lanes is left for further research.

4 Vehicles Yielding to Crossing Pedestrians

The topics related to pedestrian gap acceptance, vehicle yielding, and car-following, have been studied and analyzed separately (Zheng et al., 2015). Although, work has been conducted to investigate the decision-making process of pedestrians and vehicles, it is still not possible to describe a precise picture of the vehicle-pedestrian interaction phenomenon at locations outside marked crosswalks.

In the previous section the gap acceptance model for pedestrians' crossing behavior was described. However, a limitation of the gap acceptance model is that it only considers the current speed of any approaching vehicles and does not factor possible vehicle acceleration or deacceleration. Therefore, once a pedestrian has started to cross, any approaching and accelerating vehicle travelling in the same road lane, may need to adjust their speed to avoid a collision. Furthermore, if a crossing pedestrian is delayed, by having to avoid other pedestrians crossing the road or due to congestion on the opposite sidewalk, it may be necessary for an approaching vehicle to stop and wait, to allow the pedestrians to pass.

To accommodate the modeling of vehicles yielding to crossing pedestrians, EXODUS places a SUMO pedestrian (Erdmann and Krajzewic, 2015) agent into the SUMO simulation and on a given road lane if the presence of the agent crossing the road will impact the movement of the vehicles. The condition for this to be performed is if the distance between the crossing pedestrian and the near vehicle approaching the pedestrian is within twice the stopping distance of the vehicle's current speed. The stopping distance is calculated as s = $vt + \frac{v^2}{2d}$ where v is the vehicle's speed, t the driver's reaction time, and d the maximum deceleration rate. It should be noted that the driver's reaction time t will be dependent on several factors, such as age and gender. However, for simplicity, a mean reaction time of 0.89 seconds is utilized here, which is based on a study of vehicle reaction times in 13 provinces in China (Lui et al., 2002).

The SUMO representation of the crossing person remains stationary, located in the middle of the lane in question. This is until EXODUS deems that the crossing pedestrian has cleared the vehicle lane. At that point, the crossing pedestrian is removed from the SUMO simulation. This functionality allows the SUMO car following model to determine whether the vehicle should slow down or where and when it may need to stop.

It should be noted that if there is a group of pedestrians crossing a given lane, then only the pedestrian nearest to the approaching vehicle or vehicles is represented in SUMO. However, if the person which is nearest to the approaching vehicles, reaches the opposite sidewalk while other pedestrians are still crossing the road then the next person closest to the approaching traffic is added to SUMO as blocking the lane.

The vehicle yielding model described here is a first implementation developed for demonstration purposes that ensures that the vehicles slowdown or come to a standstill, without colliding with the pedestrians. However, this model can be replaced by a more sophisticated model at a later date, such as the models described by Zhao et al, 2020.

5 SUMO and EXODUS Integration

As mentioned previously, EXODUS links to SUMO via the TraCI API (Wegener et al, 2008). EXODUS operates as the main controlling application having knowledge of all people, vehicles, and model geometry, such as buildings and road network. The SUMO model data is initially limited to vehicle and road network information. When necessary, EXODUS will inform SUMO about certain crossing pedestrians, which may impact the movement of vehicles, and generate a pedestrian in SUMO, if necessary, as outlined in Section 4. A summary of how EXODUS and SUMO are synchronized is provided below and is shown in Figure 5:

1) The EXODUS simulation starts SUMO and defines the time step interval. This is typically a 6th of a second, which is the pedestrian movement time step utilized by EXODUS (Galea et al., 1996).
2) At each clock tick EXODUS updates the movement of the agents and advances the SUMO simulation by the same amount of time thus achieving synchronization.
3) EXODUS retrieves the updated position of the vehicles from SUMO via the TraCI API and represents them in EXODUS.
4) Pedestrian agents use the vehicles' current positions and speed for assessing their crossing decisions.
5) If a pedestrian agent is crossing a given lane and is identified by EXOUDS as possibly impacting any approaching vehicle's travel speed, a SUMO pedestrian agent is added to the lane that the pedestrian is currently crossing, see Section 3. If there is a group of pedestrians crossing a lane, only the pedestrian nearest to the approaching vehicle or vehicles is represented in SUMO (i.e., not all agents are explicitly modelled within SUMO).
6) As a result, vehicles in SUMO become aware of people in lanes when necessary, and can adjust their speeds accordingly (i.e., slow down/stop).
7) Changes in vehicles' speed and location are then sent back to EXODUS, where they are again used by agents in their decisions to cross the roads etc.

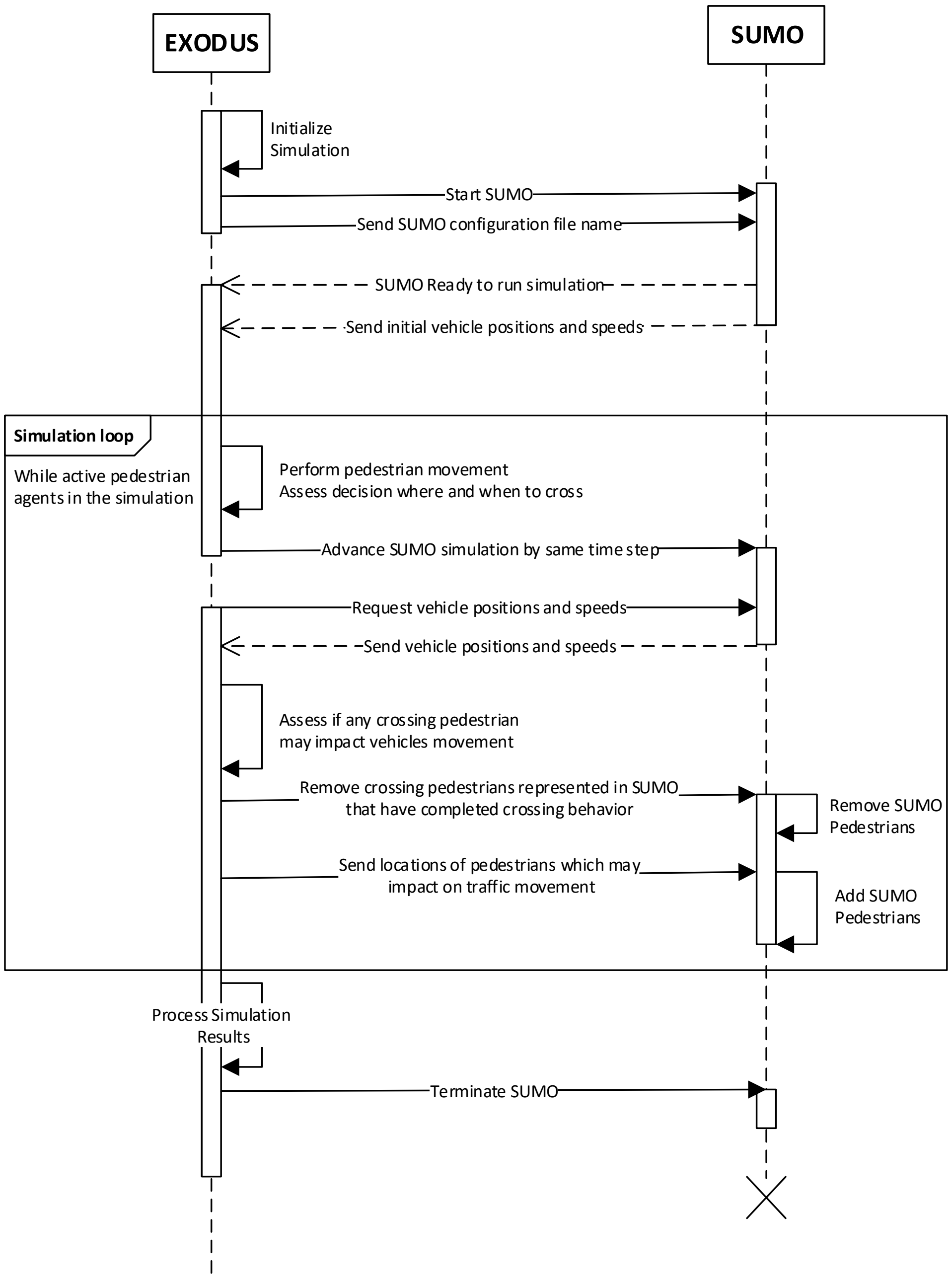

EXODUS
SUMO
Initialize Simulation
Start SUMO
Send SUMO configuration file name
SUMO Ready to run simulation
Send initial vehicle positions and speeds
Simulation loop
While active pedestrian agents in the simulation
Perform pedestrian movement
Assess decision where and when to cross
Advance SUMO simulation by same time step
Request vehicle positions and speeds
Send vehicle positions and speeds
Assess if any crossing pedestrian may impact vehicles movement
Remove crossing pedestrians represented in SUMO that have completed crossing behavior
Remove SUMO Pedestrians
Send locations of pedestrians which may impact on traffic movement
Add SUMO Pedestrians
Process Simulation Results
Terminate SUMO

Figure 5: Sequence Diagram showing messages passed between the two applications.

6 Demonstration Case

To demonstrate the proposed crossing model, a scenario where a crowd of people are leaving en masse from a station is outlined. The scenario involves a full-scale evacuation where people must reach a place of refuge located across a main road, see Figure 6. To highlight the crossing behavior no designated pedestrian crossing is provided. This is so the simulated pedestrians are forced to select a location along a section of the road and attempt to cross it, utilizing the gap acceptance model. To highlight the proposed behaviors on traffic flow and pedestrian movement, there are four different scenarios considered. The objective of these scenarios is to establish the time for the people to assemble and therefore, determine how the assembly process is affected by the presence of traffic and pedestrian vehicle interaction.

- ***Scenario 1*** No interaction between the vehicles and pedestrians. In this case the pedestrians cross the road with no regard to vehicles on the road. Vehicles and pedestrians pass through each other, so collisions do not take place in the simulation. While unrealistic, this scenario is used as a base case for comparisons and to highlight the impact of representing pedestrian and vehicle interactions, enabled, in the next scenarios.
- ***Scenario 2*** Interaction between vehicles and pedestrians is enabled. All pedestrian agents are risk-averse, so will cross both lanes of the road in one go, when deemed safe to do so.
- ***Scenario 3*** Interaction between vehicles and pedestrians is enabled. All pedestrians are assigned as risk-takers, so will cross lane by lane, when safe to do so, sometimes waiting in the middle of the road, if deemed necessary.
- ***Scenario 4*** Interaction between vehicles and pedestrians is enabled. Here the split between risk-averse and risk-takers is evenly split at 50%. Therefore, depending on the pedestrian type they will use one of the crossing behaviors described in Scenarios 2 and 3.

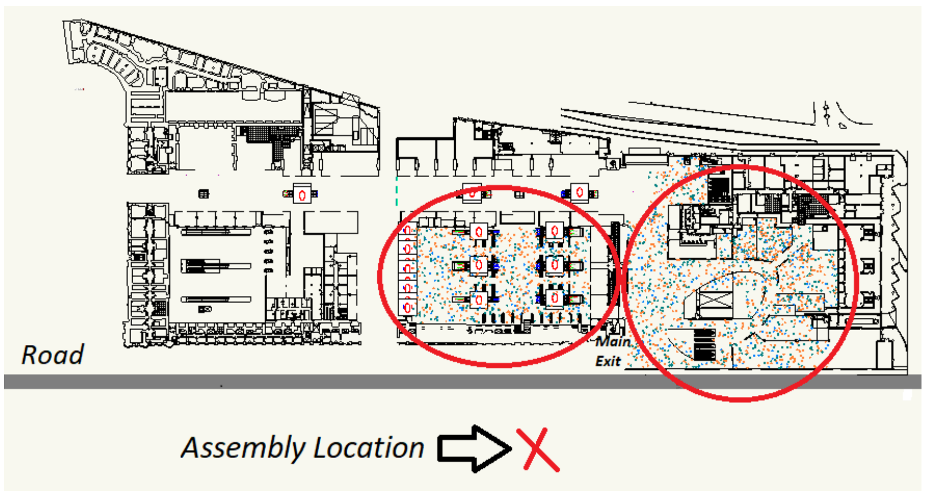


Figure 6: Station area being evacuated. The red circles indicate where people are initially located, who head through the main exit to the assembly point, marked with a red cross. Road shown in gray**.**

In all four scenarios, there are 2098 people placed within the circled areas shown in Figure 6. This population size is hypothetically based on assuming a maximum density within the circled areas of 1 person per 4m^2. In addition, the population is assigned a response time of between 60 and 90 seconds. The response time defines the time at which a pedestrian starts to evacuate the station and heads towards the assembly area. Furthermore, to eliminate the impact of different response times on the variability between simulations, each pedestrian responds exactly at the same time in all four scenarios and repeated simulation runs.

The vehicle generation rate in all simulations is also identical to minimize impact of traffic variability on results. The vehicles are generated at a rate of between 12 to 24 vehicles a minute, which represents a vehicle every 2.5 to 5 seconds, which represents a medium traffic level as defined by Hine (Hine, 1996). A bi-directional two-way road with vehicles driving on the left, as in the UK, is used for demonstration purposes. Furthermore, the SUMO's default car following model Krauss (Sumo, 2021) is utilized in all scenarios with a sigma value of zero, i.e., no randomness, to minimize variability.

The setting up of the scenarios is such that variability between the scenarios and simulation runs is artificially limited to emphasize the effect that the proposed crossing behavior has on the assembly process and how it is affected by the presence of traffic and pedestrian vehicle interactions.

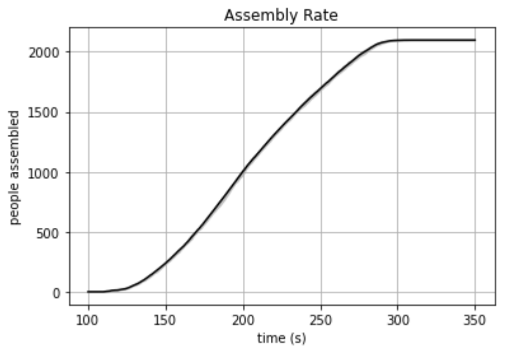
7 Results and Discussion

In Table 1 the overall simulation results from the four scenarios are presented, which include the time for the pedestrian agents to assemble at the assembly location shown in Figure 6. Each Scenario was run multiple times until an estimated accuracy of within 5% was achieved on the mean assembly time (Grandison, 2020) and shown in the column labeled "*Number of Repeated Simulations*" in Table 1. As expected, the time to assemble for the pedestrians is considerably quicker, in Scenario 1, since the pedestrians do not have to judge where or when to cross the road. Scenario 3 which included only risk-takers, pedestrians crossing lane by lane, was the next quickest (11% slower than Scenario 1), followed by Scenario 4, which had 50% risk-takers and 50% risk-averse people (55% slower than Scenario 1). As expected, Scenario 2, which only included risk-averse people took the longest time to assemble (85% slower than Scenario 1).

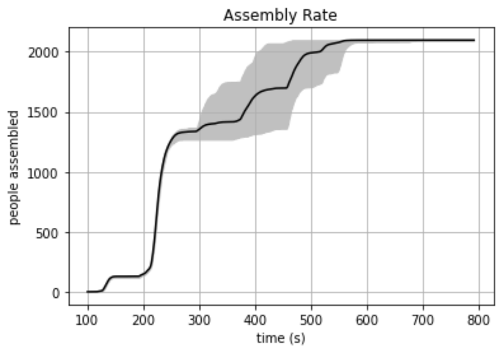
Scenario	Average Assembly Time (s)	Standard Deviation (s)	Number of Repeated Simulations
1 - *No Interaction*	302.9	2.18	10
2 - *All Risk Averse*	560.6	64.57	121
3 - *All Risk Takers*	337.3	20.40	21
4 – *50% Risk Takers*	469.7	109.43	168

Table 1: Pedestrian Results

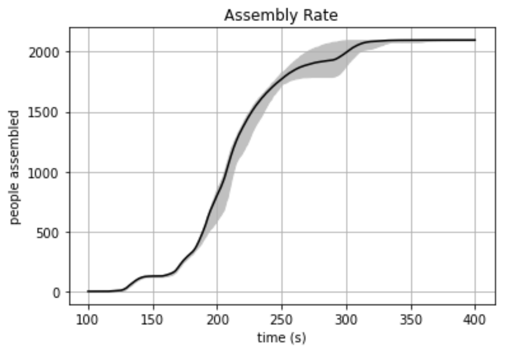
To highlight the differences in behaviors and the variability of the results, the assembly time graphs for the four scenarios are presented in Figure 7. The graphs in Figure 7 shows the average assembly time graph for each scenario together with minimal and maximal ranges.



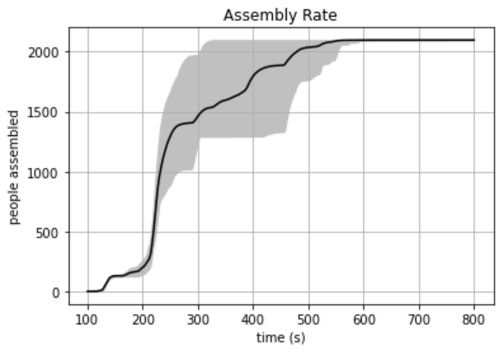
(A) Scenario 1 No Interaction



(B) Scenario 2 All Risk Averse



(C) Scenario 3 All Risk Takers



(D) Scenario 4 50% Risk Takers

Figure 7: Assemble time graphs for the four Scenarios, the shaded area represents the minimum and maximum average assembly ranges.

The graph in Figure 7A , highlights the lack of variabilty between the repeat simulations in Scenario 1. This is because the only variability in the results comes from the pedestrian-pedestrian interaction (Galea et al. 1996) in the EXODUS model, which in this scenario is very limited. Therefore, not modelling the pedestrian-vehicle interactions considerably removes significant human behavioral factors, such as when and where to cross and interaction with approaching vehicles.

The variability in assembly times for the risk-averse, Scenario 2, is shown in Figure 7B. As the pedestrian agents are risk-averse they are taking longer to find acceptable gaps in the traffic. If the probability model rejects an acceptable gap, the risk-averse pedestrian agents will take a longer time than the risk-taker agent to find an alternative acceptable gap. This is because a risk-taker only needs to consider vehicle gaps on one lane, whereas the risk-averse pedestrian needs to consider the gaps across both lanes of traffic, simultaneously. In fact, in Scenario 3, Figure 7C, the varibility is significantly less when compared to scenarios 2 and 4, i.e., Figures 7B and 7D, respectively, but larger than the non-realistic Scenario 1. Before going into detail as to the reasons behind this difference in variability between the four scenarios, it is necessary to look at vehicle behavior, which is discussed next.

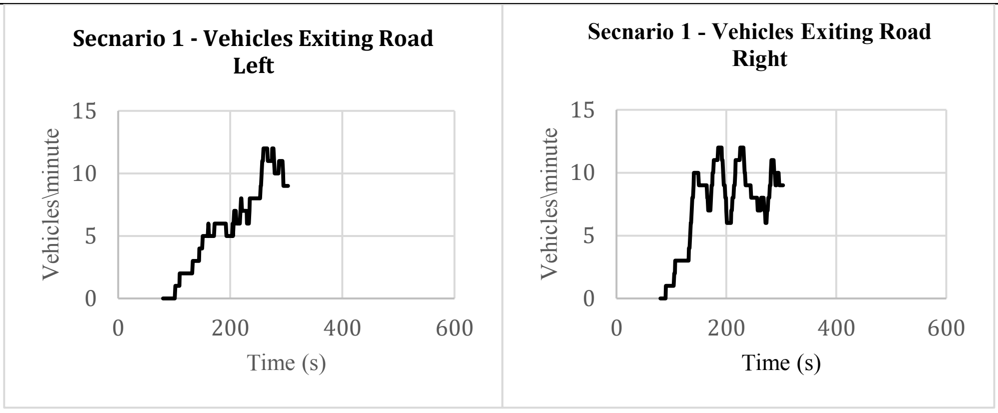
It should be noted that the same SUMO vehicle trip file, which specifies the vehicles' generation time, route, and characteristics, is used in all scenarios and across all simulations. The trip file was generated using a custom python script that generated vehicles travelling on both lanes, with a time gap between vehicles being randomly assigned a value of between 12 to 24 seconds, representing 2.5 to 5 vehicles a minute. Since the same trip file is used for all simulations and the default SUMO car following model with a sigma value of 0 is utilized, this effectively eliminates randomness from the vehicle model. Therefore, the only stochastic variability between simulations comes from the EXODUS conflict resolution algorithm (Galea et al., 1996) and the gap acceptance model described in this paper.

To highlight the impact of pedestrian crossing behaviors on traffic flows, Table 2 lists the total number of vehicles exiting from either end of the road during the simulations with the standard deviation shown in brackets. The standard deviation values are of interest here, since without pedestrian vehicle interaction the variation in assembly times between the simulations is limited. The inclusion of pedestrian crossing behavior impacts not only how long the assembly process will take, but also the variability of the data in relation to the total time for the pedestrians to assembly and the flow of traffic out of the system.

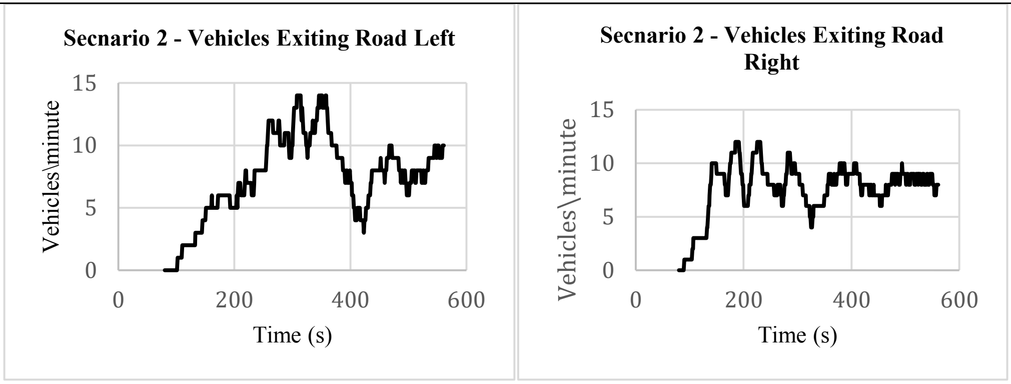
Scenario	Total Average Number of Vehicles	Average Number of Vehicles Travelling to the Right Upper Lane	Average Number of Vehicles Travelling to the Left Lower Lane
1 - *No Interaction*	53.55 (2.06)	28.44 (1.33)	25.11 (0.94)
2 - *All Risk Averse*	126.79 (19.10)	64.97 (9.03)	61.81 (10.07)
3 - *All Risk Takers*	58.95 (11.79)	29.82 (10.23)	29.14 (2.17)
4 - *50% Risk Takers*	112.11 (25.57)	58.12 (11.733)	54 (13.83)

Table 2: Vehicle Exiting Results. Note the standard deviation is shown in brackets.

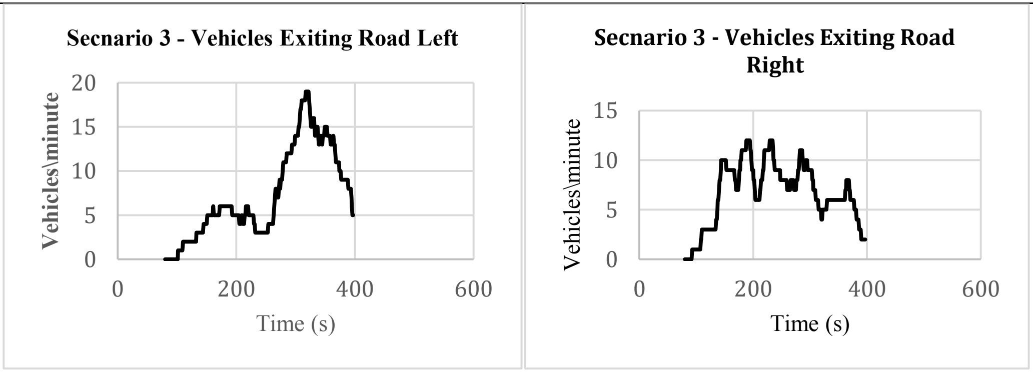
As expected, the scenarios that took the longest for the pedestrians to assemble had the greatest number of vehicles modelled. Therefore, to get some insight into the impact of pedestrians crossing behavior on traffic flow, the average exit flow rate of vehicles from either lane is shown in Figures 8 and 9, for all four scenarios. Looking at the *risk-taker,* Scenario 3, and more specifically at Figure 8C, the flow rate of the vehicles exiting the left side of the road starts to drop at around 200 seconds. This drop takes place sooner compared the other scenarios and is caused by the crossing pedestrians on the road, slowing down as the opposite sidewalk becomes congested. Consequently, the pedestrians on the road start to block the oncoming vehicles. As the vehicles start to slowdown or even stop to avoid colliding with these pedestrians, more people have the opportunity to start crossing the road, hindering further the vehicle flow, see Figure 10C. The flow rate then only picks up again once the congestion on the opposite sidewalk eases, therefore allowing the pedestrians to also clear the road.



(A) Scenario 1 – No interaction

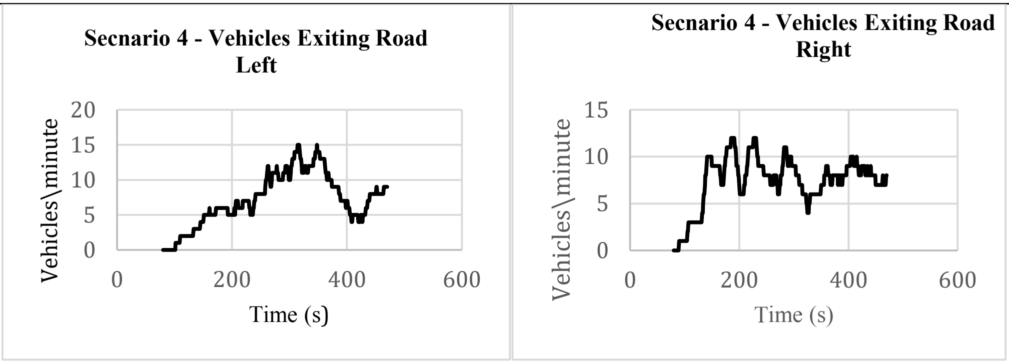


(B) Sceanrio 2 – All risk-averse



(C) Scenario 3 - All risk-takers

Figure 8: Vehicle Exit Flow Rates, Scenarios 1 to 3



(D) Scenario 4 - 50% Risk-averse and 50% Risk-takers

Figure 9: Vehicle Exit Flow Rates – Scenario 4

For comparison, snapshots from all four scenarios are shown in Figure 10, at around the 3-minute mark. As expected, there is a greater number of people waiting to cross in the case when all pedestrians are risk-averse, Figure 10B. In Figure 10D, it can be observed some pedestrians standing in the middle of the road, waiting for a safe gap in the lower lanes traffic before continuing to cross the final lane. Then in Figure 10C, as mentioned previously, risk-taker pedestrians are blocking the flow of traffic and crossing between the stationary vehicles.

In Scenario 1, where the interaction between the vehicles and pedestrians was not considered, very little variation between repeated simulation runs is observed and crossing pedestrians do not have an impact on the traffic flows. It should be noted, while pedestrians and vehicles do not interact in Scenario 1, some road crossing behaviors still take place, thus when the pedestrians reach the road, some delays occur as the agents negotiate the crossing of the road. In Scenario 2, the assembly of agents takes considerably longer compared to the other four scenarios. This is because the pedestrians are risk-averse and utilize only the *Double Gap* crossing behavior. Therefore, waiting longer before choosing when to cross compared to the other three scenarios. Hence, having less impact on the traffic flows than scenarios 3 and 4.

The simulation where all agents were risk-averse, Scenario 2, the pedestrian took 85% longer to assemble when compared to Scenario 1, where pedestrian vehicle interaction was not modelled. When comparing Scenario 1, with the risk-taker Scenario 3, there was only a 11% increase in assembly time. Therefore, while it is important to model pedestrian vehicle interactions as this improves realism of the scenario, a key component that must also be considered, is the crossing behavior of the pedestrians as this affects the overall movement of the pedestrians and the vehicles. Therefore, being able to quantify the impact that crossing behavior has on the pedestrian and traffic movement is an important factor when planning for mitigation strategies for crowd management.

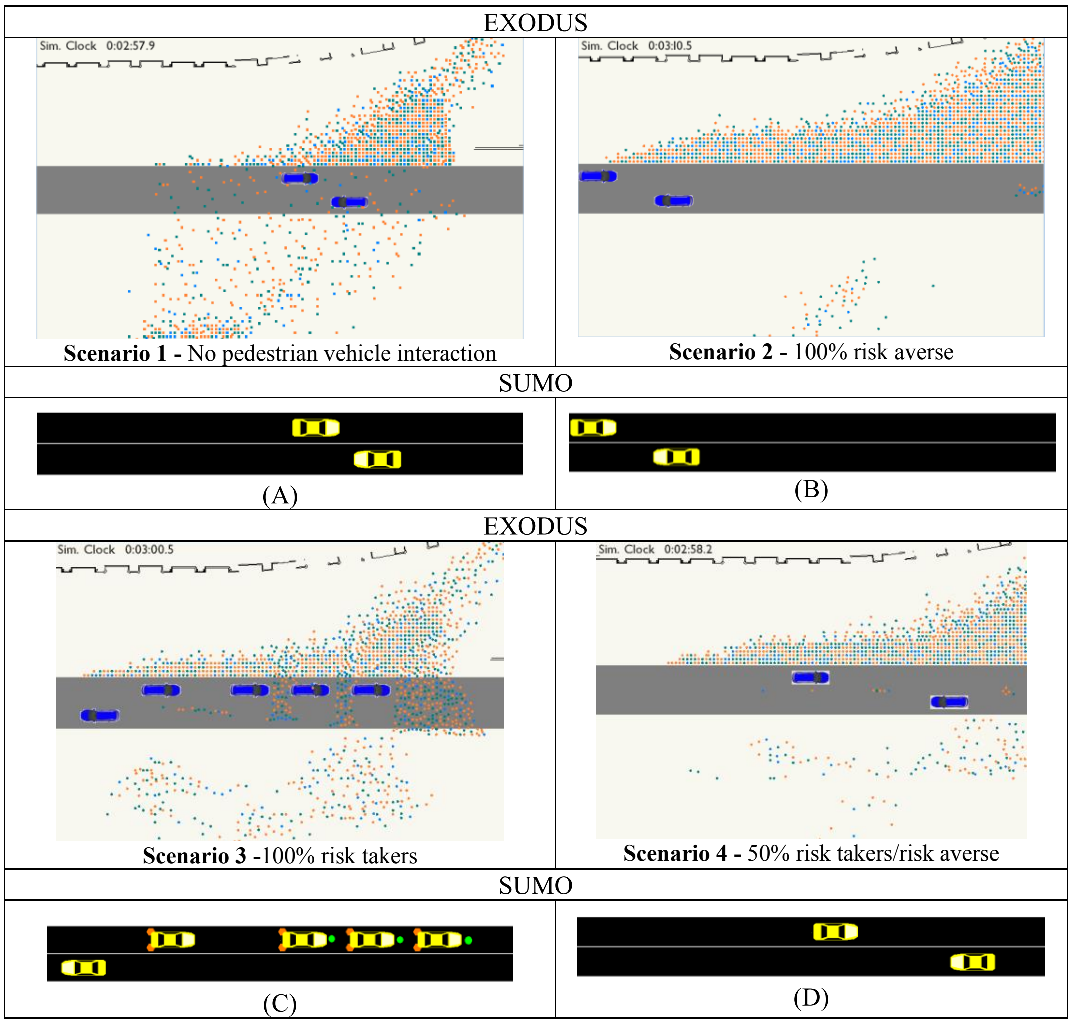

EXODUS
Sim. Clock 0:02:57.9
Sim. Clock 0:03:10.5
Scenario 1 - No pedestrian vehicle interaction
Scenario 2 - 100% risk averse
SUMO
(A)
(B)
EXODUS
Sim. Clock 0:03:00.5
Sim. Clock 0:02:58.2
Scenario 3 -100% risk takers
Scenario 4 - 50% risk takers/risk averse
SUMO
(C)
(D)

Figure 10: Snapshots of the examined scenarios at the 3-minute mark showing varying levels of pedestrians in the road. From none as in Scenario 2 (B), people waiting in the middle of the road Scenario 4 (D) and Scenario 3 pedestrians blocking vehicle and taking advantage of stationary traffic (C). The images at the top are from EXODUS while the images at the bottom from SUMO. In Scenario 3 you can see the Sumo *vType* person, the green circles, blocking the traffic in the upper lane. Red dots at the back of the vehicles indicate stopped vehicles. The colors of the individual pedestrians represent their age category, light blue ones are aged under 30, Orange 30 to 50 and Dark Green over 50.

In Scenario 3, where all the pedestrians are risk-takers and choose to utilize the *Double Gap* crossing behavior, a significant number of pedestrians choose to cross the road simultaneously. Therefore, considerable congestion can build up on the opposite sidewalk and along the center of the road, where pedestrians may choose to wait before attempting to cross the second lane of traffic. This congestion prevents other crossing pedestrians reaching the opposite sidewalk or central section of the road, resulting in pedestrians blocking the lanes of traffic. Then as the traffic slows and starts to stop, this then allows more of the pedestrians waiting to cross, the opportunity to do so. The less variability observed in the assembly time for Scenario 3, over scenarios 2 and 4, is because once the crossing pedestrians block the traffic flow, the only thing preventing them from reaching the assembly location is the congestion of pedestrians in the vicinity. The delay caused by the pedestrian congestion is considerably less than the variation in the delay caused by waiting to cross safely two traffic lanes.

In Scenario 4, where there is a 50% mixture of risk-taker and risk-averse pedestrians, the variability in assembly times is the greatest and the overall assembly time is similar to Scenario 2. This is because in parts of the simulations the risk-taker pedestrians may block traffic flows, where on other occasions this does not occur. When risk-takers slow or bring the traffic to a standstill, this provides opportunities for risk-averse pedestrians to cross. However, towards the end of the simulation, the risk-averse pedestrians will struggle to cross the road, as the traffic then starts to flow unhindered.

8 Conclusions and Final Remarks

A method of how SUMO can be connected to a third-party pedestrian model, namely EXODUS, to model pedestrian crossing and interaction at midblock locations was demonstrated. The presented model allows the user to evaluate people crossing outside designated areas using two different pedestrian crossing styles, namely *Double* or *Rolling Gap*. The ability to include a psychological *risk-taker* or *risk-averse* pedestrian attribute allows the user to consider the urgency of the pedestrians, which could be of benefit when modelling rush-hour or emergency situations, such as an evacuation from a large transportation hub.

The outlined crossing behavioral algorithm combines research from several studies to derive a unique *Gap Acceptance Model*, to accommodate the *Double* or *Rolling* Gap pedestrian crossing styles. To represent vehicles yielding to crossing pedestrians, a simplistic approach was taken, which relies on SUMO's inbuilt car following model. This simplistic approach is only a first step and research into a more suitable approach is ongoing. Therefore, it should be noted that currently this work provides a framework for experimental evaluation only, as further calibration and sensitivity of the model crossing parameters is required, before carrying out possible validation. Additional research is ongoing, looking at pedestrian choices and urgency and how this relates to the decision to use a marked pedestrian crossing (Lawrence et al. 2020). The presented demonstration case looked only at a two-way road scenario, a possible further study could be to compare these results with a two lane one-way road to identify what impact this proposed crossing behaviours may have.

Acknowledgements

Research into pedestrian crossing behaviors was partly funded under a University of Greenwich VC Research Scholarship. The development of the link between EXODUS and SUMO was performed as part of a larger EU Horizon 2020 project IN-PREP, Grant agreement ID: 740627 in conjunction with internal departmental HEIF funding 2018/19.

References

Akyol, G., Silgu, M.A., Celikoglu, H.B, (2019). Pedestrian-Friendly Traffic Signal Control Using Esclipse in SUMO, SUMO User Conference 2019, May 13-15 Berlin-Adlershof, Germany, May 2019, V.62 PP 101-106, https://doi.org/10.29007/c6k6.

Balakrishnan, N, (1992) Handbook of the Logistic Distribution, December 3, CRC Press, SBN 9780367450458.

Brewer, M. Fitzpatrick, K., Whitacre, J., Lord, D, (2006). Exploration of Pedestrian Gap-Acceptance Behavior at Selected Locations", Transportation Research Record, 1982(1), 132–140. doi:10.3141/1982-18.

Brosseau M., Zangenehpour S., Saunier N., Miranda-Moreno L., (2013). The impact of waiting time and other factors on dangerous pedestrian crossings and violations at signalized intersections: A case study in Montreal Transp. Res. F, 21, pp. 159-172.

Chooramun, N., Lawrence, P. , Galea E., (Aug, 2017), Evacuation simulation using Hybrid Space Discretisation and Application to Large Underground Rail Tunnel Station", Physical Sciences Reviews, Vol. 2, Issue 7, DOI:10.1515/psr-2017-0001.

Erdmann, J., Krajzewicz, D. (2015). Modelling pedestrian dynamics in SUMO, SUMO User Conference 2015, DLR, Berlin - Adlershof, Germany, 7-8 May 2019, V.28, pp103-118, ISSN 1866-721X.

Flötteröd, Y., Behrisch, M., Hendriks, M., Bonne, J., Vullings E., Bruining, R. Co-simulation of vehicles and crowds for rescue trials, , SUMO User Conference 2019, May 13-15 Berlin-Adlershof, Germany, May 2019, V.62 PP 56-69, https://doi.org/10.29007/qzg2

FSEG (2021) EXODUS Capabilities, Retrieved from: https://fseg.gre.ac.uk/exodus/exodus_products.html, [Accessed: 14 May 2021].

Han, L.D., Yuan, F. (2005, November) Evacuation modeling and operations using dynamic traffic assignment and most desirable destination approaches", in Proceedings of the 84th Annual Meeting Transportation Research Board, Washington, DC, USA.

Hine, J., (1996). Assessing the impact of traffic on behaviour and perceptions of safety using an in-depth interview technique, Journal of Transport Geography, 4(3), 179–197, 1996

Galea,E.R., Lawrence, P.J., Fillippidis L., Blackshields, D., Cooney, D., (2012). buildingEXODUS V 6.1 Theory Manual, Fire Safety Engineering Group, University of Greenwich, 2012.

Galea, E, Owen, M, Lawrence, P, (July, 1996). The EXODUS Model, Fire Engineers Journal, pp.26-30.

Grandison, A., (2020). Determining Confidence Intervals, and Convergence, for Parameters in Stochastic Evacuation Models, Fire Technology 56(6), DOI: 10.1007/s10694-020-00968-0.

Kadali, B. R., & Vedagiri, P. (2013). Modelling pedestrian road crossing behaviour under mixed traffic condition. European Transport - Trasporti Europei, (55), 1–17.

Islam, M. S., Serhiyenko, V., Ivan, J. N., Ravishanker, N., & Garder, P. E. (2014). Explaining pedestrian safety experience at urban and suburban street crossings considering observed conflicts and pedestrian counts. Journal of Transportation Safety & Security, 6, 335–355.

Lämmel, G., Klüpfel, H., Nagel, K., (2009). The MATSim Network Flow Model for Traffic Simulation Adapted to Large-Scale Emergency Egress and an Application to the Evacuation of the Indonesian City of Padang in Case of a Tsunami Warning, Chapter 11, pages 245-265, Emerald Group Publishing Limited, DOI:10.1108/9781848557512-011.

Lawrence, P.J., Pellacini, V., Galea, E.R. (March 2020). The Modelling of Pedestrian Vehicle Interaction for Post-Exiting Behaviour, Proceedings of the 9th International Conference on Pedestrian and Evacuation Dynamics (PED2018), Lund, Sweden, 21-23 August , 2018, Pages 271-279, Published March 2020, DOI: http://dx.doi.org/10.17815/CD.2020.60.

LEGION, (2017). Pedestrian and Traffic Simulation Integrated into a Single Software Application [Online], Available: http://www.legion.com/pedestrian-traffic-simulation-integrated-into-a-single-software-application, [Accessed: 3 Jan 2018].

Lopez, P.A., Behrisch M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y-P., Hilbrich, R., Lücken, L., Rummel, J. Wagner, P., Wiessner, E. (2018) Microscopic Traffic Simulation using SUMO, In: Proceedings of the 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4-7 Nov. 2018, DOI: 10.1109/ITSC.2018.8569938.

Liu, Y., Shi, J. and Xiong, H. (2002) The simulation technology in traffic systems (in Chinese). Beijing: China Communications Press.

Paul, M. Rajbonshi, P., (November 2014). A Comprehensive Review on Pedestrian Gap Acceptance at Unsignalized Road, International Journal of Engineering Research & Technology (IJERT), Vol. 3, Issue 11, November-2014, ISSN: 2278-0181.

Pawar, D. S., & Patil, G. R. (2014). Pedestrian temporal and spatial gap acceptance at mid-block street crossing in developing world. Journal of Safety Research, 52, 39–46. doi:10.1016/j.jsr.2014.12.006.

Serag.M.S. (2014). Modelling pedestrian road crossing at uncontrolled mid-block locations in developing countries. International Journal of Civil and Structural Engineering, 4(3), 274–285

Schroeder, B. J. (2008). A Behavior-Based Methodology for Evaluating Pedestrian-Vehicle Interaction at Crosswalks. Analysis, 332.

Song L., D. M. C. and B. J., (1993). Models of delay and accident risk to pedestrians. Transportation and Traffic Theory/ C.F. Daganzo (Editor), 295-312.

Song, L., Dunne, M.C., Black, J.A. (2003). Models of Delay and Accident Risk to Pedestrians, Transportation and Traffic Theory", Proceedings of the Twelfth International Symposium, Elsevier Science.

SUMO, (2021). Randomness [Online], Available: https://sumo.dlr.de/docs/Simulation/Randomness.html, [Accessed: 5 July 2021].

Sun, D., Ukkusuri, S., Benekohal, R., Waller, S.(2002, November). Modeling of motorist-pedestrian interaction at uncontrolled mid-block crosswalks, Proceedings of 82nd Annual Meeting of the Transportation, Research Board, Urbana.

Wang, T. (2012). Study of Pedestrian - Vehicle Interaction Behaviour by Microscopic Simulation Modelling, Doctoral Thesis, University of Southampton, Faculty of Engineering and the Environment.

Wegener, A., Piórkowski, M., Raya, M.. Hellbrück, H., Fischer, S., Hubaux, J. P. (2008) TraCI: An interface for coupling road traffic and network simulators, Proceedings of the 11th Communications and Networking Simulation Symposium CNS '08, pp. 155-163.

Xin X., Jia N., Zheng L., Ma S. (2014). Power-law in pedestrian crossing flow under the interference of vehicles at an un-signalized midblock crosswalk Physica A, 406, pp. 287-297, 10.1016/j.physa.2014.03.068

Yannis, G., & Technical, N. (2010). Pedestrian gap acceptance for mid-block street crossing. 12th World Conference for Transportation Research, 1–11

York, I., Ball, S. D., Beesley, R., Webster, D., & Hopkin, J., (2011). Pedestrian Countdown at Traffic Signal Junctions (PCaTS) - Road trial - (TfL 2481). http://content.tfl.gov.uk/PCaTS-Note-3-PCaTS-Trial-Results-Report.pdf

Zhang, C., Zhou, B., Chen, G., & Chen, F. (2017). Quantitative analysis of pedestrian safety at uncontrolled multi-lane mid-block crosswalks in China. Accident Analysis & Prevention, 108, 19–26.

Zhao, J., Malenje, J., Wu, J., Ma, R,. (2020) Modeling the interaction between vehicle yielding and pedestrian crossing behavior at unsignalized midblock crosswalks, Transportation Research Part F: Traffic Psychology and Behaviour Volume 73, August 2020, Pages 222-235.

Zheng, Y., Chase, Y., Elefteriadou, L, Schroeder, B., Sisiopoku, V.P. (2015). Modeling vehicle–pedestrian interactions outside of crosswalks, Simulation Modelling Practice and Theory, Volume 59, December 2015, Pages 89-101.

SUMO User Conference 2021
https://doi.org/10.52825/scp.v2i.87

Published: 29 Jun. 2022

Investigation of the effect of autonomous vehicles (AV) on the capacity of an urban transport network

Ronald Nippold[1], Peter Wagner[1,2], Olaf Angelo Banse Bueno[1], and Christian Rakow[2]

[1] Institute of Transportation Systems, German Aerospace Center (DLR), Germany

ronald.nippold@dlr.de

[2] Department of Transport Systems Planning and Telematics, Technical University of Berlin, Germany

Abstract

In this paper, we assess the effects of different shares of autonomous vehicles (AVs) on the traffic flow and, in particular, on the maximum possible capacity at signal-controlled intersections. For this purpose, all signal-controlled nodes in the traffic network of the Düsseldorf metropolitan area were systematically simulated and evaluated using the microscopic traffic simulation tool SUMO.

The analysis shows that defensively parameterized AVs – as envisaged in the umbrella project of this research – may decrease the maximum possible traffic at signal-controlled intersections. Moreover, the simulation runs indicate that capacity at these intersections decreases almost linearly with a growing share of AV. In a second part of this analysis, a freeway section was simulated with the same varying shares of CV and AV to investigate free-flow traffic. In this case, the simulation results of the maximum traffic flow can be approximated by a third-order polynomial fit. The minimum capacity is found for the uniform share of both vehicle types (i. e. 50 % AV and 50 % CV).

The overall intent of this project is to provide an approach to determine system-wide and long-term effects of AVs from local microscopic observations. To this end, the SUMO microscopic traffic simulation will be utilized to derive realistic flow capacities for signal-controlled intersections. In a next step, these capacities will be transferred to a mesoscopic traffic simulation. Subsequently, flow capacities can be systematically adjusted in this network-wide mobility simulation to parameterize the influence of future infrastructure and vehicle technologies.

1 Introduction

Autonomous driving is gradually becoming an operational technology today. This development is anticipated to radically change the transportation sector in the coming years (ref. [1]). A number of expectations are associated with this process. As autonomous vehicles (AVs) are equipped with a variety of sensors and communication technologies, they are now assumed to further increase road safety compared to human-controlled conventional vehicles (CVs). At the same time, the overall traffic efficiency and thus traffic flow is projected to be improved due to the more homogeneous driving style of AVs compared to human drivers.

Currently, car manufactures are testing their Advanced Driver Assistance Systems (ADAS) as the first level of automation and – in some cases – even prototype vehicles of higher levels of automation[1]. Partially, these pilot tests have already been carried out on publicly accessible

[1]https://www.sae.org/news/press-room/2018/12/sae-international-releases-updated-visual-chart-levels-of-driving-automation

roads. For this purpose, test fields with intelligent infrastructure have been set up in a number cities and regions[2].

However, there are still some open or not fully resolved issues that are subject of current research. A lot of work is spent on validation and verification of AVs [2, 3, 4]. In particular, this involves analyzing the artificial intelligence (AI) components, which are expected to replace human drivers as decision-makers in the future [5, 6]. In addition, aspects of cyber security in general or for specific components are also being intensively investigated [7, 8, 9]. On the side of (legal) regulations, there is also still a certain need for conclusive and complete regulations with regard to AVs [10, 11].

In particular, the ability of AVs to communicate with traffic infrastructure is expected to enable an "intelligent" and holistic decision-making process for traffic control. In the past, several algorithms for an optimized intersection control have been proposed in the literature [12, 13, 14, 15]. However, in the transition phase with mixed mode traffic, the capacity of roads and especially of intersections does not necessarily increase. Examples of mixed traffic can be found in the literature, in which AVs drive defensively with larger headways and thereby significantly reduce traffic flow and thus capacity at intersections [16] and on freeways [17].

There is still no clear picture of how traffic capacities will develop in the future, especially during the transition phase with an increasing share of AVs. For this reason, the paper investigates the flow capacities at signal-controlled intersections of a real-world road network of a metropolitan area with different shares of AV and CV in a systematical manner. This analysis is part of an approach to determine system-wide and long-term effects of AVs from local microscopic observations. For this purpose, the microscopic simulation "SUMO" [18] is integrated in a framework with the mesoscopic agent-based mobility-simulation "MATSim" [19] to derive realistic flow capacities for traffic signal-controlled intersections for different shares of CV and AV. This framework is applied to systematically adjust and parameterize the effect of future infrastructure and vehicle technologies in the network of the greater Düsseldorf area in Germany, which is part of the German national test field for automated and connected driving[3].

2 Scenario overview

The following section illustrates the analyzed road network and lists all required components to build and simulate the virtual representation of the scenario.

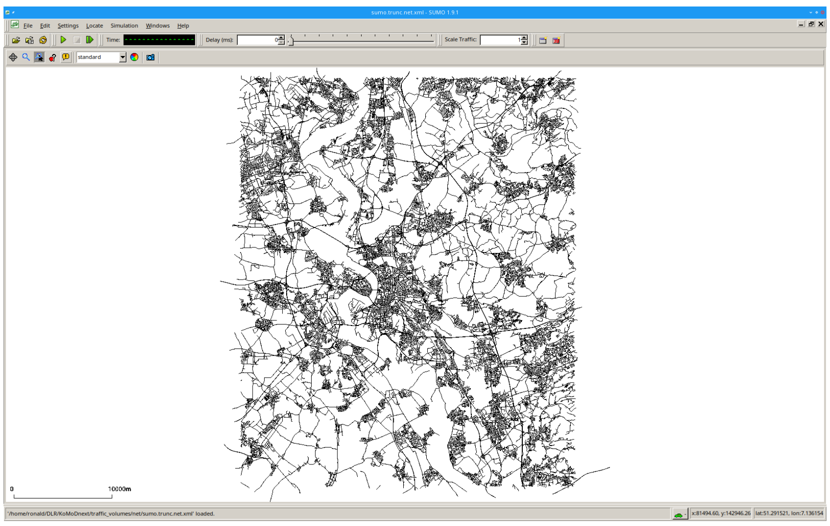
2.1 Simulation network

As the basis for this research, a raw OSM (openstreetmap[4]) road network of the greater Düsseldorf area was imported into SUMO. Due to the dimensions of this network (area: approximately 900 km^2, total length of all edges: about 13700 km, number of signal-controlled intersections: 1637), post-processing and quality control were generally omitted. Figure 1 on the facing page shows the network and its expanse. The city of Düsseldorf can be recognized by the close meshed road network in the center of the picture.

[2]For an overview of the situation in Germany cf. e. g. https://www.bmvi.de/DE/Themen/Digitales/AVF-Forschungsprogramm/Projekte/avf-projekte (German language only).

[3]https://komodnext.org/

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



'/home/ronald/DLR/KoMoDnext/traffic_volumes/net/sumo.trunc.net.xml' loaded.

Figure 1: Dimension of the simulation network under investigation. The city of Düsseldorf is situated in the center of the network. The broad, curving and road-free band running vertically through the network is the river Rhine.

2.2 Driver models parameters

For the present analysis, the SUMO implementation of the KRAUSS model was used for simulating the vehicle behavior due to its simplicity and reliability. This model, as documented in [20], is used as the default car-following model in SUMO. It is a space-continuous model based on a safe speed definition. As an advantage, the model needs only a moderate set of input parameters in comparison to other car-following models, e. g. the WIEDEMANN model used in VISSIM [21] or the standard model of the MITSIM simulator [22].

For the analysis, CVs were simulated using SUMO's default parameter set for passenger cars (cf. the second column of Tab. 1). As shown in the third column of this table, the parameters for the AV were adapted appropriately. For technical reasons, the deceleration of AV in normal

Table 1: Parameter set for modeling CV and AV. CV corresponds to the SUMO default parameter set and is only displayed for easy comparison. All other model parameters not listed in this table correspond to their respective SUMO default values.

Parameter	**CV (default)**	**AV (adapted)**
`decel`	7.5	3.0
`tau`	1	1.5
`sigma`	0.5	0.1
`speedFactor`	1	1
`speedDev`	0.1	0

mode is limited to $b = 3.0\,\mathrm{m/s^2}$. Note that there is another emergency breaking mode for AVs, which allows a much higher braking deceleration. The driver's desired (minimum) time headway τ refers to the net space between leader back and follower front. AVs were simulated rather conservative with a time headway of $\tau = 1.5\,\mathrm{s}$ reflecting the specifications used in the umbrella project. Note that τ may possibly decrease by several orders of magnitude as technology of AVs advances and the regulatory environment changes. However, the regulatory framework is still not fully defined. The driver imperfection σ reflects small deviations and defects that human driver usually introduces into car-following mode due to variations in perception or concentration. A value of $\sigma = 0$ denotes perfect driving without any deviations. The value of $\sigma = 0.1$ is again chosen rather conservative and reflects the occurrence of latencies in the internal signal processing modules of AVs. The parameters speedFactor and speedDev indicate the mean and variance of a possible speed deviation, i. e. whether drivers generally exceed or fall below the defined lane speed. According to Tab. 1, AVs drive always with the intended or permissible speed.

Any other vehicle classes – e. g. motorbikes, buses and trucks – were neglected in the investigations. Lane changing is also not integrated into the analysis, as the aim was to study vehicle flow in a single driving or turning lane in front of signal-controlled intersections. In addition, no V2X communication or other AV concepts such as platooning etc. were applied, as their use is also not planned in the umbrella project.

2.3 Simulation setup

The road network contains 1637 signal-controlled nodes, which in turn are divided into 17653 driving or turning lanes. The traffic signals in this network were modeled as traffic actuated units. In this way, the traffic signals react to the magnitude of the respective traffic flows. Each lane at these traffic signal-controlled intersections was simulated separately in SUMO with different shares of CV and AV to determine the maximum possible traffic flow. For this purpose, the SUMO interface TraCI (traffic control interface) was used, which allows interacting with a running traffic simulation and modifying objects and their behavior.

In a first step, the network was loaded and parsed with the Python module sumolib. All signal-controlled intersections were identified and their IDs were stored in a list. Internal intersections (i. e. the waiting positions within the intersection) were ignored. The actual determination of capacities was then performed iteratively using a Python script and TraCI for each of the previously stored intersection IDs. This involved looping through each possible lane of each identified intersection ID and creating a route file with a vehicle flows for a variable share of CV and AV. The overall traffic volume was set to $2000\,\mathrm{veh/h}$ for that specific lane (sum of CV and AV share). Respective "e1"-detectors with a frequency of 1 sec were defined in SUMO for this connection. Finally, the simulation was run for one hour using the route and detector files generated above as inputs. To ensure a faster procedure, the list of intersections to be processed was divided and manually distributed to the available CPU cores of a server. The size of the road network and thus the required working memory played an important role and limited the number of possible parallel runs.

The detector outputs, resulting from all the simulations performed, had all their file names corresponding to the connections they were scrutinizing. Subsequently, an R-script was developed to extract the data from all of these detector output files, determine the traffic volumes relative to an hour, and summarize the results.

3 Relationship between road capacity and headways

The following section gives a brief description of the relationship between road capacity q, speed v and headways T in the vehicle flow. The capacity c generally corresponds to the maximum traffic flow $q_{\max}$ that a traffic flow can reach at a given road section under defined environmental and traffic conditions. The traffic flow q is defined as the number of vehicles per unit of time. The smallest possible unit of q is the inverse gross headway T_g of a vehicle:

$$q = \frac{1}{T_g} = \frac{v}{v \cdot T + l}. \tag{1}$$

The gross headway T_g itself is defined as the space $d = v \cdot T + l$ that is occupied by a car; where T is the net time headway and l is the generalized car-length, i. e. physical length of the car plus the additional free space occupied in a jam.

The traffic flow q as a function of the speed v is a monotonously increasing function, as shown in Fig. 2. Eq. (1) can be rewritten into the more familiar $v(q)$-form:

$$v(q) = \frac{q \cdot l}{1 - q \cdot T}. \tag{2}$$

This corresponds to the congested arm of the FD (rf. [23]). The free arm is a declining function of speed v versus demand q, at least for heterogeneous car-fleets with different preferred speeds of the vehicles:

$$v(q) = v_{\text{free}} \left(1 - \frac{q}{}\right). \tag{3}$$

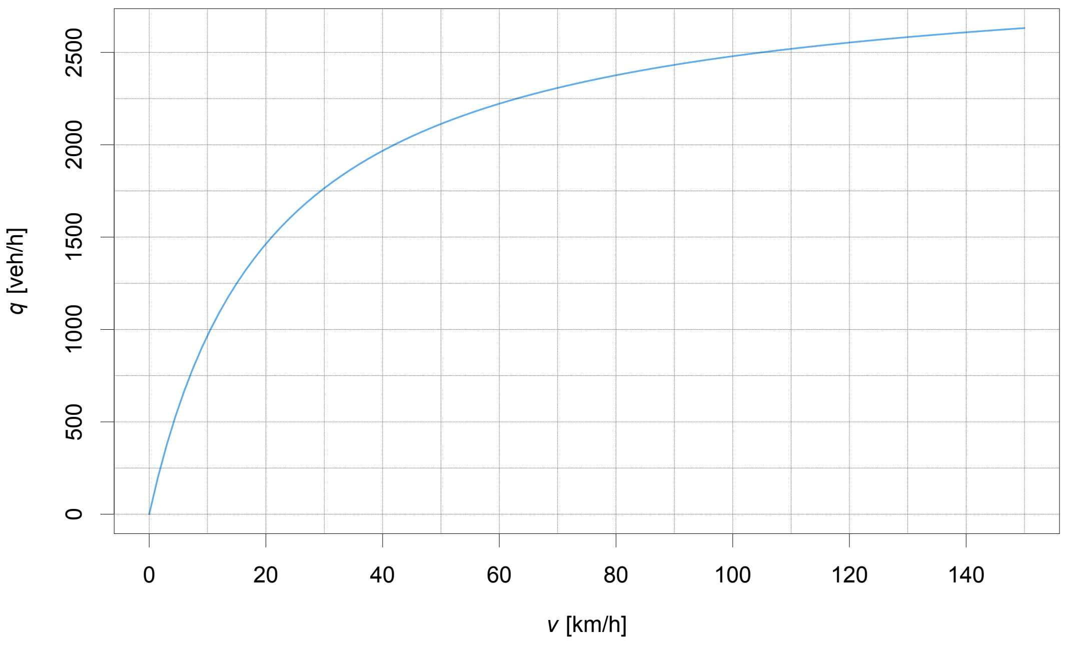


Figure 2: Monotonous increasing capacity q vs. speed v curve.

Note, eq. (3) can be simplified for homogeneous fleets as follows:

$$v(q) = v_{\text{free}}. \tag{4}$$

Therefore, the full definition of the FD as a combination of eqn. (3) and (4) reads:

$$v(q) = \begin{cases} v_{\text{free}} \left(1 - \frac{q}{q_0}\right) & \text{if} \quad v > v_c \\ \frac{q \cdot l}{1 - q \cdot T} & \text{if} \quad v \leq v_c \end{cases}. \tag{5}$$

The capacity versus speed curve in Fig. 2 is valid under the assumption that all vehicles in a given traffic stream follow each other with the optimal distance $v \cdot T$. Of course, this would require that the distance T is independent of the velocity v. However, the validity of this assumption is not guaranteed.

Eq. (1) can be easily generalized or adapted to a fleet of different vehicles; where each vehicle type i is characterized by its length l_i, the headway T_i and a respective share p_i ($\sum_i p_i = 1$). This results in a more complex formulation of Eq. (1) as follows:

$$q_c = \frac{v}{\sum_i p_i \left(v \cdot T_i + l_i\right)}. \tag{6}$$

The assumption that AVs follow each other (if this information is distributed via V2V communication) at a shorter distance, but increase their headway in case of CV as leading vehicle, can be modeled with a similar approach. At least three different distances are introduced then: T_{aa}, T_{ah}, T_{hx}; where the first index denotes the following and the second index the leading vehicle. If θ denotes the share of AV, Eq. (6) can be rearranged as follows (assuming one generalized vehicle length l):

$$q_c = \frac{v}{l + v\left(\theta^2 \cdot T_{aa} + (1-\theta)^2 T_{hh} + \theta\,(1-\theta)\left(T_{ah} + T_{ha}\right)\right)}. \tag{7}$$

4 Results

This section presents the maximum possible capacities observed at the signal-controlled intersections in the simulations for the greater Düsseldorf area. For this analysis, the share of AV was increased by 10 % steps reaching from 0 % up to 100 %. CV and AV were parameterized and modeled according to Tab. 1 in subsection 2.2.

4.1 Signal-controlled intersections

Fig. 3 on the next page gives an overview of the determined maximum capacities q on all lanes of all signal-controlled intersections in vehicles per hour for the 11 different shares of CV and AV. In general, the distributions obtained correspond to the expected assumptions, especially for the case of pure CV traffic (cf. the dashed blue curve in Fig. 3). The mean value for 100 % CV equals $\overline{q} = 1131\,\text{veh/h}$, the minimum value is $q_{\text{min}} = 24\,\text{veh/h}$ and the maximum value reaches $q_{\text{max}} = 1766\,\text{veh/h}$. There is a total amount of 20 cases (i. e. lanes of signal-controlled intersections) with unrealistic low values of $q < 100\,\text{veh/h}$. These values are caused by network errors in the unprocessed OSM network. Fig. 4 a on the facing page contains an example for an intersection with partial lane defects that significantly impact traffic flow. For the case of 100 % CV, the traffic flow exceeds a value of $q > 1600\,\text{veh/h}$ for 369 lanes. Such high values at

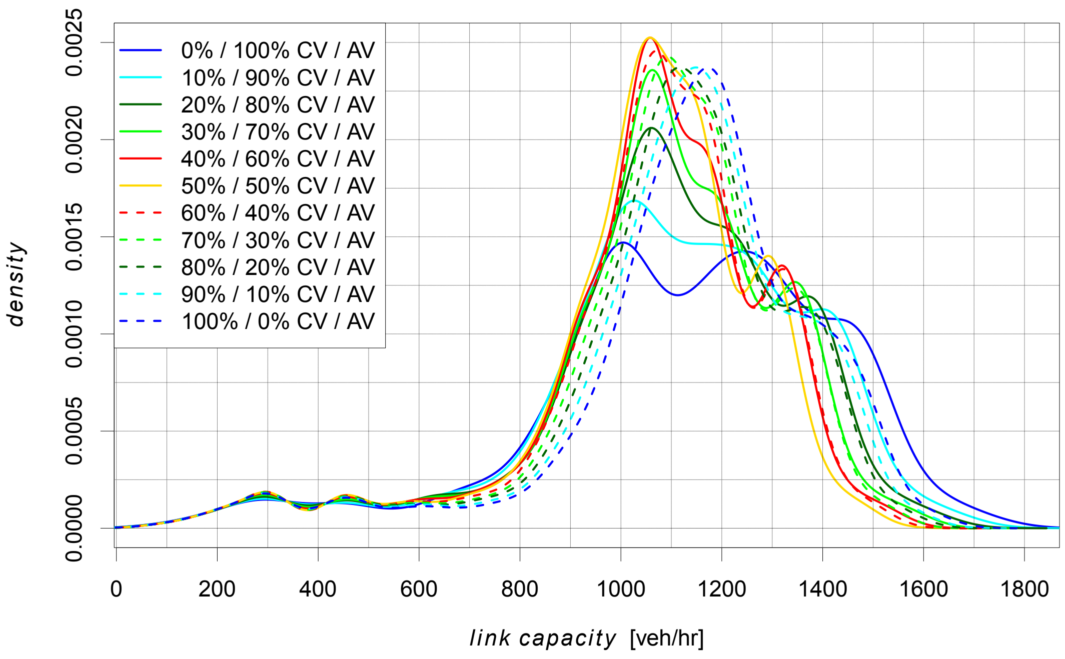
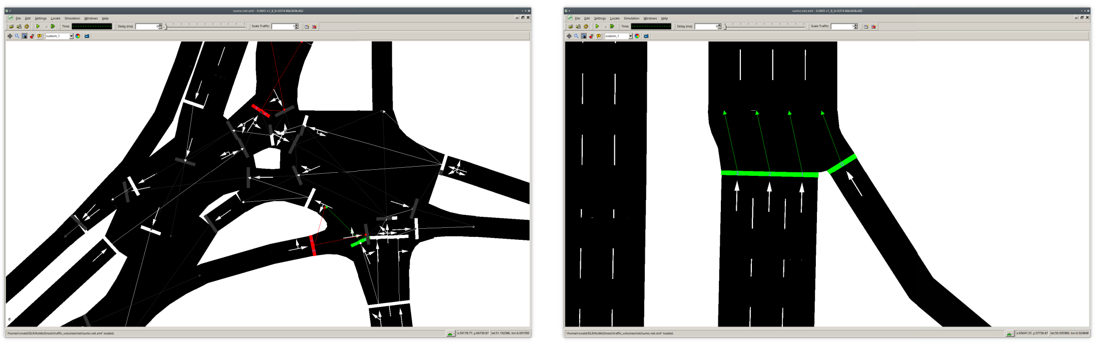


Figure 3: Density curves of the theoretical maximum traffic flows for all lanes in front of all 1 637 traffic signal-controlled intersections in the study area. The different penetration rates of AVs are reflected by different colors and line types. Note that higher percentages of AVs are shown as solid lines, while lower AV rates (and thus higher rates of CV) are drawn as dashed curves.



(a) Network defects

(b) Different/ high priorities

Figure 4: Example of an intersection with network defects and thus significantly reduced traffic flows (a). Example of traffic lights at a junction of roads with different priorities or road categories (b). Link indices 1 to 3 on the northbound main road have a higher priority and therefore a potentially higher traffic flow.

urban intersections are not quite unrealistic. Generally, these values represent special cases of intersections with different road categories. Fig. 4 b shows an example where vehicles traveling northbound on the three-lane main road can simply continue straight ahead at normal speed when the traffic light is green. These vehicle do not have to reduce their speed, as is the case at other intersections, e. g. when turning left or right.

Fig. 5 shows the range of the peak values ($1000\,{}^{\mathrm{veh}}/_{\mathrm{h}} < q < 1200\,{}^{\mathrm{veh}}/_{\mathrm{h}}$) of the density curves of Fig. 3 in greater details. With an increasing share of CV traffic (dashed lines in Fig. 5), the
p
b

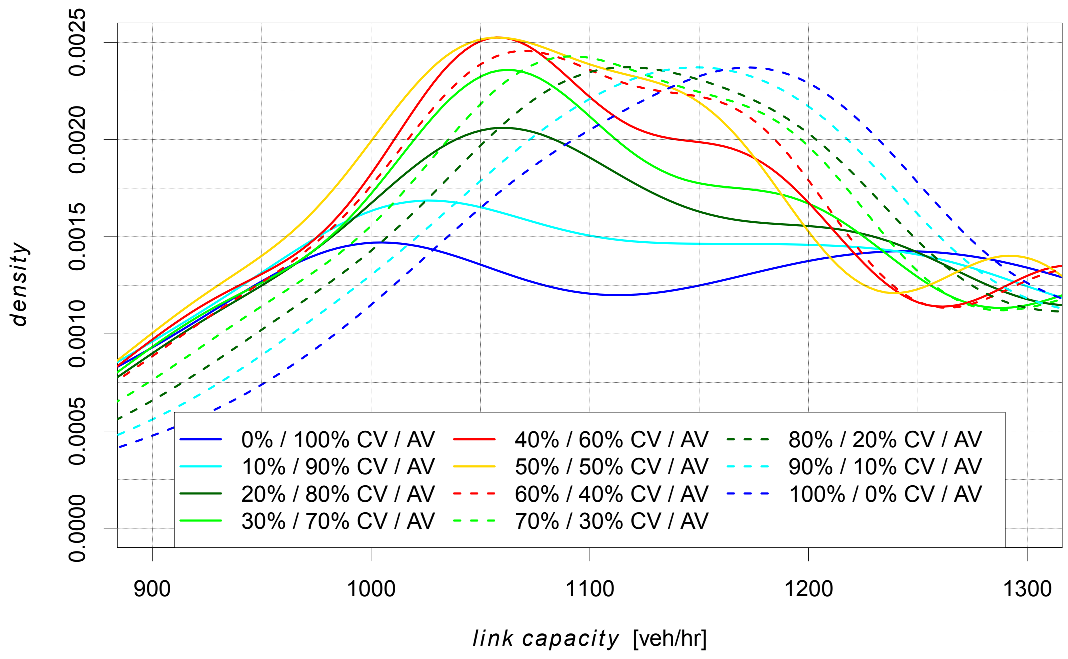


Figure 5: Detail of the peak values of the density curves of the theoretical maximum traffic flows for all lanes in front of all 1637 traffic signal-controlled intersections in the study area.

This diagram presents a linear fit of the class means of the maximum traffic flow rate q for the 11 analyzed distributions of CV and AV. The x-axis represents the share of CVs in the simulated traffic flows. The share of AVs is correspondingly reciprocal. Obviously, the class means of the maximum traffic flow rate q increase almost linearly with a growing share of CVs. Only the class mean for the uniform distribution of both types of vehicle (i. e. 50 % AV and 50 % CV) deviates slightly from this linear relationship. This behavior may be caused by the maximum degree of disruption in traffic flow introduced by the two different driving regimes of AV and CV. The class means of the maximum traffic flow rate of 100 % AV ($q = 1010\,{}^{\mathrm{veh}}/_{\mathrm{h}}$) and 100 % CV ($q = 1170\,{}^{\mathrm{veh}}/_{\mathrm{h}}$) differ noticeably by over 10 %.

As a consequence, this means that road capacities decrease as the number of AVs increases. This finding is in line with an analysis of Mattas et al. [24] who conducted a simulation study for AVs and connected autonomous vehicles (CAV) using a model proposed by Shladover et al. [25]. This study evaluates effects of connectivity and automation on the Antwerp ring road in Belgium. The results demonstrate that AVs can have a negative impact on the traffic flow q,

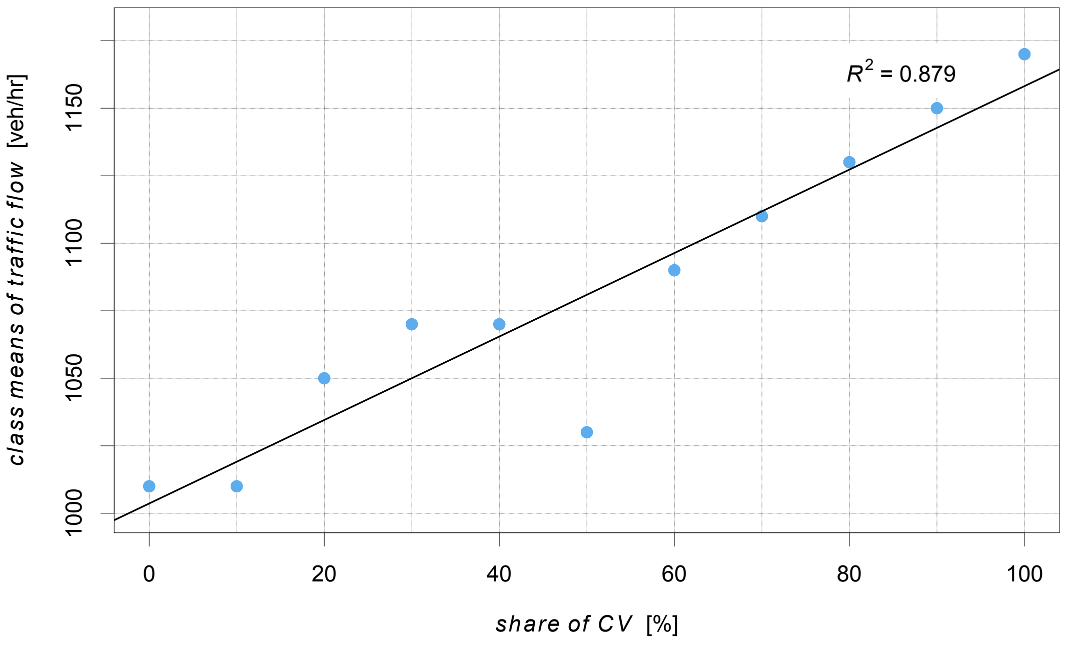


Figure 6: Linear fit of the class means of the peak capacities for the analyzed distribution of CV and AV at signal-controlled intersections. The x-axis denotes the share of CV, the share of AV is correspondingly reciprocal to it.

depending on the overall traffic demand. The connectivity and the exchange of traffic messages through CAVs have emerged as key elements for achieving higher network capacities. As a result of the reduction in capacity due to AVs, travel times for motorized individual transport also increase. Accordingly, individual transport would become less attractive overall due to the potentially greater volume of congestion. This should ultimately affect the modal split in the study area.

However, it must be pointed out that the study was concerned with the maximum possible flow. In reality, traffic is not at the limit at all times and in all places. Also, the quite conservatively dimensioned vehicle parameters of the AVs only represent a current interim status and not the final technological development. In this direction, further improvements are to be expected in the future. In addition, it must also be taken into account that there is currently no conclusive legal framework that bindingly outlines a safe parameter range for AVs.

4.2 Freeway

The situation described in the previous section is characterized by very high traffic demand at signal-controlled intersections. The traffic flow is interrupted at more or less regular intervals by the necessary activation of the transverse direction. Free traffic flow and thus car-following only occurs in certain intervals of time.

For this reason, the capacity impact of the 11 evaluated distributions of CV and AV was investigated for a car-following situation on a freeway section in the test area. The selected road section was the approx. 5.5 km long segment of the BAB 57 between the interchanges

"Meerbusch" and "Kaarst" in the west of Düsseldorf. The autobahn has three lanes, and the maximum speed is 100 km/h. The terrain is very flat, the road gradient is less than 1 %.

According to the HBS (German "Handbuch für die Bemessung von Straßenverkehrsanlagen") [26], the maximum traffic flow on a comparable road section within metropolitan areas and without heavy duty traffic is $q_{\max} = 5700$ veh/h. The Swiss "Bundesamt für Strassen" calculates with $q_{\max} = 5800$ veh/h for equivalent road sections and circumstances [27]. Therefore, this road section was simulated with the same shares of AV and CV, the same model parameter sets according to Tab. 1 and a traffic demand of $q = 5800$ veh/h. In accordance with Fig. 6, the class means of the determined maximum traffic flow rate q are presented in Fig. 7 for the freeway sc
o

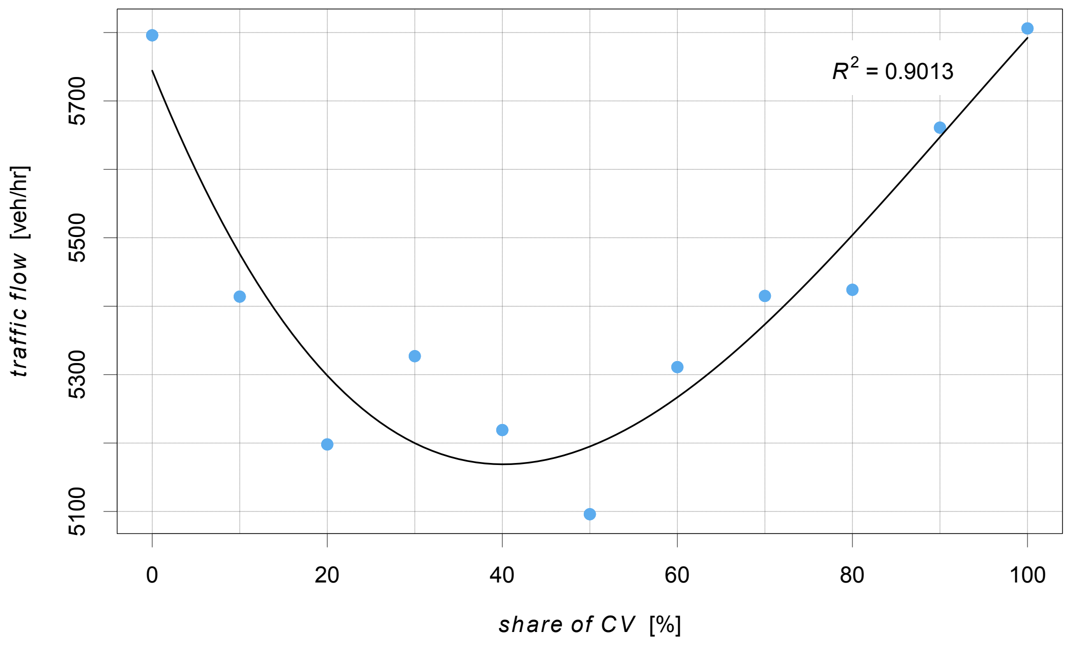


Figure 7: Third order polynomial fit of the class means of the peak capacities for the analyzed distribution of CV and AV for freeway traffic. The x-axis denotes the share of CV, the share of AV is correspondingly reciprocal to it.

In contrast to the case of the signal-controlled intersections, the resulting point cloud of class means of the maximum traffic flow q can be approximated by a third-order polynomial fit in a reasonable way. The uniform distribution of both vehicle types (i. e. 50 % AV and 50 % CV) leads to a minimum traffic flow of $q = 5096$ veh/h.

The maximum traffic flow rates can be observed for both AV-only and CV-only traffic. The maximum class mean for 100 % AV is only very slightly higher ($q = 5806$ veh/h) compered to 100 % CV ($q = 5796$ veh/h). Note that the class means of the maximum traffic flow q themselves and consequently the fitting curve are not symmetric. This implies a larger increase in road capacity resp. traffic flow rate q with a growing share of AV than for the case of an increased CV rate. Thus, a quadratic fit of the resulting point cloud only achieves a worse coefficient of determination ($R^2 = 0.8642$) than a third order polynomial fit ($R^2 = 0.9013$). The reason for this can be assumed to be the smoother traffic flow with a higher share of automated vehicles exhibiting a well defined and uniform driving behavior. The simulation of the free-flow traffic on the freeway segment shows that the larger headways of the AVs lead to decreased road capacities only in conjunction with the disrupted traffic flow at signal-controlled intersections.

5 Conclusion

The aim of this study is to assess the effects of different shares of AVs on the traffic flow and, in particular, on the maximum possible capacity at signal-controlled intersections. Therefore, all signal-controlled nodes in the traffic network of the Düsseldorf metropolitan area were systematically simulated and evaluated using the microscopic traffic simulation tool SUMO.

This analysis is the basis for an approach to determine system-wide and long-term effects of AVs from local microscopic observations. The microscopic transport simulation SUMO is utilized to derive realistic flow capacities for signal-controlled intersections. These capacities are transferred to the mesoscopic transport simulation MATSim and then systematically adjusted to parameterize the effect of future infrastructure and vehicle technologies.

The analysis shows that defensively parameterized AVs – as foreseen in the umbrella project of this research – may decrease the maximum possible traffic at signal-controlled intersections. Moreover, the simulation runs reveal that capacity at these intersections decreases almost linearly with a growing share of AV. The observed class means of the maximum traffic flow rate of 100 % CV ($q = 1170\,\text{veh}/\text{h}$) and 100 % AV ($q = 1010\,\text{veh}/\text{h}$) differ noticeably by over 10 %. Similar results can be found in the literature, e. g. in [16].

Because the traffic flow at signal-controlled intersections is interrupted at more or less regular intervals, car-following can only occur at specific, relatively short time intervals. For this reason, a freeway section was analyzed with the same varying shares of CV and AV in the second part of this analysis. The traffic flow corresponded to the theoretical maximum traffic volume of $q_{\max} = 5800\,\text{veh}/\text{h}$ for a comparable road section without heavy goods traffic. In contrast to signal-controlled intersections, a minimum traffic flow rate $q = 5096\,\text{veh}/\text{h}$ is found for a uniform share of both vehicle types (i. e. 50 % AV and 50 % CV). The maximum traffic flow rate ($q \approx 5800\,\text{veh}/\text{h}$) emerges for both AV-only and CV-only traffic; the deviations between these configurations are negligible. The simulation results of the maximum traffic flow q can be approximated by a third-order polynomial fit.

As a next step, it is planned to model intelligent traffic infrastructure concepts directly in SUMO and transfer adapted capacities to the mesoscopic simulation MATSim. This allows for an improved investigation of such technologies and provides the basis for an economic evaluation of an infrastructure roll-out. With access to the national test-field in Düsseldorf, we also plan to test some of these intelligent infrastructure concepts in practice and validate the obtained simulation results.

ACKNOWLEDGMENT

Our special thanks go to the "KoMoDnext" project founded by the German Federal Ministry of Transport and Digital Infrastructure for kindly providing the demand data set.

References

[1] Maria Alonso Raposo, Biagio Ciuffo, Fulvio Ardente, Jean Philippe Aurambout, Gianmarco Baldini, Robert Braun, Panayotis Christidis, Aris Christodoulou, Amandine Duboz, Sofia Felici, et al. The future of road transport. Technical report, Joint Research Centre (Seville site), 2019.

[2] Nijat Rajabli, Francesco Flammini, Roberto Nardone, and Valeria Vittorini. Software verification and validation of safe autonomous cars: A systematic literature review. *IEEE Access*, 9:4797–4819, 2021.

[3] Dhanoop Karunakaran, Stewart Worrall, and Eduardo M. Nebot. Efficient statistical validation with edge cases to evaluate highly automated vehicles. *CoRR*, abs/2003.01886, 2020.

[4] Juozas Vaicenavicius, Tilo Wiklund, Austė Grigaitė, Antanas Kalkauskas, Ignas Vysniauskas, and Steven Keen. Self-driving car safety quantification via component-level analysis. *SAE Intl. J CAV*, 4:35–45, April 2021.

[5] G Baldini. Testing and certification of automated vehicles (av) including cybersecurity and artificial intelligence aspects. 2020.

[6] Chih-Hong Cheng and Rongjie Yan. Continuous safety verification of neural networks, October 2020.

[7] Gurcan Comert, Mashrur Chowdhury, and David M. Nicol. Assessment of system-level cyber attack vulnerability for connected and autonomous vehicles using bayesian networks. *CoRR*, abs/2011.09436, 2020.

[8] Shahida Malik and Weiqing Sun. Analysis and simulation of cyber attacks against connected and autonomous vehicles. In *2020 International Conference on Connected and Autonomous Driving (MetroCAD)*, pages 62–70. IEEE, February 2020.

[9] Rony Komissarov and Avishai Wool. Spoofing attacks against vehicular fmcw radar, April 2021.

[10] Steven Uytsel. Testing autonomous vehicles on public roads: Facilitated by a series of alternative, often soft, legal instruments. In Steven Van Uytsel and Danilo Vasconcellos Vargas, editors, *Autonomous Vehicles*, Perspectives in Law, Business and Innovation, pages 39–64. Springer, May 2021.

[11] Markus Maurer, J. Gerdes, Barbara Lenz, and Hermann Winner. *Autonomous Driving. Technical, Legal and Social Aspects.* 05 2016.

[12] Darshan Gadginmath and Pavankumar Tallapragada. Data-driven distributed intersection management for connected and automated vehicles, July 2020.

[13] Tanja Niels, Klaus Bogenberger, Nikola Mitrovic, and Aleksandar Stevanovic. Integrated intersection management for connected, automated vehicles, and bicyclists. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, September 2020.

[14] Youssef Bichiou and Hesham A. Rakha. Developing an optimal intersection control system for automated connected vehicles. *IEEE Trans. Intell. Transp. Syst.*, 20(5):1908–1916, 2019.

[15] Joyoung Lee and Byungkyu Park. Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment. *IEEE Trans. Intell. Transp. Syst.*, 13(1):81–90, 2012.

[16] Scott Le Vine, Alireza Zolfaghari, and John Polak. Autonomous cars: The tension between occupant experience and intersection capacity. *Transportation Research Part C: Emerging Technologies*, 52:1–14, 03 2015.

[17] Mohamed Berrazouane, Kailin Tong, Selim Solmaz, Martijn Kiers, and Jacqueline Erhart. Analysis and initial observations on varying penetration rates of automated vehicles in mixed traffic flow utilizing sumo. In *ICCVE*, pages 1–7. IEEE, 2019.

[18] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wiessner. Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582. IEEE, November 2018.

[19] Kay W Axhausen, Andreas Horni, and Kai Nagel. The multi-agent transport simulation matsim, 2016.

[20] S. Krauß. *Microscopic modelling of traffic flow: Investigation of Collision Free Vehicle Dynamics.* PhD thesis, University of Cologne, 1998.

[21] Rainer Wiedemann. Simulation des Straßenverkehrsflußes. Technical report, Institut für Verkehrswesen, Universität Karlsruhe, 1974. Heft 8 der Schriftenreihe des IfV, in German.

[22] K. I. Ahmed. *Modelling Drivers' Acceleration and Lane-Changing Behavior.* PhD thesis, MIT, 1999.

[23] 75 years of the fundamental diagram for traffic flow theory - greenshields symposium, 2011.

[24] K. Mattas, M. Makridis, P. Hallac, M. A. Raposo, C. Thiel, T. Toledo, and B. Ciuffo. Simulating deployment of connectivity and automation on the antwerp ring road. *IET INTELLIGENT TRANSPORT SYSTEMS*, 12(9):1036–1044, 2018.

[25] Steven E. Shladover, Dongyan Su, and Xiao-Yun Lu. Impacts of cooperative adaptive cruise control on freeway traffic flow. *Transportation Research Record*, 2324(1):63–70, 2012.

[26] *HBS – Handbuch für die Bemessung von Straßenverkehrsanlagen.* Number ISBN: 978-3-86446-103-3. FGSV Verlag – Der Verlag der Forschungsgesellschaft für Straßen- und Verkehrswesen, 2015.

[27] H. Werdin, H. Honermann, R. Laube, and I. Belopitov. *Verkehrsqualität und Leistungsfähigkeit auf Autobahnen.* 2004.

SUMO User Conference 2021
https://doi.org/10.52825/scp.v2i.92

Published: 29 Jun. 2022

A Study of Applying Eco-Driving Speed Advisory System on Transit Signal Priority

Hsuan-Chih Wang [1]
[1] Department of Transportation and Communication Management Science, National Cheng Kung University, Tainan, Taiwan (R.O.C).
hcwang71026@gmail.com

Abstract

Transit Signal Priority (TSP) has long been seen as a cost-effective way to reduce bus delays at intersections. With Connected Vehicle (CV) technology, speed advisory system guides buses to pass intersections in an energy-saving way. The integration of TSP and speed advisory may reduce bus delays and enhance energy consumption performances. This study proposed a system of integrating eco-driving speed advisory on TSP under CV environment. A TSP strategy based on intersections passing probability is designed. In addition to signal priority, this study designed and implemented an eco-driving speed advisory algorithm. A real electric bus route in Tainan City, Taiwan is used for the case study. Intersection layout and traffic related parameters are established in microscopic traffic simulation software SUMO (Simulation of Urban Mobility) to verify the effectiveness of the proposed model. The results provide an insight of how cooperation between signals and vehicles can enhance performances of energy consumption and signal-incurred traffic delays.

1 Introduction

In metropolitan cities, transit vehicles serve as a type of transportation means to move a large number of passengers efficiently. Increasing transit system ridership has also been viewed as one of the potentially effective ways to diminish traffic congestion. Transit Signal Priority (TSP) is a collection of techniques that provide transit vehicles moving through signalized intersections smoothly. A TSP system has been seen as a cost-effective method to improve regional mobility. With the adjustment of signal settings, TSP not only reduce the delay of transit buses but enhance operation reliability (Smith et al., 2005).

In the past decades, many studies have proposed various TSP strategies. In general, they can be classified into two types: passive control or active control. Based on off-line information of bus routes and ridership patterns, passive TSP is a fixed time signal plan that favors bus operation. The objective functions of such system include maximizing the progression band or minimizing total bus delays. Most

Passive TSP strategies are developed for high bus volume systems such as Bus Rapid Transit (BRT) since frequent TSP requests may have negative impacts on side street traffic and negate the benefits of priority control (Ma et al., 2007, Cheng et al., 2015, Kim et al., 2019). Active TSP (ATSP) requires the placement of detectors to track bus real-time location information and calculates the adjustment of traffic signal plans accordingly. ATSP control can be further divided into two categories: unconditional and conditional. Unconditional ATSP conducts green time extension or red time truncation without any condition (Lee et al., 2005). Despite the effectiveness of the unconditional ATSP on improving bus efficiency, it may disturb original signal plan and deteriorates non-prioritized traffic dramatically if limitations on the amount of priority time are not considered. Thus, some studies deal with the problems by limiting activation times of TSP upon frequent request or buses ahead of schedule condition (Dion et al., 2002, Lee et al., 2005).

It is noted that two-way communications between buses and Road Side Units (RSUs) installed on traffic signals are established under Connected Vehicle (CV) environment. In such communication On-Board Units (OBUs) can give bus driver speed guidance to coordinate traffic signal plan. Deployment of TSP system with CV technology in BRT system is presented in a previous study (Wang et al., 2014). However, the study only provides signal count down information for drivers. To decrease the times of priority requests, a study that integrates Green Light Optimized Speed Advisory (GLOSA) into TSP strategy is proposed (Seredynski et al., 2019). The performance results indicate that GLOSA can partially replaces TSP under certain traffic conditions. However, the study assumes buses riding on exclusive lane, the effects of mixed traffic circumstance remain unknown.

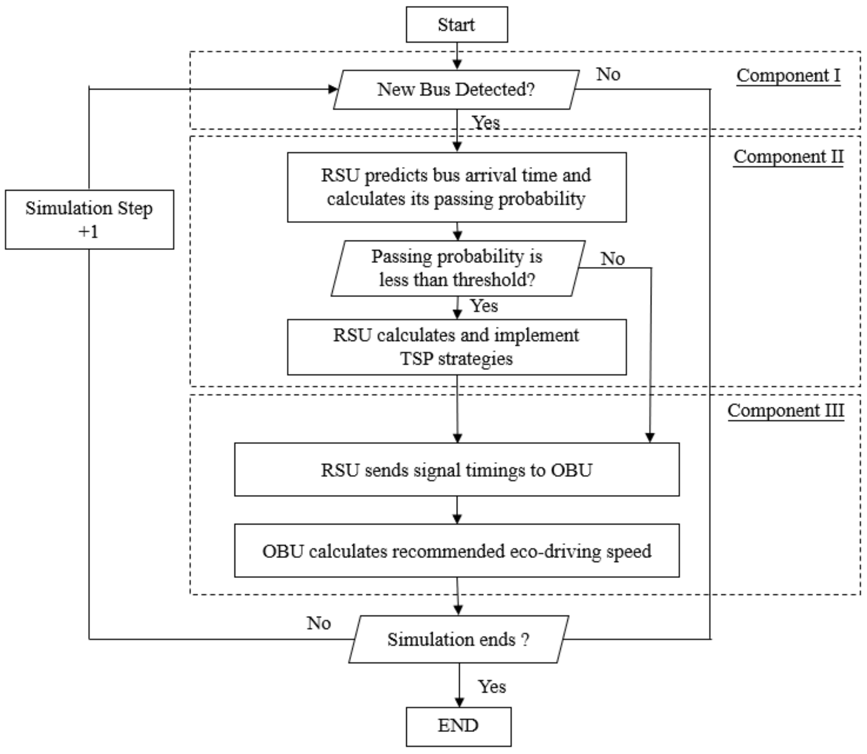
With the advent of electric vehicles, electric buses have gradually been adopted due to the advantages of low noise and air pollution. However, in most cases, electric transit vehicles can charge electricity only when returning to their terminals. It would be a problem if the vehicles run out of energy during operation. Therefore, approaches to enhance the efficiency of electric energy consumption are important. Two previous studies compare the efficiency of electric consumption of two electric bus routes (Institute of Transportation, 2014; Institute of Transportation, 2016). One is riding on expressway and the other is on general road. The results show bus riding efficiency on expressway is better than on general road due to less traffic disturbances from traffic signals. In general road, traffic signals are key control facilities which can dramatically affect the efficiency of traffic operation. This study proposes a Cooperative ATSP method (ATSP-C) which contains ATSP and eco-driving speed guidance. The objective is to improve electric transit vehicles' riding efficiency and energy consumption. This study formulates TSP and eco-driving speed advisory model respectively. A microscopic traffic simulation software Simulation of Urban Mobility (SUMO) is applied to verify the effectiveness of the proposed models.

This study is organized as follows: The assumptions and the key features of ATSP-C in the next section, followed by description of ATSP-C flow chart and model formulation. The simulation platform, simulation case study and evaluation results are then presented. Finally, conclusions and suggestions are discussed in the last section.

2 Methodology

2.1 Control framework of ATSP-C

The flow chart of ATSP-C is presented in Figure 1. The whole control framework contains three components. After the system detects buses are approaching, as shown in component I, the RSU computes intersection passing probability according to predicted bus arrival time. If the probability is less than the threshold, TSP calculation will be activated. Finally, the signal timing information will send to OBUs to perform eco-driving speed advisory computation in component II. Both RSU and OBU perform once for one bus. OBU gets latest signal timing information to perform eco-driving speed calculation.


Start
New Bus Detected?
No
Component I
Yes
Component II
RSU predicts bus arrival time and calculates its passing probability
Simulation Step +1
Passing probability is less than threshold?
No
Yes
RSU calculates and implement TSP strategies
Component III
RSU sends signal timings to OBU
OBU calculates recommended eco-driving speed
No
Simulation ends ?
Yes
END
Figure 1 The flowchart of ATSP-C

2.2 Assumptions

For model simplicity, the following assumptions are made in this study:
1. All transit vehicles are installed with On-Board Units (OBUs).
2. All traffic signals are installed with Road-Side Units (RSUs).
3. Overtaking and lane-changing behavior are not taken into consideration.
4. Bus drivers fully comply with the advisory driving speed guidance.
5. One bus request can be served per cycle on a first-come, first-served basis.
6. The effects of dwell times at bus stop are not taken into consideration.

2.3 Calculation of transit vehicle's passing probability at an intersection

The make TSP control and eco-driving advisory effective, precise prediction of buses arrivals at intersections is essential. Since transit vehicles run under mixed traffic circumstance, the prediction results may be biased if a simple calculation of arrival time (i.e. distance to the intersection divided by bus current speed) is adopted. To take the uncertainty nature of arrival time into consideration, this study assumes the arrival time of bus b at intersection i is a random variable $T_{i,b}^{B}$, which follows a normal distribution with standard error σ_i.

This idea is demonstrated in Figure 2. At time t_0, there is a bus b with a speed v_b riding towards intersection i. We assume the bus will maintain its speed to the stop bar, thus the mean value $E(T_{i,b}^{B})$ is calculated by the distance to intersection i (L_i) divided by bus current speed (v_b). The standard error (σ_i) is a function related with bus current distance to stop bar. Hence, we can calculate passing probability with cumulative density function of normal distribution by integrating green duration.

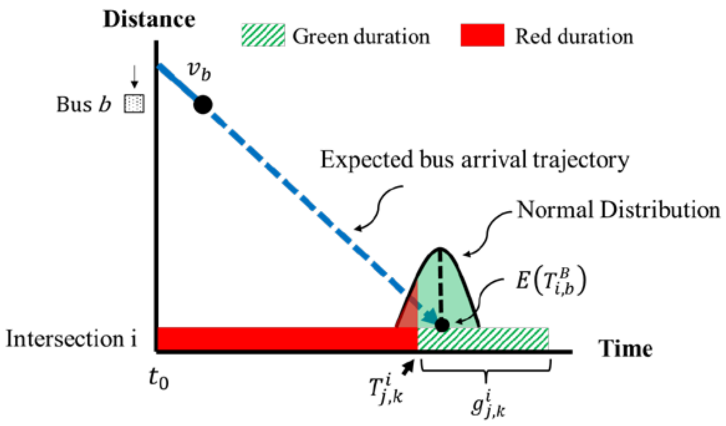


Figure 2 Concept of bus arrival formulation

2.4 TSP strategy

As shown in Figure 3, a rule-based TSP strategy is proposed. According to the arrival time range, the arrival of each bus is classified into four cases. For transit vehicles that arrives before start time of green phase, they are classified as case 1. Case 2 and 3 are buses that earliest and latest arrival time are in different phases. Case 4 refers to those arrive after green phase. The planning horizon is two cycle length.

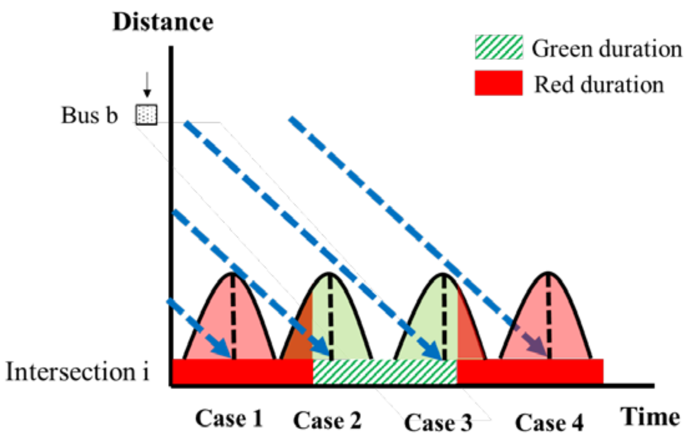


Figure 3 Cases of TSP strategies

As shown in Figure 4, two phase adjustment technique are used: green extension of intersection i, phase j and cycle k ($y^{I}_{\mathrm{i,j,k}}$) and red truncation of intersection i, phase j and cycle k ($y^{II}_{\mathrm{i,j,k}}$). For case 1 and 2, the TSP adopts red truncation. For case 3 and 4, both green extension and red truncation are used.

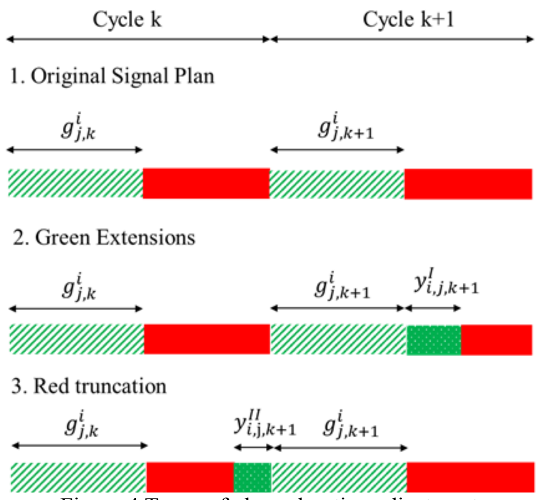


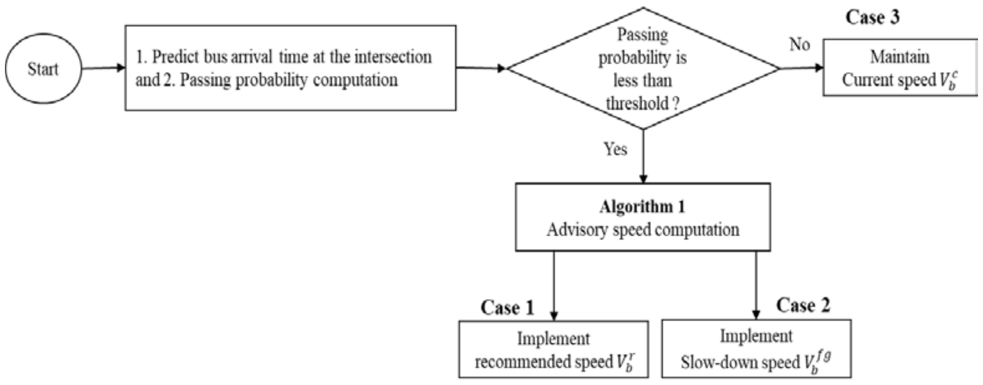
Figure 4 Types of phase duration adjustment

The process of determining priority values is listed as follows:
Step 1: The system produces arrival time range and identify the case.
Step 2: Initialize $y^{II}_{\mathrm{i,j,k}}$ and $y^{I}_{\mathrm{i,j,k}}$ as 0.
Step 3: Increment $y^{II}_{\mathrm{i,j,k}}$ or/and $y^{I}_{\mathrm{i,j,k}}$.
Step 4: Compute passing probability.
Step 5: Repeat **Step 3** to **Step 4** ***until*** one of the following conditions is fulfilled: (1) Passing probability is larger than a threshold or (2) priority value equals maximum priority value (y_{max}).

Step 6: If priority value equals maximum priority value (y_{max}) but passing probability does not fulfill the threshold, minimization red duration strategy, which truncates red duration y_{max} seconds, is adopted.

2.5 Eco-driving speed advisory

Eco-driving speed advisory aims to provide a driving speed advisory that minimizes acceleration and deceleration. The logic is formulated as a decision tree demonstrated in Figure 5. At the beginning, the OBU calculates passing probability based on signal timing plan and expected arrival time. If the probability is larger than threshold, then the system will recommend maintaining its current speed (V_b^c). Otherwise, it will compute eco-driving speed. If any solution exists, it will provide recommended eco-driving speed (V_b^r), or it will give a continuous slow down speed (V_b^{fg}).


Start
1. Predict bus arrival time at the intersection and 2. Passing probability computation
Passing probability is less than threshold ?
No
Case 3
Maintain Current speed V_b^c
Yes
Algorithm 1
Advisory speed computation
Case 1
Implement recommended speed V_b^r
Case 2
Implement Slow-down speed V_b^{fg}
Figure 5 Flow chart of Eco-driving speed advisory

The detail of the algorithm is show in Algorithm 1. The goal is to find a speed v which fulfills the threshold of passing probability. Line 1 is initialization of variables. Note that maximum passing probability is initialized as passing probability threshold. Line 2 to line 3 is the process of enumerating speed variable for speed list (i.e. $[0.7v_{lim}, 1.1v_{lim}]$). The lower and upper bound of speed list are set as 70% and 110% multiple road speed limit (v_{lim}). Through line 4 to 6, the passing probability of each speed is computed and replace maximum one if the value is larger than it. In line 7, the recommended eco-driving speed (V_b^r) is returned.

```
Algorithm 1: Finding an advisory speed
Input: bus order ($b$), bus speed ($v_b$), intersection ($i$),
distance to intersection i ($L_i$), signal plan of intersection i ($SignalPlan[i]$)
Output: advisory driving speed ($V_b^r$)
1.  Initialization: set passProb = 0, $OptimalSpeed = v_b$,
Max_PassProb = PassProbThreshold
2. For $v \in [0.7v_{lim}, 1.1v_{lim}]$ do :
3.     passProb = Passing_Probability_Computation(input: $b, v_b, L_i, SignalPlan[i]$)
4.     If  passProb > Max_PassProb then:
5.         Max_PassProb = passProb
6.         $V_b^r = v$
7. Return $V_b^r$
```

If Algorithm 1 fails to find a feasible eco-speed, the system will implement “slow-down speed” strategy (Case 2 in Figure 5). With bus initial speed (v_b), slow-down speed ($V_{b,t}^{fg}$) at time *t* can be calculated through equation 1 to 3.

$$L_i = v_b t + \frac{1}{2}at^2 \tag{1}$$

$$a = \frac{(L_i - V_0 T) \cdot 2}{T^2} \tag{2}$$

$$V_{b,t}^{fg} = V_{b,t-1}^{fg} + a \tag{3}$$

3 Experiment design

3.1 Simulation platform and settings

This study chooses Simulation of Urban Mobility (SUMO) as simulation platform (Lopez et al., 2018). A real intersection located in Tainan City, Taiwan is adopted. It’s a four-leg intersection with left-turn bay (Figure 6). The signal is four phases including a protected left-turn phase. The phase order and respective direction is illustrated in Table 2. There are two electric bus routes riding through the intersection: Bus 1 and Bus 2. The headway of Bus 1 is 900 seconds, and Bus 2 is 1200 seconds.

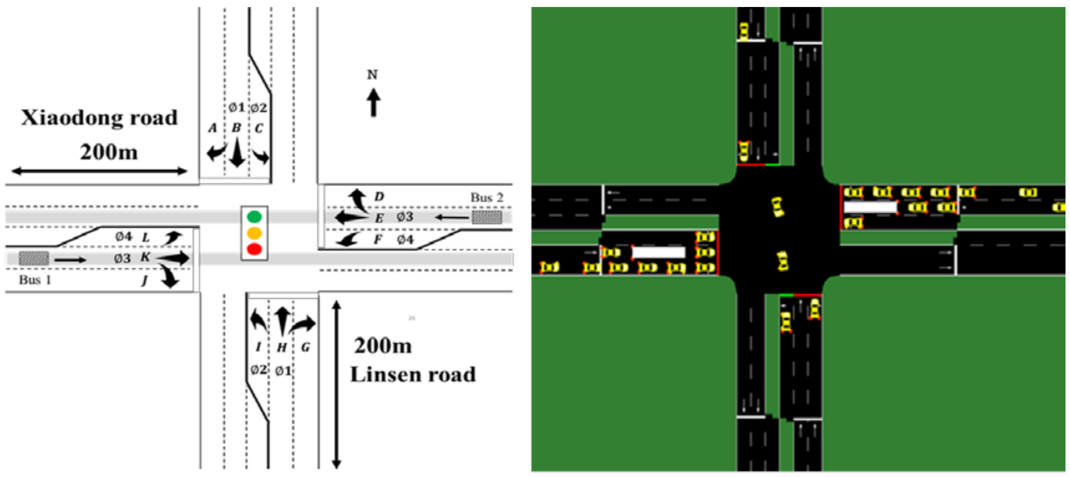


Figure 6 Layout of the intersection

The traffic volumes (veh/hr) are listed in Table 1. To test the effectiveness under different traffic congestion level, two different V/C ratio (i.e. 0.5 V/C ratio and 0.9 V/C ratio) are designed. Background signal timing settings are listed in Table 2. The green duration is 34, 11, 43, 12 seconds with respect to phase 1, 2, 3 and 4. The yellow and all-red time are 3 and 2 seconds.

Table 1 Traffic volume of each direction

Movement (veh/hr)	A	B	C	D	E	F
V/C = 0.9	45	543	83	80	1147	131
V/C = 0.5	23	272	42	40	574	66
Movement (veh/hr)	*G*	*H*	*I*	J	K	L
V/C = 0.9	82	530	161	129	814	55
V/C = 0.5	41	265	81	65	407	28

Table 2 Signal timing parameters

Phase	Ø1	Ø2	Ø3	Ø4
Movement	A, B, H, G	C, I	D, E, K, J	L, F
Green (sec)	34	11	43	12
Yellow (sec)	3	3	3	3
All-red (sec)	2	2	2	2
Cycle length (sec)	120			

The parameters of the electric buses are listed in Table 3. SUMO supports electric vehicle settings and energy consumption model. With these parameters set as input, this study retrieves corresponding energy consumption results.

Table 3 Parameters of electric transit vehicles

Parameter	Value
Length (m)	12
Width (m)	2.55
Height (m)	3.5
empty weight (kg)	11800
Full-loaded weight (kg)	16000
Average power (Kw)	206
Nominal electric capacity (Kwh)	200

3.2 Experiment scenarios

Two experiments are designed in this study:

1. Activation of TSP and eco-driving speed advisory.
2. Sensitivity test of maximum priority time.

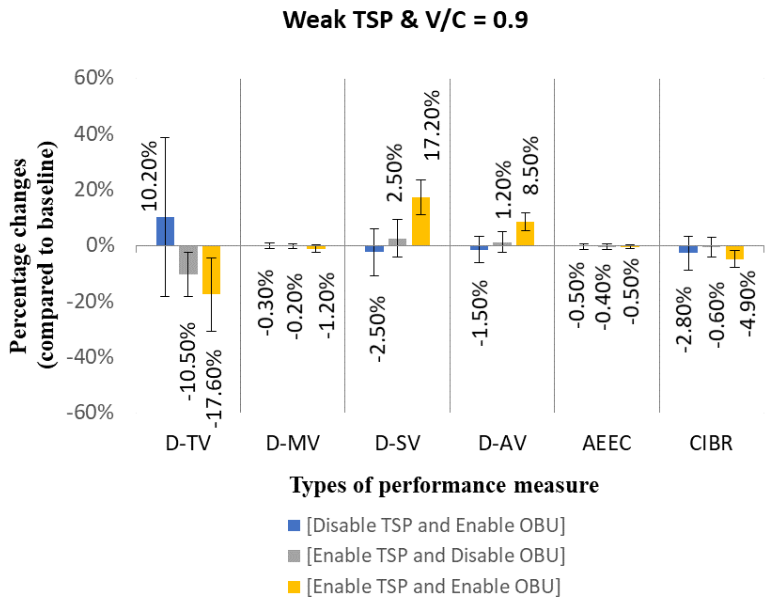
The purpose of the first experiment is to evaluate the results of various combination of activation on TSP and eco-driving speed advisory. Here we design four scenarios: Background signal plan (Disable TSP and Disable OBU), ATSP-C with only eco-driving speed advisory (Disable TSP and Enable OBU), ATSP-C with only TSP (Enable TSP and Disable OBU), ATSP-C with both TSP and OBU (Enable TSP and Enable OBU). "Background signal plan" scenario serves as the baseline. The second experiment is the sensitivity test of maximum priority time. The study designed two cases: Weak TSP and Strong TSP (i.e. weak and strong maximum priority time) to evaluate the effects. The maximum priority time of Weak TSP is set as 10 seconds, and Strong TSP is set as 20 seconds. Both experiments are tested under two V/C ratios: 0.9 and 0.5. The simulation time of each scenario is 2 hour and 5 times run, excluding 5 minutes warm-up time. A random seed is set as 8.

3.3 Performance measures

Four intersection-based performance measures are introduced: average Delay of Transit Vehicles (D-TV), average Delay of Main-street Vehicle (D-MV), average Delay of Side-street vehicle (D-SV) and average Delay of All Vehicle (D-AV). These measures aim to evaluate the operation efficiency of the intersection. Two measures are designed for bus riding: Average Electric Energy Consumption (AEEC) and Comfort Index of Bus Riding (CIBR). AEEC is the average electric energy usage of each transit vehicle. CIBR is calculated by averaging absolute acceleration and deceleration rate of each bus, which is used for indicating the level of comfort.

4 Results analysis

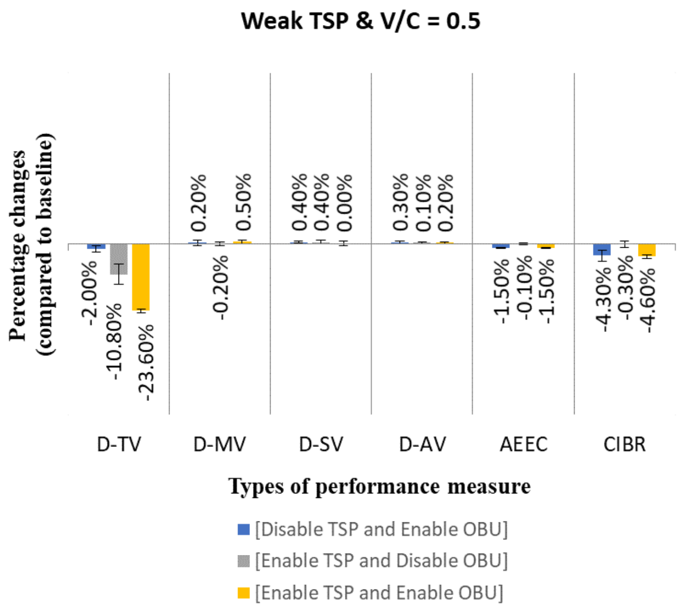
Figure 7 illustrates the results of weak TSP experiment under 0.9 V/C ratio. Compared to baseline, the delays of transit vehicles (D-TV) decrease 10.05% in “Enable TSP and Disable OBU” and 17.6% in “Enable TSP and Enable OBU” under 0.9 v/c ratio, both are statically significant at p-value < 0.5. An increase in D-TV (10.2%) is presented in “Disable TSP and Enable OBU” scenario. One possible reason for the result is that effects of queueing vehicle is not involved in the speed advisory algorithm, OBU may give less precise speed advice under near-saturated traffic and results in the increase in D-TV. Also, the increase is not statically significant at 0.5 p-value.



Note: Error bars show 95% confidence intervals

Figure 7 Simulation results - Weak TSP with 0.9 v/c ratio

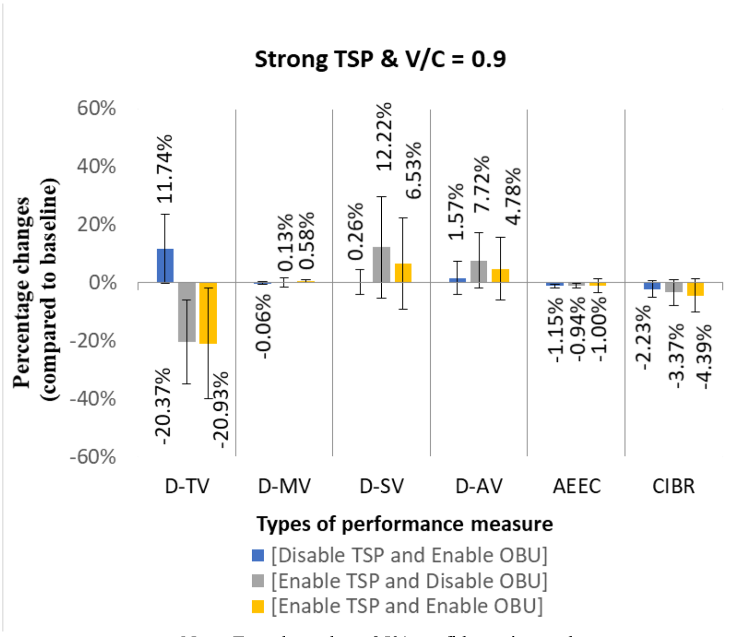
For 0.5 v/c ratio case shown in Figure 8, significant decreases in D-TV are presented in "Disable TSP and Enable OBU" scenario. In contrast to 0.9 v/c ratio, the D-TV decreases from 10.02% to -2.00%. This represents the eco-driving speed advisory has greater positive effects on decreasing delays under lighter traffic volume. The change in D-TV is -23.60% in "Enable TSP and Enable OBU" scenario, which is more than the changes in 0.9 v/c ratio (-17.60%). At the same time, the delay changes in side-street vehicle (D-SV) remains nearly unchanged. The results indicate the cooperation between TSP and OBU can enhance delays of transit vehicles without dramatically deteriorating non-prioritized traffic under 0.5 v/c ratio.



Note: Error bars show 95% confidence intervals

Figure 8 Simulation results - Weak TSP with 0.5 v/c ratio

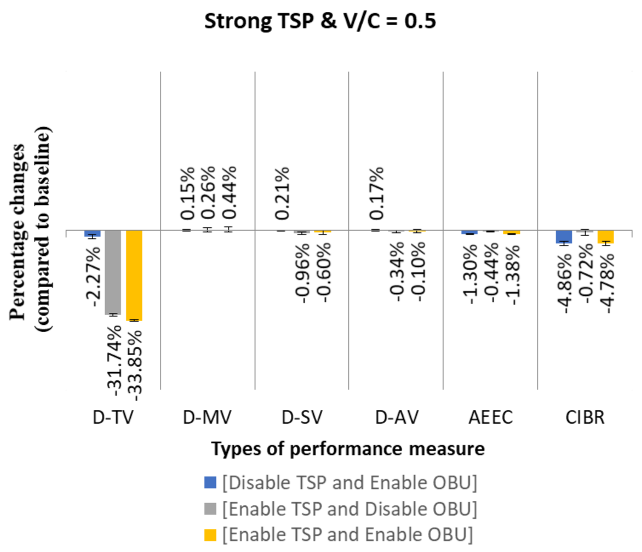
Figure 9 demonstrates the results of Strong TSP experiment under 0.9 v/c ratio. Delays of transit vehicles in"Enable TSP / Disable OBU" decrease from -10.05% to -20.37%, and "Enable TSP / Enable OBU" decrease from -17.60% to -20.93% compared to Weak TSP experiment. It is evident that strong TSP can significantly improve delays of the transit vehicles. However, negative impacts on side-street traffic (i.e. D-SV) of "Enable TSP and Disable OBU" increase from 2.5% to 12.22%, while "Enable TSP / Enable OBU" decrease from 17.20% to 6.53%. With facility of larger priority time, TSP can enhance delays of buses but deteriorate the delay performances of non-prioritized traffic. However, due to speed guidance provided by OBU, it ameliorates the negative impacts of TSP without increase delays of transit vehicles, which brings about the decrease in D-SV in "Enable TSP / Enable OBU" scenario.



Note: Error bars show 95% confidence intervals

Figure 9 Simulation results - Strong TSP with 0.9 v/c ratio

For 0.5 v/c ratio shown in Figure 10, D-TV decreases from -10.8% to -31.74% in “Enable TSP / Disable OBU” and from -23.6% to -33.85% in “Enable TSP / Enable OBU” compared to Weak TSP (Figure 8), while D-SV has little changes compared to baseline (only about -0.7%). This shows that in medium traffic volume, since TSP provides sufficient favors for buses, the delays of transit vehicles are near the same regardless of the activation of OBU.



Note: Error bars show 95% confidence intervals

Figure 10 Simulation results - Strong TSP with 0.9 v/c ratio

5 Conclusions

This study proposed an integrated system named ATSP-C, which focuses on exploring the potential benefits on integrating eco-driving speed advisory system and transit signal priority. This study assumes electric transit vehicles riding under mixed circumstance traffic and designs several performance measures and simulation-based experiments to evaluate proposed model. Experiment results indicate ATSP-C can enhance electric consumption efficiency and riding comfort in all cases. In near-saturated traffic, standalone eco-driving advisory yields worser delays of transit vehicles because of inaccurate arrival time prediction. In comparison to only TSP strategy, a significant amelioration of the adverse impacts on side-street traffic can be found when both TSP and OBU are applied, which shows great benefits of the signal-vehicle cooperative control.

Some limitations of the study may improve in future works. For eco-driving speed advisory model, the experiment results suggest that speed advisory may worsen delay performance during near-saturated traffic since levels of traffic congestion and queue saturation time are not taken into consideration. Future studies can involve traffic queueing such as Webster delay model into the algorithm to enhance the performance. The assumption of bus drivers fully comply with the speed guidance is hard to implement real world, more research on the effects of rates of compliance should be done. Finally, this study adopts rule-based TSP strategy, which may not be optimal in different traffic scenarios. Future studies can make detailed exploration on TSP designs.

Data Availability

The data is generated by a simulation program, and part of the program files included in this study are available by sending request to the author.

Conflicts of Interest

The author declares no potential conflicts of interest with respect to the research, authorship and/or publication of this article.

References

Cheng, Y., Yang, X., & Chang, G.-L. (2015). *A bus-based progression system for arterials with heavy transit flows*. Paper presented at the Transportation Research Board 94th Annual Meeting, Washington DC, United States.

Dion, F., Rakha, H., & Zhang, Y. (2004). "Evaluation of Potential Transit Signal Priority Benefits Along a Fixed-time Signalized Arterial," *Journal of Transportation Engineering*, 130(3), 294-303.

Institute of Transportation (2014), The Cost-Benefit Analysis of Battery Electric Bus, Diesel-Electric Hybrid Bus and Diesel Bus, 102-TAA005, Institute of Transportation, Ministry of Transportation and Communication, Taiwan (R.O.C).

Institute of Transportation (2016), Operating Investigation and Performance Evaluation on Electric Bus Industry for Public Transportation, 103-MDB002, Institute of Transportation, Ministry of Transportation and Communication, Taiwan (R.O.C).

Kim, H., Cheng, Y., & Chang, G. (2019). Variable Signal Progression Bands for Transit Vehicles Under Dwell Time Uncertainty and Traffic Queues. *IEEE Transactions on Intelligent Transportation Systems, 20*(1), 109-122. doi:10.1109/TITS.2018.2801567

Lee, J., Shalaby, A., Greenough, J., Bowie, M., & Hung, S. (2005). "Advanced transit signal priority control with online microsimulation-based transit prediction model," Transportation Research Record, 1925(1), 185-194.

Ma, W., & Yang, X. (2007). *A Passive Transit Signal Priority Approach for Bus Rapid Transit System.* Paper presented at the 2007 IEEE Intelligent Transportation Systems Conference.

M. Seredynski, G. Laskaris and F. Viti, "Analysis of Cooperative Bus Priority at Traffic Signals," IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 5, pp. 1929-1940, May 2020, doi: 10.1109/TITS.2019.2908521.

P. A. Lopez et al., "Microscopic Traffic Simulation using SUMO," 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, 2018, pp. 2575-2582, doi: 10.1109/ITSC.2018.8569938.

Smith, H. R., Hemily, B., & Ivanovic, M. (2005). Transit signal priority (TSP): A planning and implementation handbook.

Wang, Y., Ma, W., Yin, W., & Yang, X. (2014). "Implementation and Testing of Cooperative Bus Priority System in Connected Vehicle Environment: Case Study in Taicang City, China," Transportation Research Record, 2424(1), 48–57.

SUMO User Conference 2021
https://doi.org/10.52825/scp.v2i.93

Published: 29 Jun. 2022

Validating a parking lot assignment method by Eclipse SUMO

Levente Alekszejenkó[1] and Tadeusz Dobrowiecki[1]

Department of Measurement and Information Systems,
Budapest University of Technology and Economics,
Budapest, Hungary
alelevente@mit.bme.hu
dobrowiecki@mit.bme.hu

Abstract

The present paper shows how a novel autonomous vehicle (AV) algorithm can be tested and evaluated by multiple simulations with different levels of refinement.

As the first AVs will appear on our roads (possibly) within the upcoming decade, novel traffic problems, like parking lot assignment, will also emerge. Parking is that kind of activity that will fundamentally change. Today drivers attempt to find parking places close to the destinations to minimize the walking time. The autonomous vehicles will drop off passengers at their very destination, and then, on their own, will seek even a distant parking place. Such change will affect human activities, city traffic, and the parking infrastructure.

To investigate this problem we need an abstract city model, because an attempt to create a detailed model of the city traffic from the perspective of decades may be misleading and may lead to false emergent conclusions. We report on such an abstract model and the obtained conclusions in the paper.

A question remains how genuine is the introduced future city model. To validate it, we propose to confront it with a detailed microscopic traffic simulation on a road network borrowed from an existing city of corresponding layout and complexity. To this end, we use Eclipse SUMO simulating the traffic of an expansive Budapest district, and show that the results obtained with the simplified abstract mathematical model stay valid.

1 Introduction

The analysis of transportation in future cities is a challenging task. The principal difficulty is how to set up simulations of the future traffic when little if any may be known about it, besides futuristic assumptions and visions.

As the penetration rate of autonomous vehicles (AVs) is foreseen to reach a considerable level within the next 20-40 years, we aimed at this horizon in our research. The arising changes in vehicle ownership, technological advances, and changing habits made it hard to estimate the future traveling demands. The practical and important question was whether it is safe to use the existing microscopic traffic simulation platforms, set up for a detailed reproduction not of the future but of the existing traffic and city road networks. With such platforms, one consequently risks that the simulation results may become a mixture of emerging plausible and difficult to discern spurious phenomena.

Our research tackled, in particular, the parking problem of AVs based on their communication and assumed intelligent decision-making capabilities. To this end, and because of the above doubts, we created a semi microscopic but much more abstracted mathematical city model. Assuming that the basic city structure (including historical downtowns, buildings, utility infrastructures, etc.) will not change in the future, an abstract city model can describe future cities as well as cities of today. It can obscure the yet unknown details (like specific road networks), but it can provide sufficient information for qualitative comparisons. With such a city model we found interesting phenomena about the parking behavior of AVs. However, the most important question was whether such a model is acceptable. In an attempt to clarify these doubts, we deploy an analogy to the downward refinement consistency of [6]. More precisely, a phenomenon discovered on the abstract level shall also be observable in a more refined model when modeling the physical reality on different levels of abstraction. However, it might not be true the other way around.

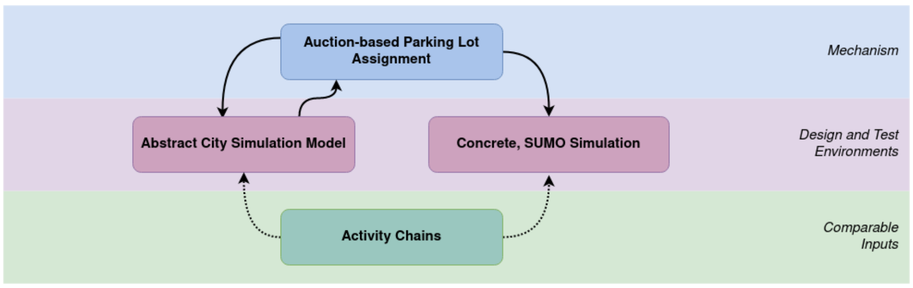

Auction-based Parking Lot Assignment
Mechanism
Abstract City Simulation Model
Concrete, SUMO Simulation
Design and Test Environments
Activity Chains
Comparable Inputs

Figure 1: Overview of the validation process applied to the parking problem of AVs.

Thus, the point is that a more refined model permits the validation of a less detailed one. A feasible choice of such validating model of a city may be one of the available traffic simulation platforms. The open-source microscopic traffic simulation tool Eclipse SUMO [18] seems to be an efficient choice for lower-level validation tasks. To make both models comparable, the Eclipse SUMO model should describe a real city with dimensions and complexity similar to that of the abstract model. Figure 1 shows the interaction of the two models when applied to the same simulation task. In the following, in Section 2, we present why and how the AV parking problem might be discussed. After that, Section 3 contains a brief overview of the works already done in the topic of city modeling, then, in Section 4, we shortly review the auction-based parking assignment method used as a pilot problem. Then in Section 5 the composition and results obtained with the abstract mathematical model are presented. In Section 6 we set up the comparable SUMO simulation, and in Section 7 the results obtained with the two models are compared.

2 Autonomous Vehicles, Parking, and the Advantages of an Abstract City Model

One of the fundamental questions facing a private car owner is where to park. Traditionally most drivers attempt to minimalize the walking distance when looking for a suitable parking place. Therefore, spiraling around the destination with an increasing radius [17] might be an effective strategy. Unfortunately, it can also be quite time-consuming [20] as well. The AVs, by definition, will drive without any human supervision, so they may find a suitable place to park on their own after delivering passengers to their destinations. Here, an important issue regarding the ownership and the utilization of the future AVs pops up. It was supposed earlier that AVs would be shared as a Mobility as a Service (MaaS) solution. However, due to the experience of the pandemic, with social distancing and disinfection problems, moreover, due to other current trends [21], a significant part of the AVs is now expected to become private property, at least for a time. Hence, we assume in our research that the AVs will be either privately owned or shared among a socially closely related group of people. The parking problem of such AVs differs greatly from the MaaS solution. After dropping off the last passenger, a privately owned AV is not speeding to the location of the next order but seeks a place to wait for being resumed, somehow optimizing the actual and the future traffic exposure. Finding an optimal parking lot is thus a kind of intelligent decision that has to be studied profoundly. As a solution to this problem, an auction-based method was proposed in [3].

To obtain a clear picture of the efficacy of the proposed parking strategy, a simple but expressive mathematical city model is sufficient. The parking infrastructure, the traveling activities, and the destination geolocations are the most relevant aspects to be modeled. Any more details (e.g. modeling the traffic flow, the interaction of the vehicles, and specific road network-related effects) would only obscure the results.

The abstract model guarantees that statistically similar city models (e.g. with similar shape and dimensions, with alike parking facility placement) can be easily defined and generated, making the human activity patterns and the temporal parking lot usage comparable by analyzing statistical averages and variances.

3 Related Literature

There are plenty of reasons to model cities. According to the short but comprehensive summary of [8], one of the most important reasons is economic research, including housing costs and firm localization. City models are also necessary for numerous engineering principles, e.g. city planners use models to evaluate new development decisions. With the emergence of powerful computers, three dimensional (3D) models also got available. Usage of 3D models ranges from shadow-casting estimations through energy demand forecasting to analyzing radio-wave propagation [9].

The level of detail of such city models depends on applications and covers a wide range. For fundamental economic research, mathematical formulas may describe a city [5]; meanwhile, there are 3D models that cover a whole city and have *centimeter* precision [15]. The level of detail of microscopic traffic simulations, such as Eclipse SUMO [18], is in-between. They may not be mathematically formalized, yet they abstract many unnecessary details.

A recent trend in microscopic traffic simulation is to model drivers or AVs as intelligent agents, and traffic simulators were extended or coupled to suitable artificial intelligence (AI) system components (e.g. Eclipse SUMO [18] was coupled to JADE, a multi-agent experimental platform [22], furthermore as an open-source program, SUMO supports the integration of intelligent agent simulators, like e.g. in [4]). Such tools may require considerable computing power [16] and were used in the practical research usually limited to simple AV scenarios based on single road segments, ramps, or intersections, aiming at the local operational decision making, like car following, simple platooning, etc.

Evaluating decision making spanning the whole city and 24 hours of activities, including e.g. repeated looking for parking lots, demands more complex scenarios. In the paper, we use two models to evaluate an extensive auction-based parking assignment scenario. The first model is abstract, using only mathematical formulations to simulate a city. The second is a detailed model, a concrete SUMO scenario consistent with the first model. SUMO is involved in many studies (e.g. [10] or [14]) focusing on optimizing classical, human-driven vehicle parking. The `PyPML` toolkit [14] even makes SUMO more convenient to assess parking-related problems.

4 Auction-based Parking Lot Assignment

The pilot problem was to optimize parking decisions of the autonomous vehicles. As we assume, after delivering passengers, AVs will not be allowed to remain in the traffic circulation; therefore, AVs seek parking lots. However, the selected parking lot shall not be too far to guarantee in-time returning to the passengers. As [3] points out, coordinated parking lot assignment can be advantageous, as it reduces the number of required parking lots and shortens the time and the distance traveled until finding a suitable parking place.

Here we focus on the validation of an auction-based parking lot assignment method introduced in [3] and elaborated in [2]. We summarize it shortly in the following. For simplicity, we assume that every simulated vehicle is an AV, they are (as mentioned earlier) privately owned and each AV participates in the proposed parking assignment mechanism. An AV shall take part in the auction mechanism after it has delivered its passengers to their destinations.[1] An AV shall participate in *rational auctions*. An auction for a parking lot is *rational* if the bid parking lot is closer than the cheapest or cheaper than the closest alternative.

Every parking lot shall act as an auctioneer, selling its free places. Starting prices can be set by either the parking lot operators or the municipalities. At each auction round, the parking prices increase by a small amount. The auctioneers, the parking lots, are asking the participants, a set of AVs, in every auction round whether they bid or not. AVs calculate the u_i utility value for the ith parking lot, according to equation (1), to determine their answer.

$$u_i = c_{fp} \cdot p_i + 2 \cdot (1 - c_{fp}) \cdot d_i \tag{1}$$

An AV shall consider the d_i distance and the p_i monetary costs to park at the ith parking lot[2]. Here, p_i includes the (current) parking fee as well as the distance-related costs to get to this parking lot (fuel or energy prices, amortization, etc.). The free-cruising parameter c_{fp} is a weighing factor between the d_i and p_i components[3]. By adjusting the $c_{fp} \in [0..1]$ value, an AV-owner can decide whether the AV shall favor the cheaper parking alternatives or park closer to the owner. The latter has the theoretical advantage of a shorter waiting time when the AV is being recalled. It also implies a reduced impact on the traffic flow, as the AV uses a smaller portion of the road network.

By introducing precise rules, see [3], it is achievable that an AV may win at most one auction. If an AV is the winner at an auction, it occupies the won parking place. Otherwise, the AV shall return to its home garage, where an AV can park for free. At the auctions, a small, discrete parking price increment was applied. The maximal amount that an AV was allowed to spend on parking was 15000 Hungarian forint (HUF) per day.

5 Working with an Abstract City Model

To test the effectiveness of the mentioned auction-based assignment mechanism, an abstract mathematical simulation environment was created. This environment represents an abstract,

[1] In our experiments, we defined a three-minute look-ahead time. Hence, if an AV arrives at the destination of its passengers, within three minutes, it shall participate in the auction mechanism.

[2] Moreover, the auction mechanism (by modifying the utility function) can also take other factors (e.g. access to an electric charging station) into account when selecting a vacant parking place.

[3] In our research, we used Hungarian forint (HUF) as currency. The current rate of HUF makes the distance (measured in meters) comparable to monetary costs. For other currencies, a new coefficient might be introduced to make monetary costs and distances be in the same range.

circular city model with a radius of $R = 5$ km. If the parking lot assignment method is found to be effective in this environment, it is worth carrying out more detailed tests.

Using a traditional parking lot seeking algorithm[4] as a baseline, we consider a parking lot assignment mechanism effective if the following statements hold:

1. Average parking prices are not increased unacceptably (e.g. the increment is less than 20%).

2. Sufficient number of parking lots are still used (e.g. at least 30% of the original amount). Too low usage rate would indicate that an assignment method causes too much conflicts between the AVs. Therefore, instead of negotiating the sharing of the parking lots, AVs would prefer a conflict-free parking alternative, e.g. going to their home garage.

3. Average vehicle kilometers traveled is not increased by more than 50%.

As the defined test metrics are relative, it is enough to build a qualitative model[5]. In the following sections, we present a brief overview of the abstract simulation environment, together with the obtained results.

5.1 Derivation of the Abstract Model

To test a parking lot assignment method, the most fundamental components to simulate are the parking lots. We model two parking lot types: curbside parking lots and (park-and-ride, P+R) parking houses. The two types, along with parking charges, differ in capacities as well as in location distributions.

We also generate a synthetic population with simple, home-based trip chains. For simplicity, we assume that the simulated population uses solely AVs to travel. After carrying passengers to their destinations, the AVs will seek parking lots. As the empty cruising can generate significant traffic, we assume that municipalities will introduce an empty cruising distance regulation (d_r) to limit it.

5.1.1 Modeling Distances

The distance plays a crucial role in the evaluation of costs and many metrics describing the traffic. Using the d_E Euclidean or the d_M Manhattan distance between two points of a city would be a simplistic approach. Unfortunately, road networks are seldom that simple. No left turns, curved streets, one-way roads, and col-du-sacs modify the actual d_d driving distances. Hence, we introduce an s (see equation (2)) parameter, which describes how regularly a city is shaped. In our experiment, we draw s values from a Gaussian-distribution $s \sim \mathcal{N}(1.3, 1.8)$. The parameters of that distribution were calibrated by driving distances measured between various points on the map of Budapest, Hungary, see equation (2) (these values are also compatible with similar findings summarized in [19]). After the reformulation of equation (2), we obtain formula (3) for driving distances between two points of the simulated abstract city. Formula (3)

[4]The parking lot searching method begins after arriving at the destination. To this end, the parking lots are listed by the order of their distance. (It is analogous to spiraling around the destination with an increasing radius.) During the search, the AV visits each parking lot in the order of their distances until it can find a free spot.

[5]Quantitative models would require more involved knowledge, more detailed data, and numerous surveys. However, precise quantitative models might be useful in industrial development and in engineering work, a qualitative model is sufficient for current research purposes.

also ensures that no driving distance between two points can be shorter than their Euclidean distance.

$$s = \frac{d_d - d_E}{d_M - d_E} \tag{2}$$

$$d_d = \max\{d_E, d_E + s(d_M - d_E)\} \tag{3}$$

5.1.2 Modeling Parking Lots

Nowadays, there are two fundamentally different types of paid parking lots. The first type is the curbside parking lots. They are close to the destination of the travelers and usually have an hourly price. (With AVs, the billing can have finer granularity, e.g. billing per seconds instead of billing per minutes or hours.) We can observe that curbside parking lots have a decaying density as we move farther from the city center. Hence, we model curbside parking lots with Gaussian distributions, described in Table 1. As streets can have various lengths and layouts, curbside parking lot capacities are modeled with a uniform distribution; for details, see Table 1.

The other parking lot type is the parking house. In the last decades, (park-and-ride, P+R) parking houses are often built at the perimeter of the cities to support efficient modal change. Table 1 describes their modeled locations[6] as an annulus around the city center with randomized radius and width. Their capacity is more frequently predetermined than that of curbside parking lots; therefore, we use constant capacity values for each parking house.

Relatively high parking charges and nearby parking places make curbside parking feasible for shorter activities, e.g. for daily shopping. We expect that AVs will prefer to park in cheaper but more distant parking houses during the more prolonged activities of their passengers.

We model parking charges for the baseline evaluation, according to Table 1. At the beginning of the auctions, those parking fees act as starting prices as well. Curbside parking is considered more expensive, if the d_c distance of the parking lot from the city center is smaller. Hence, a decaying function is used to approximate the parking fees. An example of a generated city model can be seen in Figure 2.

Moreover, many shops and working places provide free parking lots for their customers and workers. The usage of such alternatives is subject to special conditions; hence, they are out of scope of this research.

	Curbside	**Parking Houses**
Location coordinates (R is the city radius)	$x \sim \mathcal{N}(0, R)$ $y \sim \mathcal{N}(0, R)$	$r \sim \mathcal{N}(0, 0.6R) + 2R$ $\theta \sim \mathcal{U}(0, 2\pi)$
Capacity per facility [vehicles]	$\mathcal{U}(1, 10)$	const. 300
Parking fees [HUF] for t_p seconds of parking	$\max\{140, 400 \cdot e^{-.00009 \cdot d_c} + \mathcal{N}(0, 70)\} \cdot \lceil \frac{t_p}{60 \cdot 60} \rceil$	$\mathcal{N}(1200, 600) \cdot \lceil \frac{t_p}{24 \cdot 60 \cdot 60} \rceil$

Table 1: Parameters of the simulated parking lots in the abstract city model

[6]For parking houses, it is more expressive to use polar coordinates instead of Cartesians.

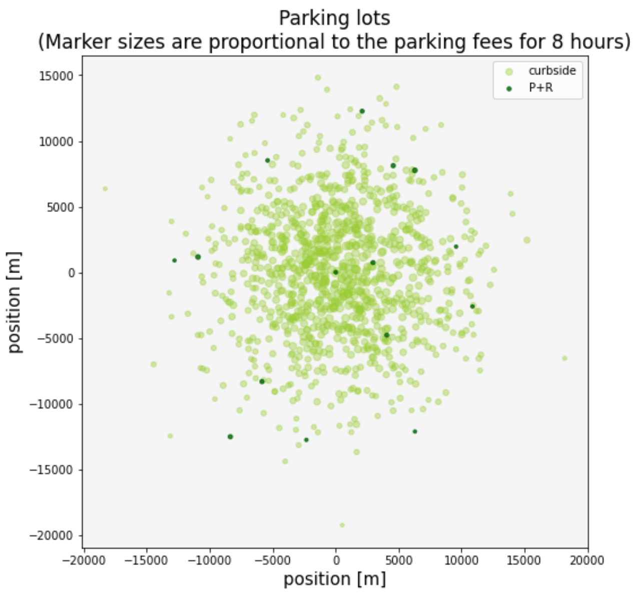


Figure 2: Locations and parking charges of different parking lot types in a generated abstract city model with radius $R = 5$ km

5.1.3 Modeling Trip Chains

In the previous sections, we described the structural aspects of the abstract city model. Now, we will focus on the dynamic components of the simulation, i.e. on the activities of the inhabitants of the abstract model. These activities form *activity chains.* We model the most frequent activity chains according to [7]. As numerous activities are undistinguishable at the abstract level, some simplification was applied:

- **Working and education:** Working and education activities are similar in various ways (e.g. both last for longer periods, normally neither requires transportation during the activity); consequently, we created a joint *working* (`w`) category.
- **Shopping and leisure:** Moreover, shopping and leisure can be similar as well, as they are likely to last for a shorter time. Hence, a common *shopping* (`s`) activity type represents them in our simulations.

With these simplifications, the most frequent activity chains and their distribution is given in Table 2. [1] surveyed shopping habits in Switzerland and summarized that most commonly, shopping time ranges from 20 to 120 minutes. We can model it with a properly scaled beta distribution, as shown in Table 3. We assume that two types of working activities exist. The first is the usual working activity that has an average duration of 8 hours per day. However, this working time is uncommon among e.g. businessmen or tradesmen. Hence, we model interrupted working activities that consist of two shorter, averagely 4 hours activity. Working times are modeled with Gaussian distributions, with these expected values, together with some random variance, as Table 3 describes them.

The trip chains of the simulated persons begin and terminate at home locations, evenly distributed in the city, see Table 4. People are likely to start their activity chains in the

Activity chain	`w`	`s`	`ws`	`ss`	`wws`	`sss`
Probability	38.60%	44.15%	3.70%	7.64%	5.70%	0.21%

Table 2: Activity chains and their probability

Activity	**Length in hour (h)**
Shopping	$\mathcal{N}(0.5\text{ h}, 0.22\text{ h}) + \beta(3,7) \cdot 2\text{ h}$
Working (interrupted)	$\mathcal{N}(4\text{ h}, 0.28\text{ h})$
Working (usual)	$\mathcal{N}(8\text{ h}, 0.28\text{ h})$

Table 3: Activity lengths

morning, as shown in Table 4. The AVs transport passengers to their destinations, and after that, they shall always look for a suitable parking place. For example, to serve the activity chain of `ws` (going to work, then doing some shopping) an AV will have to take the following actions:

1. *Leave home* in the morning.
2. *Transport passenger* to work.
3. *Seek for parking.*
4. *Stay there* for about 8 hours.
5. *Return and pick up passenger* at work.
6. *Transport passenger* to a shop.
7. *Seek for parking.*
8. *Stay there* for about 1 hour.
9. *Return and pick up passenger* at the shop.
10. *Transport passenger* to home.

In abstract simulation, we did not intend to experiment with traffic flows. Hence, we applied time offsets between each activity in an activity chain to represent traveling times.

5.2 Simulation Results

When experimenting with a new mechanism, we needed answers to some crucial questions. The most important is whether the proposed algorithm is capable of fulfilling its task. If the method seems promising, we shall check how the free parameters can influence the obtained results. And finally, we may try to fine-tune these parameters to optimize our goal functions.

The described abstract, mathematical city model is intended to answer the first two questions. In Section 5.2.1 and Section 5.2.2 we present a brief overview of the results. However, parameter fine-tuning requires more detailed simulations. For that reason, we applied Eclipse SUMO, see Section 6 for details.

Parameter	Distribution
Time of leaving home	$\mathcal{N}(7{:}30\text{ am}, 0.54\text{ h})$
Travel time offset	$\mathcal{N}(0.17\text{ h}, 0.1\text{ h})$
Home locations (in the function of R city radius)	$x \sim \mathcal{U}(-2R, 2R)$ $y \sim \mathcal{U}(-2R, 2R)$
Activity locations (in the function of R city radius)	$x \sim \mathcal{N}(0, R)$ $y \sim \mathcal{N}(0, R)$

Table 4: Miscellaneous stochastic values for simulating trip chains

5.2.1 Comparison Between the Traditional Parking Lot Seeking and the Auction-based Parking Lot Assignment Mechanism

The proposed parking lot assignment mechanism was compared to the traditional method with a simple simulation. At the beginning of every simulation run, we draw random values from the previously described distributions to create a new abstract city representation. It includes recreating parking lot locations and capacities, together with generating new trip chains. Each run simulates the trip chains of a whole (working) day by 180 s time steps. We ran 10 simulations for both parking strategies.

The size of the simulated population was 3000. To supply the parking demand, 350 curbside parking lots and 5 parking houses were placed in the model.[7] We have two free parameters for the auction mechanism. The first one is the free cruising parameter c_{fp}. We set $c_{fp} = 0.5$ in this phase of the study, implying that monetary costs and vehicle distance traveled are valued evenly. The other free parameter is the municipal empty cruising distance limitation d_r. The role of this parameter is to limit empty cruising of the vehicles; therefore, limit their adverse effect on the city traffic. We used $d_r = 2R = 10$ km, as it statistically makes 95% of parking lots reachable, due to the empirical law. We applied a 50 $\frac{\text{HUF}}{\text{hour}}$ bid increment during the auctions.

For the comparison of the two parking strategies, we evaluate the following metrics:

- *Total parking prices:* cumulated parking prices that AVs spent on parking.
- *Parking lot occupancy rate:* Portion of parking lots that was occupied during a day.
- *Empty cruising distances:* cumulated distance that AVs traveled to and from parking places.

We have already concluded [3] that the auction-based parking lot assignment mechanism has many advantages compared to the traditional parking lot searching method. According to Figure 3, parking lot assignment significantly reduces the empty cruising distances as the AVs know precisely where to find a free parking place.

There are periods when the applied auction mechanism reaches even higher parking lot occupancy rates than the traditional method, see Figure 4. Meanwhile, the highest mean occupancy rate difference is about 50% (with the traditional method, AVs occupy approximately 32% of the parking lots; and with the auction-based mechanism it is around 16%).

Total parking prices are also reduced due to the lower occupancy rate, as it is shown in Figure 5. So, if we double the total parking prices for the auction-based case, we will get a fair

[7]350 curbside parking lots have the expected capacity of $350 \cdot 5.5 = 1925$ vehicles. 5 parking houses provides parking for $5 \cdot 300 = 1500$ vehicles. Hence, the expected parking lot capacity is 3425 vehicles in this case.

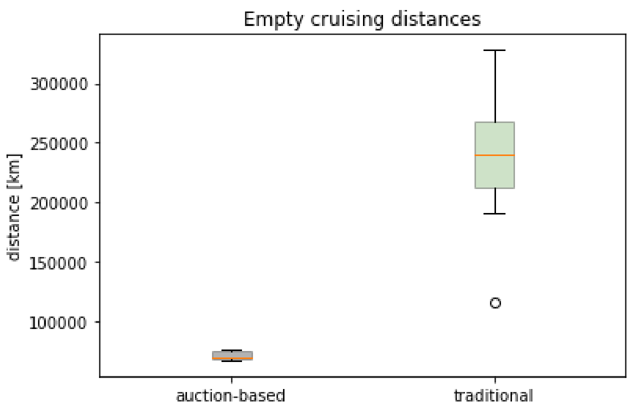


Figure 3: Comparing cumulated empty cruising distances between the case of an auction-based parking lot assignment and the traditional method

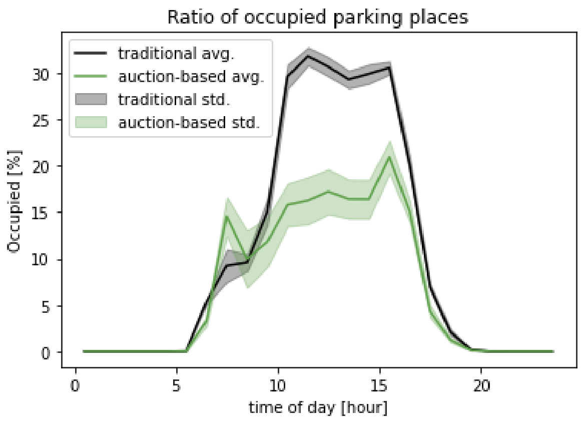


Figure 4: Comparing parking lot occupancy rates during a day between the auction-based parking lot assignment and the traditional method

upper estimate for the parking prices. It means an average total parking price of $7 \cdot 10^6$ HUF for the auction-based mechanism. Compared to the traditional method ($6 \cdot 10^6$ HUF), it is a 16.7% increment.

As an initial result, we conclude that the proposed auction-based mechanism is capable of effectively assigning parking lots to AVs. Therefore, we shall carry out further tests.

5.2.2 Testing Sensitivity to the Settings of the Free Parameters

As the second question, we studied the auction-based assignment mechanism sensitivity to the settings of its free parameters. We increased the size of the simulated population to 10000. The auction bid increment step was decreased to 10 $\frac{\text{HUF}}{\text{hour}}$.

Holistic tests were performed by adjusting the cfp $\in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ free cruising parameter and the dr $\in \{500\text{ m}, 2500\text{ m}, 4500\text{ m}, 6500\text{ m}, 8500\text{ m}\}$ municipal empty cruising

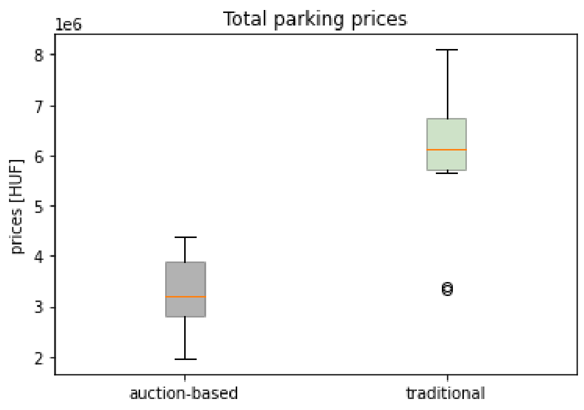


Figure 5: Comparing total parking prices between the auction-based parking lot assignment and the traditional method

distance regulation. 5 simulation runs were conducted with each $(c_{fp} \times d_r)$ parameter pair. At the beginning of the simulation runs, as discussed earlier, a new abstract city representation was generated. We focus on the total parking prices and empty cruising distances again.

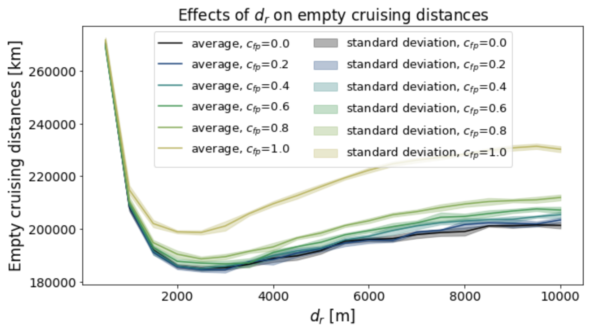


Figure 6: Cumulated empty cruising distances at different c_{fp} and d_r levels measured in abstract simulations

As we can see in Figure 6, empty cruising distances have a local minimum between 2500—3500 m of d_r. Hence, to minimize the additional load of the road network, the municipality shall select the d_r empty cruising regulation value from this range. At lower levels of d_r, many AVs are unable to find a suitable parking place; hence, they have to return to their home garages. Returning trips were not prohibited in our simulations, and in these cases, they resulted in higher empty cruising distances. At higher levels of d_r, more and more parking lots get into range. Therefore, AVs can choose cheaper alternatives instead of closer ones. Moreover, as expected, the c_{fp} free cruising parameter has a moderate positive correlation with the empty cruising distances. We may presume, it worth keeping $c_{fp} \in [0.0, 0.8)$ interval to prevent reaching too high empty cruising rates. Around $c_{fp} = 1.0$ only the (continuously changing) parking charges count when determining which parking lot is suitable, see equation (1). It may lead to unstable parking lot preference lists, causing more unsuccessful auctions. Consequently,

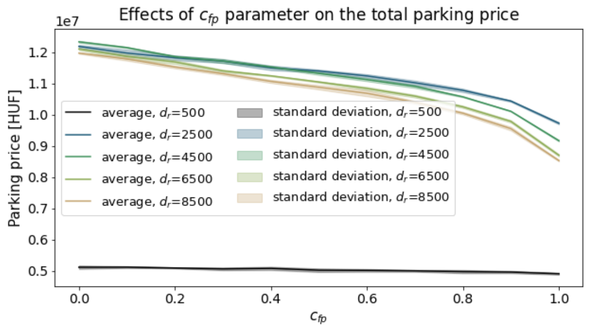


Figure 7: Total parking prices at different c_{fp} and d_r levels measured in abstract simulations

fewer parking lots are occupied, and AVs will have to cruise farther. As expected, a higher c_{fp} free cruising parameter will reduce the total parking prices for the same d_r empty cruising regulation level, see Figure 7.

6 Detailed Simulations with Eclipse SUMO

The presented abstract model intendedly lacks many details. For example, traffic flow is not represented at all. To validate the results and to verify their efficacy we will now conduct measurements with a more detailed model, involving a concrete road network and microscopic traffic simulation. To this end, to make additional tests, we will use Eclipse SUMO.

6.1 Creating a Proper Scenario

Recently, many complex scenarios have appeared for SUMO. We were looking for a proper scenario that is comparable with the abstract model. Hence, the selected scenario shall have approximately 10 km × 10 km dimensions.

We checked some publicly available scenarios for SUMO[8]. Unfortunately, Monaco [12] has many individual features. As it is located on the coast, it significantly differs from the presented abstract, circular city model. Luxembourg [11] would have both appropriate size and shape. However, this scenario does not include parking lots.

As these scenarios were not appropriate for our research, we created a new scenario. We extracted the road network of Budapest 11th district from OpenStreetMap (OSM). It resulted in a rectangular network with a diameter of approximately 10 km. There are numerous P+R parking facilities in the perimeter of this district, see Figure 8. As they were marked in OSM, `netconvert` tool of SUMO was able to recognize them as `parking areas` with higher capacities. The `netconvert` tool also recognized some smaller parking facilities.

For trip chain modeling, we used SAGA [13] to obtain activity and home locations. Instead of cross-validating the abstract model with SAGA, our goal was to test the auction-based parking lot assignment mechanism. Hence, we kept only starting positions (let us call them home locations, analogously to the above mentioned) from the output of SAGA. For each home location, we added roadside parking lots to the corresponding edges. Activity generation is

[8]In November 2020.

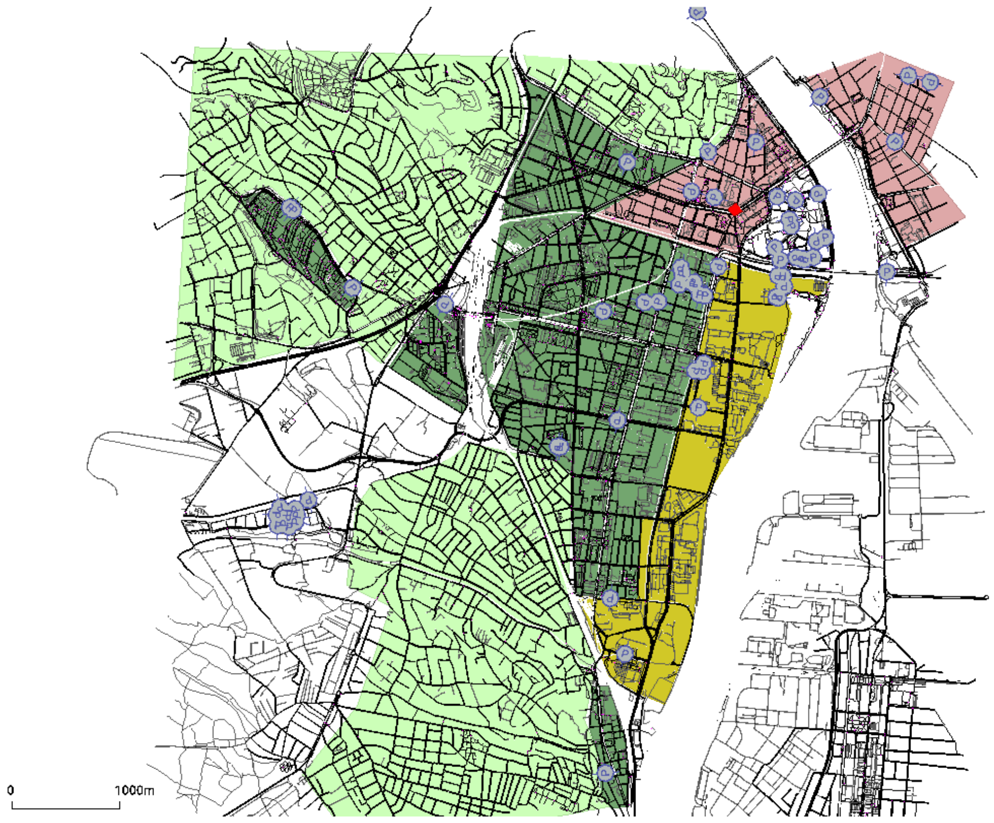


Figure 8: The simulated road network of 11th district of Budapest. The historical downtown area is brick red. High density residential areas are darker, while low density residential areas are lighter green. The yellow area is mainly occupied by industry and services. White areas either have other functionalities or are unreachable. P+R parking lots, with capacity for at least 100 vehicles are depicted by letter *P*s in blue circles. The central point for curbside parking pricing is at the red polygon.

similar to the one presented in Section 5.1.3. The only difference is that we draw activity locations evenly from edges containing parking areas.

This setup is like the abstract model of Section 5.2.2 as it covers an approximately identical area. They have almost the same expected vehicle parking capacity: the abstract model provides parking space for 10700 vehicles, meanwhile the Budapest 11th district scenario offers parking lots for 9857 vehicles.

Finally, we defined the center of the map at the intersection of Budafoki and Irinyi streets, as indicated in Figure 8. Moreover, we can use driving distances provided by SUMO.

6.2 Running Simulations with Eclipse SUMO

Our simulation setup contained a *backend*, a Jupyter Notebook, that controlled the running of the tests and carried out non-SUMO-specific tasks. Traffic Control Interface (TraCI) provided the connection between the backend and Eclipse SUMO.

The road network with its *additionals* (i.e. parking lots) was identical within each simulation run. However, the trip chains were regenerated at the beginning of each run. In this case, we used $c_{fp} \in \{0.0, 0.2, 0.6, 0.8, 1.0\}$ free cruising parameter and the $d_r \in$ {500 m, 1500 m, 2500 m, 4000 m, 7000 m, 10000 m} empty cruising distance regulation pairs. Again, we simulated a whole working day within each simulation run.

The backend, which had a 180 s timestep, controlled the SUMO simulation; conducted the actions, and also registered the occupancy of the parking lots. In every *backend step*, we collected the actual beginning and terminating activities. To perform each of these activities, a new vehicle was inserted into SUMO via TraCI. Its origin was either its home location or the parking lot it had won in the previous auction. Its destination was the position of the terminating activity. When a vehicle picks up its passengers at these locations, it transports them to their next activity.[9] After an AV has carried out these tasks, it terminates. When a passenger has additional activities, new vehicles will be inserted to serve this travel demand.

In the simulations, besides the cumulated distances and total parking prices, the following values were measured:

- *Average speed:* Average speed of the vehicles in the simulation, calculated as follows. The *cumulated distance* that AVs traveled was measured together with the *cumulated travel* times. The ratio of these two metrics results in the average speed.
- *Cumulated waiting times:* The total amount of time that AVs spent waiting (i.e. the time in which the vehicle speed was below or equal 0.1m/s[10]). This metric helps understanding how free parameters influence the traffic flow.

We emphasize that our measurements are intended to provide qualitative results only. Therefore, in our experiments with Eclipse SUMO, we did not differentiate the empty cruising for parking from the *effective movements* (when AVs are carrying passengers) when calculating the cumulated distances. As the activity chains have identical distributions, effective movements are considered to cover the same route lengths on the average. The true variance is caused by the parking lot assignment mechanism in the cumulated distances. Hence, effective movements are only an offset to the cumulated distance data.

Unfortunately, the simulation runs require a significant amount of time on an average PC[11]. Therefore, simulation runs were repeated for 3 times for each given $(d_r \times c_{fp})$ pair.

6.3 Results of Experiments with SUMO

Firstly, total parking prices, obtained by SUMO measurements, are similar to that we have seen by the abstract model, see Figure 9. It only implies that consistent distance metrics do not influence the auction method. We can also notice that parking prices are significantly lower when $d_r < 2500$ m. It is in agreement with the conclusion of Section 5.2.2, as too strict empty cruising regulation makes numerous parking lots out of range.

On the other hand, the curves of cumulated distances are not alike as we would expect because smaller c_{fp} free cruising parameters seem to enlarge them. To understand this phenomenon in Figure 10, we shall investigate the traffic flow related parameters as well. As Figure 11 shows it, average speeds are also lower when the c_{fp} is lower (at higher d_r cruising regulation levels). Higher waiting times, see Figure 12, indicate congestion forming in these

[9]If the activity is the first in the activity chain, we consider that the passenger and its AV are at their home location. Hence, we do not simulate the picking-up movements.

[10]https://sumo.dlr.de/docs/Simulation/Output/TripInfo.html [accessed: August 14, 2021] [11]Intel Core i5 4200H CPU with 8 GB of RAM.

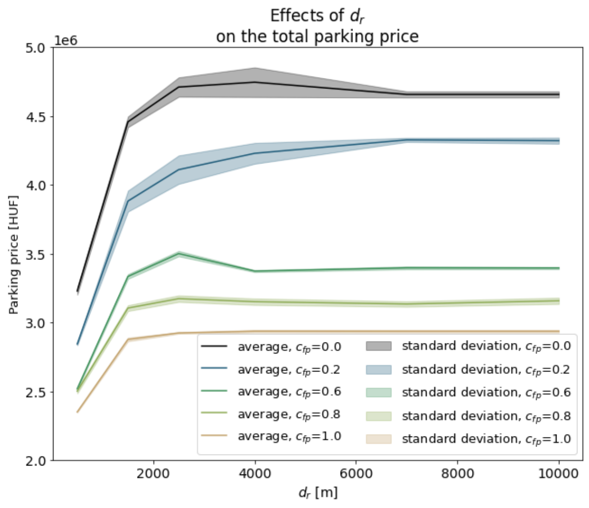


Figure 9: Parking prices for different c_{fp} and d_r levels measured in SUMO simulations

cases. As the simulated vehicles try to optimize their travel times, they are likely to take detours to avoid getting congested. That naturally results in higher traveling distances.

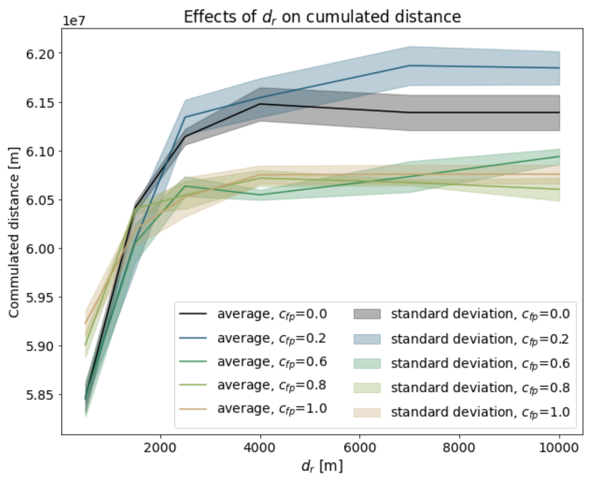


Figure 10: Cumulated distances for different c_{fp} and d_r levels measured in SUMO simulations

We suspect that this congestion has a principal cause. The low c_{fp} parameter forces AVs to find parking lots closer to the destination of their passengers. As human activities are likely to be concentrated in small areas, many AVs try to park there, thus forming congestion. When AVs can or have to[12] go further for parking, the vehicle density in the road network will be more balanced. Therefore, we conclude that c_{fp} shall have a minimum value too, e.g. $c_{fp} \in [0.2, 0.8)$.

[12]Because, for example, there are not enough parking lots within range.

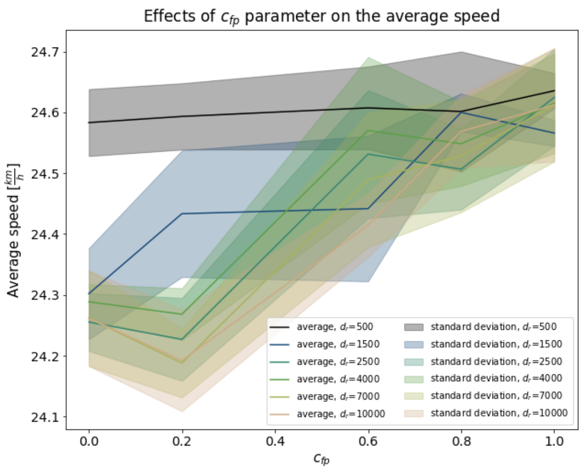


Figure 11: Average speeds for different c_{fp} and d_r levels measured in SUMO simulations

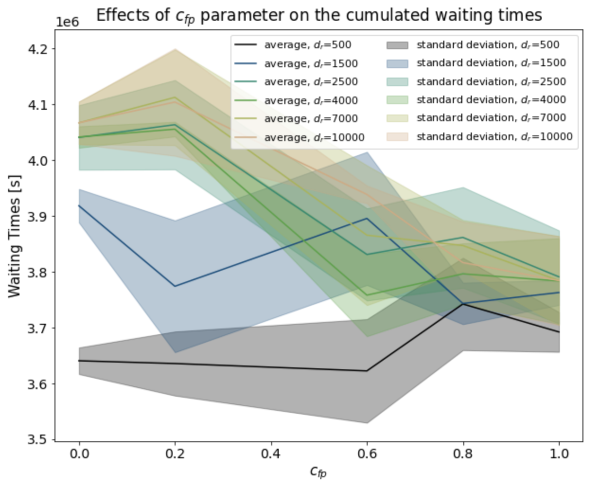


Figure 12: Cumulated waiting times for different c_{fp} and d_r levels measured in SUMO simulations

We may see another difference between the abstract and the SUMO based results in Figure 10. Namely, smaller d_r values do not necessarily cause higher cumulated distances. As parking lots are likely to have a higher density around frequented activity locations in real-world cities, it is more likely that an AV can find an empty parking place within a smaller area. However, it is still true that many parking facilities might be out of range when the empty cruising regulation is too strict (d_r is too small).

7 Conclusion and Further Research Aims

In this paper, we presented the usage of Eclipse SUMO to test an auction-based parking lot assignment mechanism developed in an abstract city model. The results of SUMO helped to analyze aspects oversimplified in the abstract simulations.

As the downward refinement consistency empirically holds for the obtained abstract and SUMO results, broadly speaking the two models are consistent for the investigated problem. However, further validation of the abstract, mathematical model may be necessary for the future research. Luckily, more and more publicly available SUMO scenarios have appeared recently. By using a variety of these scenarios, we would be able to test the abstract model more profoundly. For example, we may make hypothesis testing of the mentioned metrics (i.e. empty cruising distances, total parking prices). Then, unless being able to reject it, we shall consider the abstract model valid.

The scenario of Section 6.1 also lacks some details. E.g. the transit traffic through the 11th district of Budapest was not modeled here. As a qualitative test, instead of optimizing the traffic flow, we focused on how the c_{fp} and d_r parameters form the traffic for parking. In this interpretation, the parameter set that is less likely to cause (local) congestion is more favorable.

Our further research covers studying machine learning algorithms connected to parking lot seeking. To keep the generality of the results, and to minimize the calculation times, a more detailed mathematical city model formulation is required. When the developed method is mature enough in the abstract world, we will also test it by publicly available SUMO scenarios to show its efficacy.

Finally, we would like to point out that city planners will have to prescribe cruising limitations carefully. As we saw in Section 5.2.2, if the d_r value is too small, AVs will not find a suitable parking place. A similar precaution is advised when tuning something like the c_{fp} free cruising parameter. A small c_{fp} value[13] is more likely to cause (local) congestion. On the other hand, a higher c_{fp} can increase empty cruising distances, more vehicle kilometers traveled, consequently more energy usage. Therefore high c_{fp} values are also to be avoided.

References

[1] S. Alberton and G. Guerra. *Il comportamento dei consumatori in materia di mobilità nei principali centri commerciali del cantone Ticino*. CODE, Centro per l'osservazione delle dinamiche economiche, c/o IRE, Instituto di Ricerche Economiche, Lugano, 2008.

[2] L. Alekszejenkó and T. Dobrowiecki. Auction based parking lot assignment and empty cruising limitation of privately owned autonomous vehicles in a simple city model. (submitted to the IV21 32nd IEEE Intelligent Vehicles Symposium, Nagoya, July 11-17, 2021).

[3] L. Alekszejenkó and T. Dobrowiecki. Using auction mechanism for assigning parking lots to autonomous vehicles. In *Proc. of the 28th Minisymposium of the Department of Measurement and Information Systems Budapest University of Technology and Economics*, pages 32–35, Budapest, Hungary, February 2021. https://inf.mit.bme.hu/sites/default/files/minisymp/28Minisy_proceedings.pdf.

[4] L. Alekszejenkó and T. P. Dobrowiecki. Sumo based platform for cooperative intelligent automotive agents. In M. Weber, L. Bieker-Walz, R. Hilbrich, and M. Behrisch, editors, *SUMO User Conference 2019*, volume vol. 62 of *EPiC Series in Computing*, pages 107–123. EasyChair, 2019.

[5] W. Alonso. *Location and Land Use – Toward a General Theory of Land Rent.* Harvard University Press, 1964.

[13] We believe, human drivers have a really small c_{fp} value. That means most driver prefer paying more for parking over walking longer distances.

[6] F. Bacchus and Q. Yang. Downward refinement and the efficiency of hierarchical problem solving. *Artificial Intelligence*, vol. 71:43–100, 1993.

[7] M. Balmer. *Travel demand modeling for multi-agent traffic simulations: Algorithms and systems.* PhD thesis, ETH Zurich, 2007.

[8] M. Barthelemy. Modeling cities. *Comptes Rendus Physique*, vol. 20(no. 4):293–307, 2019.

[9] F. Biljecki, J. Stoter, H. Ledoux, S. Zlatanova, and A. Çöltekin. Applications of 3d city models: State of the art review. *ISPRS International Journal of Geo-Information*, vol. 4(no. 4):2842–2889, 2015.

[10] H. Chai, R. Ma, and H. M. Zhang. Search for parking: A dynamic parking and route guidance system for efficient parking and traffic management. *Journal of Intelligent Transportation Systems*, vol. 23(no. 6):541–556, 2019.

[11] L. Codeca, R. Frank, and T. Engel. Luxembourg sumo traffic (lust) scenario: 24 hours of mobility for vehicular networking research. In *2015 IEEE Vehicular Networking Conference (VNC)*, pages 1–8, 2015.

[12] L. Codeca and J. Härri. Monaco SUMO Traffic (MoST) Scenario: A 3D Mobility Scenario for Cooperative ITS. In *SUMO 2018, SUMO User Conference, Simulating Autonomous and Intermodal Transport Systems, May 14-16, 2018, Berlin, Germany*, Berlin, GERMANY, 05 2018.

[13] L. Codecà, J. Erdmann, V. Cahill, and J. Härri. Saga: An activity-based multi-modal mobility scenario generator for sumo. In *SUMO User Conference 2020 – From Traffic Flow to Mobility Modeling*, Berlin, Germany (Online), 10 2020.

[14] L. Codecá, J. Erdmann, and J. Härri. A sumo-based parking management framework for large-scale smart cities simulations. In *2018 IEEE Vehicular Networking Conference (VNC)*, pages 1–8, 2018.

[15] Gy. Fekete and K. Almássy. Smart city megoldások a budapest közút zrt.-nél. *Híradástechnika*, vol. LXXIII.:45 – 53, 2018. (in Hungarian).

[16] N. Kheterpal, K. Parvate, C. Wu, A. Kreidieh, E. Vinitsky, and A. Bayen. Flow: Deep reinforcement learning for control in sumo. In E. Wießner, L. Lücken, R. Hilbrich, Y.-P. Flötteröd, J. Erdmann, L. Bieker-Walz, and M. Behrisch, editors, *SUMO 2018- Simulating Autonomous and Intermodal Transport Systems*, volume vol. 2 of *EPiC Series in Engineering*, pages 134–151. EasyChair, 2018.

[17] A. Lefauconnier and E. Gantelet. La recherche d'une place de stationnement: strategies, nuisances associees, enjeux pour la gestion du stationnement en france, 2005. (research report).

[18] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE Int. Conf. on Intelligent Transportation Systems.* IEEE, 2018.

[19] D. Merchán, M. Winkenbach, and A. Snoeck. Quantifying the impact of urban road networks on the efficiency of local trips. *Transportation Research Part A: Policy and Practice*, vol. 135:38–62, 2020.

[20] J. Polak and K. Axhausen. Parking Search Behaviour: A Review of Current Research and Future Prospects. *University of Oxford, Transport Studies Unit (1. Jan. 1990).*

[21] M. Alonso Raposo, B. Ciuffo, M. Makridis, and C. Thiel. The r-evolution of driving: from connected vehicles to coordinated automated road transport (c-art). 2017. EUR 28575 EN.

[22] G. Soares, Z. Kokkinogenis, J. M. Luiz, and R. J. F. Rossetti. Agent-based traffic simulation using sumo and jade: An integrated platform for artificial transportation systems. In M. Behrisch, D. Krajzewicz, and M. Weber, editors, *Simulation of Urban Mobility*, pages 44–61, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

SUMO User Conference 2021
https://doi.org/10.52825/scp.v2i.97

Published: 29 Jun. 2022

Modeling Cellular Network Infrastructure in SUMO

Anupama Hegde[1], Ringo Stahl[1], Silas Lobo[1], and Andreas Festag[1*]

Technische Hochschule Ingolstadt
CARISSMA Institute for Electric, COnnected, and Secure Mobility (C-ECOS){anupama.hegde | ris0774 | SilasCorreia.Lobo | andreas.festag@thi.de}@carissma.eu

Abstract

Communication networks are becoming an increasingly important part of the mobility system. They allow traffic participants to be connected and to exchange information related to traffic and roads. The information exchange impacts the behavior of traffic participants, such as the selection of travel routes or their mobility dynamics. Considering infrastructure-based networks, the information exchange depends on the availability of the network infrastructure and the quality of the communication links. Specifically in urban areas, today's 4G and 5G networks deploy small cells of high capacity, which do not provide ubiquitous cellular coverage due to their small range, signal blocking, etc. Therefore, the accurate modeling of the network infrastructure and its integration in simulation scenarios in microscopic traffic simulation software is gaining relevance.

Unlike traffic infrastructure, such as traffic lights, the simulation of a cellular network infrastructure is not natively supported in *SUMO*. Instead, the protocols, functions and entities of the communication system with the physical wireless transmission are modeled in a dedicated and specialized network simulator that is coupled with *SUMO*. The disadvantage of this approach is that the simulated *SUMO* entities, typically vehicles, are not aware which portions of the roads are covered by wireless cells and what quality the wireless communication links have.

In this paper, we propose a method for modeling the cellular infrastructure in *SUMO* that introduces a cellular coverage layer to *SUMO*. This layer models cell sites in a reg-ular hexagonal grid, where each site is served by a base station. Following commonly accepted guidelines for the evaluation of cellular communication system, the method fa-cilitates standardized and realistic modeling of the cellular coverage, including cell sites, antenna characteristics, cell association and handover. In order to ease the applicability of the method, we describe the work flow to create cell sites. As a representative case, we have applied the method to *InTAS*, the *SUMO* Ingolstadt traffic scenario and applied real data for the cellular infrastructure. We validate the approach by simulating a *Cellular V2X* system with sidelink connectivity in an urban macro cell environment by coupling *SUMO* enhanced by the proposed connectivity sublayer with *ARTERY-C*, a network simulator for *Cellular V2X*. As a proof-of-concept, we present a signal-to-interference noise ratio (SINR) coverage map and further evaluate the impact of different types of interference. We also demonstrate the effect of advanced features of cellular networks such as inter-cell interference coordination (ICIC) and sidelink communication modes of *Cellular V2X* with dynamic switching between the in-coverage and out-of-coverage mode.

[*]Also with Fraunhofer Application Center VMI, Ingolstadt, Germany

1 Introduction

Communication networks are becoming an increasingly important part of the mobility system. They allow vehicles, roadside infrastructure and vulnerable road users to be connected and to exchange information. A great number of applications exist, such as online route guidance, safety hazard warnings or traffic light information. While many applications are already intensively used today, they are expected to considerably improve the road traffic efficiency and safety in the future. Also, the communication support future autonomous driving by the exchange of massive amount of sensor data and of messages for maneuver coordination.

The communication impacts the aggregated road traffic flows and the individual behavior of the traffic participants because the reception of the information changes their behavior and interaction, locations, distance and time headways, as well as velocity and acceleration. For example, online route guidance helps drivers to bypass traffic jams. A safety-related hazard warning forces a driver to break, or the reception of the traffic light status may prompt a driver to adapt the speed. For accurate modeling of road traffic, it is therefore indispensable to consider communications.

For a macroscopic representation of aggregated road traffic flows, simplifying assumptions for the communication can be made. For a microscopic model, vehicles and other traffic participants need to be modeled individually along with their specific dynamics. One possible approach is to enhance existing microscopic road traffic models and adapt their parameter settings. However, the communication strongly depends on the capabilities of the communication system and the quality of the communication links. Specifically, today's 4G and 5G networks deploy small cells of high capacity in urban areas, which do not provide ubiquitous coverage due to their short range, signal blocking etc. Also, recent developments in cellular networks facilitate a direct communication among end devices (sidelink) extending the conventional data transmission via the cellular infrastructure (up- & downlink), referred to as *Cellular-V2X*, which is particularly suitable for the exchange of V2X messages [2]. Therefore, the microscopic representation of road traffic requires a detailed modeling of protocols, functions and entities of the communication system with abstraction of the physical transmission. Specifically, we regard the communication infrastructure as integral part of a modern road traffic infrastructure similar to traffic lights.

In the microscopic road traffic simulator *SUMO* [11], the modeling of the communication system and the simulation of the information exchange are not natively supported. Instead, for studying scenarios with realistic mobility and communications, a dedicated, specialized communication network simulator is coupled with *SUMO* via the Traffic Control Interface *TraCI* [17]. The limitation of this approach is that routable *SUMO* entities, typically vehicles, are not aware which portions of the roads are covered by wireless cells and what quality the wireless communication links have.

In this paper, we propose an approach to model the cellular infrastructure in *SUMO*. We describe a generic workflow to create cellular sites (each served by a dedicated base station) containing real world data about the cellular infrastructure. Following ITU-R guidelines [5] for evaluation of wireless communication technologies, the method facilitates standardized and realistic modeling of the cellular coverage, including cell sites, antenna characteristics, cell association and handover. For validation we carry out link level simulations of a *Cellular V2X* system enabled with sidelink connectivity. This is done by coupling *InTAS*, the *SUMO* Ingolstadt traffic scenario, enhanced by the proposed connectivity sublayer with *ARTERY-C*, a network simulator for *Cellular V2X*.

The remainder of the paper is structured as follows: After a review of related work in Section 2, Section 3 provides background on *Cellular V2X*, the network simulator *ARTERY-*

C and modeling of communication aspects. Section 4 describes the modeling of the cellular infrastructure and wireless coverage in *SUMO*. Section 5 presents selected simulation results as proof-of-concept and Sec. 6 concludes the paper.

2 Related Work

Considerable research work has been carried out using *SUMO* in order to develop real world scenarios to simulate vehicular communication. Some of the best scenarios include - Luxembourg *SUMO* traffic *LuST*, *TAPAS* Cologne and Monaco *SUMO* traffic *MOST*, which model all aspects of road traffic, public transport and building structures.[1] Furthermore, the Ingolstadt traffic scenario *InTAS* [10] considers additional parameters such as traffic light conditions and also analyzes real-world data and defines a realistic traffic demand method that fits the dataset. However, the above mentioned scenarios do not model the cellular network infrastructure, which plays a major role in identifying, whether roads or road segments have cellular coverage nor provide information about the link quality. The authors believe that the introduction of the cellular coverage aspect to *InTAS* is a first major step towards developing a real world scenario, where V2X communication is natively enabled in *SUMO*.

Existing simulation frameworks for vehicular applications are capable of supporting complex vehicular traffic models along with online re-configuration and re-routing features. The hybrid simulation frameworks *iTETRIS* [7] and *Veins* [15] couple *SUMO* with the network simulator *ns3* and *OMNeT++*, respectively. Several advanced simulators for V2X communications have been developed on the basis of *ns3* or *OMNeT++*, such as *Artery*, *Artery-C*, *Cellular-VCS*, *OpenCV2X*, *Simu5G* and *SimuLTE* [14, 3, 6, 12, 16, 13]. They can be coupled with *SUMO*, either using *SUMO*'s *TraCI* interface directly or exploiting a simulation framework, such as *Veins*. *Cellular V2X*-specific simulators such as *OpenCV2X* [12] and *Artery-C* [3] are capable of supporting various application scenarios of V2X communication using the above mentioned *SUMO*-based scenarios.

However, in all the above approaches the cellular network infrastructure is modeled inside the communication network simulator (i.e., *OMNeT++* or *NS-3*) and only the vehicle-related information is exchanged between the communication network simulator and *SUMO*. This corresponds to the assumption that vehicles already know that cellular coverage is available. An approach to model various types of cellular environments for e.g. urban macro cell directly inside *SUMO* serves as a step forward towards setting up a close-to-real simulation environment where the vehicle is completely unaware of the availability of cellular network coverage when it enters the simulation.

3 Coupling of a Network Simulator with SUMO

This section presents some of the relevant Cellular V2X concepts such as communication interfaces namely - uplink (UL), downlink (DL) and sidelink (SL), an introduction to the network simulator along with protocol stack description, channel description and a brief overview of types of interferences that we have taken into account.

[1]See https://sumo.dlr.de/docs/Data/Scenarios.html

3.1 Cellular V2X for Backend and Sidelink Communications

V2X communication based on cellular networks enables the continuous exchange of information among vehicles, roadside infrastructure and other road traffic participants. It facilitates the conventional communication between vehicle and network (V2N) to provide backend services and, in addition, realizes direct communication among end devices, i.e., vehicle-to-vehicle (V2V), vehicle-to-pedestrian (V2P) and vehicle-to-infrastructure (V2I) communication. The direct communication, also referred to as device-to-device (D2D), allows two physically close end devices to use a sidelink without sending the data via the cellular access and core network. Compared to the regular cellular communication via up- and downlink, the sidelink shortens the latency, which is a critical aspect of use cases for vehicle safety and automation [9]. For road traffic efficiency and safety applications, various message types, such as the cooperative awareness message (CAM), the decentralized environmental notification message (DENM), vulnerable road user awareness message (VAM), can be transmitted over Cellular V2X links [1].

3.2 Artery-C for Network Simulation of Cellular V2X

For simulation of the *Cellular V2X* system, we use the network simulator *Artery-C* [3] and couple *SUMO* with it. *Artery-C* models both the control and the user plane of *Cellular V2X* and implements all layers as depicted in Figure 1. The top layer of the protocol stack is for generation and reception of V2X messages. *Artery-C* shares the same upper V2X messaging layer as implemented in *Artery* [14].

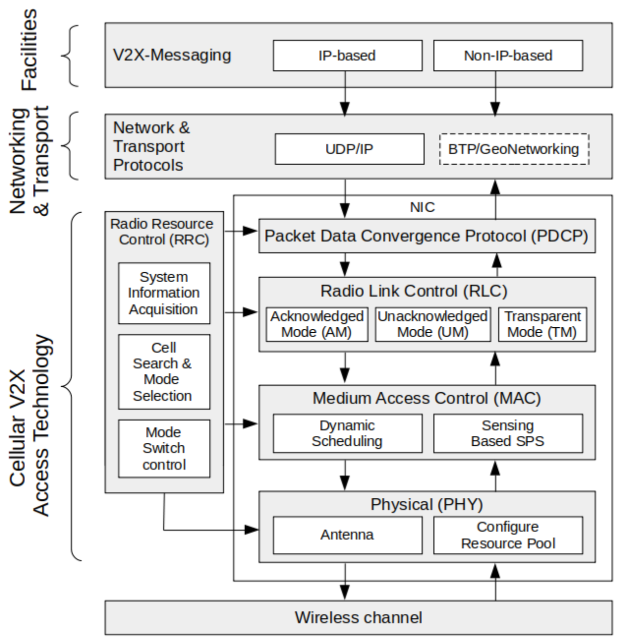


Figure 1: Implementation design of the *Cellular V2X* stack in *Artery-C* [3]

The protocol stack implementation in *Artery-C* is further enhanced by ITU specified stochastic channel models in order to carry out simulations for validation of the cellular network coverage model implemented in *SUMO*. The user plane is directly linked with the *SUMO* traffic model via the `TRaCI` API in order to continuously track the position of the vehicle and identifies whether it is located in the region of cellular coverage or not. The cellular base stations (eNodeB in 4G LTE) and road side units (RSUs) are modelled as stationary modules whereas the vehicles are modelled as dynamic modules. For details of the protocol stack modeling we refer to [3]. The vehicle user equipment (UE) in the simulation have been categorized into two types based on their connectivity to the base station and their geographical location namely (*i*) in-coverage and (*ii*) out-of-coverage. This classification of the vehicles directly corresponds to the two operational resource allocation modes in sidelink – in-coverage and out-of-coverage, which are also referred to as mode 3 and 4 in *Cellular V2X* specifications.

In-coverage mode. The vehicles which are inside the cellular coverage region can dynamically connect to the nearby base station which are in turn connected to the enhanced packet core (EPC). From Figure 4 it can be inferred that vehicles in overlapping cell regions have an option to connect to any of the eNodeBs. A vehicle measures the wireless signal strength from all possible eNodeB and eventually chooses to associate with the eNodeB with the best value. Vehicles communicate to the eNodeB using the up- and downlink interfaces. The sidelink interface is used when two vehicles want to directly communicate with each other. In the in-coverage mode of operation, the eNodeB manages the wireless resources and exchanges control information with the UE; however, the vehicle exchange application data directly via the sidelink [3, 2].

Out-of-coverage mode. In case road segments do not have cellular coverage, *Cellular V2X* enables vehicles to communicate autonomously with each other using the sidelink interface. The resource allocation mode is configured as unmanaged, where the UE dynamically selects its resources from a pre-defined pool of resources. We note that the received signal-to-interference-noise ratio (SINR) at the cell borders are also regions where vehicles can operate in out-of-coverage mode because the signal strength from the eNodeB might not be sufficient for reliable V2X communication. The decision to switch to the out-of-coverage mode on such road segments is determined by measuring the SINR, assessing the data traffic load and continuously monitoring the radio resource utilization of the wireless link.

3.3 Channel Modeling

Typical channel models consider multipath propagation and fading for narrow band radio communication where they assume that similar delays are experienced by the propagating waves; for example in *Artery-C*, Rayleigh and Ricean fading models can be applied. Additionally, the *Artery-C* simulator also incorporates the power delay profiles of multi-tap channel models specified by ITU including the pedestrian A / B and vehicular A / B channel models [5]. The power delay profiles along with the root mean square (RMS) delay spreads characterize the frequency selectivity of the channel. In V2X communication, the fast movement of the vehicles causes the channel to exhibit time variance (selectivity), which is further modeled as Jakes spectrum. While adhering to the abstraction at link level, we consider the transmit/receive power measured directly at the antenna terminals of base stations and UE.

While simplified channel models are sufficient to get an understanding of the overall statistics of vehicular communications, it is important to consider the geometrical aspects in order to study accurately the propagation characteristics in a complex vehicular network. Such approaches for geometry-based channel modeling can be classified into two categories – deterministic and stochastic.

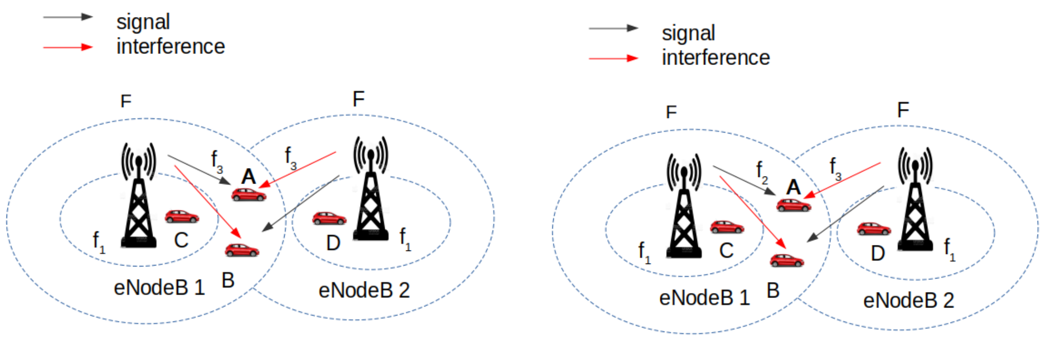
In the deterministic approach, the geometrical aspects of specific sites, including vehicles, buildings, terrain and foliage, are taken into account. Although this approach is advantageous in terms of accuracy, it consumes a lot of computational resources and the results are applicable only to that specific site or geographical area. In order to simplify the computation for larger scenarios such as *InTAS*, we consider a statistical approach to model the small scale and large scale fading/propagation characteristics. The Winner II channel model [8] is a good choice for geometry-based stochastic channel modeling, which considers both line-of-sight (LOS) and non line-of-sight (NLOS) wireless propagation conditions for various types of environments. Owing to non-homogeneity in the obstructions such as buildings, foliage and movement of vehicles in the crowded part of the city, we have considered it as a NLOS scenario where the path loss is calculated according to

$$PL = 44.9 - 6.55log(h_{BS})log(d) + 34.46 + 5.83log(h_{BS}) + 23log(f_c/5) \tag{1}$$

where h_{BS} is the height of the base station and d is the distance between transmitter and receiver for a given carrier frequency f_c.

3.4 SINR Model

In order to study the effects of multipath propagation in a densely crowded urban environment, we apply an SINR model which considers several factors, i.e., losses due to time-frequency selectivity of channel, thermal noise and inter-cell interference. For the out-of-coverage mode of *Cellular V2X*, we encounter only the effects of multipath propagation and thermal noise, but in the in-coverage mode, the effect due to inter-cell interference needs to additionally be taken into account.



(a) No inter-cell interference coordination (ICIC) (b) With ICIC

Figure 2: Modeling of inter-cell interference in the network simulator

Inter-cell interference occurs when users in neighbouring cells attempt to simultaneously use the same radio resources. This is caused by the fact that eNodeBs optimize the resource allocation for the UEs in their cell and are not aware of the allocation in neighbouring cells, which can be specifically harmful for UEs located at the border of overlapping cells: Figure 2a shows that although the vehicles C and D use the same frequency f_1, they do not experience any interference because they are located close to the base stations and hence use low transmit power. On the other hand, vehicles A and B experience interference on the same frequency

f_3. Figure 2b illustrates the use of inter cell interference coordination (ICIC), a standardized mechanism in cellular networks, where eNodeBs are tightly time synchronized and exchange information among each other about the resources utilized by their UEs. This helps to mitigate the interference problem caused by spatial reuse of frequencies.

4 Modeling Cellular Coverage in SUMO

The concept of vehicular networking has lead to the need to establish reliable communication between vehicles, infrastructure and other traffic participants. In addition to developing road traffic models in *SUMO* to model vehicle and pedestrian behaviour, there also arises a need to model network infrastructure. With a long term goal to study different application scenarios of *Cellular V2X* communication in the city of Ingolstadt, we hereby extend the *InTAS* scenario by modeling cellular coverage sites in one of the densely crowded area (the urban district *Ingolstadt Mitte*), which is a good choice for modeling an urban macro cell environment. Specific reasons to introduce the modeling of cell sites in a road traffic scenario are as follows:

- Get a clear understanding of the portions of the road covered by cellular network, i.e., which providers are operating, frequency bands, etc.
- Retrieve the number of vehicles currently served by a certain base station in a specific cell. This information is helpful to study the occupancy of a cell at a given time during the day and to further optimize the allocation of radio resources.
- Develop a better understanding of the cell coverage and interference characteristics at different points in a cell site, e.g., within the cell close to base station, near cell borders, overlapping regions etc.

The importance of having bi-directional coupling between the network simulator *OMNeT++* and the microscopic traffic model in *SUMO* is well explained in [15]. Continuing with the same architecture, the addition of the cellular layer helps us to generate a record consisting of mobility events (interactions between road traffic participants and eNodeBs) for routable entities such as vehicles and pedestrians that can be further utilized to analyse the vehicle behaviour, accuracy of message transmission & reception in case of events taken place in the recent past. In the following subsections, we further discuss about the structure of the cell sites, how to model them in sumo environment and also the types of antennas deployed.

4.1 Cell Sites

For modeling of the geographic environment and for defining the application scenario, we rely on [5], which defines detailed test environments for the evaluation of cellular networks, specifically for enhanced mobile broadband (emBB), ultra-high reliable and low-latency communications (URLLC) and massive MTC, namely *(i) indoor hotspot-emBB*, *(ii) dense urban-emBB*, *(iii) rural-emBB*, *(iv) urban macro-mMTC*, *(v) urban macro-URLLC*. These test environments correspond to the major application scenarios in the 3GPP specifications for 5G cellular networks and define define target key performance indicators for every scenario. In this paper, we have developed a *dense urban-emBB* test environment which is characterized by high density traffic load comprising of fast moving vehicles and pedestrians.

In the proposed cellular coverage layer in *SUMO*, each cell site comprises three transmission reception points (TRxP) that are placed in a regular hexagonal grid (Figure 3). A cell site is

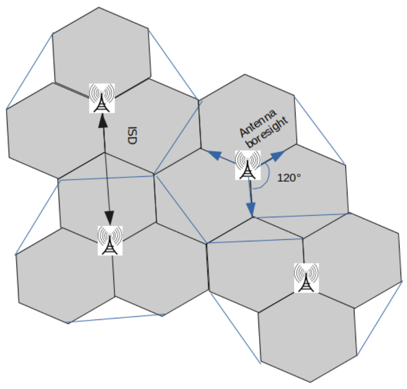
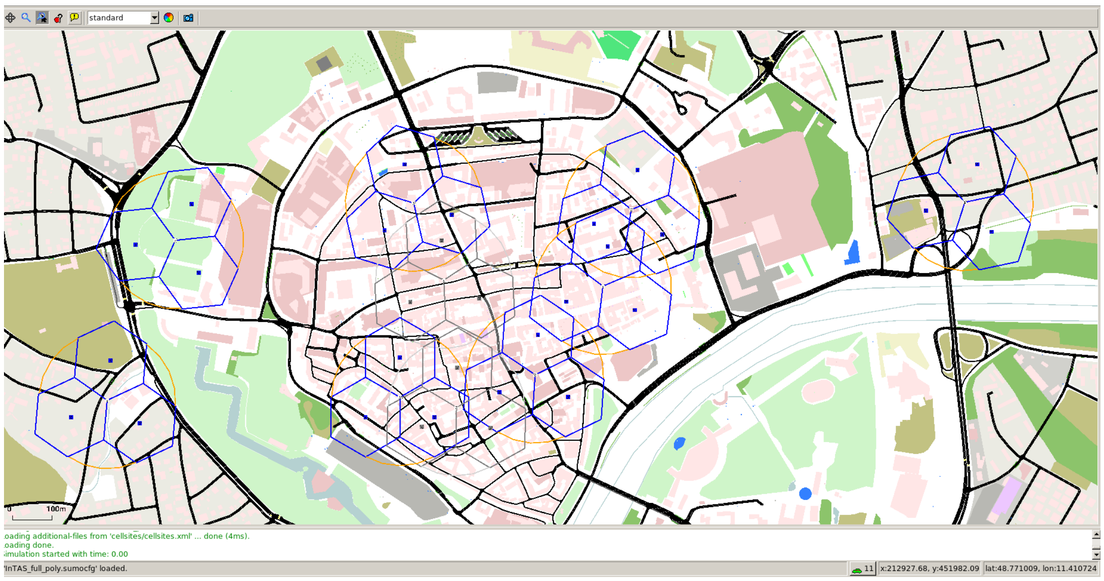


Figure 3: Dense urban macro cell layout

served by a base station (eNodeB) covering three cells in the sector at an angle of 120 degrees. The transmitter-specific parameters are defined in Table 1. We note that the inter-site distance (ISD) is 500 m for an urban macro cell. A screenshot of the cell sites in *InTAS* is depicted in Figure 4.



```
oading additional-files from 'cellsites/cellsites.xml' ... done (4ms).
oading done.
imulation started with time: 0.00
'InTAS_full_poly.sumocfg' loaded.
11  x:212927.68, y:451982.09  lat:48.771009, lon:11.410724
```

Figure 4: Cell sites in the *SUMO* Ingolstadt traffic scenario *InTAS*

4.2 Generation of Cell Sites in SUMO

The workflow to create cell sites in a *SUMO* scenario is illustrated in Figure 5, using *InTAS* with `Ingolstadt.net.xml` as an example. In general, the process of modeling cellular coverage involves the following aspects:

- Definition of eNodeB as a `Point-of-interest (POI)`. The geospatial coordinates of the eNodeBs (latitude, longitude) are obtained from the crowdsourcing database *Cellmapper*.[2]

- Creation of a hexagonal layout (Figure 3) utilizing the `polygon` feature in *SUMO*: With the knowledge of the coordinates of eNodeB, we obtain the vertices of the hexagonal grid.
- Creation of boundaries to distinguish different areas such as cell borders and sectors: this helps to distinguish among the lanes having varying degrees of cellular coverage.
- Assigning attributes to eNodeB and the corresponding cells in a `.xml` file.

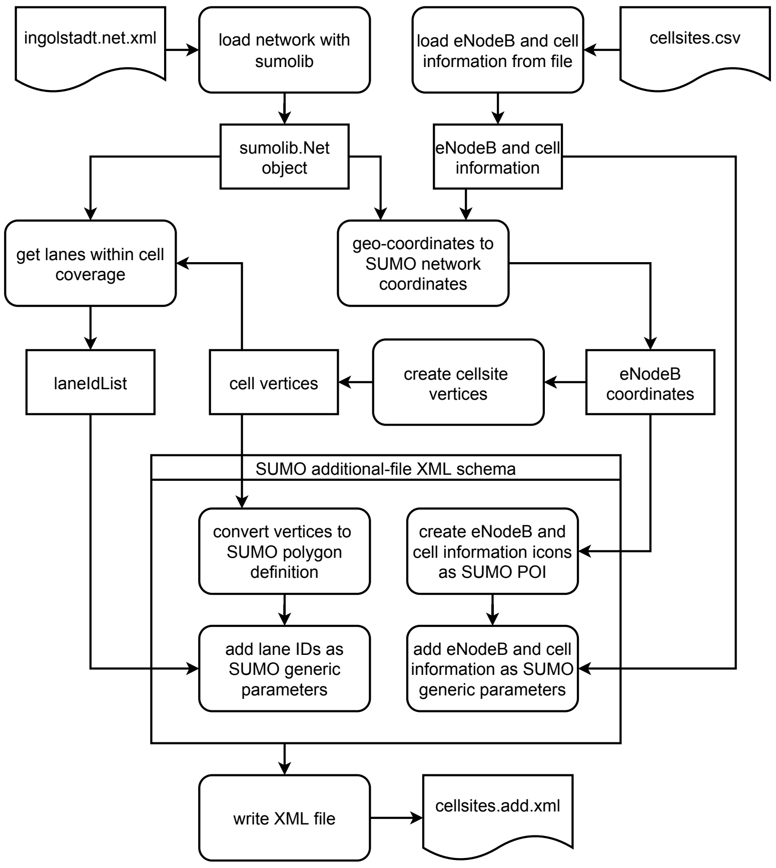

ingolstadt.net.xml
load network with sumolib
load eNodeB and cell information from file
cellsites.csv
sumolib.Net object
eNodeB and cell information
get lanes within cell coverage
geo-coordinates to SUMO network coordinates
laneIdList
cell vertices
create cellsite vertices
eNodeB coordinates
SUMO additional-file XML schema
convert vertices to SUMO polygon definition
create eNodeB and cell information icons as SUMO POI
add lane IDs as SUMO generic parameters
add eNodeB and cell information as SUMO generic parameters
write XML file
cellsites.add.xml

Figure 5: Workflow for the creation of cell sites

In Figure 5, the first step is to create a cell information file (`cellsites.csv`) that consists of the base stations and cell related attributes (refer to Figure 6a and 6b) . Then, we convert the geospatial coordinates of the eNodeB into *SUMO* recognizable network coordinates. The logic for the generation of cell vertices is written in a separate `Python`-based script, which takes the coordinates of the eNodeB as an input parameter and generates the cell site vertices to construct the hexagon grid (`polygon`) inside `Ingolstadt.net.xml` file. We use different colors to indicate the cells, cell boundaries and sectors. We add the lane IDs of the lanes traversed by the hexagonal cells as generic parameter because later this information is useful to obtain

[2]https://www.cellmapper.net

the details of the vehicles, which are inside the cellular coverage area and their mobile event log details w.r.t to the eNodeB. The eNodeB and cell information is subsequently written into the `cellsites.add.xml` file, which gets loaded along with the `intas.sumocfg` file during the start of simulation. This workflow allows the user to include any number of attributes possible in the `cellsites.csv` file, generate the hexagon grid of appropriate cellular environment and automatically generate the `cellsites.add.xml` file.

A screenshot of the specific parameters of the eNodeB and cells inside the *SUMO* environment is shown in Figure 6a and 6b respectively. In order to retrieve these attributes into the network simulator, we use the `TraCI` interface in the `POI` scope.

In order to determine the data traffic load in each cell, we first retrieve the list of lane IDs traversed by the polygon and subsequently obtain the list of vehicle IDs present in that lane. Since vehicles can dynamically enter and exit the cellular coverage region, it becomes important to monitor the road traffic flow in order to get an idea of the number of users (vehicles, pedestrians, etc.) connected to a given base station and the radio resources utilized by the users in the particular cell site.

4.3 Antenna Characteristics

Base station (BS) antennas are modeled having one or multiple antenna panels, where an antenna panel is composed of one or multiple antenna elements placed vertically, horizontally or in a two-dimensional array. An antenna panel has $M \times N$ antenna elements, where N is the number of columns and M is the number of antenna elements with the same polarization in each column. The $M \times N$ elements may either be single polarized or dual polarized.

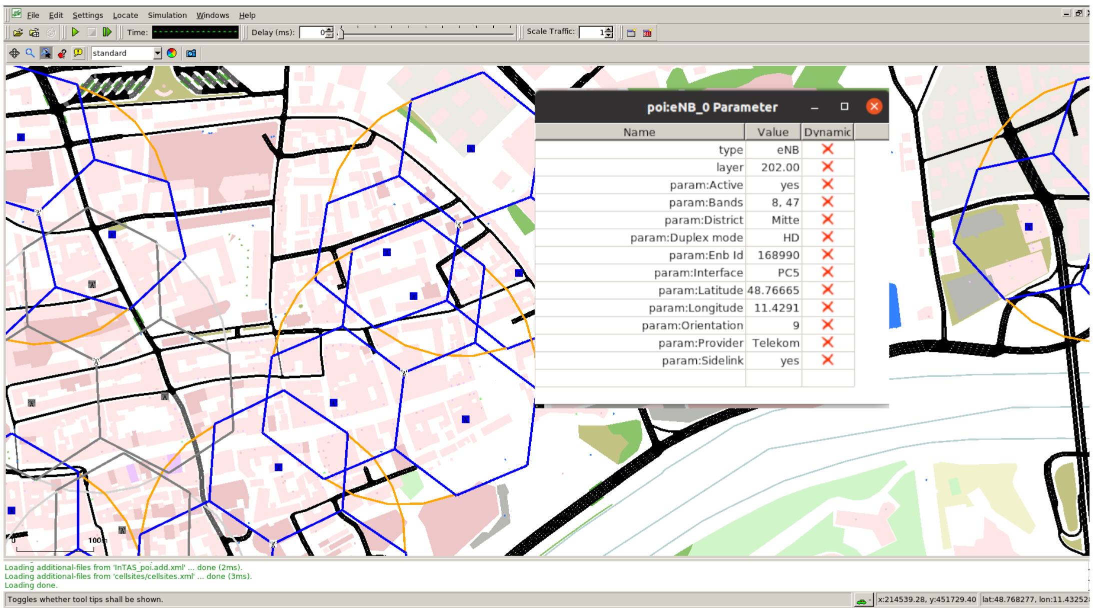
The antenna bearing is defined as the angle between the main antenna lobe centre and a line directed due east given in degrees. The bearing angle increases in a clockwise direction. Figure 5 shows the hexagonal cell and its three TRxPs with the antenna bearing orientation proposed for the simulations with three TRxP sites. The centre directions of the main antenna lobe in each TRxP point to the corresponding side of the hexagon. The antenna parameters for base station and end devices, i.e., eNodeB and UE, are listed in Tab. 1.

Table 1: Antenna-related simulation parameters

Parameter	**Equippment**	
	eNodeB	UE
Antenna gain [dBi]	18	0
Antenna profile	Omnidirectional and anisotropic	
Height [m]	25	1.5
Configuration (Tx, Rx)	$4X4$	$2X4$
Noise figure [dB]	7	5
Thermal noise level [dBm/Hz]	-174	
Mobility model	Stationary	Dynamic
Transmission bandwidth	20MHz, FDD	

4.4 Cell Association and Handover

When the vehicles enter the regions of cellular coverage, they associate with the appropriate base station dynamically by taking into account of two factors, namely *(i)* the distance from the



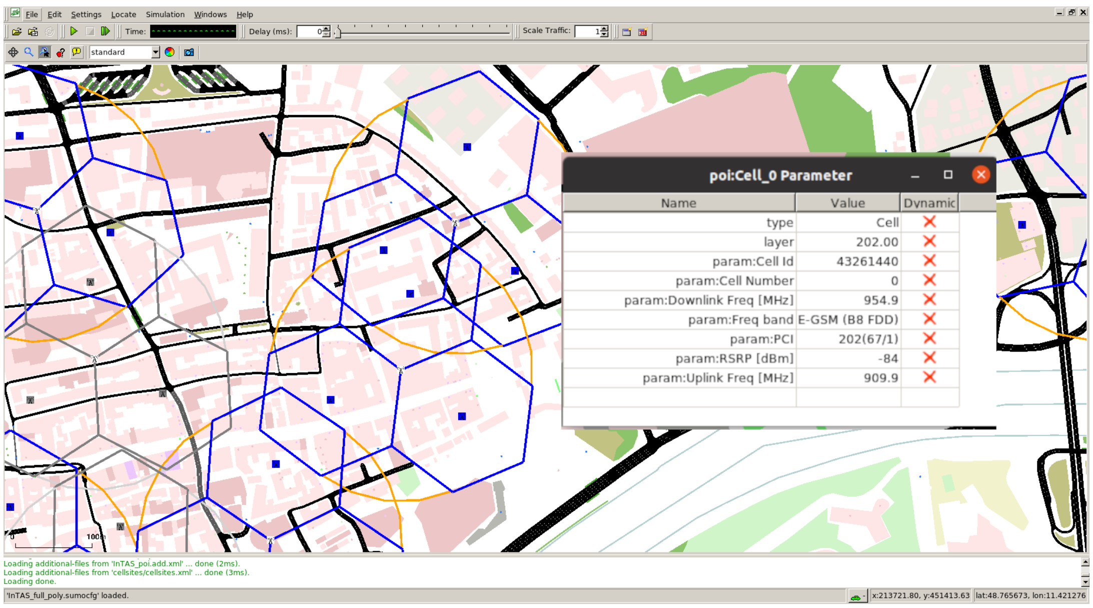
poi:eNB_0 Parameter

Name	Value	Dynamic
type	eNB	✗
layer	202.00	✗
param:Active	yes	✗
param:Bands	8, 47	✗
param:District	Mitte	✗
param:Duplex mode	HD	✗
param:Enb Id	168990	✗
param:Interface	PC5	✗
param:Latitude	48.76665	✗
param:Longitude	11.4291	✗
param:Orientation	9	✗
param:Provider	Telekom	✗
param:Sidelink	yes	✗

```
Loading additional-files from 'InTAS_poi.add.xml' ... done (2ms).
Loading additional-files from 'cellsites/cellsites.xml' ... done (3ms).
Loading done.
```

Toggles whether tool tips shall be shown. x:214539.28, y:451729.40 lat:48.768277, lon:11.43252

(a) eNodeB specific parameters



poi:Cell_0 Parameter

Name	Value	Dynamic
type	Cell	✗
layer	202.00	✗
param:Cell Id	43261440	✗
param:Cell Number	0	✗
param:Downlink Freq [MHz]	954.9	✗
param:Freq band	E-GSM (B8 FDD)	✗
param:PCI	202(67/1)	✗
param:RSRP [dBm]	-84	✗
param:Uplink Freq [MHz]	909.9	✗

```
Loading additional-files from 'InTAS_poi.add.xml' ... done (2ms).
Loading additional-files from 'cellsites/cellsites.xml' ... done (3ms).
Loading done.
```

'InTAS_full_poly.sumocfg' loaded. x:213721.80, y:451413.63 lat:48.765673, lon:11.421276

(b) Cell-specific parameters

Figure 6: Cell site-related simulation parameters of the cellular coverage layer in *SUMO*

base station and *(ii)* the received signal strength on the downlink interface. The vehicle attaches to the closest base station that has the highest received signal strength indicator (RSSI) value. The test environment shown in Figure 4 supports simulating the mechanism of handover using the direct interface among eNodeBs. In Figure 4, we can use different colors to depict the cells of different operators (e.g., blue color for Telekom and grey for Vodafone). The eNodeB and the cell-specific parameters are periodically retrieved from the simulation as the vehicles move from one cell to another. It is observed that in regions of overlapping cells and along the borders, the handovers are reported to be more frequent.

5 Proof-of-Concept

In order to extend the *InTAS* scenario for *Cellular V2X* communication, we have simulated a V2V sidelink broadcast communication network, in which both the sidelink modes – in-coverage and out-of-coverage – are operational and a vehicle can dynamically switch between these modes based on the availability of cellular network coverage. In the first step, we develop an SINR coverage map and further evaluate the impact of different types of interference in a given cellular network coverage area.

5.1 Vehicular Network Model

In the scenario, the road traffic flow with cars, buses and other vehicles is modeled according to the routing optimization methodology presented in the original *InTAS* scenario [10]. We set the speed of the vehicles to be in the range of $20 - 50\,\text{km/h}$ and do not consider pedestrian and other slow moving traffic. The smallest unit of simulation step is one millisecond, which corresponds to one transmit time interval (TTI)/subframe in the 3GPP standards for LTE. The simulation parameters related to the vehicular network model are listed in Table 2.

Table 2: Simulation parameters related to the vehicular network model

Use case	V2V sidelink broadcast
Mobility	Both Tx and Rx are moving
Interfaces	Up-/Downlink U_u and sidelink PC-5
Velocity [kmph]	20 to 50
Sidelink modes	In-coverage (mode 3) & out-of-coverage (mode 4)
Mode switching	Up-/downlink $\leftrightarrow$ sidelink (in-coverage), in-coverage $\leftrightarrow$ out-of-coverage
Carrier frequency	V2I: 5.9 GHz, V2V: 5.9 GHz
Channel model	Winner II
Selectivity	Time-frequency selective
Inter-cell interference	Only in-coverage mode

When a vehicle is located inside the region of cellular coverage, previous studies [4, 3] have shown that in-coverage mode is the preferred mode for sidelink communication. The reliability of packet reception is influenced by the quality of received signal strength from the associated base stations. The first step of validation aims at analyzing the received signal strength for an UE at various locations in a cell site – within the cell (inside the hexagon borders), at the cell borders and in the areas of overlapping cell borders. The factors influencing the signal-to-interference-and-noise-ratio (SINR) are distance between transmitter and receiver, multipath propagation and fading effects and inter-cell interference.

Based on the antenna characteristics for a single element antenna configuration and the chosen pathloss model from Winner II, we generate an SINR map as shown in Figure 7. We can observe that the regions with the maximum signal strength are the places in close proximity to the base stations along the antenna boresight angle. The red and the orange regions of the map indicate a value of SINR of around 10 to 15 dB. The blue shades correspond to areas with SINR in the range of 5 dB to -5 dB. The areas without any color on the map represent regions where SINR is below -5 dB threshold.

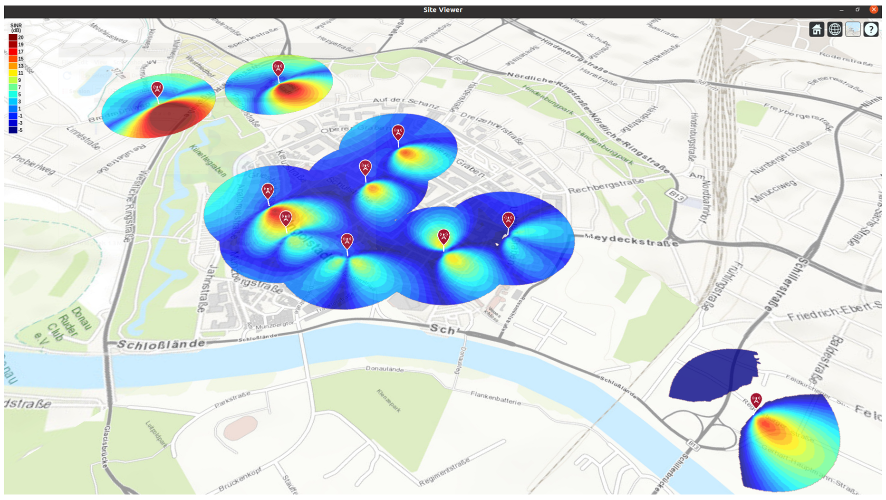
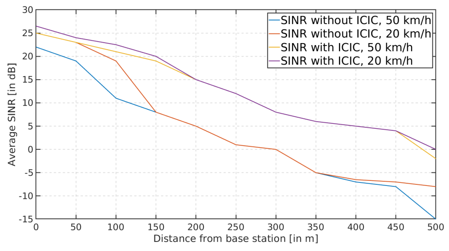
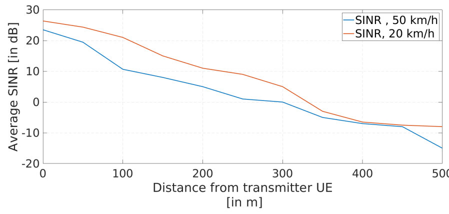


Figure 7: SINR for a single antenna element



(a) In-coverage mode



(b) Out-of-coverage mode

Figure 8: SINR analysis for sidelink modes for different vehicle speeds and considering ICIC

Based on the data collected from simulations, we generate plots to study the effect of SINR in the presence of Doppler shifts for both sidelink modes considering the boundary conditions listed in Table 2. For in-coverage operation, the vehicle has to associate with the base station and obtain radio resources for sidelink communication. The uplink (UL), downlink (DL) and sidelink (SL) SINRs play a significant role in the calculation of SINR for the in-coverage operation. The results in Figure 8a indicate that as the distance from the base station increases, the SINR experienced by the vehicles starts degrading. Especially at distances greater than 400 m, the vehicles experience inter-cell interference caused by neighbouring base stations. The overlapping of the cellular sites can be observed in Figure 7. When there is no inter-cell interference coordination (ICIC) employed, we observe that SINR reduces as low as -15 dB thereby making it difficult to associate with the base station and establish communication with other vehicles. By employing suitable ICIC mechanisms as proposed in *SimuLTE* [16] we observe the average SINR to be around -8 dB as indicated in Figure 8a.

In out-of-coverage mode as depicted in Figure 8b, the effects of inter-cell interference is not applicable. Here, we refer to the SINR computed on the sidelink interface in a V2V broadcast scenario. By taking the average value of SINR experienced by the receivers of the sidelink broadcast, we observe that when the distance between transmitter and receiver vehicle is greater than 400 m, the average SINR at the receiving vehicles is reported to be around -15 dB for vehicle speeds of 50 km/h and around -8 dB for vehicle speeds of 20 km/h. Although the effects of inter-cell interference does not come into discussion for the out-of-coverage mode, owing to losses due to multipath propagation in a dense urban environment with many obstructions such as buildings, bridges, signboards etc, the quality of SINR is highly affected.

For an UE moving along the cell borders, it becomes a difficult decision to whether continue operating in the in-coverage mode or switch to out-of-coverage mode because it is difficult to associate with eNodeB owing to bad link quality (low SINR). By carefully examining certain factors like distance from the transmitter, latency requirements of the V2V message and resource pool availability for both the modes, the mode switch decision has to be accomplished.

6 Conclusion

In this paper, we have presented a first approach to natively model an urban macro cell environment in *SUMO*. Unlike other approaches for coupling network simulators with *SUMO*, we have introduced a cellular coverage layer in *SUMO*, which enables *SUMO* to be aware of the cellular coverage and the link quality. The presented solution comprises a set of scripts to generate the hexagonal grid and files with details of base stations, cells and providers. We have added base station- and cell-specific parameters as attributes, which help to simulate different aspects of V2X communications such as cell association, handovers, radio resource monitoring, allocation, and utilization. We have coupled the *Cellular V2X* network simulator *Artery-C* with *SUMO* enhanced by the cellular coverage layer. *Artery-C* realizes different types of stochastic channel models which accurately calculate the SINR at various regions of the cellular grid. Having studied a *dense urban-emBB* test environment in this paper, the scripts developed for the generation of the hexagonal grid layout can be easily refactored to model other cellular test environments such as *rural-emBB* and *urban macro-mMTC*. Also, they can be directly applied to other existing *SUMO* road traffic scenarios. Last but not least, the proposed method can model other types of network infrastructure than LTE or 5G as in this paper such as WLAN. The cellular coverage layer in *SUMO* enables new ways for simulating microscopic road traffic by directly incorporating cellular networks as a integral part of the road traffic infrastructure comparable to traffic lights.

Acknowledgments

This work was supported by the German Research Foundation DFG within the priority program Cooperative Interacting Cars (CoInCar) (SPP 1835) and by the the Bavarian Ministry of Economic Affairs, Regional Development and Energy (StMWi) in the project “IN2Lab – Ingolstadt Innovation Lab”.

References

[1] 5G Automotive Association (5GAA). The case for Cellular V2X for safety and cooperative driving, 2016. Whitepaper, https://bit.ly/3fiK7kK.

[2] Anupama Hegde and Andreas Festag. Mode switching strategies in Cellular V2X. *IFAC Symposium*, 52(8):81–86, September 2019.

[3] Anupama Hegde and Andreas Festag. Artery-C: An OMNeT++ based discrete event simulation framework for Cellular V2X. In *ACM MSWiM*, November 2020.

[4] Anupama Hegde and Andreas Festag. Mode switching performance in Cellular-V2X. In *IEEE VNC*, December 2020.

[5] ITU-R. Guidelines for evaluation of radio interface technologies for IMT-2020, 2017. Technical Report M.412-0, https://www.itu.int/pub/R-REP-M.2412-2017.

[6] Sebastian Kühlmorgen, Patrick Schmager, Andreas Festag, and Gerhard Fettweis. Simulation-based evaluation of ETSI ITS-G5 and Cellular-VCS in a real-world road traffic scenario. In *IEEE VTC-Fall*, August 2018.

[7] Vineet Kumar et al. iTETRIS: Adaptation of ITS technologies for large scale integrated simulation. In *IEEE VTC-Spring*, 2010.

[8] Pekka Kyösti et al. WINNER II channel models, February 2008. Technical Report IST-4-027756 WINNER II D1.1.2 V1.2.

[9] Kwonjong Lee, Joonki Kim, Yosub Park, Hanho Wang, and Daesik Hong. Latency of Cellular-based V2X: Perspectives on TTI-proportional latency and TTI-independent latency. *IEEE Access*, 5:15800–15809, July 2017.

[10] Silas C. Lobo, Stefan Neumeier, Evelio M. G. Fernandez, and Christian Facchi. InTAS – The Ingolstadt traffic scenario for SUMO. In *SUMO Conference 2020*, October 2020. https://arxiv.org/abs/2011.11995.

[11] Pablo Alvarez Lopez et al. Microscopic traffic simulation using SUMO. In *IEEE ITSC*, November 2018.

[12] Brian McCarthy and Aisling O'Driscoll. OpenCV2X Mode 4: A simulation extension for cellular vehicular communication networks. In *IEEE CAMAD*, October 2019.

[13] Giovanni Nardini, Dario Sabella, Giovanni Stea, Purvi Thakkar, and Antonio Virdis. Simu5G – an OMNeT++ library for end-to-end performance evaluation of 5G networks. *IEEE Access*, 8:181176–181191, 2020.

[14] Raphael Riebl, Hendrik Günther, Christian Facchi, and Lars Wolf. Artery: Extending Veins for VANET applications. In *IEEE MT-ITS*, June 2015.

[15] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally coupled network and road traffic simulation for improved IVC analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15, 2011.

[16] Antonio Virdis, Giovanni Stea, and Giovanni Nardinis. Simulating LTE/LTE-Advanced networks with SimuLTE. In M. S. Obaidat, T. Ören, J. Kacprzyk, and J. Filipe, editors, *Simulation and Modeling Methodologies, Technologies and Applications*, pages 83–105. Springer, Cham, 2015.

[17] Axel Wegener et al. TraCI: An interface for coupling road traffic and network simulators. In *ACM CNS Symposium*, 2008.

SUMO User Conference 2021
https://doi.org/10.52825/scp.v2i.100

Published: 29 Jun. 2022

Simulation of Demand Responsive Transport using a dynamic scheduling tool with SUMO

Maria Giuliana Armellini

German Aerospace Center (DLR), Berlin, Germany
maria.armellini@dlr.de

Abstract

Demand responsive transport (DRT) has been increasingly tested and applied in recent years as a new form of transportation that seeks to address mobility problems in cities and rural areas. The planning of DRT systems is a challenging task for transport planners since the performance of the service depends significantly on the demand, how the scheduling is made, and how the routes are computed. Transport simulations are a useful option to evaluate these systems. The paper presents a Python tool, which aims to simulate di-verse DRT services using the software package for microscopic simulations Eclipse SUMO (Simulation of Urban MObility) as a framework. The fleet and requests of the DRT are handled dynamically by the scheduling module of the tool. This module is also responsible for calling a solver algorithm for the Dial-a-Ride-Problem (DARP), processing its results, and dispatching the DRT vehicles according to them. The tool also enables easier imple-mentation of other methods to solve the DARP. To demonstrate the use of the tool, a DRT service operating in two central neighborhoods of the city of Brunswick (Germany) is presented. The tool is called *drtOnline.py* and is included in SUMO since version 1.9.0.

Contents

1 Introduction

Demand responsive transport (DRT) has been increasingly tested and applied in recent years as a new form of transportation that seeks to address mobility problems in cities and rural areas. DRT seeks to serve trip requests from passengers using a fleet of vehicles without fixed routes. Depending on how the system operates and which constraints are considered, different types of DRT services can be given. For example, DRT can operate as a shared system, allowing the passengers to share their trips. DRT has also been studied to support public transport, being used as a transportation mode for the first or last mile. Depending on how the requests are made and managed by the service, the DRT system can be online (requests arrive and are managed in real-time), offline (requests are known in advance), or a mix of both [9].

Regarding the different forms of the DRT services, its planning is a demanding task for transport planners. The use of micro-simulations, like the open-source Eclipse SUMO (Simulation of Urban MObility) [2], constitutes an important tool for this aim. Since version 1.5.0 SUMO supports the simulation of demand responsive transport (DRT) via the taxi device. Vehicles with this device can receive trip reservations from persons in the simulation. The requests are processed by a dispatch algorithm that manages the route of each vehicle to serve the larger amount of requests under certain constraints. This problem statement is referred to in the literature as the Dial-a-Ride-Problem (DARP). SUMO counts with four different dispatch algorithms (greedy, greedyClosest, greedyShared, and routeExtension). However, due to the vast diversity of DRT services, these are sometimes not sufficient. The user has then the option to implement their dispatch algorithms using the TraCI API.

The taxi device is under continuous development. Since this year in version 1.9.0, the TraCI API enables the re-dispatching of taxis or DRT vehicles. This was an important step since it allows simulating shared DRT with a dynamic dispatcher. If the dispatcher finds a better route that can serve more requests, the vehicles can now change their route while driving.

In the literature, different methods to solve the DARP have been investigated. In [5] a study of different models and algorithms used in the literature to solve the DARP is presented. For large scenarios, finding an exact solution for the DARP can lead to long calculation times. For these cases, approximate solution methods, like metaheuristics are advisable. In [8], three different metaheuristics (Adaptive Large Neighborhood Search, Hybrid Bees Algorithm with Simulated Annealing, and Hybrid Bees Algorithm with Deterministic Annealing) were compared to solve a multi-depot and multi-trip heterogeneous DARP. [6] applies a multi-objective optimization to solve the DARP with time windows, whereas [7] implements an online rejected-reinsertion heuristics to solve a multi-vehicle DARP. Finally, other heuristics methods to solve the static and dynamic DARP are presented in [10]. The performance results of the DRT can depend significantly on the method used to solve the DARP, for which its election plays an important role.

To simulate DRT not only a DARP solver is needed, but also a scheduling module for the management of the fleet and reservations.

Based on this, a Python tool to simulate different types of DRT in SUMO was developed. The tool uses the taxi device and TraCI to implement a scheduling module that calls a DARP solver to simulate DRT services. The same scheduling module can be used with different DARP solvers to simulate different DRT services. It also allows users to implement their dispatching algorithms more easily and faster, and to compare their results with other methods. Since the dispatcher algorithms that are currently included in SUMO are not well suited to simulate shared DRT modes, a DARP solver to simulate these services is implemented as a first option.

In the next section, the tool and its modules are explained. After this, a user example is shown and finally, the conclusion and further work are presented.

2 Methodology

The aim of the proposed Python tool is not only to simulate shared DRT services but to allow the users to compare the performance of different DARP solvers and to enable easier implementation of its solvers. The tool was written in Python3 and uses the TraCI API to control the DRT requests and vehicles in real-time.

The tool consists of two modules: the scheduling module and the DARP solver. The inputs are the standard SUMO files. The tool needs at least information about the DRT vehicles, which are the ones equipped with a taxi device, and information about the requests, which are modeled as persons having a ride element with the DRT lines. The scheduling module starts the simulation via TraCI and manages the DRT requests and fleet. This means that the module detects when a new request arrives or requests are waiting to be served and it calls the DARP solver to find the best route for each DRT vehicle. What makes a route the best, depends on the model of the DARP solver. The solver returns the routes to the scheduling module and dispatches the DRT vehicles with them. The tool has with a default DARP solver, which allows it to simulate shared DRT services (persons can share a trip).

The tool also accepts adding the surrounding traffic and other SUMO elements to the simulation. This allows obtaining more realistic results for the performance of the DRT service.

In the following sections, the scheduling module and the DARP solver of the tool are explained in detail.

2.1 Scheduling module

Figure 1 shows a flowchart of the scheduling module. The first step is to call TraCI with the given inputs. The required inputs are the SUMO network, and the DRT vehicles and requests. As mentioned before, other SUMO inputs (e.g. surrounding traffic) can be given. The tool also supports the use of SUMO *configuration-files* to manage the simulation options.

Once TraCI has been called, the simulation starts. For each simulation step, the tool checks if new reservations have arrived or if some reservations from the last steps have not been yet assigned to a vehicle. If this is not true, then TraCI calls the next step.

The new reservations are retrieved using the TraCI call '*traci.person.getTaxiReservations(1)*'. If new reservations have been made, the tool processes these and adds them to the reservation pool. The process of a reservation consists not only in reading the minimal required parameters (departure time and origin and destination edges) but also in adding new parameters that are required for the simulation. At least four new attributes are added to each reservation element:

- direct attribute: the travel time from origin to destination taking a direct route,
- the vehicle attribute: includes the ID of the vehicle, when the reservation is assigned to one, and
- the time windows for pick up and drop off: set the allowed times for the DRT vehicle to pick up and drop off the person.

Depending on the DRT service, the definition of the requests may require other attributes. This can be defined using SUMO *GenericParameters*. After a reservation is processed, it is added to the reservation pool, which contains all current reservation elements.

Next, the DRT fleet is retrieved using '*traci.vehicle.getTaxiFleet()*'. This retrieves the ID of all vehicles in the simulation that are equipped with a '*taxi.device*'.

As a further step, the module checks if reservations have been served or rejected during the last step of the simulation and if so, these are removed from the reservation pool. The rejected reservations or persons are then removed from the simulation. The remaining reservations are then classified by:

- new: if the reservation has been made during the current simulation step,
- unassigned: if the reservation has not been yet assigned to a vehicle,
- assigned: if the reservation is included in the route of a DRT vehicle, and

- picked up: if the reservation has already been picked up by a vehicle and it is on the route to its destination.

The reservations that have been already picked up but not yet dropped off are retrieved using the TraCI call '*traci.person.getTaxiReservations(8)*'.

This classification plays an important role for the DARP solver, which is called in the next step. The DARP solver finds the best routes for each vehicle, based on the DRT fleet, the reservations, and the defined constraints.

It is possible that the DARP solver only finds all possible routes, with their respective costs, for each DRT vehicle and then, the best routes should be found using an ILP (integer linear programming). For this purpose, the Python LP solver PuLP [1] is included in the scheduling

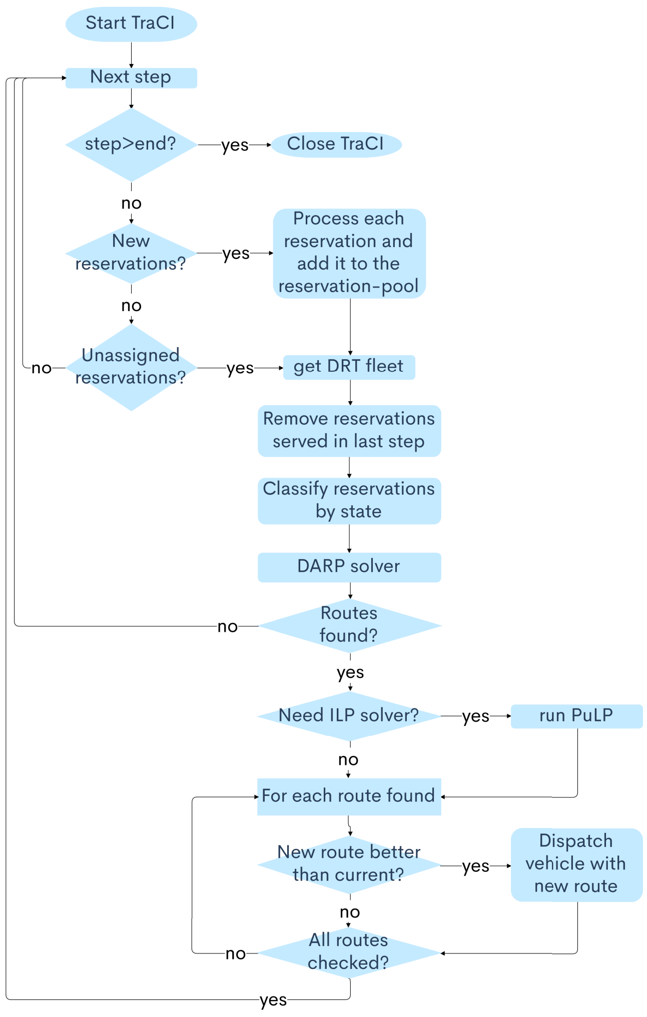


Figure 1: Flowchart of the scheduling module

[1]https://coin-or.github.io/pulp/ last access 17. May 2021

module. After calling the DARP solver, the module checks if an ILP is needed and if so, the PuLP solver is called.

Once the best routes have been found, the module will compare them to the current ones. This step is relevant since many DARPs are not solved with exact methods. As a result, the new best route of a vehicle can contain the same reservations but with a different order, which may not be better than the current one. To avoid this, the module compares the current route of each vehicle with the new best route found. If the new route is better, then the vehicle is rerouted using the TraCI call '*traci.vehicle.dispatchTaxi()*'.

When a vehicle is dispatched with a new route, the ID of the vehicle is added to the attribute '*vehicle*' of the reservations that are on the route.

After all the found routes are checked, TraCI calls for the next step and the loop starts again.

When the simulation step reaches the given end-time, the simulation is ended and TraCI is closed.

2.2 DARP solver

Having a group of reservations and vehicles that should serve them, the DARP solver will find the best route for each vehicle under certain constraints. There are different methods to solve a DARP, starting with the differentiation of an exact or approximate method, to which constraints should be considered (e.g., time windows constraints) and which objective should be optimized (e.g. passenger waiting times and vehicles idle times). Depending on the solver, the simulation results of the same DRT service can be different.

The goal of this tool is therefore not only to offer certain DARP solvers but also to allow the user to implement its algorithms. To solve the multiple vehicle DARP, many methods find as a first step all possible routes with their costs and then call an ILP solver to find the best ones. For this, the ILP solver PulP has been included as an option in the scheduling module.

The tool counts with a default DARP solver that simulates shared DRT. The solver applies the method of [1], which was also implemented in previous DRT studies [4, 3], but for solving the static or offline DARP case. This means that all reservations were known in advance and are not being processed in real-time or online, as is the case for the present tool.

The solver is called from the scheduling module and as input, the reservations, the DRT fleet, and the constraints are given. The solver searches first for all possible combinations between a vehicle and a reservation and between two reservations. A combination is possible if the pick-up and drop-off time windows can be regarded. For example, if the origin of reservation 2 wants to be paired with the destination of reservation 1, but the latest drop-off time of reservation 1 is at 10:00 and the earliest pick-up time of reservation 2 is at 10:05, then the pair is not possible. Reservations that have already been assigned or picked up by a vehicle can no longer be combined with another vehicle.

After searching all possible pairs, a pair-wise graph with each pair and its travel time is saved. The travel time is calculated using the TraCI call '*traci.simulation.findRoute()*', which considers the surrounding traffic for the calculation.

Next, the pairs of the mentioned graph are explored to find feasible trips for each vehicle. This proceeds incrementally in trip size (i.e. number of stops), starting from the vehicle-request pair. For each size, an exhaustive search is conducted. The time complexity of the algorithm depends on the number of vehicles and requests, and their shareability. If all vehicles can serve all requests and all requests can be combined with each other, the complexity will be $O(mn^v)$[1]. To allow for a faster (but not exact) solution, a timeout for the search of trips of each size can be set.

A trip is primarily feasible if the requests are picked up and dropped off between the specified time windows and if the capacity of the vehicle is not surpassed. Since the simulation of the DRT is dynamic, this means that vehicles can be rerouted, other considerations have to be made. If some reservations have been assigned to a vehicle in a previous step, then these reservations have to be included in the new possible routes. Nevertheless, the order of the stops may be changed and new reservations may be added.

Idle vehicles with the same current edge and capacity will have the same possible trips. To avoid searching for the same trips multiple times, the exhaustive search will be performed only once and then the trips will be transferred to the other vehicles. In scenarios with large fleets and repeated origin or destination edges, this will speed up the simulation time notably.

Once all possible routes with their travel times are found, the best route for each vehicle that optimizes the entire DRT service has to be found. This optimization problem can be solved with an ILP solver. As the last step, the routes are written in a format that can be processed by the ILP solver module PuLP and are returned to the scheduling module.

3 Application example

As a study case, a DRT service operating in the city center (Innenstadt) and in the Nordstadt neighborhood of the city of Brunswick (Germany) was simulated. The operating area of the DRT service is shown in figure 2. The demand and sumo network were adapted from the publicly available Brunswick scenario [2], which simulates the motorized individual traffic in the city for a typical working day. For the DRT simulation scenario, a simulation time window between 6 a.m. and 10 a.m. was considered. An extra warm-up and cool-down time of 30 minutes each was adopted.

The requests to the DRT service stem from the demand of the Brunswick scenario. First, only persons that have made all their daily trips inside the DRT operating area were considered to use the DRT service instead of the private car in the simulation. For this, each of its trips was written as a DRT request. Between 6:00 and 10:00 in the morning, 758 persons made 956 trip requests. Once the reservation is made, the person should be picked up in the next 15 minutes. If there is no DRT vehicle available for this, the reservation is rejected and the person is removed from the simulation. In addition, to offer a trip with the DRT service, the estimated travel time should not be longer than twice the direct travel time with the private car (equivalent to a detour factor of 2) or surpass it by 10 minutes.

The DRT fleet consists of 50 vehicles with a capacity of 6 passengers each, allocated in three different depots (see figure 2). When the simulation starts, the vehicles wait for the requests at the given depot. The simulation step was set to 10 seconds, which means that the reservations are also processed within this time-step. Figure 3 shows the simulation. The red vehicles represent the DRT fleet and the blue points are the persons making the requests. The surrounding traffic was hidden in the figure for a better overview.

The described simulation scenario was run 5 times. Since the proposed DRT tool only controls the routing of the DRT vehicles and passengers via TraCI, all standard SUMO outputs (i.e. trip-information, stop-information, and emissions output) can be generated. In the following, some examples of the analysis that can be carried out with the tool are shown.

[2]https://github.com/DLR-TS/sumo-scenarios/tree/master/brunswick/miv last access 17. May 2021

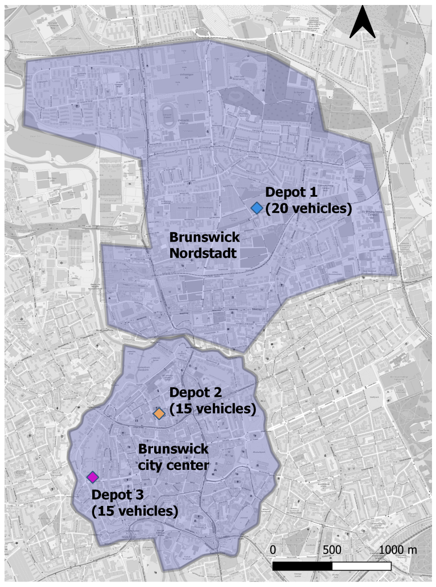


Figure 2: DRT operating area (background image from OSM)

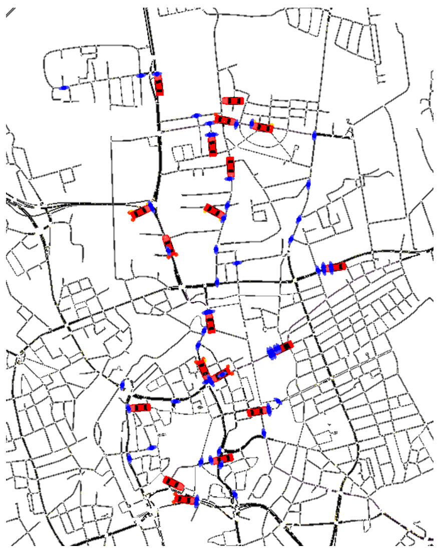


Figure 3: DRT sumo-simulation

3.1 Results

The results presented below were obtained from the simulation-log, trip-information, and stop-information outputs. Table 1 shows the results regarding the performance of the simulation and the DRT fleet for each time the scenario was run.

The five hours simulated (from 5:30 to 10:30) had an actual average duration of 95 minutes (1.58 hours), which is equivalent to a real-time factor of 3.20. According to the results, not all 50 DRT vehicles were used. The maximum number of vehicles used was 46 vehicles for the third simulation run. The mileage of all DRT vehicles was on average 2765.78 km. The total mileage of the first simulation run was 2757.73 km. The sum of the direct route lengths of the passengers served in this run was 1534.19 km. Hence the mileage of the DRT service is equivalent to 1.79 the mileage of the private cars. But the number of vehicles used is reduced from 758 in the case of all passengers using the private vehicle to only 43 DRT vehicles. In the five simulation runs, not all 956 trip requests were able to be served. This can be related to

Simulation run	1	2	3	4	5
Duration [min]	113.96	91.56	87.00	98.20	82.18
Real time factor	2.63	3.27	3.44	3.05	3.65
DRT vehicles used	43	42	46	43	41
Mileage DRT fleet [km]	2757.73	2709.89	2811.28	2799.67	2750.33
Max. passengers at same time	4	3	3	4	3
Requests served	871	835	869	876	884
Requests rejected	85	121	87	80	72

Table 1: Simulation performance and DRT fleet

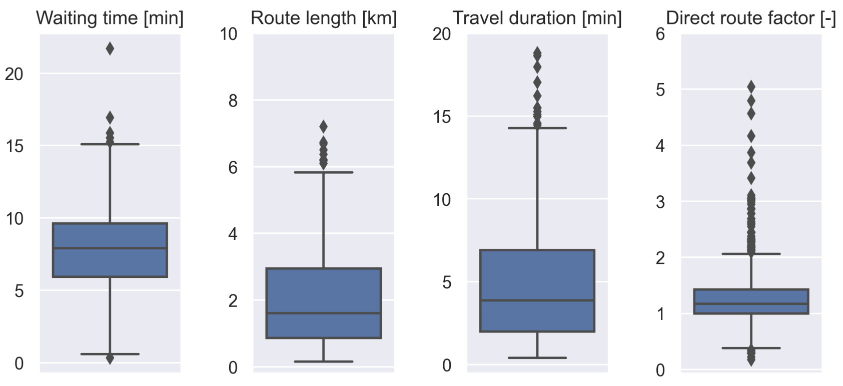


Figure 4: Simulation results for passengers as mean value of simulation runs

a longer than 15 minutes travel time from the depots to the pick-up of these passengers. But in the case of simulation run 3, where more requests were rejected, these extra rejections are most probably related to the ability of the proposed tool to find a compatible route (solve the DARP problem). This can be improve by setting a higher timeout to solve the DARP. The DRT vehicles have a capacity of 6 passengers, but as it is shown in the table, the maximum number of passengers at the same time was only 4.

Figure 4 shows the waiting time, route length, travel duration, and the direct route factor (relation between the travel duration with the DRT and the private car) for each passenger as box plots. The results for each passenger represent the mean value of the 5 simulation runs. The first plot shows the waiting time for the pick-up. As it was mentioned previously, this time should not exceed 15 minutes. Only in a few cases, this was not the case and it can be related to delays and congestion. On average the waiting time for pick up was 7.8 minutes. The direct route factor (last plot) was limited to a value of 2 or for the case of short routes, to 10 minutes longer travel time than with the private car. On average, the direct route factor was 1.28. But in some cases, it was greater than 2, which is related to the cases with short travel duration.

4 Conclusion and future work

This paper presents a SUMO Python tool to simulate Demand Responsive Transport (DRT) with a dynamic dispatcher using the SUMO taxi device and TraCI. The tool consists of two modules: a scheduling module, where the fleet and requests are managed and vehicles are dispatched, and a DARP solver module. The latter calculates the possible routes for the DRT vehicles by solving a Dial a Ride Problem (DARP).

The tool aims to simulate complex DRT systems and to make it easier for the users to implement their methods for solving the DARP.

The tool includes a DARP solver that enables the simulation of shared DRT. As an application example, a DRT service operating in the Innenstadt and Nordstadt neighborhoods of the city of Brunswick was simulated. The DRT service consists of a fleet of 50 vehicles with a capacity of 6 passengers each. A total of 758 persons made 956 trip requests between 6:00 and 10:00. The tool supports the global configurations and outputs that SUMO allows for persons and vehicles with taxi devices. For the application example, the surrounding traffic was considered in the simulation for a more realistic result. Only 43 of the 50 DRT vehicles available were used to serve 871 requests. 85 requests of the 956 (8.89%) had been rejected by the service. The reason for the rejections is most probably related to the limited time of 15 minutes for the pick-up. These results are only an example of the analysis that can be made with the proposed DRT tool. The simulation time was on average 1.58 hours, which is equivalent to a real-time factor of 3.20.

The current DARP solver is not a good solution when dealing with large scenarios, for example, for the simulation of DRT services at a city scale. According to the literature, other heuristics methods like Variable Neighborhood Search (VNS) or Adaptive Large Neighborhood Search (ALNS) are a better solution for this purpose. As further work, the implementation of such methods, as well as the extension of the tool to manage other DRT options (e.g. first/last mile DRT, multiple person booking) are planned.

References

[1] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. January 2017.

[2] P. Alvarez Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[3] Maria Giuliana Armellini. A tool for simulating demand responsive transport systems in sumo. In *7th International Conference on Models and Technologies for Intelligent Transportation Systems, MT-ITS 2021*, 2021.

[4] Maria Giuliana Armellini and Laura Bieker-Walz. Simulation of a demand responsive transport feeder system: A case study of brunswick. 2020.

[5] Jean-François Cordeau and Gilbert Laporte. The dial-a-ride problem: models and algorithms. *Annals of operations research*, 153(1):29–46, 2007.

[6] Francesca Guerriero, Ferdinando Pezzella, Ornella Pisacane, and Luigi Trollini. Multi-objective optimization in dial-a-ride public transportation. *Transportation Research Procedia*, 3:299–308, 2014. 17th Meeting of the EURO Working Group on Transportation, EWGT2014, 2-4 July 2014, Sevilla, Spain.

[7] Ying Luo and Paul Schonfeld. Online rejected-reinsertion heuristics for dynamic multivehicle dial-a-ride problem. *Transportation research record*, 2218(1):59–67, 2011.

[8] Mohamed Amine Masmoudi, Manar Hosny, Kris Braekers, and Abdelaziz Dammak. Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. *Transportation Research Part E: Logistics and Transportation Review*, 96:60–80, 2016.

[9] Yves Molenbruch, Kris Braekers, and An Caris. Typology and literature review for dial-a-ride problems. *Annals of Operations Research*, 259(1-2):295–325, 2017.

[10] André Luyde S Souza, Jonatas BC Chagas, Puca HV Penna, and Marcone JF Souza. A hybrid heuristic algorithm for the dial-a-ride problem. In *International Conference on Variable Neighborhood Search*, pages 53–66. Springer, 2019.

SUMO User Conference 2021
https://doi.org/10.52825/scp.v2i.107

Published: 29 Jun. 2022

A comparison of SUMO's count based and countless demand generation tools

Michael Behrisch[1] and Pauline Hartwig[2]

[1] German Aerospace Center, Berlin, Germany
michael.behrisch@dlr.de
[2] TH Wildau (University of the Applied Sciences), Wildau, Germany
paha2034@th-wildau.de

Abstract

There are already several tools available to generate traffic demand for the microscopic simulation suite SUMO. This paper focuses on setting up a simulation scenario for the peak hour in a small conurbation when there are vehicle counts available for the major streets. We describe tools which are part of SUMO or available as open source and compare their results with the real traffic counts as well as with the outcome of countless demand generation.

1 Introduction

Setting up a microscopic traffic simulation scenario is hard work. While it is easy to get started with a network derived from OpenStreetMap, fixing the network to fit microsimulation purposes and getting a proper traffic demand are two tasks which can consume considerable amounts of time. Fortunately there are tools to support both tasks which can ease the work. While the network adaption is still largely a manual process supported only by better editing facilities in tools such as netedit, traffic demand generation can be done in an automated way and still deliver good quality results. This paper focusses on several demand generation tools usable with the SUMO simulation suite [2]. It does not cover "traditional approaches" such as setting up an origin destination matrix, generating trips from there and then doing traffic assignment. These tools are described in detail in the available literature for an overview see [3].

We instead focus on the quick (and sometimes dirty) solutions to give people an immediate but still somehow realistic travel demand for their area of interest assuming no additional input beyond the OpenStreetMap network and a few counting locations (which are even optional for some of the tools). The motivation is to replace as much of the manual work as possible why still generating plausible routes, realistic counts and a good coverage of the network.

The paper is structured as follows. After describing the solution approaches of the various tools with and without counting input and give a brief account on their strengths and weaknesses, we compare their results to the counting data we used as input and two additional counting locations not used as input. Furthermore we address the problem of routes primally running on the main network by evaluating the network coverage. We end with conclusions and an outlook for further research.

The underlying scripts and results are all publicly available in the sumo scenario github repository [1].

2 The scenario

The tools we evaluate need a realistic yet small use case such that all basic features can be demonstrated but it is still feasible to do manual adaptions to the network and evaluate the data if needed. For these reasons (and also because of geographic proximity) we used the network of the town Wildau (Germany, Brandenburg), which was extracted from OpenStreetMap using the OSM WebWizard. Wildau is a small town in Brandenburg/Germany which has around 10000 inhabitants and a surface of 909 ha. The generated cutout as visible in Figure 1 includes a total edge length of 104.06 km and 645 nodes and 1426 edges.

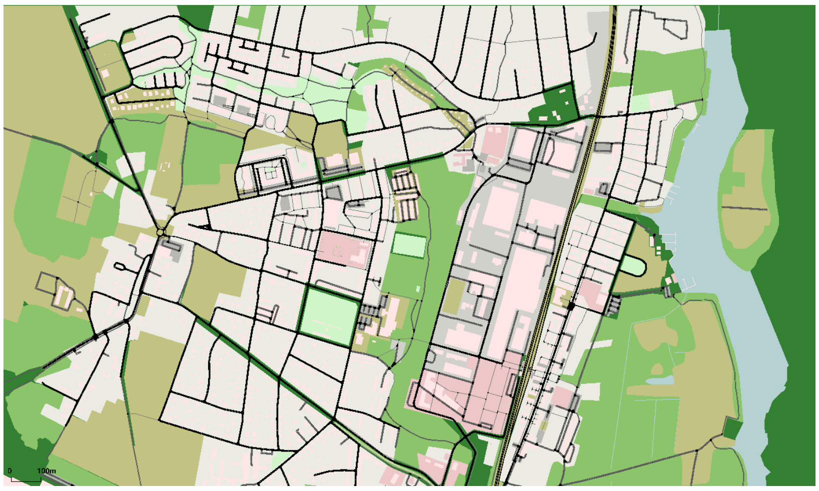


Figure 1: SUMO scenario

After the automated extraction from OSM the next step was to clean up the network mostly for faulty or missing lane to lane connections at the junctions. This is a necessary step in most simulation setups. If there is no knowledge of the local situation the easiest way to find the wrong connections is to generate a large random demand and look for the junctions which jam first. This already gives a usable network for our purposes. Since we strived for more realism, the maximum speed and the allowed vehicle classes have been adapted too (especially concerning bicycles).

The generated demand only covers the afternoon peak hour (15:00 to 16:00). The reference data for the number of (passenger) cars has been collected at nine counting points provided by local authorites and one manual counting point as visible in Figure **??**. The initial approach (and baseline for our studies) was a manually generated flow file, which includes routes based on assumptions about the local demand situation. The number of vehicles using the different routes had been scaled using Excel based optimization tools to adapt them to the counts. This resulted in a total demand of 2254 vehicles in the initial route set.

3 Countless tools

The easiest way to generate a demand would be if the user needed no extra data or could at least enter just a single number as the expected population or the total amount of vehicles per hour. We call these tools *countless* and evaluated them first for two reasons. First they are obviously easier to use and second while their output is not perfect it may still serve as a base for optimization in further steps.

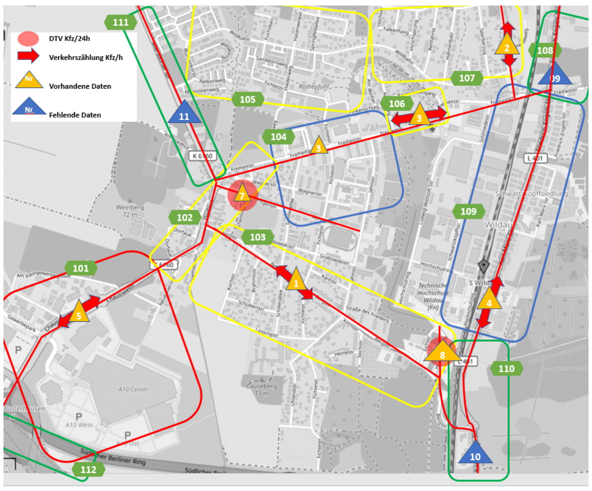


Figure 2: Counting locations

3.1 randomTrips.py

The tool randomTrips.py is part of the core SUMO suite and probably for most people the first contact with demand generation, because it also works behind the scenes in the osmWebWizard tool. As the name says it randomly generates trips (origin destination pairs of edges) and with the right options also completes them to full routes and validates them for drivability (mostly connectivity) within a network.

The selection process can be tailored heavily to prefer edges with more lanes, higher speeds or greater length but in our experiments we stuck to the defaults for most of these values. The subsequent route generation is done using a hidden call to the duarouter which runs a shortest path calculation to find the route set and discards all disconnected pairs of origin and destination.

To adapt the traffic volume two major parameters have been changed, first the fringe factor (increases the probability that journeys will end / start at a boundary edge), and the period (how often new vehicles are generated). Already the first runs showed that randomTrips is able to generate a better network coverage than the manual routes which only focussed on counting locations. This is not surprising since a random choice will obviously cover more than just the major streets.

Subsequently, the value of the "period" was reduced in order to generate more traffic. The procedure was repeated until the first heavy jams occured at the junction Bergstrasse / Dorfaue. In the second step, the boundary edges were examined, which initially showed a low level of utilization. In order to be able to adapt this, the "fringe factor" was increased. The value was increased until there were no more vehicles teleported in combination of the "period". In the end, with the values $f = 7$ and $p = 1.850$, a relatively good distribution picture of the traffic was generated.

Using the option "random" (to modify the random seed and hence the sequence of generated start and end points) several different demands have been generated to get an average result for the evaluation. To calculate the total counts on the edges with counting locations the meanData mechanism of SUMO has been used which gives aggregated data for the whole network.

3.2 SAGA

SumoActivityGen (SAGA) is a tool which takes only the extracted OpenStreetMap file as input and tries to generate the complete demand automatically. Public transport, motorized vehicles and pedestrians are generated here. The time span comprises a complete daily cycle. The route generation is based on possible work routes. The tool “looks” where workplaces, houses, etc. are located and determines the routes based on this information from OSM. The tool was used on Wildau, however the ”raw” network was used imported by OSM WebWizard. To let SAGA generate any demand at all for the small network we decided to start it with the ”single-taz” option because it otherwise tries to generate demand only between districts (in the case of Wildau between the city itself and the surrounding ones) but not inside the district.

To calculate the total counts on the edges with counting locations we again used the meanData mechanism of SUMO focussing on the peak afternoon hour. The total number of vehicles generated by SAGA was much less than in the manual demand file, details will be dicussed in section 5.

3.3 randomActivitygen

A tool similar to SAGA is ”randomActivityGen”. It serves as a randomizing wrapper to ”activitygen” which is part of the core SUMO suite. It takes the SUMO network and the number of inhabitants as input and generates random work and home locations to induce a traffic demand.

When evaluating this tool, it became apparent that the number of vehicles generated was very low. The number of inhabitants was initially given as 10000, which is close to real population. whereby the number was increased to 50,000 when called up again. The number of vehicles increased slightly, but the traffic generated was still implausible. This is probably due to the fact that the tool does not take into account surrounding cities / factors. Due to its close location to Berlin, Wildau has a high proportion of through traffic / commuter traffic, which means that the traffic flows would have to be correspondingly higher.

4 Tools using traffic counts

4.1 dfrouter

In addition to the tools already used, other methods were also used. First, the ”dfrouter” was implemented, which receives defined (induction) loops with the relevant edge IDs and a vehicle file in semicolon-separated format with the number of vehicles as input. The basic idea behind this tool is that incoming and outgoing flows can be recorded almost completely using induction loops. With the help of the information, the dfrouter rebuilds the number of vehicles and routes. This is done in four processing steps. First of all, it must be specified in the loop file whether it is a source, target or intermediate detector. In the next step, routes between the detectors are calculated, whereupon the flow rates are generated. To generate the flows dfrouter starts at every source with the demand measured there and distributes it according to the downstream detectors untila sink detector is reached.

Except for determining manually which detectors are considered sources and sinks there is no easy way to influence the behavior of the process so the results are basically stemming from a straight invocation of the tool. The dfrouter generated the routes and vehicles as output files and the result has been evaluated again by running them through a scenario and measuring edge usage using meandata files.

4.2 flowrouter

In a second method, the "flowrouter" was used, which works in a similar way as the "dfrouter". The flow router tries to find a number of routes that maximize the overall flow in the graph theoretic sense. So it treats the measuremnts as capacities for the edges and maximises the number of vehicles in the network which travel from source to sink edges. The input remained the same, but instead of individual routes, "flows" are generated for the vehicles. The traffic flows received were evaluated using a further output file and transferred to Excel. It must be mentioned here that both tools mainly use the edges covered by detectors (and the ones connecting them) and do not generate any traffic into the secondary network.

4.3 cadyts

Another option for route selection within a network, which is based on counting data and route alternatives, is offered by the "cadyts" tool. Cadyts performs 100 iteration steps in which the probabilities of the route selection from the route files are tried to match the counting data. For this purpose, the results of the previous iteration are evaluated in each iteration step and the probabilities are adjusted if necessary, in order to get close to the counting data. In order to be able to execute cadyts, however, duaIterate must be executed first. DuaIterate creates an alternative route file for each iteration (the number of iterations depends on the time slot), which is then passed on to cadyts. There are a variety of parameters to be adjusted when using duaIterate as well as cadyts to influence the calculated user equilibrium. Due to the limited time the calibration work could not be finished and as a result the deviations were still very high.

4.4 routesampler

In a further attempt, the "routesampler" was used to assign the counting data to the corresponding edges so that the randomly selected routes match the counting data. In contrast to the other tools the routesampler already receives a route file as input together with the usual "counting data". Its algorithm chooses iterativela routes from the route file to match the counting data. While the usage of an exisiting route file sounds like a severe limitation it is possible to use the output file from "randomtrips" as input for the model and still get proper results. This route file can be used together with a counting file in the meandata format (which has been generated manually) to start the calibration. By calling up the changed batch file again (now with the "routesampler"), the new scenario was generated and could be evaluated using a new output file. The result was that the routesampler can match the counting data, which were given as input.

5 Results

The results were evaluated with respect to two different indicators which are the precision of the fit (denoted by the root mean squared error of the measured counts in relation to the simulated values) dureing the peak hour and the coverage of the network (total length of streets used by simulated cars divided by overall length of car edges in the network).

With the help of the root mean square it became clear that the routesampler was the most suitable tool to achieve the counting data and a plausible route distribution. A python script was created for the evaluation of the individual tools, which calculates the root mean square of the counting data. Here, the count data were considered with the output data of the edges and the quadratic deviation was calculated. In the end, the sum of the squared deviations was calculated. The lower the value, the better the tools were. During the evaluation the routesampler and the manual route data achieved the lowest values, which means that these methods were the best during the test. On the other hand, SAGA and randomActivityGen were not suitable for generating demand due to their high values. However, the value of randomActivityGen with 50.000 inhabitants was slightly lower than 10.000 inhabitants.

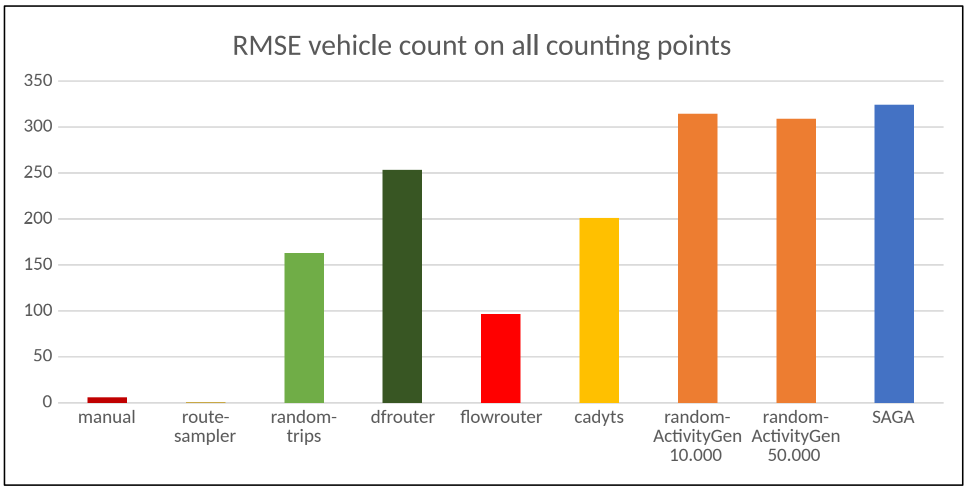


Figure 3: Root mean squared error of the vehicle counts

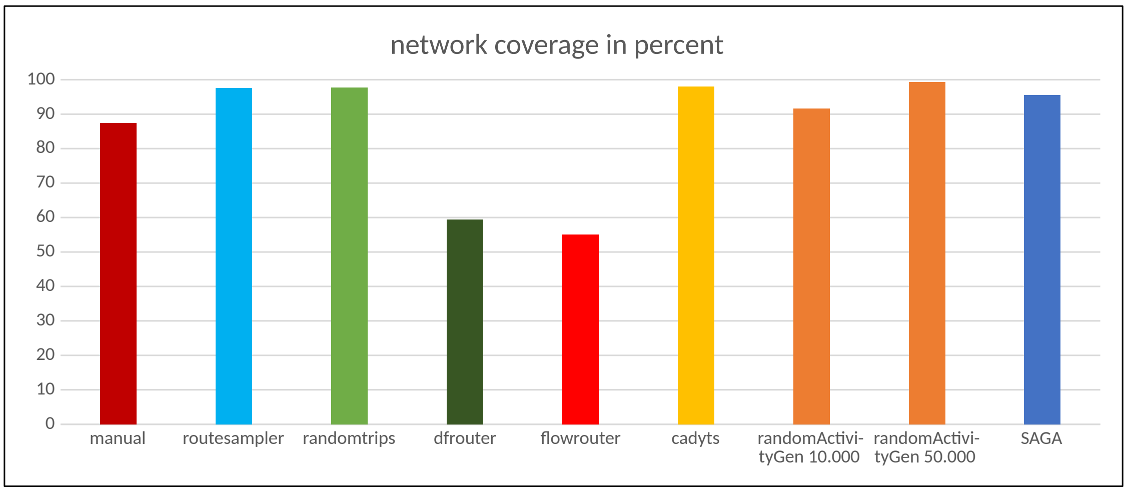


Figure 4: Percentage of the street network covered by vehicle traffic

The conclusion is that the routesampler is the best way to simulate the demand however the route distributions (main and secondary network) are balanced.

6 Future Work

In the future it is planned to expand the simulation on github. On the one hand with the local public transport and on the other hand with the bicycle and pedestrian traffic. Furthermore, the presented tools are to be further tested and evaluated with the help of this paper. Here it is conceivable to change the parameters, in which it is possible to find better results. The paper is an invitation to work with the tools interactively. Maybe some users find other methods to work with them or have some ideas for new definitions for parameters.

Furthermore the results should be used to foster automatic creation of realistic scenarios. The manual steps which were still necessary for instance when calibrating the randomTrips scenario could be partly automated at least the step of increasing the traffic until the first teleports (i.e. heavy jams) occur. It is probably not possible to automate the choice of the fringe factor though because it deponeds more onthe surrounding network than on the chosen network itself.

Last but not least further insights in the process of the user equilibrium would allow to do a joined process between duaIterate/cadyts and the routeSampler or do a better comparison of both tools and processes in the future.

7 Acknowledgments

We want to thank the city and the administration of Wildau as well as the University for providing the necessary data to perform this study. Furthermore this project has been initiated by a student project together with Ewald Bader and Andesson Wafo who contributed to the first scenario.

References

[1] SUMO scenarios. https://github.com/DLR-TS/sumo-scenarios. Accessed: 2022-06-17.

[2] P. Alvarez Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems.* IEEE, 2018.

[3] Dieter Lohse and Werner Schnabel. *Grundlagen der Straßenverkehrstechnik und der Verkehrsplanung: Band 2-Verkehrsplanung.* Beuth Verlag, 2011.

SUMO User Conference 2021
https://doi.org/10.52825/scp.v2i.120

Published: 29 Jun. 2022

Automated Calibration of Traffic Demand and Traffic Lights in SUMO Using Real-World Observations

Michael Harth[1,2], Marcel Langer[1,2] and Klaus Bogenberger[2]
[1]AUDI AG
[2]Technical University of Munich
michael.harth@audi.de, marcel.langer@audi.de,
klaus.bogenberger@tum.de

Abstract

Virtual traffic environments allow for evaluations of automated driving functions as well as future mobility services. As a key component of this virtual proving ground, a traffic flow simulation is necessary to represent real-world traffic conditions. Real-world observations, such as historical traffic counts and traffic light state information, provide a basis for the representation of these conditions in the simulation. In this work, we therefore propose a scalable approach to transfer real-world data, exemplarily taken from the German city Ingolstadt, to a virtual environment for a calibration of a traffic flow simulation in SUMO. To recreate measured traffic properties such as traffic counts or traffic light programs into the simulation, the measurement sites must first be allocated in the virtual environment. For the allocation of historical real-world data, a matching procedure is applied, in order to associate real-world measurements with their corresponding locations in the virtual environment. The calibration incorporates the replication of realistic traffic light programs as well as the adjustment of simulated traffic flows. The proposed calibration procedure allows for an automated creation of a calibrated traffic flow simulation of an arbitrary road network given historical real-world observations.

1 Introduction

The release of a fully functional automated driving system implies a high effort in testing and development. As real-world tests are time-consuming, expensive, not replicable and potentially dangerous, testing is primarily shifted towards a virtual environment. In order to replace real-world tests by virtual experiments, there are high demands towards a realistic representation of the real world in simulation. Therefore, the approach has been pursued to couple a submicroscopic vehicle simulation and a microscopic traffic simulation, e.g. [1]–[4]. In the approach, traffic simulation provides a temporally and locally realistic traffic system for an autonomous vehicle under test. Submicroscopic simulation controls the specific behavior and interactions of the involved traffic participants only in the direct environment of the tested vehicle. For achieving valid, realistic traffic surroundings, the traffic system must be accurately calibrated with real-world data. Therefore, traffic properties like traffic counts as well as traffic light programs must be obtained and assigned to the simulation, in order to recreate realistic traffic conditions. In order to prove simulation validity and especially enable the comparison of simulated and real-world results, an accurate representation of the real world conditions is inevitable.

The main contribution of this work is a scalable calibration process of a microscopic traffic simulation using the microscopic traffic simulator SUMO [5]. We present a tool chain, which enables real-world data acquisition for a requested time interval, the allocation of these measurements in the virtual world as well as the calibration of traffic flows and traffic lights. The validation of this comprehensive approach takes into account the comparison of simulated measures to the corresponding real world observations as well as the ability of the calibrated traffic lights to cope with the provided traffic demand. The developed tool chain allows for an application of the calibration method to any arbitrary traffic simulation in SUMO given necessary historical data. As our focus is mainly on the replication of a historical traffic scenario, we will not take into account calibration of driver models.

Section 2 provides an overview on related work regarding map association, traffic light emulation and traffic assignment. In Section 3, the allocation of real-world data and the calibration procedure are described in detail. The evaluation of the procedure is presented in Section 4. Section 5 summarizes the main contributions and provides an outlook regarding future work.

2 Related Work

The proposed approach to calibrate the traffic flow simulation in SUMO using real-world observations can mainly be separated into three predominant tasks. The first task consists of the representation of the virtual road network and allocation of real-world data to the virtual world. In a next step, traffic light programs must be created and implemented to the simulation. In a last step, traffic must be assigned based on observed traffic measurements. The following literature review addresses these relevant steps.

2.1 Map association

The allocation of real-world data in the virtual environment can be broken down to a map-matching problem. According to Quddus et al. [6], map-matching procedures can be classified to four different categories. Geometric map-matching algorithms take into account geometric information of a road network by neglecting logical connections between roads. Bernstein and Kornhauser [7] distinguish point-to-point matching, point-curve-matching and curve-to-curve matching as representatives of geometric map-matching algorithms. In contrast to geometric algorithms, topological map-matching techniques consider the geometry as well as the linkage of network elements. The third category of map-matching algorithms, probabilistic algorithms, create an elliptical or rectangular confidence region around a coordinate to be matched. For the determination of the corresponding network element, the confidence region is superimposed on the network. Advanced map-matching algorithms apply several methods like Kalman Filter or an Extended Kalman Filter, a flexible state-space model in combination with a particle filter, a fuzzy logic model or an implementation of Bayesian inference.

2.2 Traffic light emulation

In the literature, there are two approaches for the creation of simulated traffic lights. The first approach treats the emulation of existing traffic light programs for a realistic representation in simulation [8]–[14]. Besides, the second concept is the optimization of existing traffic light programs in order to improve certain measures, e.g. traffic flow, using reinforcement learning or evolutionary algorithms [15]–[18]. As our purpose is on building a detailed representation of the real world, we only provide an overview on literature on the first approach to estimate and emulate traffic light programs.

Schönfelder et al. [8] reconstruct traffic light programs of fixed-time traffic light cycles using camera data from probe vehicles approaching an intersection with a known cycle time. The procedure requires only few and less quality input data for high accuracy results. Axer and Friedrich [9], [10] provide an approach for the estimation of traffic light programs and the cycle durations of fixed-time as well as traffic actuated traffic lights based on sparse simulated floating car data (FCD). Besides, Wang and Jiang [11] use recorded FCD trajectories of vehicles passing a signalized intersection to derive traffic light programs and cycle times. Rostami-Shahrbakaki et al. [12] apply a machine learning approach for determining signal timings of traffic lights based on FCD of connected vehicles. They achieve high accuracy in estimating cycle time at low percentages of connected vehicles (~ 5 - 10 %). For detecting green time durations, higher penetration rates of connected vehicles (~ ≥ 15 %) are necessary to obtain higher precision estimations. The patent of Wolf et al. [13] introduces a procedure using a genetic algorithm to emulate future traffic light signals from historical signal timing information accessed by an interface to the traffic light backend. Weisheit [14] applies a Support Vector Machine on historical signal timing data for predicting future traffic light states.

2.3 Traffic assignment

For the assignment of traffic to a simulation network, trip-based as well as activity-based methods are applied. The four-step model [19] describes a trip-based approach, which divides a simulated region into several traffic assignment zones. These zones serve as origins and destinations of trips in the network. The first step of the model represents the trip generation, in which the frequency of originating and arriving trips is determined for each zone by different trip purposes based on socio-economic statistical measures. In the second step, called trip distribution, trip interchanges between zones are defined in order to meet the frequencies of the first step. The result of the second step is an origin-destination (OD) matrix incorporating the number of trips from each origin to every destination in the network. There are various methods to improve an OD matrix based on observed traffic counts. Van Zuylen and Willumsen [20] introduce information minimization and entropy maximization for estimation of an OD matrix using traffic counts. Further procedures apply generalized least squares [21], [22], a Bayesian estimator [23] and maximum likelihood estimation [24]. The third step of the four-step model implies the transportation mode choice for each trip. Finally, the last step incorporates the route assignment for each trip according to the mode-specific network. For route assignment, Wardrop introduces two concepts regarding the optimization objective. The user equilibrium defines the state, in which all vehicles are assigned to their individual fastest route. Besides, the system optimum describes an equilibrium, in which the total travel times are minimal.

Activity-based models consider traffic assignment on a person- and household-related level rather than on a zone-based level such as the four-step model. The route for each person is affiliated with certain activities and comprises all trips over an entire day. Apart from general socio-economic statistical data, household surveys are an essential basis in order to realistically represent individual travel behavior. Lobo et al. [25] introduce an activity based approach for a simulation of Ingolstadt called InTAS. The city is divided into several districts. According to regional statistics regarding the number of inhabitants, commuters, workplaces, education possibilities and demography, traffic is assigned to the network. However, the traffic assignment does not use traffic counts to generate the demand. Additionally, the traffic lights in the network are created manually instead of using real-world data.

3 Methodology

As introduced in the previous section, calibration of a traffic flow simulation is split up to map association tasks, traffic light program emulation and traffic flow calibration. For the calibration of the traffic flow simulation in SUMO, we use a network comprising the entire city of Ingolstadt, Germany, with an extent of about 10 km by 10 km. It covers more than 12000 edges, including urban, rural and highway roads, as well as 5600 junctions with 120 traffic light actuated intersections. The network is based on an import from OpenStreetMap converted to a SUMO network. Figure 1 depicts a section of the road network.

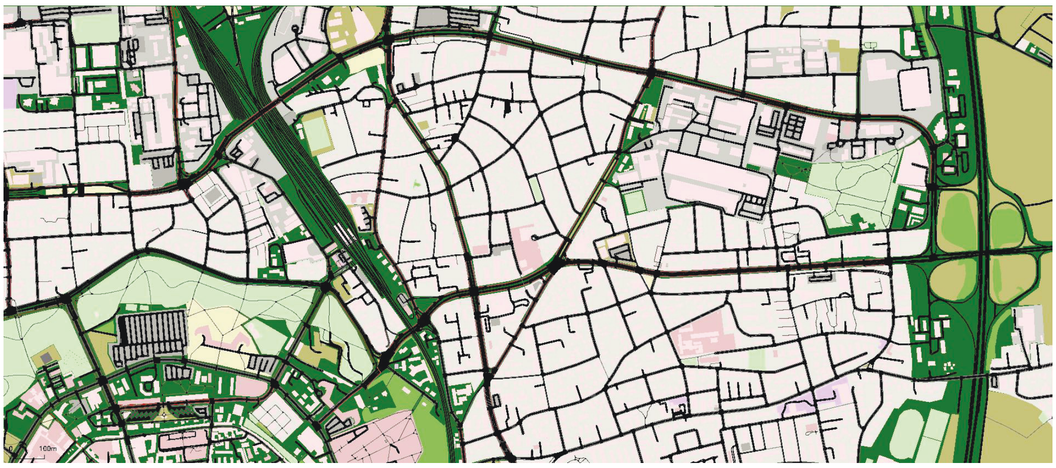


Figure 1: Section of the road network of Ingolstadt

The following Figure 2 provides an overview on the necessary steps for the calibration purpose, which will be explained in detail in the following sections.

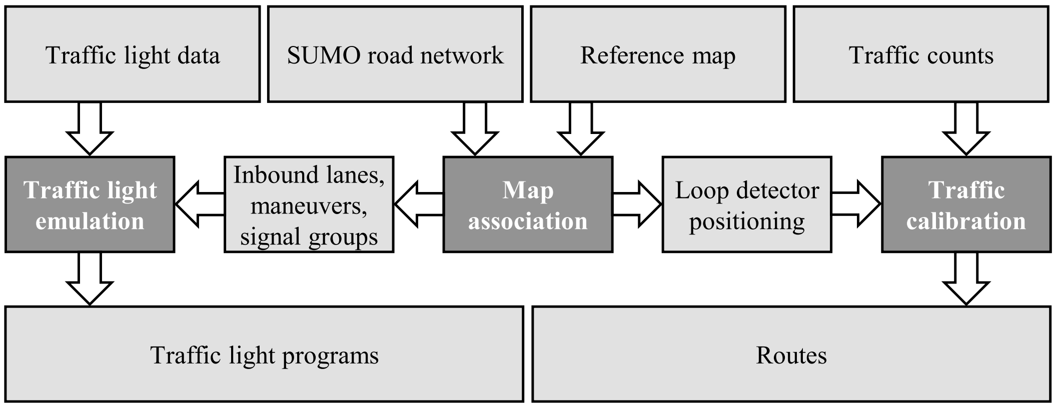


Figure 2: Overview on the calibration procedure

3.1 Map association

A key part of the calibration procedure of the traffic simulation is the linkage between real observations and their representation in the virtual world. For the map association tasks, we apply a framework based on a parsed representation of the SUMO network towards a GeoDataFrame, introduced by the Python library geopandas [26]. The framework enables map-matching tasks, which exceed SUMO basic functionalities, by not only applying geometrical matching, see section 2.1, but also considering the logical connections and properties of the map, e.g. lane successors, lane types etc. The application range of the framework is not only limited to matching tasks related to the calibration of a traffic simulation. Furthermore, the framework can also be used for applications such as vehicle trajectory matching or public transport routing. We plan to provide the developed framework as an open-source Python package.

For the calibration of the simulation, real signal groups of traffic lights must be mapped to the inbound lanes and in a second step to the succeeding connecting lanes of traffic light actuated junctions in SUMO. Signal groups provide signal state information for multiple associated lanes. Furthermore, loop detectors are allocated to the corresponding lanes given their geographical positions. For both applications, allocation is based on a reference topology representing all inbound lanes for each traffic light actuated intersection in Ingolstadt including associated stopping lines, signal groups as well as possible maneuvers. Besides, the geographical positions of all loop detectors can be extracted from the reference map.

In order to assign signal groups to the SUMO network, in a first step, each traffic light actuated junction from SUMO must be identified in the reference topology by simply comparing the geometrical centers of the respective junctions considering small deviations (~10 m) of both maps. As signal group association in the reference topology is stored as an attribute of inbound lanes, the attribute in SUMO is linked to connecting lanes within a junction. For the association of signal groups to SUMO, the inbound lanes of the respective junction in the SUMO network must be derived and matched to the inbound lanes from the reference. To prevent errors in the allocation process, the number of inbound roads and lanes of each traffic light actuated intersection must be the same for both maps. Therefore, these numbers are extracted and compared for the inbound roads and lanes. As there are deviations between the reference and the SUMO network, corresponding lanes from each map do not generally match geometrically. Therefore, the matching of the junction geometries is performed in two steps. In the first step, the inbound roads of the topology are matched with the inbound roads of the SUMO map. Once the pairing is done on the road level, the lanes within each individual pair of roads are matched between both maps. In order to complete this two-step matching, detailed information regarding the association between individual lanes belonging to the same roads is required.

For the SUMO network, the number of inbound roads and lanes can be determined directly by the unique road identifier. The intersection reference map also provides information on which lanes form the associated roads. In order to group inbound lanes from an older version of the reference map without any knowledge on inbound roads given the number of inbound roads from SUMO, additionally hierarchical agglomerative clustering is applied. A comparison of both techniques shows that the clustering produces the same groups of inbound lanes given in the reference map with knowledge about inbound roads. In case of different numbers of inbound roads or lanes due to deviations of both maps, these differences are logged for a manual inspection of both maps and for corrections, which are essential to any of the following steps to allocate signal group information in SUMO. For an association of the inbound roads of both maps, the aggregated geometric centers of the stopping lines of all lanes of each inbound road are compared. The pairing between the two maps is carried out by minimizing the Minkowski distance (p-norm) between each pair of inbound roads from both maps. The Minkowski distance enables an emphasized weighting of shorter distances by applying an exponent, e.g. squaring, in order to weight larger distances with an exponentially higher value. Finally, for the allocation of inbound lanes, all lanes within each pair of inbound roads are considered and again the overall Minkowski distance between the inbound lanes of both the junction in the reference topology and in the SUMO network is minimized. The overall distance is selected as the quality parameter of the matching process as large deviations occur between lanes of the SUMO and the reference map. Furthermore, the application of the overall distance enables an optimal matching for the combination of all corresponding lanes rather than only for several specific lanes. Figure 3 shows the results of the clustering and pairing procedure for an exemplary intersection in the SUMO network and the reference map.

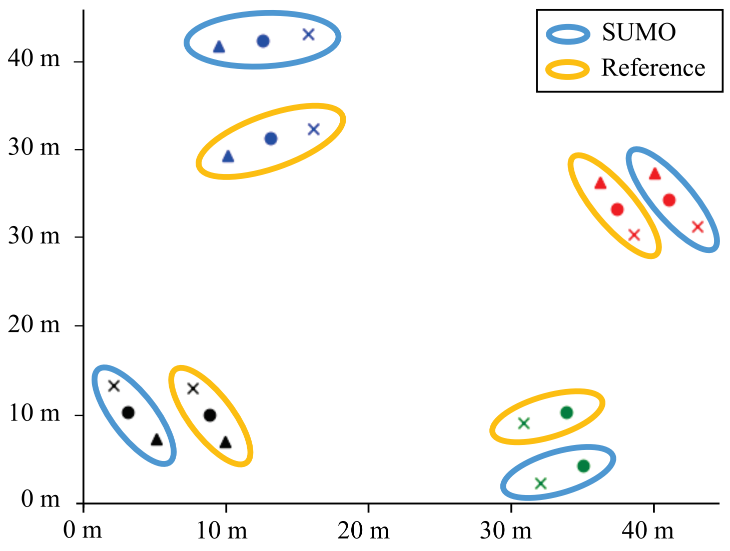


Figure 3: Clustering of inbound roads (blue and yellow ellipses) and pairing of inbound lanes (colored symbolic markers)

In Figure 3, the symbolic markers represent the geographical positions of the centers of the stopping lines of the inbound lanes. The inner markers are derived from the SUMO network and the outer markers result from the reference map. The color of the markers indicates the association to inbound roads, i.e. markers with equal color indicate the ends of all lanes from a single road. Furthermore, equal symbols having equal colors provide information on related pairs of inbound lanes comprehending both maps.

For the placement of loop detectors, a similar procedure of grouping and pairing is applied. Loop detectors are grouped by their inbound road. For the aforementioned older version of the reference map without information on inbound roads, detectors are grouped by using the agglomerative hierarchical clustering on the geographical positions. After grouping, the detectors are matched to their nearest SUMO inbound lanes by minimizing the Minkowski distance of the whole group. In contrast to the intersection matching, detectors are also matched and paired if the number of detectors and the number of inbound lanes differ. In case of more detectors than lanes, multiple detectors are placed on the same lane according to their geometrical position.

3.2 Traffic light emulation

In order to cover the strong influence of traffic control on traffic behavior, a realistic representation of traffic light programs is inevitable. Real traffic light programs in Ingolstadt are controlled traffic adaptively and incorporate features like green phase prolonging as well as public transport prioritization. For the proposed approach, average green times observed in reality are represented as fixed-time traffic light programs in the simulation. The individual effects on traffic of green phase prolonging and public transport prioritization are ignored, but their impact is still considered as average green time adaptions. Furthermore, different traffic light programs over a day are taken into account by hourly generating specific traffic light programs for each traffic light. Therefore, we provide an approach recreating these programs based on the same historical signal timing data used in [13]. Signal timing data can be requested for any traffic light actuated intersection in Ingolstadt for a defined time interval via an interface to the traffic light backend. The data are available for 75 traffic lights and contain the temporal intervals of green and red phases for each signal group of every specific traffic light with a resolution of one second. All remaining traffic lights in the network cannot be calibrated and are therefore adapted manually with static traffic light programs in order to cope with the traffic demand.

A realistic estimation of the traffic light programs requires determining the switching order and mean green time durations of all signal groups. For each signal program, one signal group is declared as the reference. For the requested time interval, this reference signal group is the first in the data to switch to a green state on the condition that all other signal groups have switched to red at least in the time step before. Next, the most common switching order, in which all signal groups have switched to green at least once and which starts with the reference signal group, is determined. Furthermore, the mean green and red time as well as the overall green time during the one-hour interval are calculated for each signal group. Incomplete signal timing phases in the recordings are ignored in the calculation. Based on the number of occurrences n_i in the most common order as well as the mean green time $\bar{t}_{g,i}$ and the proportion of the green time in the whole one-hour interval $p_{total,i}$, the cycle time $t_{cycle,i}$ for the emulated traffic light program is estimated for each signal group i individually according to equation 1.

$$t_{cycle,i} = \frac{n_i \cdot \bar{t}_{g,i}}{p_{total,i}} \tag{1}$$

Based on the individual cycle time estimations, the overall cycle time t_{cycle} for the emulated traffic light program is calculated as the mean of all $t_{cycle,i}$ weighted by $p_{total,i}$.

$$t_{cycle} = \frac{\sum(t_{cycle,i} \cdot p_{total,i})}{\sum p_{total,i}} \tag{2}$$

The weights are applied to estimate a general cycle time, which takes into account the relevance of certain signal groups for the whole cycle with respect to the green time in the entire one-hour interval. In order to fit into the determined cycle time, the emulated green times $\tilde{t}_{g,i}$ of each signal group are adopted based on their average green times $\bar{t}_{g,i}$ and red times $\bar{t}_{r,i}$.

$$\tilde{t}_{g,i} = \frac{\bar{t}_{g,i}}{(\bar{t}_{g,i} + \bar{t}_{r,i})} \cdot t_{cycle} \tag{3}$$

The signal switching order is created by applying the emulated green times under consideration of average overlaps of green times in the whole interval. If there is no overlap between two consecutive signal groups, a clearing time of 6 s is included in between.

For the implementation of the estimated traffic light programs in SUMO, all inbound lanes, which are linked with the relevant signal groups, must be associated to the SUMO map as explained in the map association section. As a result, the related inbound lanes of the reference topology and the SUMO net are available. As traffic light state information in SUMO is stored as an attribute of connecting lanes within each intersection, the succeeding connections of each inbound lane are determined. Furthermore, the feasible maneuvers for each connection are compared to the maneuvers linked to the signal groups in order to achieve a unique allocation of connections to a related signal group from reality. The generation of the necessary SUMO additional file, which incorporates the traffic light program, requires an iteration through one cycle of each traffic light. Yellow phases are accounted as a part of the red signal in the historical traffic light data. The yellow light duration is implemented after switches from green to red light according to German legislation [27]. For every change in state of any signal group, a new line in the SUMO additional file is created covering its duration as well as the current states of the corresponding connecting lanes.

3.3 Traffic assignment

For the calibration of simulated traffic flows, a two-step procedure is considered, which covers the determination of an initial pool of routes from OD relations and further calibration based on observed link counts. In order to obtain regional traffic flows, according to the first two steps of the four-step model [19], an initial OD matrix for Ingolstadt is determined based on statistical information. The initial OD matrix includes regional statistics from Ingolstadt regarding the distribution of inhabitants and workplaces in different municipalities as well as information on the attractiveness of municipalities with respect to free-time opportunities, shopping areas and educational establishments. Furthermore, we access aggregated data from Audi regarding residences and working time models of all 44000 Audi employees in Ingolstadt [28] considering data protection requirements that ensure no transparency of individual residences. The data from Audi are highly accurate and cover a significant number of trips in Ingolstadt having 136000 inhabitants [29]. As there are no local distributions regarding other trip purposes than work, their temporal distributions and trip lengths, as well as modal split, also Germany-wide statistics [30] are included. Finally, statistics of commuters traveling from and to Ingolstadt every working day are considered. Aggregating all this information, a procedure is established to create OD matrices taking into account the Ingolstadt municipalities, different trip purposes that are work, business, free-time, shopping and education, the temporal distribution of these trip purposes as well as the length of each trip with respect to the modal split. Modal split is separated to vehicles, public transport, bicycles and pedestrians. As a result, the procedure generates one OD matrix for every hour of an average working day and each mode of transport. The elements of each OD matrix can be converted to SUMO trips and routes using built-in executables *od2trips* and *duarouter* according to the route assignment of the four-step model. As the pool of resulting routes strongly affects the quality of the calibration outcome, the selection of roads in traffic assignment is weighted based on the road priority. Nevertheless, the pool of resulting routes is inaccurate due to statistical deviations from real local conditions and daily traffic characteristics making further traffic calibration inevitable. The calibration is conducted only for vehicle routes, as no further observation data regarding the other modes are available.

The pool of routes generated from statistical data is used as a basis for an hourly traffic assignment based on observed traffic counts in the next step of the calibration. The real-world counts are aggregated for each road with induction loop detectors. Additionally, the number of vehicles that would pass the same roads in the routes generated from statistical data is calculated. For the calibration, routes are copied two times and then successively removed until no further improvement can be achieved. The difference between the evaluated counts from routes and the observed counts for each road serves as a quality measure for the calibration. High values of the quality measure indicate a large error. This removal of routes is carried out in two steps. In a first step, only routes, which exclusively pass roads with an overestimation with respect to the observed counts, i.e. the worst quality measure, are removed at random. This procedure is repeated until no more routes meeting the criterion are found. In this step of the route selection, all routes are removed that pass only locations with an overestimation of traffic. In the second step, routes are also excluded if they have a positive impact on the overall error. This leads to a balancing of over- and underestimation of the error across all roads. The second step is repeated iteratively until any removal would lead to an increasing overall error value. The final set of routes contains anything between zero to three copies of each route taken from the initial pool of routes generated from statistical data. Since calibration is conducted on an hourly basis, routes for each hour are temporally shuffled inside the interval to prevent copied routes to start at the same point in time.

4 Results

For a validation of the calibration process, simulated observations are compared to their corresponding real-world counterparts. The results of the simulation are therefore analyzed in comparison to the input measurements. The evaluation has the objective to verify the functionality of the proposed calibration process. In the following, the quality of map association, traffic light emulation as well as traffic assignment is assessed.

4.1 Map association

For the map association of the signal groups to the virtual environment, all of the 75 considered traffic light actuated intersections can be fully matched showing the same number of inbound roads and lanes as well as the corresponding maneuvers for traffic light emulation. 18 intersections require slight manual changes in the SUMO network beforehand regarding inbound lanes or maneuvers in order to match the logic of the reference topology.

The association of the detectors between the two maps ends up with more errors than the intersection association due to geometric divergences between the two maps. Furthermore, the application of the agglomerative hierarchical clustering of detectors is error prone due to the missing number of clusters and strongly varying distances between detectors. Without knowledge on inbound roads, detectors are misplaced to outbound roads on occasion. Therefore, only the results of the method incorporating knowledge about inbound roads are presented. From the 529 detectors available in the road network, all are placed in the network. 210 of the 529 detectors must be replaced manually due to misalignment, while 319 are assigned correctly. In case of an equal number of detectors in a group of inbound lanes, the association of the detectors to the corresponding lanes works without errors as every lane gets occupied by one detector in the pairing process. Misalignments result from a higher number of lanes on a road than to be matched detectors in combination with map deviations. Shifted groups of detectors are paired with the closest set of inbound lanes, which may result in a shifted placement on the lanes. All in all, the 529 detectors are placed on 254 roads. For future simulation setups, the use of high definition map data is suggested as this makes an accurate geometric matching of detector locations to the simulation map possible.

4.2 Traffic light emulation

In the described approach, traffic light programs are reconstructed as fixed-time programs based on average green and red time durations of the different signal groups. In reality, traffic lights in Ingolstadt apply green phase prolonging for traffic adaptive control and public transport prioritization resulting in a discrepancy between real and simulated traffic lights. However, simulated traffic lights implicitly take into account the impact of these effects on the average green times. For the validation, emulated and real traffic lights are compared regarding the percentage of green time of each signal group with respect to the cycle time of the corresponding traffic light. In order to cover varying switching behavior of the real traffic lights along with cycle time deviations, the green time ratio of each real signal group is determined over the whole one-hour interval in which the calibration takes place. It is assumed that similar green time ratios lead to a comparable impact on the traffic.

Validation of emulated traffic lights is conducted for one defined week from Monday to Friday. Figure 4 indicates the distributions of absolute green time ratio errors in percent between emulated and real signal groups. All weekdays are aggregated and evaluated hourly for time intervals from 7 a.m. to 8 a.m., 9 a.m. to 10 a.m., 4 p.m. to 5 p.m. and 6 p.m. to 7 p.m.

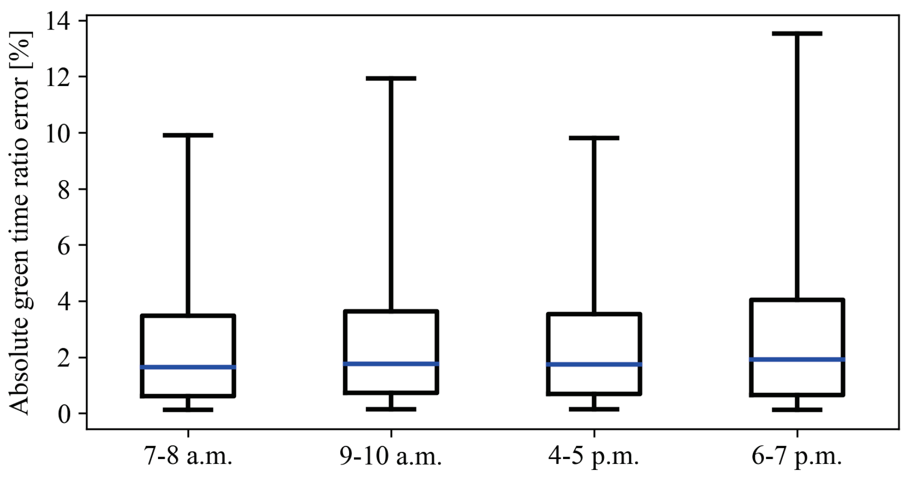


Figure 4: Distributions of absolute errors of the ratio of green time duration with respect to the cycle time

The determined median values of the deviation between simulated and observed green time ratios lie between 1.6 % and 1.9 %. The values indicate that the emulated traffic light programs are capable to adapt a realistic traffic light behavior for different time intervals. The distributions of errors in peak traffic hours from 7 a.m. to 8 a.m. and 4 p.m. to 5 p.m. range from 0 % to 10 %. In time periods with a lower traffic demand between 9 a.m. and 10 a.m. as well as 6 p.m. and 7 p.m., the proportion of public transport, which strongly influences the switching order and green time duration, is larger compared to the other hours. This increased variety of real traffic light programs leads to a wider distribution of the error value.

In the calibration process of traffic volumes in microscopic traffic simulation, traffic lights are often tuned to match the calibrated demand. In this publication, the traffic lights are created from real-world data together with the traffic counts. Inaccuracies in the calibration procedure will therefore quickly show in the simulation, since an overestimation of traffic volumes or an underestimation of the capacities at signalized intersections will lead to unrealistic congestions. However, the calibrated simulation shows that the emulated traffic lights are able to cope with the traffic demand even in peak hours without leading to excessive congestion or deadlock situations in the simulated network. This capability further indicates a realistic representation of traffic lights in Ingolstadt.

4.3 Traffic assignment

Traffic volumes generated by the proposed approach are validated by a comparison of simulated traffic counts with their real counterparts both aggregated on road level. In reality, there exist only measurements at specific loop detector locations so that only the counts of these measurement points from reality and simulation can be compared. Although these data are used for calibration of the simulation, there is no further available data for the validation purpose of traffic flows.

For the validity analysis of the simulated traffic counts, the absolute value of the relative error e_{rel} is calculated according to equation (4):

$$e_{rel} = \frac{|count_{real} - count_{sim}|}{count_{real}}, \tag{4}$$

where $count_{real}$ represents the real-world traffic count and $count_{sim}$ is its simulated counterpart both aggregated road-wise for each of the 254 roads containing detectors.

The proposed calibration approach uses detector counts of one hour to calibrate routes starting within the corresponding hour in order to meet the counts again. Consequently, the procedure leads to an offset of routes, which start in the corresponding hour but cannot finish during the one-hour interval. The issue especially arises in peak traffic, as the higher number of vehicles potentially leads to more congestion, which increases travel time and causes incomplete trips. Therefore, simulated traffic counts are evaluated in a prolonged timespan, which enables all generated vehicles to complete their journey through the network. Figure 5 shows the relative errors for one week from Monday to Friday which are separated hourly in time spans from 7 a.m. to 8 a.m., 9 a.m. to 10 a.m., 4 p.m. to 5 p.m. and 6 p.m. to 7 p.m.

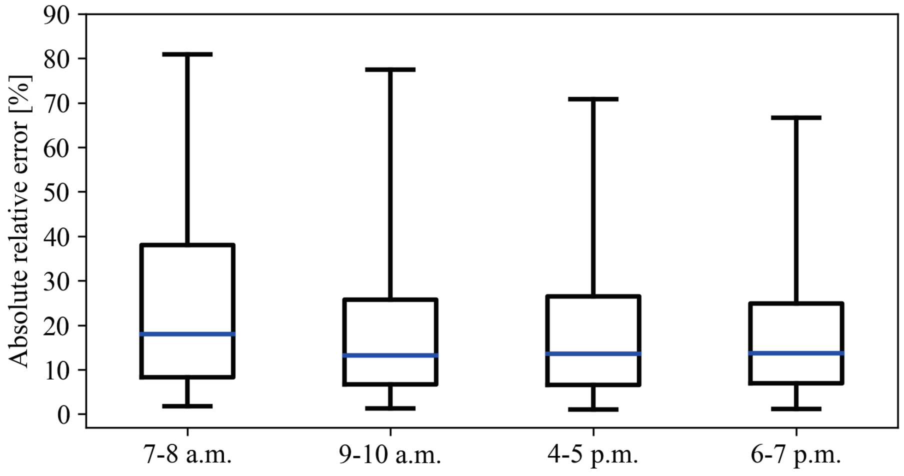


Figure 5: Distributions of absolute relative errors of simulated traffic counts aggregated to road level after a clearing period

All distributions presented in Figure 5 show median values between 13 % and 18 %. The error distribution of values of peak traffic in the afternoon between 4 p.m. to 5 p.m. shows similar results compared to the hours from 9 a.m. to 10 a.m. and 6 p.m. to 7 p.m. with lower demand. Only the distribution for peak traffic in the morning from 7 a.m. to 8 a.m. reveals higher error values. There are mainly two reasons for the remaining errors, which result from the calibration, compared to reality, i.e. corrupt input data and rerouting. The calibration procedure balances the over- and underestimation of routes passing the detector loops in order to achieve an overall error of 0. However, the observed traffic counts show several unrealistically low values, which strongly influence the calibration. Given an exemplary road network with three junctions linking three successive main roads in a line each containing detectors, unrealistically low detector counts of the road in between the two other roads would induce the traffic assignment algorithm to adjust an underestimation of the two roads with correct detector counts and an overestimation of the road with the lower counts. This would lead to a balanced overall error but reveals in an increased absolute error. Especially in peak traffic hours, overestimation of traffic exceeds the capacity of a share of roads resulting in congestion and deadlock situations. In order to cope with the overestimation induced by inaccurate input data, the rerouting probability of SUMO vehicles is adapted to 20 %, which enables 20 % of the vehicles to take different routes than assigned to reach their destinations. In comparison, Lobo et al. [25] apply a more than four times higher rerouting probability to handle congestion in the network.

The absolute values of the observed traffic counts vary over a wide range. However, absolute errors of small traffic count values add to higher percentage deviations than equal absolute errors of high counts. Therefore, GEH analysis is applied additionally in order to assess the calibration performance under consideration of a non-linear weighting of errors. The GEH statistic is given by

$$\mathrm{GEH} = \sqrt{\frac{2 \cdot (\mathrm{count}_{\mathrm{real}} - \mathrm{count}_{\mathrm{sim}})^2}{\mathrm{count}_{\mathrm{real}} + \mathrm{count}_{\mathrm{sim}}}}. \tag{5}$$

Figure 6 depicts the distributions resulting from GEH analysis regarding a clearing period.

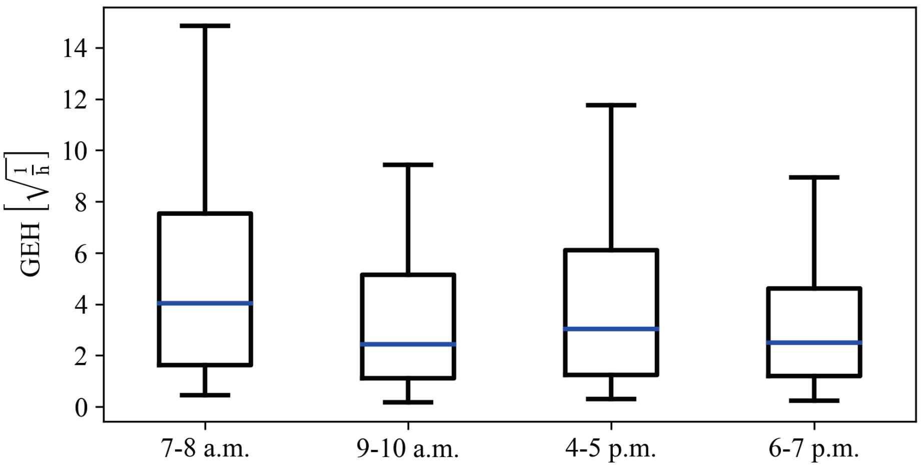


Figure 6: Distributions of GEH analysis of simulated traffic counts aggregated to road level after a clearing period

Figure 6 reveals wider distributions of the GEH statistic in peak traffic hours from 7 a.m. to 8 a.m. and 4 p.m. to 5 p.m. than for the remaining hours with less traffic. The median GEH values between 2.4 and 4 indicate a good performance of the calibration procedure.

In order to tackle the issue regarding the offset of routes when starting the simulation under initial conditions without any traffic, additionally, the impact of a one-hour warm-up phase without a clearing period is investigated. The warm-up on the one hand fills up the network at the beginning of each analyzed interval. On the other hand, the overlapping traffic is originally calibrated for the previous time interval. Figure 7 displays the resulting distributions of the absolute value of relative errors of the traffic counts for the previously specified time intervals regarding a warm-up phase.

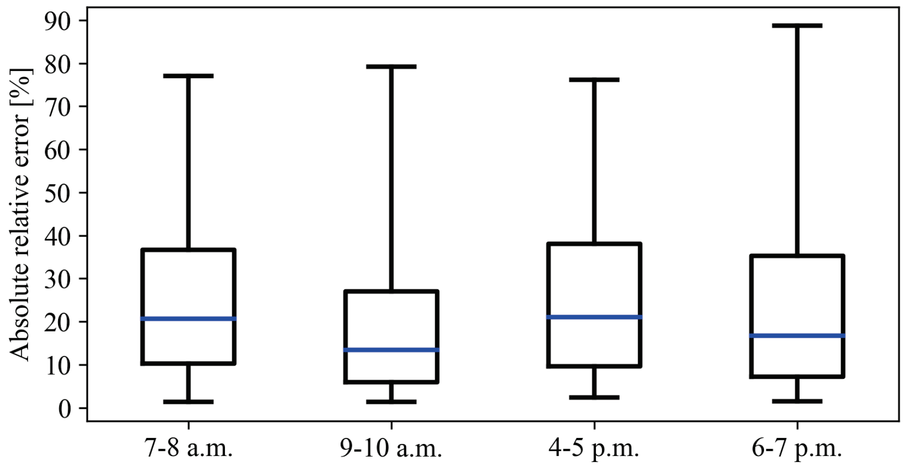


Figure 7: Distributions of absolute relative errors of simulated traffic counts aggregated to road level with a previous warm-up phase

With an initial warm-up phase, small deviations between traffic in peak hours and traffic with lower demand are visible. Compared to Figure 5, the slightly higher error values when considering a warm-up phase might result from different traffic volumes in the corresponding two consecutive time intervals, which increase the observed error. Hence, the traffic from 9 a.m. to 10 a.m. shows similar characteristics compared to the previous hour from 8 a.m. to 9 a.m. and therefore yields the lowest error. In order to improve the presented calibration method, the temporal offset should be considered by a partial overlap of calibrated routes beyond the corresponding hour.

5 Conclusion

In this work, a scalable approach for the calibration of a microscopic traffic simulation based on real-world observation data for a requested time interval was presented. Therefore, information of traffic lights and loop detectors for this time interval was assigned to the virtual world. The inbound roads and lanes to intersections were associated correctly to the map by a robust determination of inbound roads and pairing of lanes. However, the detector placement was error prone affected by map inaccuracies and the number of lanes exceeding the number of detectors. Traffic light programs were emulated as fixed-time traffic lights and implemented to the simulation. Simulated traffic lights were evaluated regarding the ratio of green times with respect to the cycle times showing median errors between 1.6 % and 1.9 %. Furthermore, emulated traffic lights proved the ability to cope with the calibrated traffic demand. For the calibration of traffic flows, a two-step procedure was introduced. In a first step, an initial pool of routes was generated based on OD relations determined from statistics. In a second step, the routes in the pool were copied twice and subsequently removed in order to meet the observed traffic counts in the network. The procedure exhibited an offset of a share of routes calibrated with traffic counts of a one-hour interval, which could not be completed within the interval. Considering a clearing period for vehicles to finish their trips, the evaluation resulted in median absolute error values between 13 % and 18 % with respect to observed traffic counts.

In the future, we will apply a time component to the traffic flow calibration in order to shift calibrated routes to the corresponding previous hour. Further improvement can potentially be achieved by a more comprehensive and diverse initial pool of routes. Additionally, different optimization algorithms like Cadyts [31]–[33] can be applied for a comparison of traffic flow calibration procedures. In order to increase the accuracy of the simulation, fixed-time traffic lights must be replaced by a traffic adaptive traffic light control using rule-based or data-driven techniques. Finally, individual driver behavior has a strong impact on the whole traffic system. Therefore, we will analyze and calibrate driver models with real-world data to achieve a more accurate representation of the real world.

We plan to publish the simulation of Ingolstadt as an open-source project to make it available for further applications especially regarding testing and development of automated driving. Furthermore, we intend to publish the code framework for map-matching to the SUMO network as a Python library based on geopandas [26], which provides a broad range of possible matching applications.

6 Acknowledgements

This work is supported by the German Federal Ministry of Transport and Digital Infrastructure (BMVI) within the Automated and Connected Driving funding program under grant no. 16AVF2145A (SAVe) with a funding amount of 3.885.755,00 €.

The authors thank the colleagues from GEVAS, TTS and AUDI AG for their support and for providing the traffic count and traffic light data, on which the calibration is based on.

References

[1] M. Semrau, “Untersuchung zur Modellierung von chinesischem Fahrverhalten auf Autobahnen für den Test pilotierter Fahrfunktionen,” TU Braunschweig, 2018.

[2] M. Semrau, J. Erdmann, B. Friedrich, and R. Waldmann, “Simulation framework for testing ADAS in Chinese traffic situations,” *SUMO 2016 Conference Proceedings*, pp. 103–113, 2016.

[3] J. Olstam and R. Elyasi-Pour, “Combining traffic and vehicle simulation for enhanced evaluations of powertrain related ADAS for trucks,” *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 851–856, 2013.

[4] D. Krajzewicz, A. Richter, M. Behrisch, and J. Erdmann, “Abbildung des Umgebungsverkehrs in einem Fahrsimulator,” pp. 1–10, 2013.

[5] P. A. Lopez *et al.*, “Microscopic Traffic Simulation using SUMO,” *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 2575–2582, 2018.

[6] M. A. Quddus, W. Y. Ochieng, and R. B. Noland, “Current map-matching algorithms for transport applications: State-of-the art and future research directions,” *Transportation Research Part C: Emerging Technologies*, vol. 15, no. 5, pp. 312–328, 2007.

[7] D. Bernstein and A. Kornhauser, “An introduction to map matching for personal navigation assistants,” *Technology*, pp. 587–602, 1996.

[8] M. Schönfelder, V. Protschky, and T. Bäck, “Reconstructing fixed time traffic light cycles by camera data analytics,” *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 1–7, 2018.

[9] S. Axer and B. Friedrich, “A Methodology for Signal Timing Estimation Based on Low Frequency Floating Car Data: Analysis of Needed Sample Sizes and Influencing Factors,” *Transportation Research Procedia*, vol. 15, pp. 220–232, 2016.

[10] S. Axer and B. Friedrich, “Signal timing estimation based on low frequency floating car data,” *Transportation Research Procedia*, vol. 25, pp. 1645–1661, 2017.

[11] C. Wang and S. Jiang, “Traffic signal phases’ estimation by floating car data,” *2012 12th International Conference on ITS Telecommunications, ITST 2012*, pp. 568–573, 2012.

[12] M. Rostami-Shahrbabaki, K. Bogenberger, A. A. Safavi, and A. Moemeni, “Intersection SPaT estimation by means of single-source connected cehicle data,” *TRB 99th Annual Meeting*, 2020.

[13] J. C. Wolf, M. Zweck, and C. Rucker, “Traffic Signal Emulation Using Genetic Algorithm,” US Patent 10,255,805, 2019.

[14] T. Weisheit, “Algorithmenentwicklung zur Prognose von Schaltzeitpunkten an verkehrsabhangigen Lichtsignalanlagen,” *Heureka*, vol. FGSV 002/1, pp. 320–339, 2014.

[15] Z. Shen, K. Wang, and F. Zhu, “Agent-based traffic simulation and traffic signal timing optimization with GPU,” *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 145–150, 2011.

[16] R. Blokpoel and J. Vreeswijk, “Uses of Probe Vehicle Data in Traffic Light Control,”

Transportation Research Procedia, vol. 14, pp. 4572–4581, 2016.

[17] S. S. Mousavi, M. Schukat, and E. Howley, "Traffic light control using deep policy-gradient and value-function-based reinforcement learning," *IET Intelligent Transport Systems*, vol. 11, no. 7, pp. 417–423, 2017.

[18] A. Hussain, T. Wang, and C. Jiahua, "Optimizing Traffic Lights with Multi-agent Deep Reinforcement Learning and V2X communication," no. 1, pp. 1–6, 2020.

[19] M. G. McNally, "The Four Step Model," *Handbook of Transport Modelling*, pp. 35–52, 2000.

[20] H. J. Van Zuylen and L. G. Willumsen, "The most likely trip matrix estimated from traffic counts," *Transportation Research Part B: Methodological*, vol. 14, no. 3, pp. 281–293, 1980.

[21] M. G. H. Bell, "The estimation of origin-destination matrices by constrained generalised least squares," *Transportation Research Part B: Methodological*, vol. 25, no. 1, pp. 13–22, 1991.

[22] E. Cascetta, "Estimation of trip matrices from traffic counts and survey data: A generalized least squares estimator," *Transportation Research Part B*, vol. 18, no. 4–5, pp. 289–299, 1984.

[23] M. J. Maher, "Inferences on trip matrices from observations on link volumes: A Bayesian statistical approach," *Transportation Research Part B: Methodological*, vol. 17, no. 6, pp. 435–447, 1983.

[24] H. Spiess, "A maximum likelihood model for estimating origin-destination matrices," *Transportation Research Part B: Methodological*, vol. 21, no. 5, pp. 395–412, 1987.

[25] S. C. Lobo, S. Neumeier, E. M. G. Fernandez, and C. Facchi, "InTAS -- The Ingolstadt Traffic Scenario for SUMO," in *SUMO User Conference 2020*, 2020, pp. 1–20.

[26] "Geopandas," 2021. [Online]. Available: https://geopandas.org/. [Accessed: 17-May-2021].

[27] A. V. Forschungsgesellschaft für Straßen- und Verkehrswesen -FGSV- Köln, Ed., *Richtlinien für Lichtsignalanlagen. RiLSA. Lichtzeichenanlagen für den Straßenverkehr.* Germany, 2010.

[28] AUDI AG, "Audi Werk Ingolstadt." [Online]. Available: https://www.audi-mediacenter.com/de/ingolstadt-195. [Accessed: 31-Jul-2020].

[29] Stadt Ingolstadt, "Ingolstadt in Zahlen 2018/2019," 2019. [Online]. Available: https://www.ingolstadt.de/media/custom/465_1995_1.PDF?1533818349. [Accessed: 31-Jul-2020].

[30] infas Institut für angewandte Sozialwissenschaft GmbH, "Mobilität in Tabellen (MiT 2017)," 2017. [Online]. Available: https://mobilitaet-in-tabellen.dlr.de/mit/login.html?brd. [Accessed: 20-Jul-2020].

[31] G. Flötteröd, M. Bierlaire, and K. Nagel, "Bayesian calibration of dynamic traffic simulations," 2009.

[32] G. Flötteröd, Y. Chen, and K. Nagel, "Behavioral Calibration and Analysis of a Large-Scale Travel Microsimulation," *Networks and Spatial Economics*, vol. 12, no. 4, pp. 481–502, 2012.

[33] G. Flötteröd and R. Liu, "Disaggregate path flow estimation in an iterated dynamic traffic assignment microsimulation," *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, vol. 18, no. 2, pp. 204–214, 2014.

SUMO User Conference 2021
https://doi.org/10.52825/scp.v2i.91

Published: 12 Feb. 2026

Simulation based method for the analysis of energy-efficient driving algorithms using SUMO

Benedikt Buhk[1] and Rasmus Rettig[1]

[1]Hamburg University of Applied Science, Urban Mobility Lab, Hamburg, Germany
Faculty of Engineering and Computer Science
benedikt.buhk@outlook.de,rasmus.rettig@haw-hamburg.de

Abstract

The limited possibilities to evaluate the energy efficiency of driving algorithms for connected and autonomous vehicles (CAVs) make it very difficult for policymakers to decide on the potential of autonomous driving. This study is introducing a method to analyze the energy performance of a driving algorithm under various simulated traffic conditions using the microscopic traffic simulator SUMO. The method can also be used to optimize driving algorithm parameters for chosen traffic scenarios. Therefore, a tool-chain is developed that can simulate a CAV under many traffic scenarios in SUMO systematically. In those scenarios, one or more vehicles are controlled by the implemented driving algorithm. The resulting driving cycles are then analyzed by a forward-facing energy model to calculate the consumed energy. To validate the model, three measurement cycles under real urban traffic conditions were taken and the speed and state of charge (SOC) data of the test vehicle, a 2017 Tesla Model S 75D, were collected. The energy model was shown to be highly accurate and the simulated road network and traffic, which were chosen to represent the same urban traffic scenario as the measured cycles, were shown to result in similar statistics as the measurements. A simple driving algorithm that is already implemented in SUMO's Kraus car-following model was chosen to verify the model's applicability. For different values of the algorithm parameters acceleration and deceleration, a range of random driving cycles was simulated. In the simulations and the measurements, the effect of higher and lower use of auxiliary systems was also analyzed. The results show that the analyzed driving algorithm achieves similar results for the energy consumption as the human driver in the measurements with the best performing parameters. Also, the significance of auxiliary system use on the energy consumption and its effect on a driving algorithm's parameter to remain energy efficient due to the higher impact of the trip duration is pointed out.

1 Introduction

In the controversial debate about the pros and cons of CAVs, one of the most commonly used arguments to advocate for their use and their further development is their promising increase in energy efficiency. However, such an increase in energy efficiency under urban traffic conditions is very difficult to estimate. This study is introducing a simulation-based method for the analysis of the energy efficiency of a driving algorithm for a certain electric vehicle (EV).

To generate realistic traffic scenarios and analyze them down to a single vehicle, microscopic traffic simulation tools are already widely used today for various objectives. One of those simulation tools is the open-source Simulator for Urban Mobility (SUMO) of the German Aerospace Center (DLR) [11]. SUMO comes with an application programming interface called Traffic Control Interface (TraCI) ([19]) which can be used to control certain vehicles inside the simulation with an external driving algorithm.

To calculate the energy consumption of a vehicle, microscopic traffic simulators are usually based on backward-facing energy models. Those backward-facing energy models such as the

VT-CPEM proposed by Fiori et al. [3] or the one of T. Kurczveil et al. [10] implemented in SUMO can deliver instantaneous energy consumption estimations requiring only a few vehicle-specific parameters. They remain computationally efficient by calculating the consumed energy based on simple equations from the current speed "backward" to the energy source. However, forward-facing models such as ADVISOR [13] can deliver more accurate instantaneous energy consumption estimations for the cost of being computationally more complex. They also take the speed-over-time data of a driving cycle as input. This speed is then used as a reference for a driver model which controls the torque request to the engine. From this torque request, the drive train is calculated "forwards" to the wheels resulting in an actual speed of the vehicle. Since all of the vehicle aggregates and the drivetrain components can be simulated, the consumed energy to achieve the actual speed is calculated in a causal way. Therefore, forward-facing models are the better choice when it comes to a detailed analysis of a single vehicle's energy consumption.

R. Galvin analyzed the influence of different speed and acceleration values on the energy consumption over a short driving scenario without traffic in [4] for eight commonly used EVs. One of the key findings was the significant reduction of energy efficiency with modest to high acceleration. To reduce the energy consumption of CAVs different methods are proposed in the literature. They can mainly be divided into two groups: eco-routing approaches, and eco-driving approaches. Eco-routing focuses on the energy-efficient routing of one or more vehicles. For example, the traffic light control can be optimized on the energy consumption of all vehicles on the road as Luin et al. demonstrated in [12]. In their study, they used SUMO together with a backward-facing energy model to verify their results. In this case, a more accurate, forward-facing model would be unfavorable since its computational complexity would result in very long simulation times when calculating the energy consumption for a whole vehicle fleet. Eco-driving approaches focus on the optimal speed for a single-vehicle. For the evaluation of the energy performance of a driving algorithm for CAVs, J. Han et al. proposed an eco-driving control system for energy-optimal acceleration and deceleration and simulated the performance in [6]. Their simulation was based on the assumption, that the preceding vehicle stays the same and there is no other vehicle joining the gap between the controlled vehicle and its leader over a whole driving cycle. However, this assumption is usually not fulfilled in real urban traffic.

The method presented in this study is coupling SUMO with a highly accurate, forward-facing energy model implemented in Simulink [14]. This coupled model allows the implementation of a driving algorithm for one or more vehicles in the SUMO simulation and the following analyses of the energy consumption for the resulting driving cycles with the energy model. Lots of realistic traffic scenarios are generated with SUMO and the performance of different driving algorithms and parameters can be analyzed. With the forward-facing energy model also power-train optimizations of the considered vehicle would be possible for the cycles generated with the driving algorithm.

To validate the Simulation results, this study also presents the speed and state-of-charge (SOC) data collected from a test vehicle in real urban traffic. Therefore, a 2017 Tesla Model S is driven three cycles around the test track for automated and connected driving (TAVF) in Hamburg [5]. A SUMO model of the same road network, the TAVF, is used together with route files representing realistic traffic demand on those streets. With this model, driving cycles for the test vehicle following the same route, as in the measurements are simulated. Thereby, the test vehicle is controlled by a driving algorithm and the resulting cycles are analyzed for the vehicle's energy consumption.

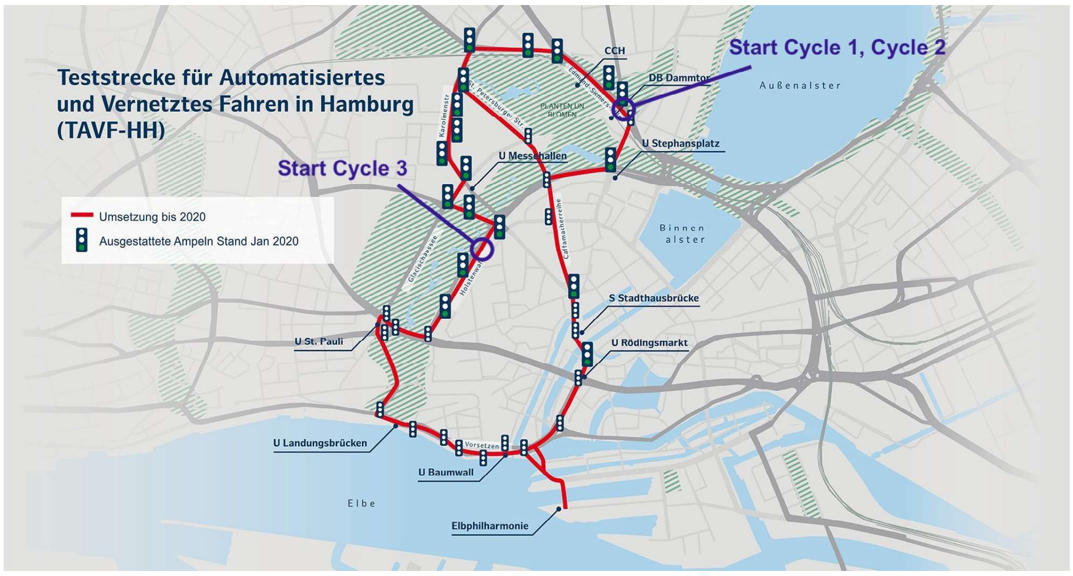


Figure 1: The TAVF with the starting points of Cycle 1, 2, and 3 [2]

2 Experimental Setup

2.1 Route

For the data collection, three driving cycles were measured: Cycle 1, Cycle 2, and Cycle 3. All of them had the same route. The test vehicle was driven one circle on the TAVF in Hamburg [5]. In Cycle 1 and Cycle 2, the starting- (and ending-) point was in front of the train station Hamburg Dammtor and the Cycle 3 started (and ended) at Holstenwall (next to the park Planten un Blomen). The TAVF is shown in Figure 1. Also, the starting points of the cycles are marked. In all three cycles, the big, counterclockwise round on the TAVF was driven over Stephansplatz and Dammtor. At the Elphilharmonie, a turn was performed to continue the route.

The TAVF is located in the center of Hamburg and represents real urban traffic. On the route there are 37 traffic lights, the streets have between 2 and 3 lanes and the speed limit is 50 km/h.

2.2 External Influences

The energy consumption for a cycle on the TAVF has many external influences that do not depend on the driving behavior of the car such as traffic, traffic lights, and temperature. The traffic depends highly on the weekday and time. The experimental data of Cycle 1 and Cycle 2 were taken consecutively on the 16th of December 2020 between 14:00 and 15:00. The ambient temperature was 8 degrees Celsius. During those measurements, the cabin of the car was already well-tempered to around 18 degrees Celsius. Cycle 3 was measured on the 13th of January 2021 between 15:00 and 16:00 at an ambient temperature of 4 degrees Celsius. At the beginning of this cycle, the cabin was still cold and the air conditioning systems were used to a higher extent to warm up the cabin, as shown in the next subsection 2.3.

2.3 Auxiliary Systems

The auxiliary systems with the most significant energy consumption of the test vehicle were analyzed in a previous study [15]. Those values are also taken into account in the energy model (subsection 3.4) with a factor representing the percentage of use of each of those systems. Table 1 shows those factors for each system in the three cycles approximately.

Note, that the AC and ventilation were set to a higher power during the measurements of Cycle 3 compared to the measurements of Cycle 1 and 2.

Measurement	AC	Ventilation	Seat Heater	Headlights	Infotainment
Cycle 1	0.3	0.3	0	1.0	0.1
Cycle 2	0.3	0.3	0	1.0	0.1
Cycle 3	0.8	0.8	0	1.0	0.1

Table 1: use factor of auxiliary systems during the measurements of Cycle 1 to 3

2.4 In-Vehicle Setup

The used test vehicle is a 2017 Tesla Model S 75D [17]. The internal systems of the car communicate over a Controler Area Network (CAN) bus. On the CAN3 bus, data regarding the powertrain is shared. This CAN3 bus can be accessed over pin 18 and 19 of the Tesla Diagnostic Connector under the Touchscreen of the middle console. During the driving cycles on the TAVF, the bus messages are read and logged using the software CANalyzer Mini3 [7]. To ensure that no signals are disturbed and nothing can be sent on the bus, a certified CAN-diode, the CAN-Bus Iso-Koppler [8], is used for the connection.

All messages from the CAN3 are logged in a CSV file. The speed data is published on the bus with the identifier 0x256 with a frequency of 10Hz. The SOC-value has the identifier 0x302 and is published with a frequency of 1 Hz.

2.5 Measurement Results

The collected speed and SOC data for the three cycles are shown in Figure 2. The duration in seconds, the distance in meter, the average speed in km/h and the difference in the SOC in percent of the battery charging state over the whole cycle are given in Table 2 for each cylce respectively:

Measurement	duration (s)	distance (m)	average speed (km/h)	Δ SOC (%)
Cycle 1	1442	8491.4	21.2	2.2
Cycle 2	1406	8605.2	22.03	2.3
Cycle 3	1462	8240.0	20.29	2.7

Table 2: Key values of the measurements of three TAVF cycles with human driver

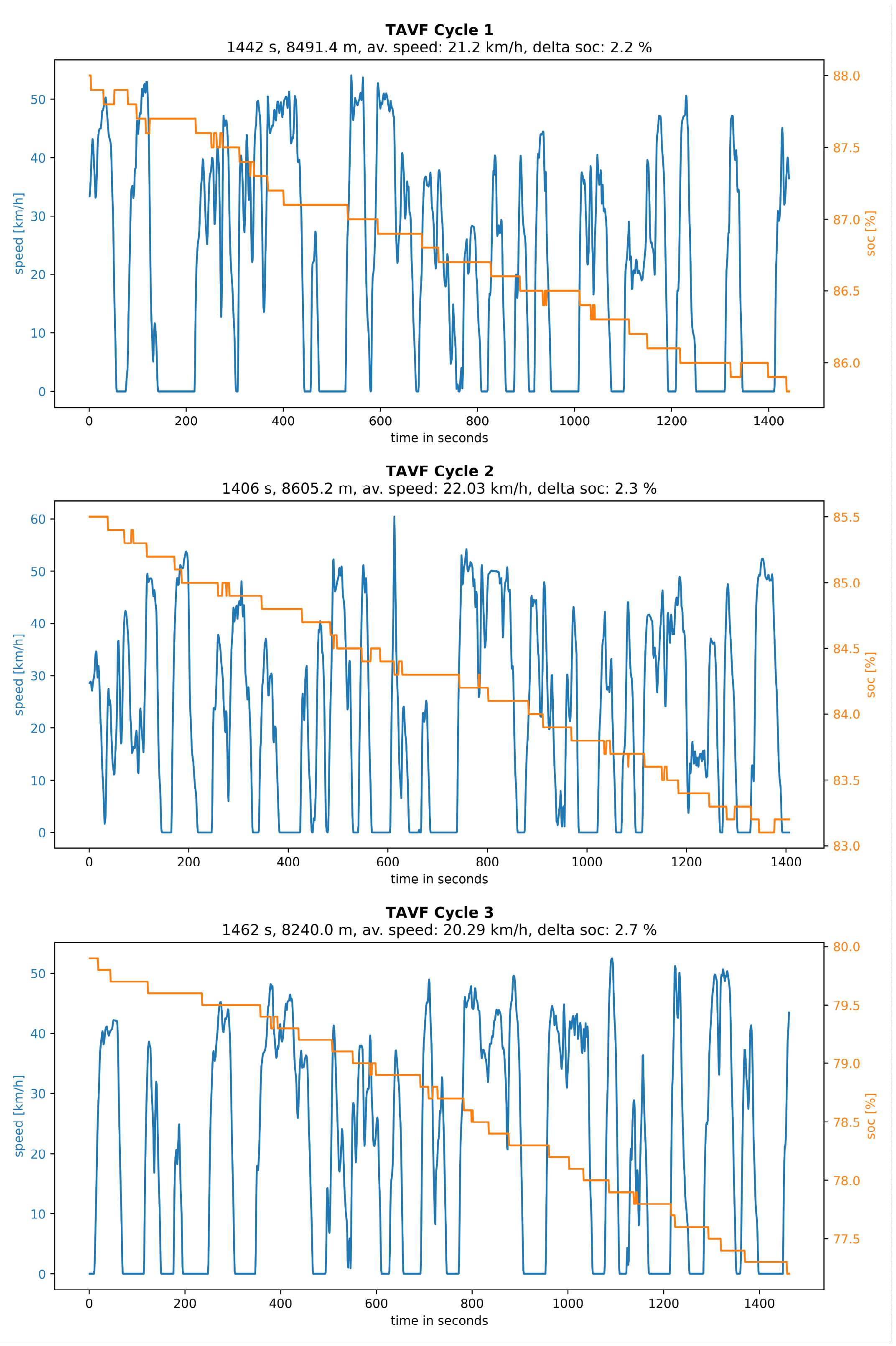


Figure 2: Measurement data from Cycle 1, Cycle 2 and Cycle 3

3 Simulation Setup

3.1 Simulation Toolchain

To test different driving algorithms for the test vehicle in a large number of driving scenarios on the TAVF, the toolchain of different simulations is realized as a python script. The script couples both simulation tools SUMO and Simulink runs a chosen number of random scenarios and saves the resulting speed over time and SOC over time data of the test vehicle. Also, the driving algorithms are implemented in python.

At the beginning of a simulation run, the number of cycles to be simulated and the driving algorithm of the test vehicle is specified. Then, for the chosen number of cycles, SUMO is started as a server using the Traffic Control Interface (TraCI), and the simulation is executed. The time-step length of the simulation was set to 1 second for this study. Between every simulation step, a driving algorithm implemented in python can calculate the speed for the test vehicle in the simulation for the following step. Therefore, the driving algorithm can access data from the current situation of the test vehicle in the SUMO simulation. Once the test vehicle has finished its round on the TAVF, the logged speed over time data is passed to the Simulink energy model. The Simulink model is started and controlled from the python script with the Matlab engine for python.

After the model has calculated the energy consumption for the cycle, the SOC over time array is returned to the python script. The SOC and speed data for the cycle are saved as a file and as a plot figure.

The script runs until the chosen number of scenarios is generated. Each of those scenarios is different since each of the SUMO simulations is initialized with another seed value for the pseudo-random number generator.

3.2 Required Software

The top layer python script is implemented in python version 3.6.8 [18]. This is the newest version that is still compatible with the Matlab engine for python [14], which is also used. The energy model used in this study requires Simulink, the Powertrain Toolbox, and the Parallel Computing Toolbox of MathWorks from version 2020b or newer. In the python code, the cycle data is analyzed and saved using the NumPy [16] library. For the SUMO simulation the SUMO version 1.7.0. is used [11].

3.3 SUMO Simulation

SUMO is used to generate random scenarios for the route of the test vehicle. With SUMO, the chosen driving algorithm can be analyzed under a big range of different traffic models for example on highways, in urban traffic, or certain cities or districts. The only requirements are accurate network and routing files for the whished traffic situations and enough computer power and time to run the simulations. The driving algorithm can be used to control one or more vehicles in the simulation. Note that especially the analyses of the energy consumption with the forward-facing energy model is computationally expensive, which increases the run time of this toolchain significantly if the energy consumption is calculated for multiple vehicles of the same scenario. Also, the energy model has to be parameterized for the exact vehicle it is representing (see subsection 3.4).

In this study, all generated cycles are done on the same road network, which is a model of the TAVF and only one vehicle in the simulated scenarios is controlled by the driving algorithm and analyzed with the energy model.

This vehicle is representing the test vehicle of the measurements. It virtually drives the same route on the TAVF as the real test vehicle in the measurements with the starting point at Hamburg Dammtor (as in Cycle 1 and 2 of the measurements) in every simulation scenario. Also, the other vehicles in the simulation - the traffic demand - are the same for every simulation. SUMO allows pseudo-random changes in the driving of the other vehicles and a random departure offset of every vehicle based on an integer seed value. In one simulation run, for every SUMO simulation, the seed value is increased by one. This way, many different cycles are generated with the same route files.

The used network file and route files of the TAVF were provided from the German Aerospace Center (DLR). The route files are based on induction loop count data of the streets of the TAVF from the authorities of the state of Hamburg. Cars and trucks were generated from this data using a GEH statistic so that they represent weekday traffic between 15:00 and 16:00 with an accuracy between 95 and 99 %. The cars and trucks have a sigma value of 0.5, which means their driving behavior includes random deviations from a perfect one with a factor of 0.5. Additionally, buses are considered in the route files. The bus routes are based on the schedules from the Hamburger Verkehrsverbund (HVV, the public transport provider in Hamburg) from autumn 2019.

With the network file also the traffic lights are defined. In the provided network file of the TAVF, three of the 37 traffic lights of the TAVF are implemented based on the real phases of their counterparts, namely LSA 34, 560, and 94. The rest of the traffic lights were generated automatically by SUMO and partly optimized to guarantee realistic traffic flow during the simulations.

The different cycles simulated with SUMO from the python script all have the same configuration file except for the seed value, which is increased with every simulation by one. Therefore, the simulated scenarios are repeatable, but the behavior of the cars and their starting time varies randomly in every single scenario. The seed value for the random number generator (RNG) of SUMO is influencing the random behavior of other vehicles and the random depart offset of each vehicle, which is set in the simulation configuration to a maximum of 30 s. Therefore, every vehicle departs with a random offset between 0 and 30 s to their programmed departure time in every simulation.

3.4 Energy Simulation

The used energy model is based on the Electric Vehicle Reference Application from MathWorks [14] which was parameterized for the 2017 Tesla Model S and extended for the most energy-consuming auxiliary systems of the Tesla in a previous study [15].

The model takes a driving cycle in the form of a two-dimensional vector as an input. In the first column, the time in seconds is passed and the second column contains the corresponding speed in km/h. This driving cycle with one speed value per second is seen as the target speed cycle. The car model calculates a torque request based on the actual speed and the target speed. From this torque request, the whole drive train is calculated in a forward and causal way, resulting in an actual speed of the vehicle.

Besides the driving cycle input, the extended model also takes the percentage of use of the auxiliary systems over time as an input. Those percentages result in factors that are multiplied with the maximum current measured for the respective system in the test vehicle.

In [15] the maximum current of the air-conditioning (AC), the ventilation, the seat heater, the headlights, and the infotainment (speaker, radio) of the 2017 Tesla Model S was measured.

The AC and the ventilation are supplied from a 400V system meanwhile the rest of the considered systems run with a 12V power supply. The maximum current of the systems was measured to be for the AC 6A, the ventilation 2A, the seat heating 0.6A, the headlights 0.4A and the infotainment systems 0.2A.

For simplification, the use factors for each of those systems were not changed over time in this study. Therefore, they were assumed to stay unchanged for the whole cycle. Also, the atmospheric temperature is considered by the model and taken as an input.

Since the model calculates the whole drive train from the motor to the wheels, the performance of all internal systems can be accessed at any time point of the cycle. For this study, the SOC data in % of the battery is used to compare the energy consumption of different cycles. This data is outputted as a vector of the same length as the input drive cycle containing the SOC value in % of the battery for every second over the cycle time.

3.5 Driving Algorithms

With the method presented in this study, any longitudinal driving algorithm that is based on input data available in the SUMO simulation can be implemented in the python script to control the speed of one or multiple vehicles in the simulation. The performance of an implemented algorithm can then be tested over lots of random scenarios and under different traffic conditions. Furthermore, with the method presented in this study, the algorithm can be analyzed in its energy consumption for a certain vehicle and its parameters can be optimized for the chosen traffic conditions.

Over the Traffic Control Interface (TraCI) of SUMO, the algorithm can access data of the vehicle's surroundings from the SUMO simulation in between each simulation step. A selection of the information that can be accessed with TraCI is given below.

- The current speed of the vehicle
- The current lateral speed of the vehicle
- The current position of the vehicle
- A list of traffic lights on the vehicle's route with their current state and distance
- The leading vehicle of the car and its distance
- The current speed of any vehicle in the simulation

Based on this input data, the algorithm can control the vehicle's speed over the whole simulation. The route of the controlled vehicle can be programmed before in the route file so that the algorithm has to take care only of the longitudinal movement. The vehicle can be controlled by the algorithm by setting either its speed or its position for the next simulation step. It is also possible to consider lane changes in the driving algorithm. In that case, it can be implemented as a car-following model in SUMO.

To show the applicability of the presented method, the energy efficiency of the driving algorithm of the Kraus car-following model [9] is analyzed. This algorithm is already implemented in SUMO and calculates the speed of a vehicle for the next simulation step based on the parameters of its acceleration, deceleration, emergency deceleration, the minimal gap it should have to a leading vehicle, a value sigma representing the imperfection in the driving behavior and a value tau representing the reaction time a driver needs to react on changing conditions. The values sigma and tau are implemented to simulate non-perfect, human driving behavior in SUMO. Since in this study, the algorithm is used to simulate an autonomous vehicle, sigma is chosen to be 0 (no random variation in speed) and tau to the shortest possible value of the simulation step length, which is set to one second for the simulations in this study.

The algorithm decelerates the vehicle with a deceleration corresponding to the deceleration parameter value d to stop if a red or yellow traffic light is coming up or to respect the minimal gap to any leading vehicle. Otherwise, the vehicle is accelerated depending on the acceleration parameter a.

4 Model Validation

4.1 Energy Model Validation

The energy model was run for each measurement cycle with the required inputs: atmospheric temperature, the percentage of use of the considered auxiliary systems, and the speed cycle. Those values for the Cycles 1 to 3 are shown in section 2.

In Figure 4 the resulting SOC calculations for Cycle 1, 2, and 3 are plotted in red on top of the corresponding measurements.

One can see that the SOC values calculated with the energy model are ruffly following the measured ones. Note that the measured SOC was logged with a precision of 0.1 and the calculated values were rounded to a precision of 0.001. For Cycle 1, the delta SOC, so the difference of SOC at the beginning of the cycle to the end of the cycle, is measured to be 2.2 % and calculated to be 2.219 %.

The absolute and relative differences between measured and calculated Δ SOC % consumed by each cycle are listed in Figure 3.

TAVF Cycle	measured [%]	calculated [%]	difference abs. [%]	difference rel.
Cycle 1	2.2	2.219	0.019	0.86 %
Cycle 2	2.3	2.291	-0.009	-0.39 %
Cycle 3	2.7	2.673	-0.027	-1.00 %

Table 3: Δ SOC [%] measured and calculated for each TAVF cycle

Cycle 3 shows the biggest gap with an absolute difference of 0.027 % SOC. In all three Cycles, the energy model comes to a Δ SOC with a relative deviation of 1 % or less.

In Figure 3 the difference in the SOC value in % from the energy model (calculated) to the measured one is shown over the whole cycle for TAVF Cycle 1, 2, and 3. In the plots, the highest resolution of the measurement values of 0.1 is marked in yellow.

One sees that even though the calculated SOC value is quite close to the measured one at the end of the cycles, during the cycles the difference is significantly higher. It can also be seen that this difference follows a bit of a pattern, where it is slightly increasing over more or less the whole cycle and then at some point dropping at a comparably high rate. In Cycle 1 and Cycle 2, this drop happens at about the same period of around 500 to 700 and 600 to 800 seconds meanwhile in Cycle 3 the drop can be seen between 150 and 350 seconds approximately.

This phenomenon can be explained by the road grade of the route. The TAVF is comparably flat over most of its length but has a higher negative road grade between U St. Pauli and U Landungsbrücken. Since the road grade is not considered in the energy model, in this part of the cycle the test vehicle consumes less energy than calculated. This results in higher calculated energy consumption over that period than measured, and consequently the difference between

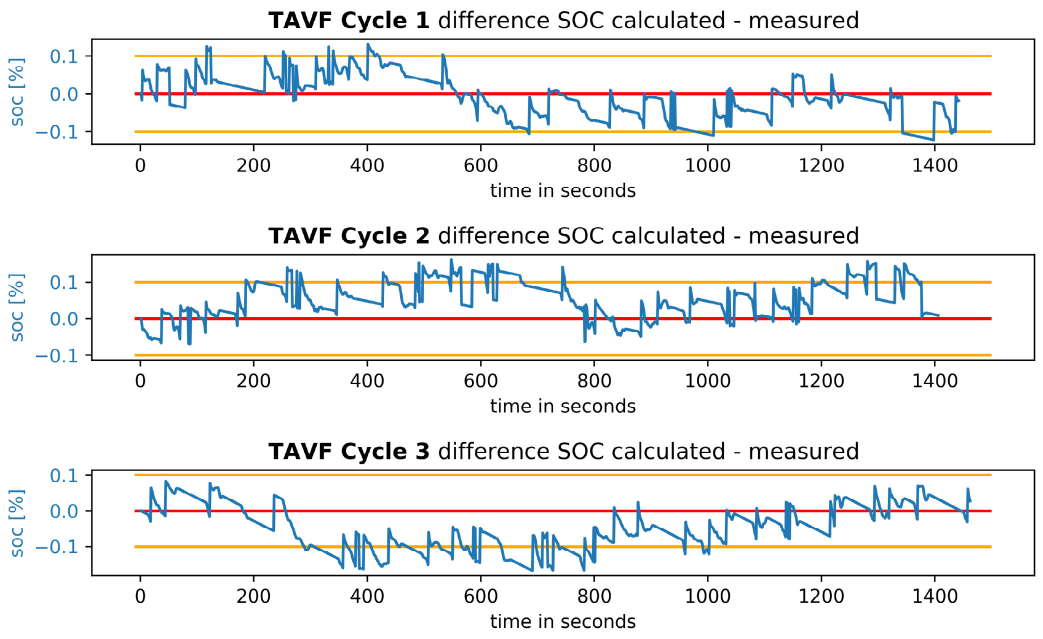


Figure 3: Difference from calculated SOC to measured SOC

calculated and measured SOC is decreasing. This explaination also agrees with the results of the study of S.C. Yang et al. [20], in which the influences of different road grades on an EVs energy consumption was analyzed.

The road grade of the rest of the TAVF does not have such significance at any other part of its length. Evidently, it does have a positive road grade at other parts to come back to the same height after a full round. Therefore, since the measured cycles were all going over a full round on the TAVF, during the rest of the cycle this effect is canceled out by the model calculating a lower energy consumption at the positive road grade parts.

The time period of the drop in the difference of calculated and measured SOC also meets the assumptions for the three cycles. Cycle 1 and 2 both have the same starting point at Dammtor. And they both show the drop event in about the same time period between 500 to 800 seconds from the cycle start.

Cycle 3 on the other hand started at Holstenwall, which is closer to the negative road grade part. Therefore, this cycle also shows the drop event earlier.

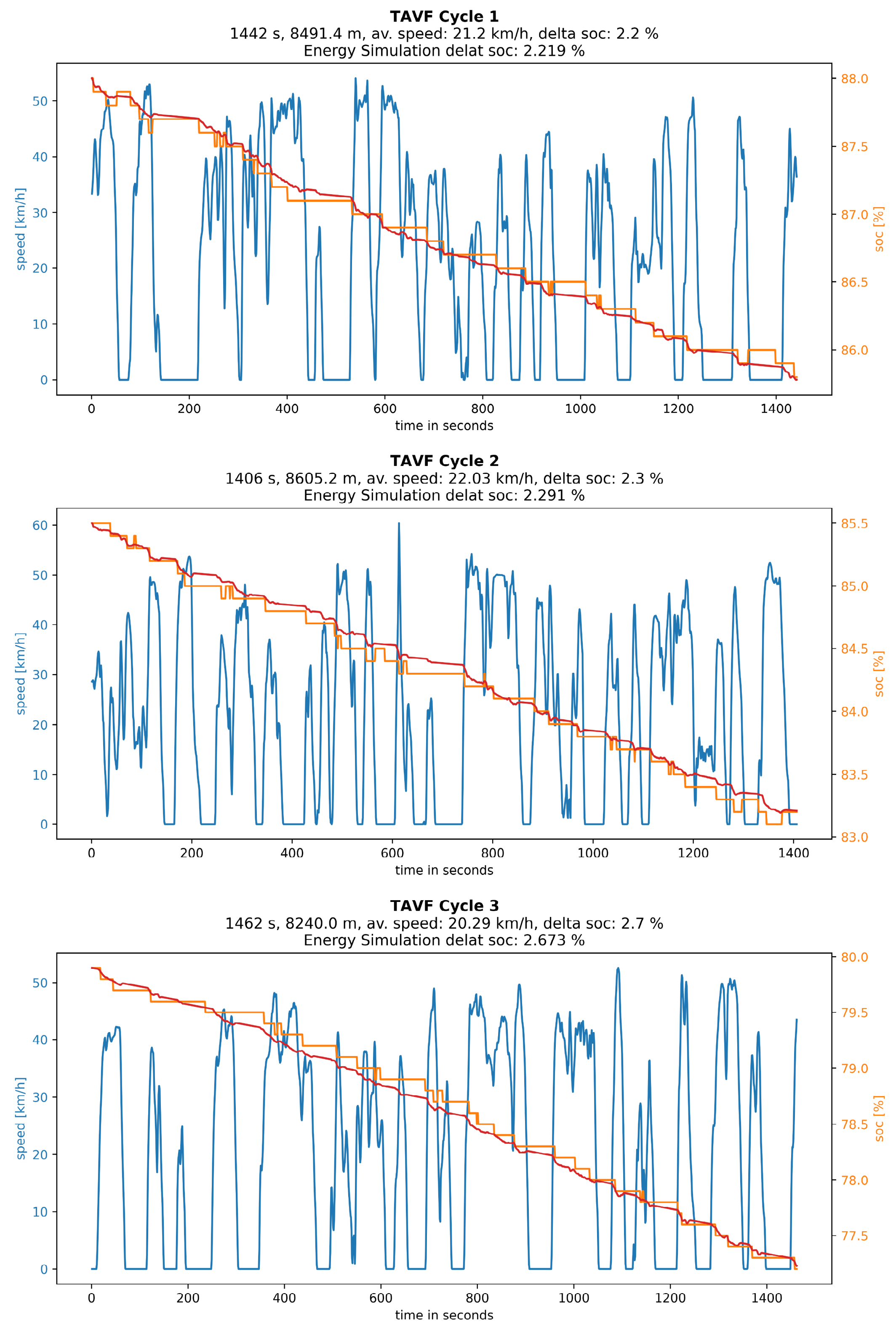


Figure 4: Cycle 1, Cycle 2 and Cycle 3 measurements and the SOC of the energy model (red)

4.2 TAVF Driving Cycle Generation with SUMO Model

For the method presented in this study, SUMO is used to generate realistic driving scenarios for a vehicle under fixed traffic conditions. With the used SUMO model for the TAVF and the traffic presented in subsection 3.3, the resulting driving cycles of the test vehicle of 100 pseudo-random simulations were analyzed and compared to the measured driving cycles of the TAVF.

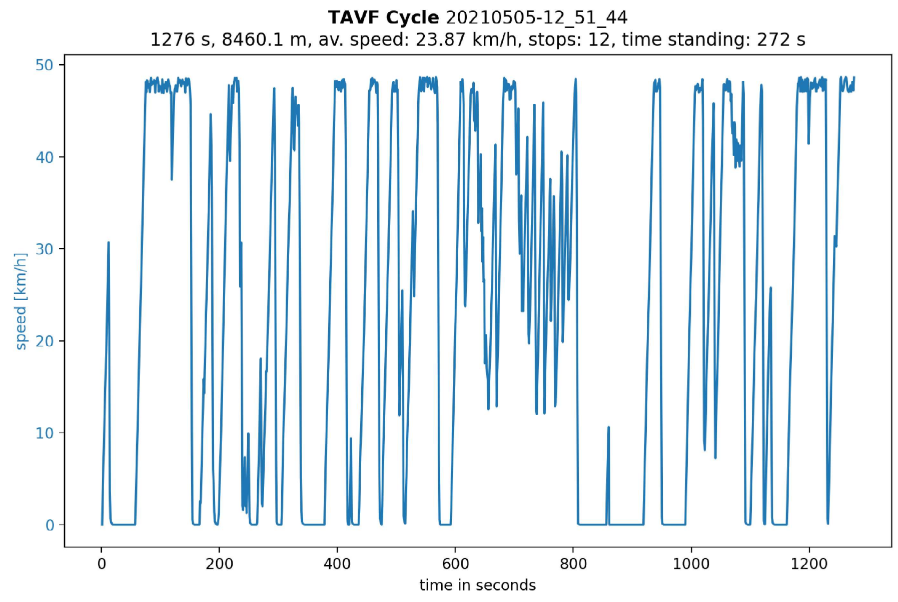


Figure 5: One of 100 random TAVF cycles generated with SUMO

For the 100 cycles, the test vehicle was simulated with the Kraus car-following model in SUMO. Its acceleration was set to 1.0 m/s^2, the deceleration to 5 m/s^2 and the value for sigma - which represents a driving imperfection in SUMO - was chosen to be 0.5. In Figure 5 one of those 100 cycles is shown. When compared to the measured driving cycles from Figure 4, one can see in the example plot that the driving behavior simulated in SUMO does not match the real human driving behavior from those cycles very closely. Nevertheless, the generated cycles do deliver some clues about the suitability of the TAVF model in SUMO. Therefore, the values of duration, distance, average speed, the total time standing, and the number of stops are observed for the generated cycles and compared to the measured cycles.

All of these values are approaching a normal distribution over the 100 random cycles. Their average, maximum, minimum, and standard deviation for the 100 simulations are shown in Table 4 together with the data of the measurements. The time standing is the number of seconds where the vehicle had a speed of 0 km/h and for the number of stops, the occurrences of standing situations (consecutive speed of 0 km/h) are counted where short movements of less than 5 seconds (up to four consecutive seconds with a speed other than 0 in between two standing situations) are respected as one single stop. This prevents a stop at a single traffic light where the vehicle moves forward again after stopping to close the gap to the leading vehicle

TAVF Cycle	duration	distance	average speed	stops	time standing
Cycle 1	1442 s	8491.4 m	21.20 km/h	15	428 s
Cycle 2	1406 s	8605.2 m	22.03 km/h	16	340 s
Cycle 3	1462 s	8240.0 m	20.29 km/h	15	489 s
100 generated cycles:					
average	1394 s	8464.2 m	22.08 km/h	15.95	384 s
max	1772 s	8472.1 m	28.57 km/h	23	662 s
min	1067 s	8450.8 m	17.19 km/h	9	127 s
std. dev.	137.11 s	4.35 m	2.22 km/h	3,1	98,47 s

Table 4: key values of random SUMO cycle generation compared with measurements

in the same red phase to be counted as two stops.

The statistics of the simulated cycles show that the model matches the values from the measured Cycle 1, 2, and 3 in the chosen categories. The values for duration, average speed, and stops of all three measured Cycles are inside one standard deviation from the mean value over the 100 simulated cycles. For the standing time, only Cycle 3 is outside this boundary with 489 seconds, being 1.35 % higher than the one standard deviation over the mean. Also, this Cycle is well below the maximum standing time of all 100 generated cycles which is 662 seconds. The only category in which the value range of the simulated cycles does not cover the values of the measurements is the distance of the cycle. Here, the standard deviation over the values of 100 simulated cycles is very little with 4.35 m. The difference in the distance of the simulated cycles is, therefore, a lot smaller than the difference between the distance of the measured cycles, which have a maximum difference of 365.3 m. This could partly be caused by the driving behavior of the vehicle in the simulation. Here, the test vehicle performs only the necessary lane changes for its route, which is the same in every simulation. In the measured cycles, the human driver also performed some lane changes that were not necessary to complete the route. This effect explains a greater difference in distance between the measured cycles compared to the simulation. It can also explain a higher total distance in the measured cycles because of more (unnecessary) lane changes as one sees in Cycle 1 (27.2 m more than the average of the simulated cycles) and Cycle 2 (141,0 m more). What it can not explain is a measured distance that is lower than the lowest simulated one, like the one of Cycle 3 with 210,8 m less. One can therefore assume that the route of the used TAVF model in SUMO is at least about this distance longer than the real route on the TAVF (that would be 2,56 %).

Overall, the randomly generated cycles show that the TAVF model used in this study matches the expectations of the three measured cycles of the real TAVF. Excepts for the distance where the lowest simulated one is 2.56 % larger than the lowest measured one, the values of the 100 simulated cycles are all distributed over a range that contains the measured ones. If one assumes that the used model was representing the TAVF perfectly, the measured values of Cycle 1 to 3 for the duration, average speed, stops, and standing time do not only represent possible but also very probable quantities when compared to the distribution of the values from the simulations. Vice versa, the presented results can be seen as evidence that the network and routing files used in the simulation are an accurate model of the real TAVF with realistic traffic.

5 Simulation Results

In this section, the simulation results for the energy consumption of the driving algorithm of the Kraus car-following model are presented. With the method described in section 3, the test vehicle is simulated to drive always the same route on the TAVF. During the simulation, its speed is controlled from the algorithm of the Kraus car-following model described in subsection 3.5 with different parameters for acceleration and deceleration. Then, the performance of the algorithm is analyzed for the energy consumption, duration, stop time and the number of stops.

The acceleration parameter a was set to 0.5, 1.5, 2.5, 3.5 and 4.5 m/s^2 and the decelartion parameter d to 1.5, 2.5, 3.5, 4.5, 5.5 and 6.5 m/s^2. For each possible combination of a and d in the driving algorithm, the test vehicle was simulated over 20 random cycles and its energy consumption has been calculated.

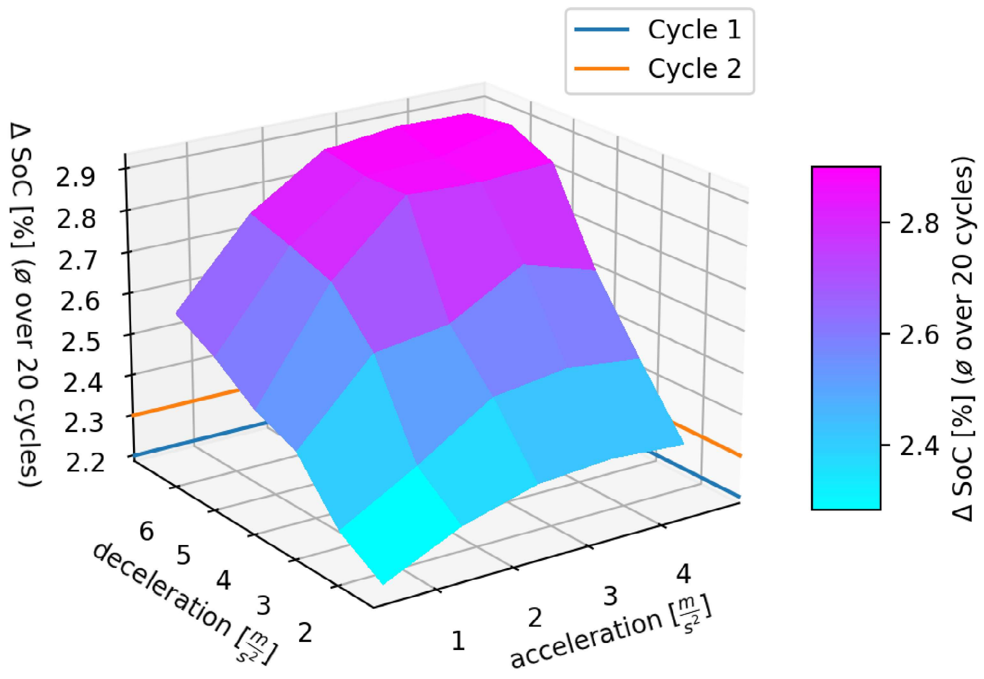


Figure 6: Average Δ SOC over random 20 TAVF cycles for different acceleration and decelaration parameters with low auxiliary system use

In Figure 6 the average difference in SOC per cycle over 20 cycles is plotted for different combinations of the driving algorithm parameters of a and d. The energy model was set to the lower auxiliary system use of Cycle 1 and 2 of the measurements (see subsection 2.3). One can see a trend for higher energy consumption if the algorithm is set to higher a and d values.

Figure 7 shows the average duration and average speed per TAVF cycle over 20 cycles for the same a and d values. Since the average cycle distance does not vary more than 10 m between the simulated parameters (the minimum is 8457 and the maximum 8466 m), duration and average

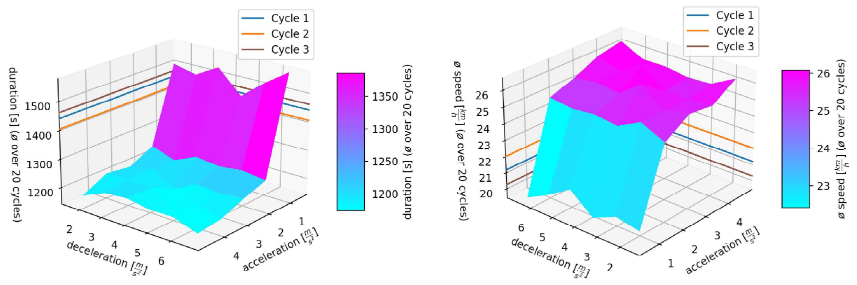


Figure 7: Average duration (left) and average speed (right) over random 20 TAVF cycles for different acceleration and decelaration parameters

speed data show the same trend. One can see that for a values of 1.5 m/s^2 and higher the average cycle duration remains very similar around 1200 seconds. For those parameters, the maximum duration is 1237 s (at $a = 1.5$ m/s^2 and $d = 3.5$ m/s^2).

Meanwhile a values above 1.5 m/s^2 do not result in significant changes in the cycle duration, the simulation results show significantly higher values for a lower a of 0.5 m/s^2. Here, the duration is between 1444 and 1568 s, which is between 16,7 % and 26,8 % higher, than the highest result for other parameters (1237 s). The parameter d does not influence the cycle duration considerably over the whole range of simulated values.

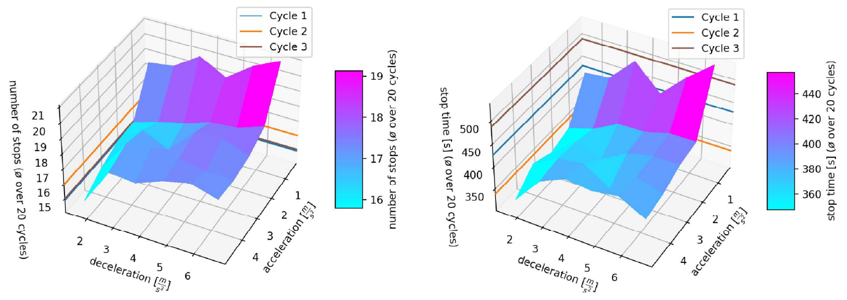


Figure 8: Average number of stops (left) and average stoping time (right) per cycle over random 20 TAVF cycles for different acceleration and decelaration parameters

The average number of stops and the average stopping time per cycle are shown in Figure 8. The plots indicate that generally for the simulated scenarios and over all the acceleration and deceleration parameters, more stops lead to longer stopping time. Also, they show that for a low a of 0.5 m/s^2 the stopping time is significantly higher than for a values of 1.5 m/s^2 and above. This can explain the longer cycle duration for those simulations and the lower average speed. The results for the average number of stops and stopping time in the simulated TAVF cycles for different acceleration and deceleration parameters of the driving algorithm also show a light trend of lower values for a lower d meanwhile the average cycle speed does not show this trend. Therefore, for the lower deceleration values, lower stopping time and fewer stops do not result in a higher average speed or lower cycle duration.

Regarding the average energy consumption per cycle for lower auxiliary system use in Figure 6, the cycles with an a value of 0.5 m/s^2, which have a longer cycle duration, still show a lower energy consumption than the cycles with higher a values that have shorter trip duration.

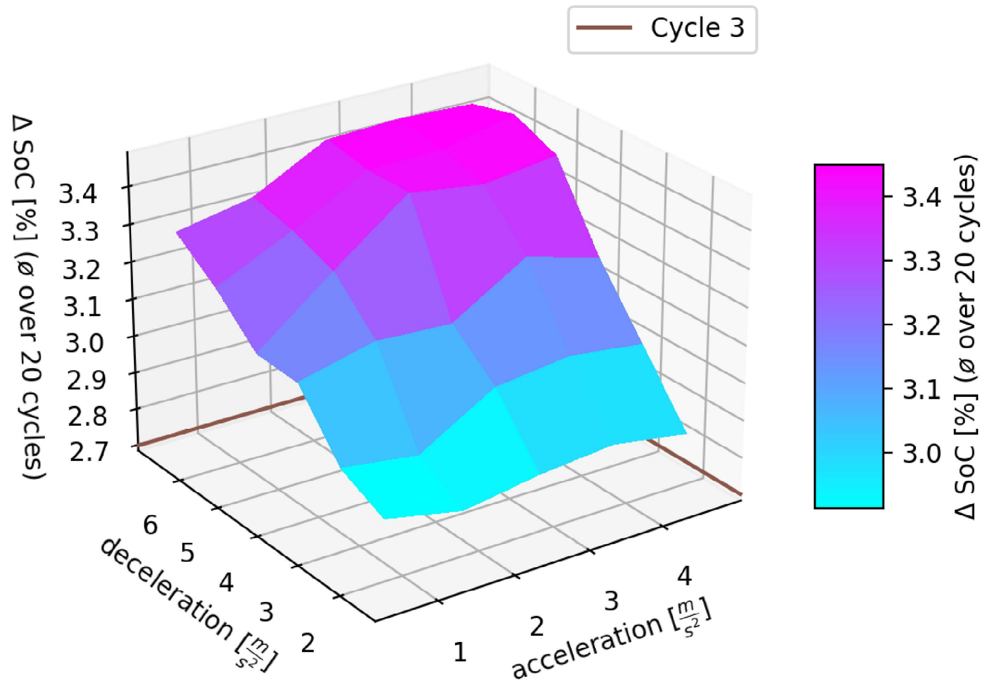


Figure 9: Average Δ SOC over random 20 TAVF cycles for different acceleration and decelaration parameters with high auxiliary system use

In Figure 9, the energy consumption for the same cycles is shown with higher auxiliary system use. Therefore, the energy model is set to the parameters for auxiliary systems and atmospheric temperature of the measurements Cycle 3. One can see that with this auxiliary system use, the energy consumed in all cycles is higher. The average Δ SOC per cycle over all the cycles is here 3.196 %. For the energy calculations with lower use of auxiliaries in Figure 6, the average over all the cycles is 2.610 %. Therefore, the energy consumption per TAVF cycle increased by 22,4 % on average when changing from a low auxiliary system use as in measurement Cycles 1 and 2 (AC and Ventilation to 30 %) to a higher one as in the measurements of Cycle 3 (AC and Ventilation to 80 %, see subsection 2.3).

One can also see when comparing Figure 6 and Figure 9, that the longer cycle duration for simulations with an a value of 0.5 m/s^2 leads to a larger increase in energy consumption for the high auxiliary system use than for other a values, that correspond to lower cycle durations. In the cases for $d = 1.5$ m/s^2 and $d = 2.5$ m/s^2, one can see that with the high auxiliary system use, the energy consumption is even larger for an a of 0.5 m/s^2 than for one of 1.5 m/s^2, due to the longer duration of those cycles. In fact, in the simulations with $d = 1.5$ m/s^2, the average Δ SOC value over 20 cycles for different acceleration was even the second highest for an accelarion of 0.5 m/s^{2}with 2.905 % (only with $a = 3.5$ m/s^{2}the resulting energy consumption was slightly higher with a Δ SOC of 2.912 %).

The lowest average energy consumption per cycle achived by the algorithm is 2.862 % SOC for high auxiliary system use with the parameters $a = 1.5$ m/s^2 and $d = 1.5$ m/s^2. For the simulation with low auxiliary system use (Figure 6), the best value of 2.204 % is achived with the lowest simulated parameter values of $a = 0.5$ m/s^2 and $d = 1.5$ m/s^2.

6 Conclusion

A method to evaluate the energy consumption of an EV controlled by a driving algorithm in realistic urban traffic was developed in this paper. The simulation model couples the microscopic traffic simulator SUMO with a forward-facing energy model in Simulink. The energy model was validated with speed and SOC data collected under real urban traffic on the TAVF in Hamburg, Germany with a 2017 Tesla Model S 75D. The energy model with the parametrization of [15] was shown to calculate an energy consumption with a difference of 0.86 %, -0.39 % and -1.00 % to the measured one for the three measured TAVF cycles, respectively.

The used network and route files of the TAVF in SUMO were shown to deliver realistic cycle statistics when simulating driving cycles with the same virtual route on the TAVF in SUMO, that was driven for the measurements.

With the introduced method, a driving algorithm for a vehicle can be analyzed on the vehicle's energy consumption and also on other statistics such as trip duration, trip distance, average speed, number of stops, and stopping time. The compatibility for algorithms is so far limited to input variables available in SUMO and the vehicle can not be controlled by a torque request, but by setting its speed directly. However, in this direction, the model can possibly be extended (see section 7).

To demonstrate the applicability of the simulation model, the driving algorithm of the Kraus car-following model was analyzed. Therefore, multiple, random scenarios for different values of the acceleration and deceleration algorithm parameters were simulated. In each of the simulations, the driving algorithm controlled the speed of the test vehicle during one cycle on the TAVF. It was shown that the introduced simulation model can serve as an energy consumption analysis and optimization tool for a driving algorithm. With the model, a driving algorithm can be used to control one or multiple cars in any traffic scenario implemented in SUMO. The energy consumption for a simulated driving cycle can be calculated for any given vehicle, that the energy model is parameterized for.

The simulation results show that, for the Kraus driving algorithm on the TAVF with low use of auxiliary systems, the cycles with the lowest values for the acceleration and deceleration parameters of 0.5 m/s^2 and 1.5 m/s^2 respectively have the lowest energy consumption with a Δ SOC of 2.204 %. This is very close to the energy consumption of the measured, human-driven cycles with the same auxiliary system use, namely Cycle 1 and Cycle 2. Here, the consumed energy was measured to be 2.2 % and 2.3 %, respectively. It was also shown that the trip duration for one cycle on the TAVF does not depend on the deceleration parameter of the driving algorithm. For the acceleration parameter a, the cycles simulated with a value of 0.5 m/s^2 had durations up to 26.8 % higher than the one simulated with higher values for a. Meanwhile, the duration of cycles simulated with $a = 1.5$ m/s^2 did not vary from the ones simulated with higher a values by more than 6.68 %.

Furthermore, the energy simulations with higher use of the auxiliary systems AC and ventilation show that, for the same cycles, higher steady energy use results in a larger reduction of energy efficiency for cycles with a higher duration. Therefore, in this simulation with the higher auxiliary system use, the parameters deceleration $= 1.5$ m/s^2 and acceleration $= 1.5$ m/s^2 resulted in the lowest average energy consumption due to their lower cycle duration. With these parameters, the driving algorithm achieved an average Δ SOC of 2.862 % for a TAVF cycle.

This is relatively 6 % more than the measured Δ SOC of 2.7 % of Cycle 3 with a human driver, which had the same, higher auxiliary system use.

The energy simulations demonstrate the significant effect of auxiliary systems on a vehicle's energy consumption. Over all the 600 TAVF cycle simulations, the consumed energy per cycle increased by 22.4 % when changing the use factor for the AC and ventilation from 30 % to 80 %. This finding also overlaps with the results of [3].

7 Future Work

The authors intend to extend the model presented and couple it with the open-source robot simulator Webots [1]. This software can generate a 3D simulation environment corresponding to the street network of the SUMO Model including all vehicles from the SUMO simulation. In Webots, real sensors like lidar and camera sensors can be simulated for the test vehicle. It is planned to implement a driving algorithm that controls the car based on simulated vehicle cameras in Webots. Therefore, the driving algorithm in the simulation could use the same input data to control the car as the real 2017 Tesla Model S is using for automated driving.

References

[1] Cyberbotics Ltd. Webots.

[2] Geschäftsstelle der Teststrecke für automatisiertes und ver-netztes Fahren Hamburg c/o ITS mobility e. V. Tavf streckenkarte. https://tavf.hamburg/fileadmin/user_upload/Bilder/TAVF_Streckenkarte_quer_de_200108.jpg. Last accessed on 2021-05-16.

[3] Chiara Fiori, Kyoungho Ahn, and Hesham A Rakha. Power-based electric vehicle energy consumption model: Model development and validation. *Applied Energy*, 168:257–268, 2016.

[4] Ray Galvin. Energy consumption effects of speed and acceleration in electric vehicles: Laboratory case studies and implications for drivers and policymakers. *Transportation Research Part D: Transport and Environment*, 53:234–248, 2017.

[5] Geschäftsstelle der Teststrecke für automatisiertes und vernetztes Fahren Hamburg c/o ITS mobility e. V. Teststrecke für automatisiertes und vernetztes fahren in hamburg. https://tavf.hamburg.

[6] Jihun Han, Antonio Sciarretta, Luis Leon Ojeda, Giovanni De Nunzio, and Laurent Thibault. Safe-and eco-driving control for connected and automated electric vehicles using analytical state-constrained optimal solution. *IEEE Transactions on Intelligent Vehicles*, 3(2):163–172, 2018.

[7] HMS Industrial Networks. cananalyzer mini.

[8] IPETRONIK GmbH & Co. KG, Parsevalstraße 9a. *SAM-CAN-ISO011*.

[9] Stefan Krauß. Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics. 1998.

[10] Tamás Kurczveil, Pablo Álvarez López, and Eckehard Schnieder. Implementation of an energy model and a charging infrastructure in sumo. In *Simulation of Urban MObility User Conference*, pages 33–43. Springer, 2013.

[11] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582. IEEE, 2018.

[12] Blaž Luin, Stojan Petelin, and Fouad Al-Mansour. Microsimulation of electric vehicle energy consumption. *Energy*, 174:24–32, 2019.

[13] Tony Markel, Aaron Brooker, Terry Hendricks, Valerie Johnson, Kenneth Kelly, Bill Kramer, Michael O'Keefe, Sam Sprik, and Keith Wipke. Advisor: a systems analysis tool for advanced vehicle modeling. *Journal of power sources*, 110(2):255–266, 2002.

[14] The Mathworks, Inc., Natick, Massachusetts. *MATLAB version 9.9.0.1467703 (R2020b)*, 2020.

[15] Christopher Oellerich and Rasmus Rettig. Entwicklung eines modells zur simulation der energiebilanz beim betrieb eines elektrofahrzeugs, 2020. unpublished.

[16] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[17] Tesla Model S 75D. *Tesla Model S characteristics*, 2017.

[18] Guido Van Rossum et al. Python programming language., 2007.

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

[20] SC Yang, M Li, Y Lin, and TQ Tang. Electric vehicle's electricity consumption on a road with different slope. *Physica A: Statistical Mechanics and its Applications*, 402:41–48, 2014.