SUMO Conference Proceedings

3 | 2022

TIB OPEN PUBLISHING

SUMO User Conference

09 – 11 May 2022
Virtual Event

SUMO
User Conference
May 09-11, **2022**

Editors:

Pablo Alvarez Lopez
Olaf Angelo Banse Bueno
Michael Behrisch
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

Volume 3

SUMO User Conference 2022

09 – 11 May 2022, Virtual Event

Conference papers

https://doi.org/10.52825/scp.v3i

Editors

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

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

Michael Behrisch, 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 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

TIB OPEN PUBLISHING

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.

SUMO User Conference 2022
https://doi.org/10.52825/scp.v3i.115

Published: 29 Sept. 2022

Extension and Validation of NEMA-Style Dual-Ring Controller in SUMO

Max Schrader[1], Qichao Wang[2], and Joshua Bittle[1][https://orcid.org/0000-0003-4524-3316]

[1] University of Alabama, Tuscaloosa, Alabama, USA
mcschrader@crimson.ua.edu, jbittle@eng.ua.edu

[2] National Renewable Energy Laboratory,
Golden, Colorado, U.S.A.
qichao.wang@nrel.gov

Abstract

Until recently, SUMO users could not model the behavior of a ring-and-barrier traffic signal via existing signal types, which left North American SUMO users without the direct ability to capture traffic dynamics of their local networks. This work presents the meth-ods, implementation overview, and validation of a 'dual-ring' NEMA style traffic controller which has recently been added to the main SUMO code base. A brief explanation of the 'dual-ring' implementation is also provided as context for those new to this type of traffic controller. The foundation for this work was presented at the SUMO User Conference 2021 by researchers at the US Department of Energy's National Renewable Energy Laboratory, but was not integrated into SUMO code base at the time. Following the initial inclusion of the controller to SUMO, the authors began validation of the SUMO controller against an Econolite software-in-the-loop (SIL) traffic signal controller configured with actual setup parameters from controllers in a real-world three-intersection corridor in Tuscaloosa, Al-abama, USA. This paper documents the process of adding new features to the controller code as well as validating their implementation through simulation-based and automated grey-box testing is presented in this paper. Key features such as fully-actuated operation, various timing offset plans, proper next-phase fit algorithms and more, have been added and validated against this SIL system. Though not an exhaustive demonstration of fea-tures, this work is intended make more users aware of this extension of SUMO capabilities.

1 Introduction

Prior to Wang, Li and Jones's presentation at the 2021 SUMO User's Conference, SUMO could not capture the dynamic behavior of a standard traffic signal in North America [7, 4] . Users that wanted to model a network with 'dual-ring' traffic signals had to either sacrifice speed by building a custom SIL simulation or sacrifice accuracy by approximating the dual-ring controller in limited capacity through the existing SUMO traffic signal controller types. The remaining alternative for prospective SUMO users was to forgo it altogether in favor PTV Vissim, which has a built-in dual-ring controller module, as well as an add-on package for software-in-loop simulation with an Econolite traffic signal controller [2]. It is clear that a native implementation of the control logic within SUMO's core code base would be desirable for many current and potential future users.

Enabled by the open-source model of SUMO, the integration of the dual-ring controller into SUMO's main branch allowed for the extension of its capabilities. In both literature and practice, there are several terms used synonymously for the ring-and-barrier signal controller. Two such terms that will be used throughout this paper are "dual-ring controller" or "NEMA-type controller", which is a reference to the National Electrical Manufacturing Association (NEMA) Standards to which the controllers adhere. When testing the controller against an

Econolite software-in-the-loop (SIL) controller, it became clear that the code from [7] would have to be extended to capture the broad range of behavior possible for NEMA-based traffic signal controllers. This paper aims to describe the process of extending the ring-and-barrier controller presented in [7] to model additional operation modes. First a brief explanation of the dual-ring controller is presented. This is followed by a description of the test setup / environment which was aided by a SIL traffic signal controller. Finally, the results of validation are presented along with a summary of current features of the NEMA type controller within SUMO code base along with known features that may be added in the future.

2 Background

2.1 NEMA Dual-Ring Controller

Traffic signals in the United States adhere to the NEMA Standards, which enforce a concept of rings and barriers on the traffic signal switching logic. For succinctness, only a standard, four-way intersection is discussed below; however, the same logic can be applied to any intersection configuration.

Under NEMA standards, a phase is used to represent a certain movement at the intersection. A phase is named by a number which is usually between 1 and 8. Conventionally, the even numbers represent the through movements and the odd numbers represent the left-turn movements. The right-turn movements usually share the same phase numbers as the associated through movements. Figure 1 shows standard phase numbering for a four-way intersection. As foundation of the control logic, there are typically two barriers, which represent the separation between serving 'side' or 'main/major' streets [6]. The main side of the barrier is denoted as the side that serves the most traffic volume.

Using the intersection in Figure 1 as reference, the dual-ring phase diagram can be drawn as Figure 2. The top row in the figure comprises one ring, phases {1, 2, 3, 4}), and the bottom the other, phases {5, 6, 7, 8}. The horizontal axis in the dual-ring diagram represents cycle-time and the barriers are denoted by the double vertical grey lines. They must not be crossed unless both rings move across the barrier at the same time. On either side of a barrier, the top ring may be served with any combination of the bottom ring. For example, phases {[1, 5], [1, 6], [2, 5], [2, 6]} are all potential combinations on the mainline side of the barrier. In the same manner, phases {[3, 7], [3, 8], [4, 7], [4, 8]} are all valid for the side street barrier. Intuitively, it is clear that control should not serve both main and side streets simultaneously for safety reasons

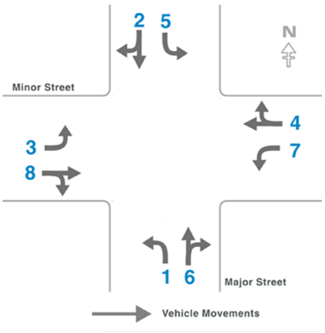


Figure 1: Typical phase numbering for a four-way intersection. Adapted from [5].

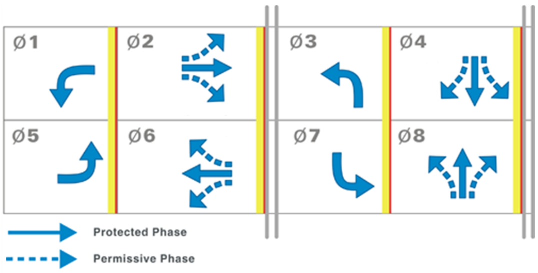


Figure 2: Typical ring-and-barrier diagram for the intersection in Figure 1. Adapted from [5].

and thus the barrier crossings must be synchronized between the two rings. In determining transitions, vehicle detectors are used in combination with a minimum and maximum time that a given phase should be served, though maximum time can be extended in certain situations.

All the phase combinations presented are valid, but the NEMA controller does constrain the transitions between states, and it depends on various manufacturer-specific settings as well as operation mode. Two of the most common operation modes are 'coordinated' and 'free' operation.

The goal of coordination is to synchronize multiple intersections, which will ideally minimize vehicle stops on the mainline roads. Coordination happens by enforcing a cycle length on the NEMA controller. The coordinated phases must be served at a regular interval equal to the cycle length, though the specifics of when and how to return to this coordinated state can vary. Figure 3 displays a ring-and-barrier diagram for the intersection in Figure 1, with added coordination annotations. It is important to note that Figure 3 is drawn with all phases at their maximum duration. In coordinated mode, each phase has a maximum and a minimum duration. Whether it lasts for the maximum, minimum or somewhere in between depends on the vehicle extension timer, which will be explained below. The cycle length is equal to the sum of each phase's maximum duration plus its transition time (yellow and red time) per ring, however it is often the case that the side street phases are not served for their maximum time. If this happens, the additional cycle time is returned to the coordinated phases and they are actually served longer than their 'maximum' duration. In this example, phases 2 & 6 are the coordinated phases (i.e. main road through movements).

The differences between three common NEMA controller conventions are displayed in the bubble callouts [5]. The ring-and-barrier diagram in Figure 3 has a leading left turn on the mainline street, meaning that phase 1 is served in conjunction with phase 6 (one of the coordinated phases) before phase 2 turns green. This is a more complex example than Figure 2, but it is helpful in illustrating the different offset types. For example, TS1 style-offsets designate the offset reference point (0 cycle time) as the time when **both** coordinated phases must be green, so the offset reference point in Figure 3 is not until phase 2 turns green as well. A TS2-syle offset designates the start of the coordinated cycle as the point when the **first** phase should be green. In the case of Figure 3 below, the first coordinated phase is 6. A Type-170 style offset sets 0 cycle time as the beginning of yellow on the earliest coordinated phase to end.

Having the offset reference point at the beginning of yellow makes the coordination easy to identify in the field. In the case of TS1 and TS2, the offset is referenced to the start of green, but only when all phases have been served their maximum allotted time. In the case when all phases haven't been served their maximum duration, the controller will return to green on the coordinated phases before the offset point. In TS2-style controllers, the coordinated phases can

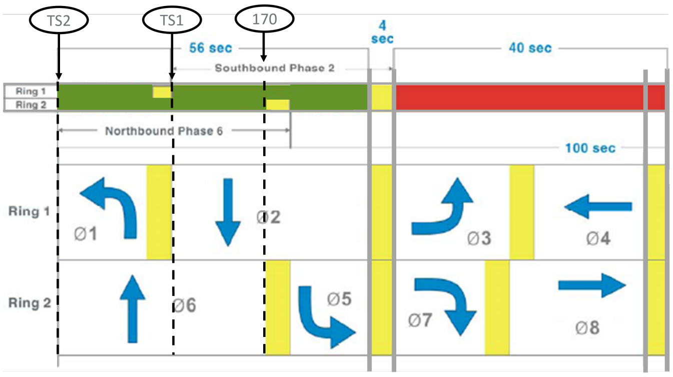


Figure 3: Dual-ring diagram for intersection in Figure 1 displaying the 3 different types of coordination with a 100 second cycle length. Adapted from [5].

also 'rest in green' if there are no vehicles detected on the side streets, which further obfuscates the coordination. While this discussion is not comprehensive, it provides some context to the motivation for including options for each style of offset.

As alluded to above, the non-coordinated phases may vary in duration and occurrence depending on controller settings. If phase skipping is possible, the controller can go directly from [2, 5] to [4, 8] if there is not a vehicle detected on either the 3 or the 7 phases' actuating detector. If phase skipping is disabled, the controller must progress from [2, 5], to [3, 7] for at least the minimum green time and then finally [4, 8]. The duration of non-coordinated phases can vary between the minimum and maximum green time, depending each phase's vehicle extension timers. Vehicle extension timers are also referred to as passage gap or passage timers. They serve to extend a phase past its minimum green time. When a phase is active and a vehicle crosses it's actuating detector, the duration of the phase is extended by the extension timer amount, as long as the addition of the extension timer to current phase duration will be greater than the minimum phase green time and less than the maximum time.

When the NEMA traffic signal controller is used at a stand-alone intersection or where traffic is sparse, traffic engineers will often use the controller in 'fully-actuated' or 'free' operation. It varies only slightly from coordinated operation, with the main difference being there is no cycle length. There is also more variation allowed in phase transition, as well as the phases which are typically coordinated ([2, 6] in case of Figure 1) being actuated. When there is infrequent traffic on the side-streets, a traffic signal in free operation will "rest-in-green" on designated phases (typically the mainline straight). In free operation and assuming that [2, 6] has been served for at least its minimum time, a transition from [2, 6] to [2, 5] or [1,5] is always valid, which is not the case in coordinated operation. During coordinated operation, a transition from [2, 6] to [1, 5] will have to wait until the possibility of serving [3, 7] or [4, 8] is exhausted. Put another way, [1, 5] cannot be served in coordinated mode unless the latest possible start time of the prior phases in the sequence has past.

The target of the initial development by Wang, Li and Jones was a coordinated, Type-170 Dual-Ring traffic signal [7]. As this controller was applied to other simulation networks, it became apparent that certain dual-ring settings and operation modes were missing. The term 'ring and barrier' traffic light describes only the core of each traffic light controller, and does not necessarily capture the additional functionality that each controller manufacture bundles with the core logic.

3 Methods

3.1 SUMO Integration

At a high level, the NEMA logic was incorporated into SUMO as a subclass of the SUMO `MSSimpleTrafficLightLogic` class, and is called `NEMAController`. The code is located at `src/microsim/traffic_lights/NEMAController.cpp` relative to the SUMO repository. The `NEMAController` logic fundamentally operates as a state machine, with the numbered phases being the state space. There are sets of transition conditions, depending on the mode of operation. Further details on the code layout are omitted for brevity.

A SUMO user indicates that a traffic light should utilize the NEMA logic by providing `type="NEMA"` in the traffic light configuration file. Indicating the traffic light is of type NEMA gives the user access to the NEMA controller settings. A typical configuration is displayed in the code block below. Information about detectors, cycle length (if coordinated), ring mapping, barrier phases, specific minimum/maximum and transition timing for each phase, and more. More details of features of the NEMA controller implementation in SUMO are provided in Section 5. There is also further explanation of the configuration parameters on the NEMA page of SUMO's website.

```
<tlLogic id="2881" offset="0" programID="NEMA" type="NEMA" offset="10">
    <param key="detector-length" value="20"/>
    <param key="detector-length-leftTurnLane" value="10"/>
    <param key="total-cycle-length" value="130"/>
    <param key="ring1" value="3,4,1,2"/>
    <param key="ring2" value="7,8,5,6"/>
    <param key="barrierPhases" value="4,8"/>
    <param key="coordinate-mode" value="true"/>
    <param key="barrier2Phases" value="2,6"/>
    <param key="minRecall" value="2,6"/>
    <param key="maxRecall" value=""/>
    <param key="whetherOutputState" value="true"/>
    <param key="fixForceOff" value="false"/>
    <phase duration="99" minDur="5"  maxDur="25" vehext="2" yellow="3" red="2" name="3" state="rrrrrrrrGrrr"/>
    <phase duration="99" minDur="5"  maxDur="25" vehext="2" yellow="3" red="2" name="7" state="rrGrrrrrrrrr"/>
    <phase duration="99" minDur="5" maxDur="30" vehext="2" yellow="3" red="2" name="4" state="GGrrrrrrrrrr"/>
    <phase duration="99" minDur="5" maxDur="30" vehext="2" yellow="3" red="2" name="8" state="rrrrrrGGrrrr"/>
    <phase duration="99" minDur="5"  maxDur="20" vehext="2" yellow="3" red="2" name="1" state="rrrrrGrrrrrr"/>
    <phase duration="99" minDur="5"  maxDur="20" vehext="2" yellow="3" red="2" name="5" state="rrrrrrrrrrrG"/>
    <phase duration="99" minDur="5"  maxDur="35" vehext="2" yellow="3" red="2" name="2" state="rrrrrrrrrGGr"/>
    <phase duration="99" minDur="5"  maxDur="35" vehext="2" yellow="3" red="2" name="6" state="rrrGGrrrrrrr"/>
</tlLogic>
```

3.2 SIL Setup

Both the development and validation of the SUMO NEMA dual-ring controller were aided by Econolite's EOS virtual controller. The virtual controller emulates a Econolite Cobalt or ATC controller running the Econolite EOS signal control software. Using the virtual controller running on local PC, configurations used by real intersections could be loaded into the virtual controller and importantly - confidently used as a ground truth.

To compare the behavior of the SUMO controller vs. Econolite, a software-in-the-loop (SIL) simulation framework was developed that coupled the Econolite EOS to SUMO. Similar to [1], the SIL framework is a Python program that maps detector calls in SUMO to the Econolite EOS and the traffic light state in the Econolite EOS to SUMO. Figure 4 depicts the SIL framework in more detail.

Communication between the Econolite EOS and the python script uses RFC 6455, also known as a websocket. The Econolite EOS broadcasts its traffic light state at a regular interval

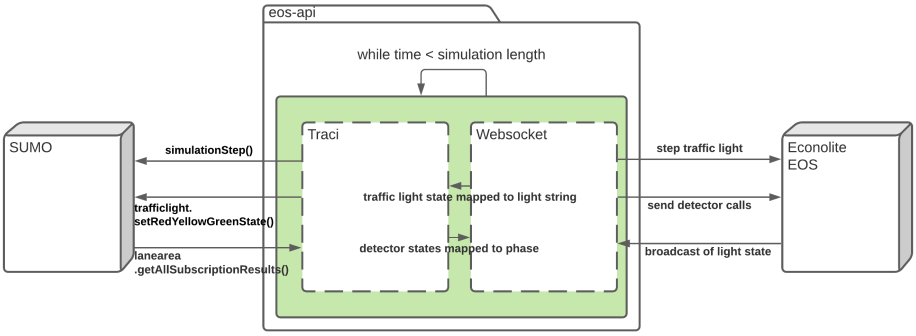

eos-api
while time < simulation length
SUMO
Traci
Websocket
Econolite EOS
simulationStep()
trafficlight.setRedYellowGreenState()
lanearea.getAllSubscriptionResults()
traffic light state mapped to light string
detector states mapped to phase
step traffic light
send detector calls
broadcast of light state

Figure 4: Schematic of the SIL framework.

and the message queue must be consumed quickly to ensure that the most up-to-date information is used. The Econolite EOS has both a pause and a 'step' feature, which allows the middleman python script to keep SUMO and the Econolite in sync.

The framework is scaleable to multi-intersection networks by running multiple instances of the Econolite EOS software. Care must be taken as the Econolite EOS defaults to using the host computers date-time and must be configured to match the real-world time that the simulation begins. The Econolite EOS's settings can be configured via the Python script, which sends websocket messages that match user inputs on the Econolite EOS GUI. Both date and time are important, as the EOS can have different day plans (similar to the 'Wochenschaltautomatik' is SUMO's traffic light logic). Further information about the Econolite implementation specifics can be shared upon request.

Developing the SIL framework and using it as a ground-truth against the SUMO NEMA controller was essential to enable building the additional features referenced in this paper. Significant time was spent investigating the various Econolite EOS settings and their corresponding responses to simulation traffic. Without coupling the Econolite EOS to the simulation, it would have been difficult to capture the true behavior of many of the features implemented and perhaps several other features such as cross-phase switching or locking detectors would have gone unnoticed. Again, an overview of currently features and remaining features to be implemented are provided in Section 5.

3.3 Test Description

Validation tests for the SUMO implementation of the NEMA controller were split into two different categories: realistic, simulation based tests and automated fuzz testing. The simulation-based tests show that the state machine transitions adhered to NEMA switching logic, or more specifically the Econolite EOS switching logic. Fuzz testing was used to send a barrage of random detector call combinations at the NEMA controller, with the intention of breaking the logic if bugs were present.

3.4 Simulation-Based Tests

For the simulation-based tests, a calibrated SUMO network representing a three intersection corridor of Tuscaloosa, Alabama was utilized. This was advantageous as the authors had access to the physical controllers in the network, and so were able to download and copy the configurations to the virtual controllers introduced in Section 3.2. The simulation encapsulated traffic from 7AM to 9AM on a representative work day where real-world detector logs allow for generating simulated traffic volumes at appropriate volume per hour at all network edges.

Figure 5 shows the SUMO model of the target network overlaid on geo-located satellite images. Each of the three intersections in the network have a different layout. The left-most is a three-way intersection, whereas the other two are four-way intersections. The two four-way intersections were specifically selected to test various scenarios, as each receives different volumes on its non-coordinated (side-street) phases from shopping centers and residential areas.

Ultimately, the goal of the simulation-based tests was to match the NEMA controller's phase and duration to that of the SIL controller exactly. The initial efforts sparked the inclusion of many new features to the Dual-Ring controller in SUMO and the testing was highly iterative. As the controller in SUMO matured, comparison on a multi-intersection network scale became possible. The results are presented in Section 4.

In the early stages the development and validation cycle, a one intersection cutout of the three intersection network was used to drive development. Figure 5 shows this one-intersection cutout as the intersection inside of the white-dashed box. The traffic signals impact traffic flow and thus comparing the SIL behavior to the SUMO-native traffic lights in a multi-intersection network is difficult unless the traffic signals operate in a very similar manner. The development of the NEMA controller was accompanied by SUMO test cases which are available in the SUMO repository, with the relative path being `/tests/sumo/basic/tls/NEMA`. Each feature added to the NEMA controller in SUMO has a corresponding test case that subsequent changes can be compared against, preventing regression. SUMO's documentation provides information on how to run each test.

3.5 Fuzz Testing

The NEMA traffic light agent interacts with the larger SUMO simulation in two fundamental ways: detector states and simulation time. Because the simulation time is intrinsically tied to the progress of the simulation, the behavior of the traffic light at any particular time is easy to analyze. On the other hand, when the NEMA traffic light has some level of actuation, different combinations and durations of detector calls are what trigger state transitions. Adding addi-

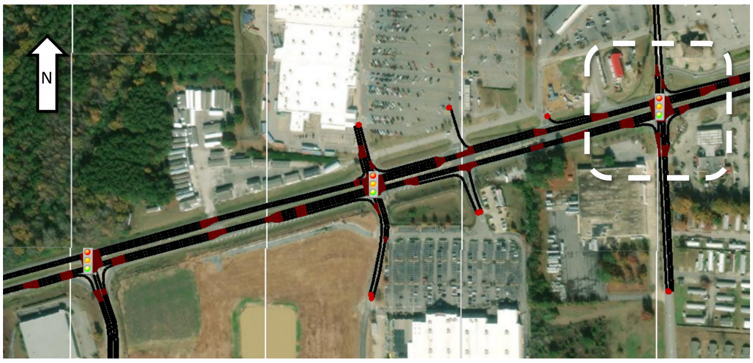


Figure 5: SUMO model of the simulated network including three intersections. Initial development was completed with the outlined sub-network.

tional features to the NEMA controller during the iterative design process scaled the complexity of the state transition logic. Though the simulation-based NEMA tests described in Section 3.4 give an idea of how the controllers perform under common traffic situations, they cover only a subset of potential traffic situations. To combat the lack of test coverage, a modified method of functional grey-box testing was employed.

There are numerous methods of grey-box testing utilized during software development. One such method is all-pairs testing, but considering the combinatorial detector state space quickly makes a comprehensive detector sweep unmanageable. In the same way, analyzing only 'critical' detector situations was considered, but designing such a test would likely been subject to the same logical issues that the code could contain. As such, an approach similar to a comprehensive detector sweep was employed, except the detectors on/off times and combinations were chosen at random. In software testing, this approach is sometimes referred to as targeted fuzz-testing, where automated tests generate random inputs to find software bugs and vulnerabilities [3, 8].

The automated detector tests were implemented using TraCI and a newly built API to override detector calls in SUMO. Prior to running the simulation, a set of randomly generated detector 'on' (corresponding to 1 vehicle on the detector) and 'off' (0 vehicles on the detector) times for each detector in the network were generated by iteratively sampling from a uniform distribution. Equations 1 and 2 below show how a series of detector calls was generated. Starting with the detector off (Off[0] = 0), then generating first on time (On[0]), then the second off time (Off[1]), and so on.

$$\mathrm{On}[i] = \sum_{k=0}^{i} U(0,N] + \mathrm{Off}[k], \text{ for } i = 0, 1, \ldots \mathrm{Off}[i] \geq T \tag{1}$$

$$\mathrm{Off}[i] = \sum_{k=1}^{i} U(0,N] + \mathrm{On}[k-1], \text{ for } i = 1, 2, \ldots \mathrm{Off}[i] \geq T \tag{2}$$

$U(0,N]$ represents a sample of a uniform distribution between 0 and N (the cycle length). Off and On represent vectors of simulation times where the detector should turn off and on respectively. The summation continues until the calculated detector off time is greater than the specified simulation time.

Assertions were added to the NEMALogic code to forcefully highlight bugs in the logic. In fully-actuated tests, there were two basic assertions:

- Active phases must be on the same side of the barrier, i.e. in Figure 2, phases 2 & 7 should never be served together.
- Each phase must last at least as long as it's minimum time.

In coordinated mode, an additional assertion was added which ensured that:

- The coordinated phases must be green at the start of their coordinated period.

This pass-fail logic was then applied to the tests in the SUMO repository, which include various intersection layouts as well as combinations of configuration s ettings. T he fuzz testing was also applied to a single intersection cut-out of the network presented in Section 3.4. It should be noted that proper testing would also sweep all combinations of user configurations. While the authors have such tests planned, the results are not included in this paper.

4 Validation

This section includes validation results of the SUMO NEMA Controller. Results are presented broken down into the two primary operation modes (coordinated and free) and for both simulation-based and the grey-box fuzz testing.

4.1 Coordinated Operation

The NEMA controller was developed and extended with all operation modes in mind, but the main target was coordinated mode. Coordination is favored by traffic engineers where there are several intersections that are close together and is the operation mode utilized by the target simulation network in the field. Coordination is enabled in the SUMO NEMA controller by passing `<param key="coordinate-mode" value="true"/>` in the traffic light configuration file.

4.1.1 Simulation-Based Testing

In the real network presented in Section 3.4, the three intersections operate almost exclusively in coordinated mode, with a 20 second offset between each controller. Figure 6 presents the result of controller development: identical response to traffic as a time history of the active green/yellow phases and coresponding detector calls for both controllers. The SUMO controller's behavior is shown above the phase on the y-axis, and the Econolite SIL controller is below. The dark vertical lines show the configured controller's cycle reference-point, which is a TS2 style offset. The offset type can be set via `<param key="cabinetType" value="TS2"/>` parameter in the traffic light configuration file.

In Figure 6, there are several side street phase progressions that occur. At 3250 seconds into the simulation, the controller progresses from [2, 6] to [3, 8] and back to [2, 6], which indicates light traffic on the side street. At 3700 seconds, the controller goes from [2, 6] to [3, 7], then to [3, 8], then [2, 5] and finally [2, 6]. This progression shows the "green transfer" functionality, as phase 3 stays green going from [3, 7] to [3, 8]. There are no detector calls on either phases 3 or 4 during this transition, so the controller behavior is to leave the existing phase (3) green. Phase 1 is never served explicitly in this simulation period, but by analyzing the detector calls in Figure 6 it is clear that there were no detector calls on phase 1 during transition periods.

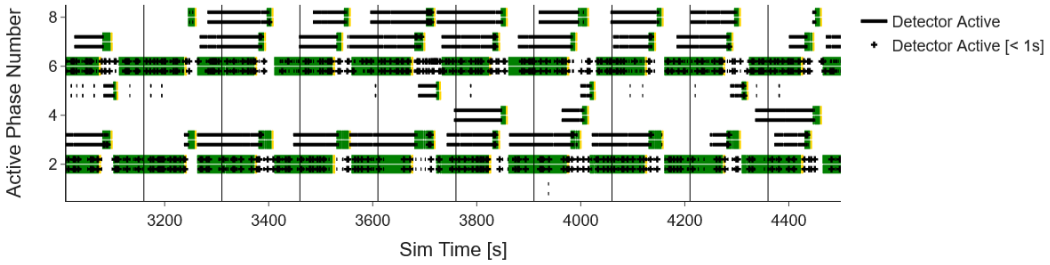


Figure 6: Visual comparison of SUMO NEMA Logic vs. Econolite in the presence of the same traffic demand. SUMO behavior is displayed slightly above the phase number and Econolite EOS below. Detector calls are shown as black crosses when their duration is less than one second and as a horizontal black line when longer than 1 second.

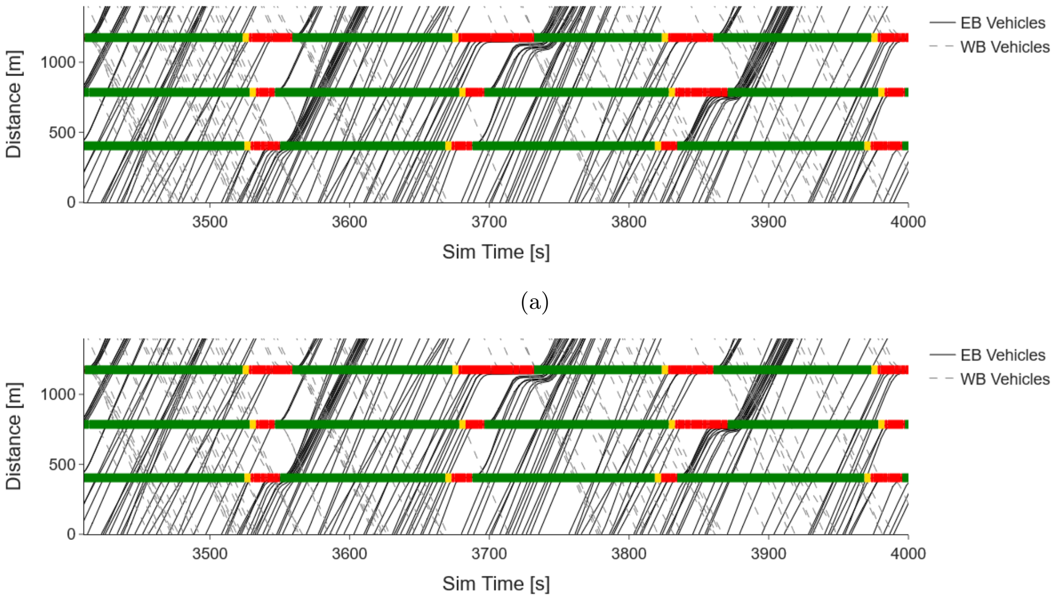
While it was unreasonable to present a plot comparing all phases of all intersections, the test will be added to SUMO as a test case, meaning that the results can be reproduced and analyzed by all SUMO users.

In addition to looking at all 8 phases of a single intersection, it was important to verify that all three NEMA controllers in the SUMO simulation worked together in coordinated mode. The effects of coordination are frequently viewed through the use of space-time diagrams, which show the effect of multiple traffic signals on traffic flow. Figure 7 presents the space-time diagrams of two simulations: one with SUMO NEMA controllers and one with SIL traffic signal controllers. In both Figure 7b & Figure 7a the eastbound (EB) vehicles are shown as the solid black lines and westbound (WB) as dashed. Only phase 6 of each of the three intersections is plotted, with the color of the horizontal line corresponding to the three intersections' light state. The light states are plotted at the distance each intersection is from the EB network edge.

The benefits of coordination on traffic flow are clear, with traffic progressing with constant velocity through the network during periods of all green. Comparing the two sub-figures reveals little to no difference, which gives the authors confidence that TS2-style offsets and coordination is working as expected in the SUMO NEMA controller. In fact, the two simulations are indistinguishable in the period analyzed.

4.1.2 Fuzz Testing

Fuzz testing the controller in coordinated mode surfaced several bugs in the logic that have been addressed. As an example, the algorithm which computes whether a phase will 'fit' inside of the cycle time was incomplete. This becomes important with phase-skipping functionality.



(a)

(b)

Figure 7: Space-time diagram of SUMO traffic lights (a) and the Econolite traffic lights in SIL (b) described in Section 3.4. The space-time diagrams are virtually identical.

In addition to the phase fit algorithm, certain combinations of detector calls during a yellow to red transition were found to cause the controller to 'reverse' away from a barrier, meaning that in Figure 2, phase 8 was transitioning to 7, which should not happen in coordinated mode.

4.2 Free Operation

In free mode, the controller has more freedom than in coordinated mode, thus the SIL controller and SUMO NEMA controller diverge more frequently, especially when the divergence of one is propagated through the three intersections of the network under study.

4.2.1 Simulation-Based Testing

An example of differences between the SIL controller and SUMO during simulation testing of free operation can be seen in Figure 8. As with Figure 6, the SUMO NEMA controller phases have been plotted above the phase number and Econolite SIL below. In the first half of plotted simulation time (3500 - 3800s), some of their behavior looks quite different such as during the period encircled with the red-dashed overlay. Inspection of the detector calls inside of the overlay reveal that vehicles cross the phase 6 detector in the SIL simulation around 3740 seconds into simulation, which extends the [2, 6] phases. Those vehicles do not cross the detector in the SUMO-native simulation, which is likely due to a difference in the upstream behavior of a different traffic signal.

Because of a limitation in the SIL controller implementation, there is a one simulation step delay on detector calls. This lag between SUMO detectors and what the SIL controller sees leads to differences in the vehicle extension timer and then ultimately the phase length. Knowing the limitations of the SIL setups and the degrees of freedom that a free dual-ring controller has, the authors are confident that SUMO is capturing the behavior of the SIL controller correctly.

4.2.2 Fuzz Testing

As in Section 4.1.2, fuzz testing the controller in free mode also surfaced bugs. For example, combinations of detector calls that occurred during a transition from [2, 5] to [1, 6] could ultimately lead to the barrier being crossed by one ring and not the other. The bug was since fixed by enforcing stricter logic on barrier cross transitions.

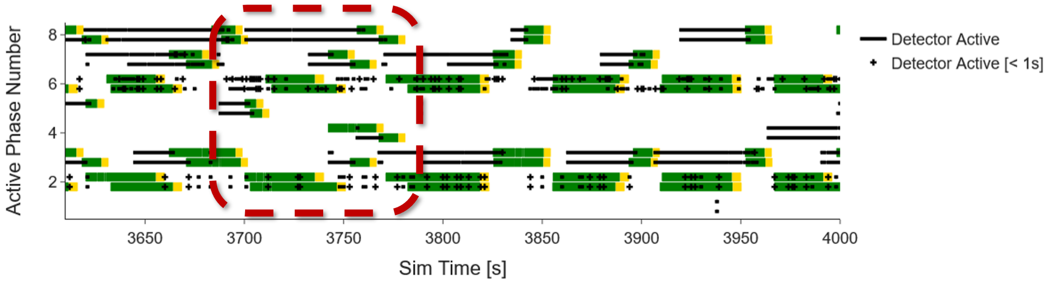


Figure 8: Visual comparison of SUMO NEMA Logic vs. Econolite for a select period of the simulation. The phase are not identical, but the behavior in response to detector calls is.

4.3 Simulation Speed

Preserving SUMO's standalone simulation speed was one of the main goals of integrating the NEMA controller into SUMO. As discussed, there were SIL alternatives, but SIL simulations are slower and more resource intensive. Table 1 presents a comparison of simulation real-time factor for the two options, which is the equivalent to $\frac{\text{simulated time}}{\text{computation time}}$. The two columns (1 Intersection and 3 Intersections) represent the simulations discussed in Section 3.4. Each simulation was ran with a 0.1 second step length for and lasted 6900 seconds. The route file and random seed was the same for each 1 Intersection and 3 Intersection simulation.

Table 1: Comparison of simulation real time factor for the SIL and SUMO-native NEMA methods.

	Network Size	
Controller Method	**1 Intersection**	**3 Intersections**
SIL With EOS	31.5	21.6
SUMO NEMA	205.2	180.0

While it's not a comprehensive simulation speed test, the brief comparison of the Econolite SIL simulation presented in Section 3.2 against the built-in NEMA controller makes the speed penalty of the SIL implementation clear. With three intersections, the standalone SUMO simulation has a real-time factor of 180.0, which is roughly 8.5x faster than the same SIL simulation. The one intersection simulation is 6.5x faster in standalone mode. The ratio between SUMO-standalone real-time factor and the SIL real-time factor will continue to increase as intersections are added to the simulation network.

5 Conclusions and Future Work

As this paper has shown, the authors have attempted generality in their implementation of the NEMA controller. Care was taken to test against multiple configurations and intersection layouts. At the same time, the only virtual traffic signal controller available to the authors was the Econolite EOS and thus there is potential that the SUMO integration is 'overfit' to the Econolite EOS traffic signal controller software.

Table 2 presents some of the features implemented, as well as features that may be useful for other users but have not been implemented yet.

The authors are hopeful that the SUMO community will see the newly-integrated controller as a big step forward for North American users and will be willing to contribute to the code-base or reach out to the authors when they see a missing feature. In addition to the features not included in Table 2, one of largest outstanding tasks at the time of writing is to incorporate the NEMA controller configuration into netedit.

6 Acknowledgments

Incorporating the NEMA controller would have been impossible without the support of the entire SUMO team. Specifically, the authors would like to thank Jakob Erdmann for his guidance integrating the NEMA Controller into SUMO's source code as well as being a sounding board for new features and parameters.

This work was supported by The University of Alabama's Center for Advanced Vehicle Technology (CAVT). This material is also based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the the Vehicle Technologies Office (VTO) and through Energy Efficient Mobility Systems (EEMS) program award number DE-EE0009210. The views expressed herein do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

Table 2: Coverage of NEMA-type controller settings

Feature	**Included**	**Notes**
Green Rest	✓	Econolite Implementation
Green Transfer	✓	Econolite Implementation
Fully-actuated	✓	
Latching Detectors	✓	Basic Implementation
Cross-phase Switching	✓	Econolite-style Implementation
Detector Delay		
Detector Lock-In Time		
Phase Recall	✓	Min/Max Recall. Detector Recall Missing
Fix/Float Force Off	✓	Bool on/off, not per phase
Dual Entry		
Red Revert		
Type-170 Offset	✓	
TS1 Offset		
TS2 Offset	✓	

References

[1] Mirko Barthauer and Bernhard Friedrich. External signal control: Integrating LISA+ into SUMO. In *SUMO User Conference*, pages 143–153, 2017.

[2] Martin Fellendorf. VISSIM: A microscopic simulation tool to evaluate actuated signal control including bus priority. In *64th Institute of Transportation Engineers Annual Meeting*, volume 32, pages 1–9. Springer, 1994.

[3] Patrice Godefroid. Fuzzing: Hack, art, and science. *Communications of the ACM*, 63(2):70–76, 2020.

[4] 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 *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[5] Tom Urbanik, Alison Tanaka, Bailey Lozner, Eric Lindstrom, Kevin Lee, Shaun Quayle, Scott Beaird, Shing Tsoi, Paul Ryus, Doug Gettman, Srinivasa Sunkari, Kevin Balke, and Darcy Bullock. *Signal Timing Manual - Second Edition*. The National Academies Press, Washington, DC, 2015.

[6] Qichao Wang and Montasir Abbas. Optimal Urban Traffic Model Predictive Control for NEMA Standards. *Transportation Research Record*, 2673(7):413–424, 2019.

[7] Qichao Wang, Tianxin Li, and Wesley Jones. Augmenting SUMO with Ring-and-Barrier Structured Traffic Signal Controller Module [Conference Presentation]. In *SUMO User Conference*, 2021.

[8] Andreas Zeller, Rahul Gopinath, Marcel Böhme, Gordon Fraser, and Christian Holler. The Fuzzing Book, 2019.

SUMO User Conference 2022
https://doi.org/10.52825/scp.v3i.116

Published: 29 Sept. 2022

A Comparison of Reinforcement Learning Agents Applied to Traffic Signal Optimisation [*]

Jacobus Louw[1], Louwrens Labuschange[1], and Tiffany Woodley[1][https://orcid.org/0000-0002-3651-6426]

[1]ByteFuse AI, Stellenbosch, South Africa

cobus.louw@bytefuse.ai, louwrens.labuschange@bytefuse.ai, tiffany.woodley@bytefuse.ai

Abstract

Traditional methods for traffic signal control at an urban intersection are not effective in controlling traffic flow for dynamic traffic demand which leads to negative environmental, psychological and financial impacts for all parties involved. Urban traffic management is a complex problem with multiple factors affecting the control of traffic flow. With recent advancements in *machine learning* (ML), especially *reinforcement learning* (RL), there is potential to solve this problem. The idea is to allow an agent to learn optimal behaviour to maximise specific metrics through trial and error. In this paper we apply two RL algorithms, one policy-based, the other value-based, to solve this problem in simulation. For the simulation, we use an open-source traffic simulator, *Simulation of Urban MObility* (SUMO), packaged as an OpenAI Gym environment [3, 9]. We trained the agents on different traffic patterns on a simulated intersection [24]. We compare the performance of the resultant policies to traditional approaches such as the Webster and *vehicle actuated* (VA) methods. We also examine and contrast the policies learned by the RL agents and evaluate how well they generalise to different traffic patterns.

Contents

[*] Other people who contributed to this document include Greg Newman, Etienne Barnard, Ruan van der Merwe and Tessa Smit.

Accepted at SUMO conference 2022. This paper contains no competing interests. All authors made equal contributions.

1 Introduction

Traffic light intersections are responsible for directing large volumes of traffic flow, making their optimal control critical. Negative environmental, psychological, and financial impacts are associated with traffic build up, resulting from inefficient control [14]. Currently, traffic lights predominantly use control methods which require resource intensive configuration[18, 15]. This configuration requires a large amount of engineering hours, making these systems costly to implement and maintain [5].

To improve financial viability, we propose a low-cost control system that can obtain similar or improved functionality in an automated manner. This system is implemented using an RL agent that can be trained in a simulated environment, thereby removing the need for costly configurations. Practically, implemented systems predominantly control a single intersection. Our approach is trained and tested on a single intersection, providing a low maintenance alternative to current solutions.

The promise of using RL for the application of traffic light control has been demonstrated in previous research predominantly utilising Deep Q-learning [24, 1, 8]. There has been emergent research into using *proximal policy optimisation* (PPO) due to its ease of implementation and encouraging results [12]. The main focus of this paper is to directly compare Deep Q-learning and PPO on the traffic light optimisation application. A comprehensive comparison is completed by evaluating the method's policy performance, not only in terms of average performance, but also the robustness of the chosen strategy.

For direct comparison to current implementations, our achieved results are compared to traditionally used methods on a simulated intersection. These traditional methods include a gap-based actuated setup, a time-loss actuated setup, and a fixed-time approach which makes use of the Webster equation [17, 2].

2 Related work

Traffic signal timing optimisation is a complex task which necessitates a high level of expertise in the industry. Many methods varying in complexity have been developed for this application. Despite much work on the development of these methods, even more sophisticated methods struggle to accurately predict traffic operation at intersections. This difficulty is due to a wide variety of influencing factors.

A commonality across methods is the evaluation of the model's performance. The performance can be quantified by using either a performance index or a measure of the level of service. These are generally calculated using metrics such as number of vehicle stops at the intersection, vehicle delay etc.

Traditionally used methods can be split into two types: actuated and fixed-time approaches. Both methods have limitations: actuated approaches perform best at isolated intersections and fixed-time approaches are unable to dynamically respond to unexpected increases in traffic. These limitations have opened up new research branches into adaptive methods and RL which, whilst still new, show much promise for this application.

2.1 Fixed-time control

There are several methods for controlling the timing of traffic signals which require manual configuration. Fixed-time methods fall into this category where cycle stages of the different traffic light phases are set ahead of time and once set their operation will not deviate. In order to set the cycle stages different evaluations can be used, the most common one is based on an equation developed by Webster [22]. His formula can be used to calculate the optimal cycle length that would minimise the total delay at an intersection

$$C_o = \frac{1.5 \cdot L + 5}{1 - \sum Y_i}, \tag{1}$$

where C_o is the optimum cycle length in seconds, L is the total lost time per cycle in seconds, and Y_i is the volume/saturation flow ratio per critical movement in stage i.

Webster further elaborates on the fact that the equation's proposed cycle lengths have some leeway. Cycle lengths chosen within the range of $0.75C_o$ to $1.50C_o$ should not significantly increase the delay. Although robust to small deviations in the cycle length, this equation is very sensitive to the accuracy of the metrics of lost time and saturation flow given to the equation. It further is unable to account for pedestrian traffic.

2.2 Vehicle-actuated control

Vehicle-actuated approaches allow for deviation in their response based on how many vehicles are detected as well as pedestrian buttons pushed [20]. This deviation in cycle times allows for a more tailored approach to current traffic patterns. When at an isolated intersection where traffic follows a more sporadic pattern, vehicle-actuated control can provide considerable reductions in delay when compared to fixed-time approaches [13]. This is due to the fact that fixed-time approaches cannot readily adapt to the randomness in vehicle arrivals.

To implement such an approach, first a fixed-time approach is established; the actuated control strategy can then extend the set cycle times if there are queues at the receiving green lanes. If these queues were cut off too soon by a red light, the overflow of vehicles would have to be discharged at the next green light, causing delays. The queues can be established by measuring the gap between vehicles, if there is a long gap between vehicles it is an indication that the queue has been discharged.

2.3 Reinforcement learning

Adaptive controllers, for example *split cycle offset optimisation technique* (SCOOT) and *Sydney coordinated adaptive traffic* (SCATS), have been shown to improve upon actuated methods, however the difficulty of the initial manual tuning process still remains resource consuming and costly [6, 10, 18]. Both the advancements in computational methods and amount of traffic data stored have made *reinforcement learning* (RL) a feasible solution to the problem. Since RL can be trained in a simulated environment, it removes the initial costly barrier.

RL has the ability to learn optimal control in dynamic and uncertain environments [19]. This ability makes it well suited to the traffic light optimisation problem which needs to dynamically react to changing traffic patterns. Across emergent research into this approach the traffic light simulation tool, SUMO, has been used to train and evaluate implemented RL systems [24].

A relatively simple approach is to apply tabular RL methods to this problem [21]. Although effective on smaller case studies such as single intersections, these methods do not scale well to problems with larger state spaces. When we apply RL to our application, the state space of more complex intersections becomes far too large to represent with a table [23]. The solution to addressing the large state space lies in interpolation and approximation – being able to use a small subset of states to generalise and make decisions over a much larger subset. This can be achieved by using function approximation to approximate the value function, allowing problems with high-dimensional state spaces to be solved. Various function approximation techniques can be used in combination with RL, but recently *artificial neural networks* (ANNs) have become the commonly method used to represent value functions and policies. The combination of ANNs and RL is known as *deep reinforcement learning* (DRL).

Deep Q-learning is a popular DRL algorithm that utilises an ANN in conjunction with Q-learning [11]. The resultant model is known as a *deep Q-network* (DQN). The benefits of using a DQN are highlighted by Wei et al [24]. These authors develop a DQN for the purpose of optimal traffic control, the DQN is improved by implementing state-of-the-art techniques such as *experience replay* (ER) and a target network. Their paper further investigates how robust the DQN is to different traffic patterns. They suggest that further improvement can be made to the performance of the DQN in terms of convergence and stability.

Another popular RL approach is using policy-based methods. Traditional policy-based methods tend to destabilise if not constrained. *Trust region policy optimisation* (TRPO) methods were developed with the aim of solving this problem by restricting the magnitude of the changes allowed to be made to the policy. PPO is an emergent method which has all the same benefits as TRPO methods, with the added advantages of easy implementation and better sample complexity. This method is also data efficient, as it is able to run multiple epochs on a single sample and has been shown to outperform other policy gradient methods on applications such as Atari [16]. Mousavi has demonstrated that PPO can achieve comparable results to a DQN in the context of traffic light optimisation, making it a promising area for further research [12]. This paper makes a direct comparison between the two methods in terms of robustness and strategy implementation to help formulate the direction of future research.

3 Experimental setup

The traffic light intersection is simulated using *Simulation of Urban MObility* (SUMO), an open source traffic simulation package [9]. The intersection geometry and traffic patterns used to simulate the simple intersection used for our experiments are described in Sections 3.1 and 3.2. Phase changes can be made in the simulation to directly control the generated traffic; these phase changes are detailed in Section 3.3.

3.1 Intersection layout

SUMO was used to simulate the intersection used in Wei et al's paper which can be visualised in Figure 1 [24]. This simulated intersection was used to both train and test the RL agents, as well as test the traditional methods used as benchmarks.

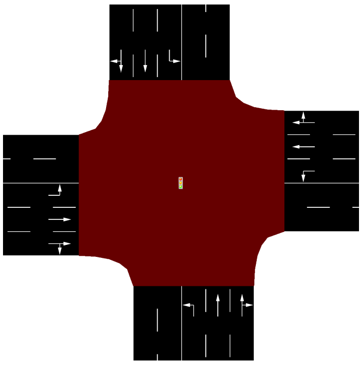


Figure 1: The intersection geometry used throughout this paper [24].

3.2 Traffic patterns

Four different synthetic traffic patterns are employed by Wei et al. to capture the spatial-temporal aspects of urban traffic [24]. The four traffic patterns were simulated in the SUMO environment, and include:

- A traffic distribution which simulates higher traffic flow patterns on the major roads over the minor roads (P1).
- A traffic pattern which simulates higher traffic flow on the left-turn lanes over the through lanes (P2).
- A tidal traffic pattern, where two perpendicular lanes have higher traffic flow (P3).
- Time based varying traffic patterns, where one of the major lanes has time varying traffic patterns, where the other lanes kept a steady traffic flow (P4).

These synthetic traffic patterns were generated using a binomial distribution with the arrival rates specified in Table 4. Since all these are feasible traffic patterns, the methods were tested on all four patterns. Testing on the different traffic patterns gives us a more complete idea of how the agent operates.

3.3 Traffic light phase definitions

Phase changes in the simulation can be made to directly control the generated traffic; these phase changes are detailed in Section 2. The links between lanes represent the possible directions a vehicle can take from a given lane. If the link is green the vehicle can move through the green link and leave the intersection. If the link is red the vehicle will have to wait for a phase change where the corresponding link is green. Purple links indicate a stop for right turning vehicles. These vehicles may turn right when it is safe to do so.

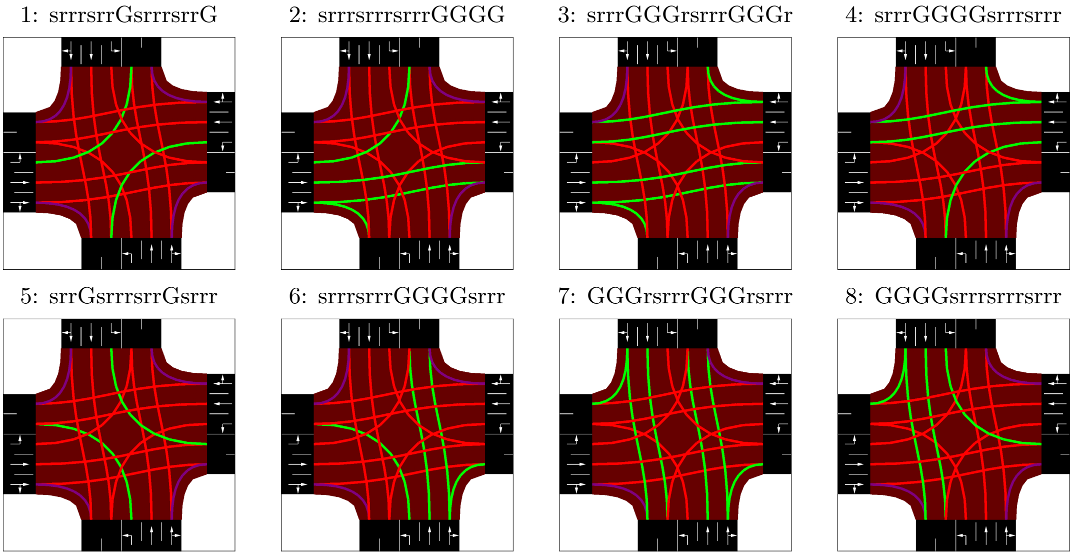


Figure 2: The phase definitions of the intersection [24].

A yellow time between consecutive green phases is implemented. This entails turning all green links yellow for a specified time before switching to the next phase. In order to prevent the agent from making rapid changes between phases a minimum green time is implemented where the agent is forced to wait a predefined duration before being able to make another phase change. The parameters used for the simulation are shown in Table 3.

4 Methodology

One goal of this study is to develop a policy for dynamically controlling the traffic light that optimises flow at the intersection described in Section 3 . To control the traffic light optimally requires efficient decision making regarding phase changes. This requires careful consideration of how the problem should be structured in order to apply RL techniques such as Deep Q-learning and PPO.

4.1 Reinforcement learning approach

In order to apply RL, the traffic light optimisation problem should be framed as a *Markov decision process* (MDP). The MDP can be referred to as the environment, where the state of the environment consists of the current traffic conditions and phase of the traffic light. Furthermore an agent can interact with the MDP by performing actions in the environment. In this case, the set of actions available is phase changes of the intersection (as detailed in Section 3.3). By taking actions which change the phase of the traffic light the agent is able to directly affect the traffic flow. After an action (phase change) has been implemented a reward is returned to the agent. The reward signal can be set up based on the exact objectives of the intersection and serves as indication of how well the agent is performing. The goal of the agent is to accumulate the largest possible return (sum of rewards) over time. Figure 3 illustrates the interaction between the agent and the environment. In this case the environment is a SUMO simulation wrapped in a Gym wrapper. The wrapper allows the agent to treat the environment like any other Gym RL environment, enabling quick changes to both agent and intersection.

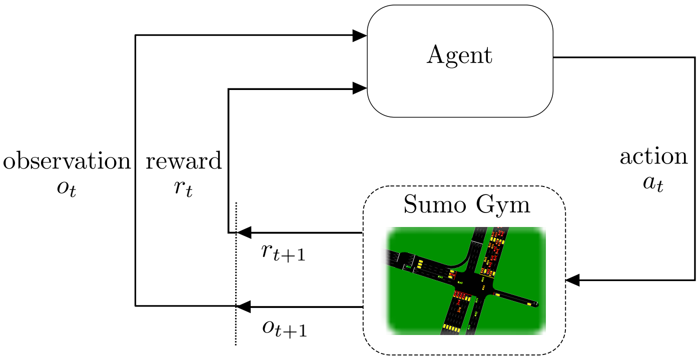


Figure 3: The interaction between the agent and Sumo Gym environment. At each time step the agent receives a reward r_t and observation o_t. The agent performs an action a_t in the environment based on the observation received. In return the environment returns a new observation o_{t+1} and reward r_{t+1}. The interaction between the agent and environment is recurrent.

With the problem framed as an MDP, different RL techniques can be used to solve the problem of finding an optimal decision path through the environment. Q-learning is a popular value-based RL method that learns the value function of an optimal policy. By recurrently alternating between estimating the Q-values of the current policy and improving the policy by acting greedily with respect to the estimated Q-values, the agent indirectly moves towards an optimal policy. The greedy policy can be followed by choosing the action with the highest Q-value in each state. Q-learning is classified as a value-based method since it requires a value function to be estimated.

In contrast to Q-learning, *policy gradient* (PG) methods aim to improve upon a policy directly. Instead of predicting the value of an action, the agent directly predicts the action(s) that will yield the most return (the sum of the rewards). Schulman et al. [16] state that improvements are made to the policy by updating the policy in the direction of a calculated gradient

$$\hat{g} = \hat{\mathbb{E}}_t[\nabla_\theta log \pi_\theta(a_t|s_t)\hat{A}_t]. \tag{2}$$

The advantage $\hat{A}_t$ is an indication of how much better the true value of being in a state s_t given the current policy π_θ is compared to an estimated value. If $\hat{A}_t$ is positive it means the value was underestimated and the probability of taking that action in the policy is increased.

Making large steps based on these calculated gradients can lead to the policy becoming unstable and diverging. To mitigate against this, PPO aims to keep deviations from the previous policy small by clipping them to ensure stability. Since the updated policy will not differ too significantly from the previous policy there will be lower variance when training.

4.2 Agent observations

The observations returned by the environment have a significant effect on the agent's performance. This is because the agent's actions are based on the observations it receives. For example if the environment is only partially observable, it cannot be classified as an MDP and presents new challenges to the agent.

The environment used in this study can be easily modified by using wrapper functions, by allowing for the information returned by the environment to be adapted to suit the application's specific requirements. In our experiments we made use of an observation wrapper which returned a vector of queue lengths for each lane [24]. The queue lengths are determined by the number of halting vehicles. SUMO defines a halting vehicle as a vehicle moving at a speed $\leq$ 0.1 m.s^{-1} [9]. After the wrapper has been applied the observations returned to the agent take the following form

$$\boldsymbol{o}_t = \left[q_t^1 q_t^2 \cdots q_t^{12}\right]^T, \tag{3}$$

where q_t^i is the queue length (or number of halting vehicles) in lane i at time step t. Furthermore, the cardinality of the observation space is $|\boldsymbol{o}_t| = 12$, since the intersection in Figure 1 has 12 incoming lanes.

4.3 Reward function

The reward function is a crucial component when solving an RL problem. It specifies the metric that the RL agent will strive to maximise in the long run. It is therefore critical to design the reward function to align with the goal the designer wants the agent to achieve. Sutton et al. advise against implicitly encoding domain knowledge into the reward function, as this may cause the agent to learn behaviour that results in a large reward without achieving the desired goal [19].

For this reason, we kept the reward function as simple as possible. Our reward function gave the agent a penalty (negative reward) for the aggregate of the waiting time of vehicles at the intersection [4]. The reward at a time step can be calculated using

$$R_t = -\sum_{i}^{n} w_t^i, \tag{4}$$

where n is the number of vehicles halting at the traffic light and w_t^i is the waiting time of vehicle i at time step t. The waiting time of a vehicle only starts when it begins to halt i.e. when its speed is below 0.1 m.s^{-1}. Since a vehicle's waiting time is correlated with delay, by maximising this reward we direct the agent to learn how to reduce delays.

4.4 DQN implementation

Since traffic light optimisation is complex, the state space of the environment becomes extremely large and calculating a value for every possible action-value pair becomes infeasible [24]. To account for this an ANN can be used to approximate the action-value function resulting in a DQN. When using an ANN in conjunction with the off policy Q-learning algorithm, some instabilities occur. RL works in a sequential manner, therefore observations are highly correlated. This conflicts with traditional supervised learning where data samples are *independent and identically distributed* (IID). In order to address this issue an *experience replay* (ER) buffer is utilised. As the agent steps through the environment it stores its experience in the replay buffer. The agent then samples random mini batches from the buffer to train the DQN. The second problem is that the targets used to update the DQN are not stationary. This is because the DQN self is used to compute its targets. As a result, as the DQN's weights change, so do the new target values. To address this, a frozen copy of the DQN is created, known as the target network. The target network is now rather used to compute the agent's targets. The parameters of the target network are held constant and only updated to the new parameters after a set period [11]. This further helps to stabilise training.

The ANN was set up as follows, the network was made up of two hidden layers with 64 neurons in the first layer and 32 in the second. The activation function used was *rectified linear unit* (ReLU) after each hidden layer. The networks were trained using an *adaptive moment estimation* (Adam) optimiser, an extension of *stochastic gradient descent* (SGD), which has been shown to work well for problems which are noisy [7]. The hyperparameters used can be found in Table 1.

Hyperparameter	**Value**
Replay memory size (M)	10000
Mini-batch Size (B)	128
Starting (ε)	0.9
Ending (ε)	0.05
Target network update interval (ΔT)	1800
Discount factor (γ)	0.999
Learning rate (α)	0.01

Table 1: Hyperparameters used for DQN

4.5 PPO implementation

To adapt PPO to the complex environment, two separate ANNs where created. The first ANN was used to capture the policy by outputting a probability distribution over the agents' actions. The second network was used to estimate the state action value function. The same architecture was used for both of the networks. The networks were setup with an input layer with the number of neurons equal to the number of lanes in the observation, two hidden layers containing 128 neurons and an output layer where each neuron represents a possible action. The hyperparameters used are outlined in Table 2.

Hyperparameter	**Value**
Clipping parameter (ϵ)	0.2
Mini-batch size (B)	32
GAE parameter (λ)	0.95
Num epochs	4
Horizon (T)	128
Learning rate actor (α_a)	2.5×10^{-4}
Learning rate critic (α_c)	2.5×10^{-5}

Table 2: Hyperparameters used for PPO

5 Experiments and results

In this section we evaluate the performance of the PPO and DQN agents across the four traffic patterns defined by Wei et al. [24]. Furthermore, we compare the performance of the RL approaches discussed above to three benchmarks: gap-based actuated, delay-based actuated, and Webster. The parameters for the gap-based and delay-based traffic lights are set to the default values found in SUMO, as shown in Table 7 and Table 8, respectively. The configuration for the Webster traffic light using the parameters in Table 5 results in the timings shown in Table 6.

The gap-based actuated traffic light is implemented by SUMO [9]. This method entails to prolong the green time of any phase whenever a continuous stream of traffic is detected. The gap-based implementation of SUMO also supports dynamic phase selection. This entails assigning priorities to different phases of the traffic light depending on various factors. The phase with the highest priority is then selected as the next phase of the traffic light.

The delay-based actuated traffic light prolongs the current green phase if there are vehicles with accumulated time loss. A vehicle's time loss begins as soon as it enters the detector range. If the accumulated time loss exceeds the minimum time loss value, the corresponding green phase is requested to be extended if it is active. The instantaneous time loss of a vehicle is defined as

$$1 - \frac{v}{v_{\max}}, \tag{5}$$

where v is its current velocity and $v_{\max}$ the allowed maximal velocity.

We trained each RL agent for 200 episodes on each of the four traffic distributions P1 through P4 (see Table 4). Each episode consists of 1800 simulation time steps, where each time step is equal to one real-time second. All experiments are seeded with the same seed to ensure that all agents are presented with the same traffic and initial policies. After training, we evaluate the eight trained agents (4 PPO, 4 DQN) on each of the four traffic distributions using a different seed than used during training. This results in 32 total agents being evaluated (16 PPO, 16 DQN).

For the benchmarks, there is no training involved and as such we only evaluate them once using each of the four traffic distribution resulting in 12 benchmarks - 4 actuated, 4 delayed and 4 Webster.

SUMO, [9], offers a plethora of output statics which we generated at the end of each testing episode in order to gain an understanding into the workings and efficiencies of the benchmarks versus the DQN and PPO RL agents.

5.1 Performance during training

We present three metrics: vehicle waiting time, mean queue length and mean speed to evaluate the performance of each agent over consecutive training episodes. These metrics are defined by SUMO [9], as:

- Waiting time: The time in seconds which the vehicle speed was below or equal 0.1 m.s^{-1} (scheduled stops do not count).
- Speed: The mean speed of all vehicles in the network in m.s^{-1}.
- Queue length: The mean queue length of all lanes in meters.

To ensure that our agents do indeed converge, we ran the training process 10 times, each time seeding the traffic and agents with a different seed. The results is the learning curves of in

Figure 4, Figure 5 and Figure 6, with the line showing the mean across the 10 runs, and the shaded area indicates a 95% confidence interval.

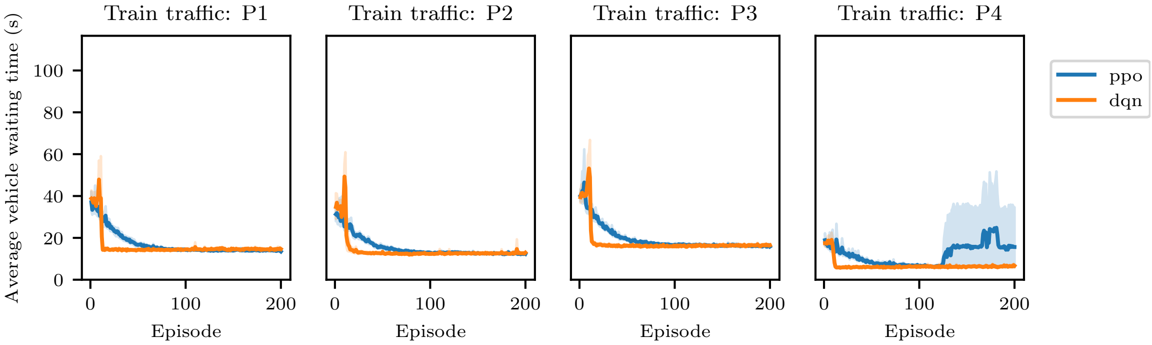


Figure 4: The average vehicle waiting time of DQN compared to PPO over training episodes. Since the goal was to minimise waiting time, this also serves as a proxy for the learning curve.

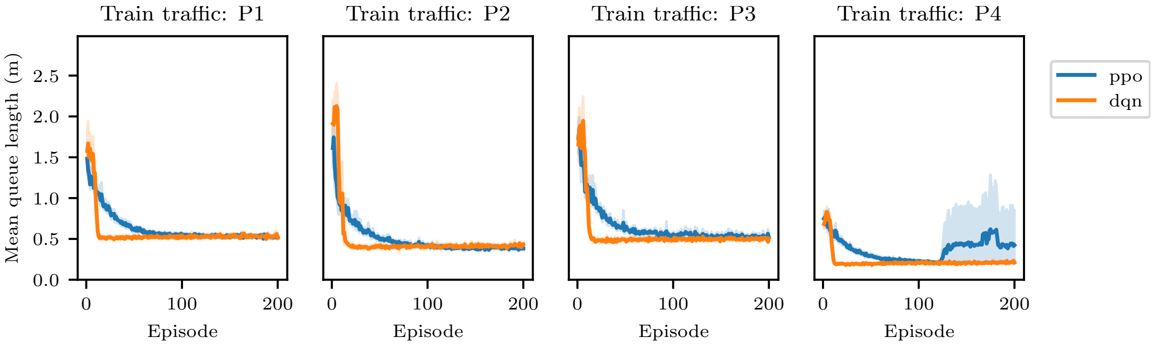


Figure 5: The mean queue length of DQN compared to PPO over training episodes.

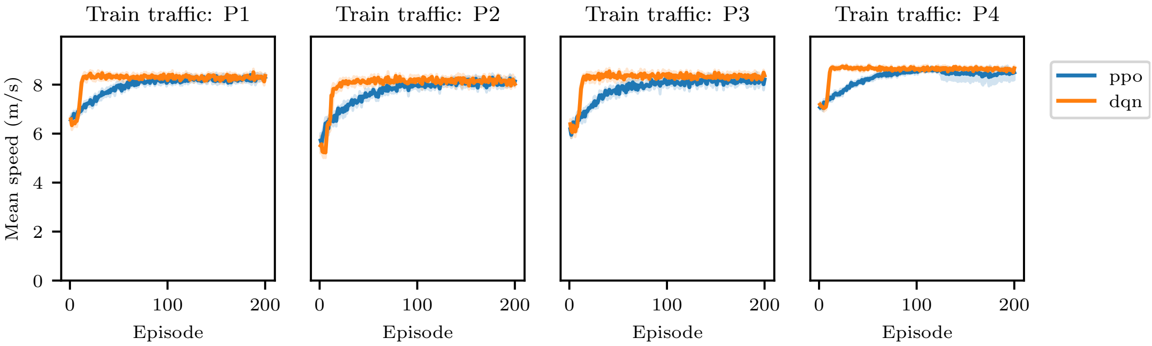


Figure 6: The mean vehicle speed of DQN compared to PPO over training episodes.

For all three metrics it is clear that the DQN agent convergences much faster to a stable solution compared to the PPO agent. Further, the PPO agent seems to have a few runs that become unstable after the 130 episodes mark on traffic pattern P4. Why this is we leave for future work and for the remainder of this paper we use both a DQN and PPO agent that remained stable during the entire training process.

In addition to policy performance, we measured the computational cost of training the various agents. We present similar figures to those shown above, but this time they are plotted over relative training time. All agents were trained on AWS using ml.m5.large instances. As illustrated in Figure 7, both the DQN and PPO agents learn a high-performing policy in a matter of minutes.

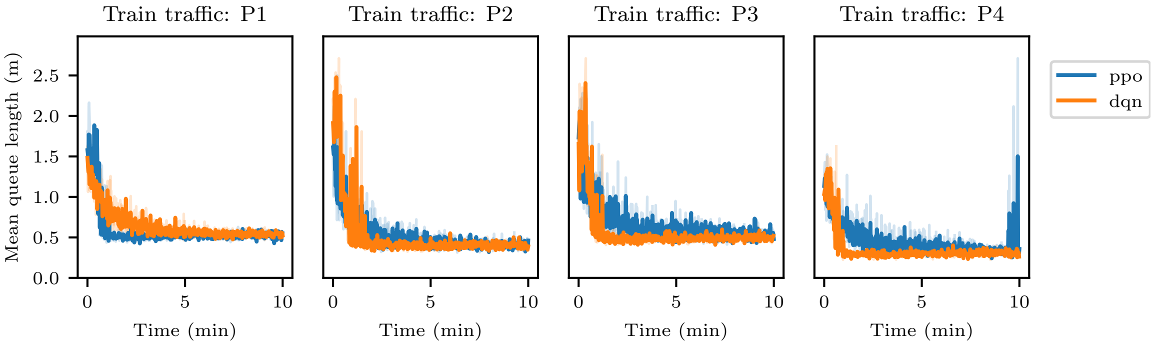


Figure 7: The mean queue length of DQN compared to PPO over relative training time. Both agents converge to optimal policies in a few minutes of training.

5.2 Performance comparisons

To evaluate the two RL agents against the three benchmark agents, we zoom in on the distribution of the waiting time, speed and queue lengths. Using SUMO's output functionality, we are able to export the waiting time and speed for each vehicle and the queue length for each edge at each second during the simulation.

5.2.1 Waiting time at the intersection

Figure 8 shows the distribution of speed for each of the three benchmarks. From the figure it is clear that the Webster algorithm performs the worst of the three benchmarks against all four test traffic patterns which is expected as it is unable to adapt to any dynamic traffic patterns. The delay-based method outperforms the gap-based method on all the traffic patterns except P4. This can be explained by the varying nature of the P4 traffic which the actuated method can handle effectively in a manner which reduces the tail (max waiting) of the distribution.

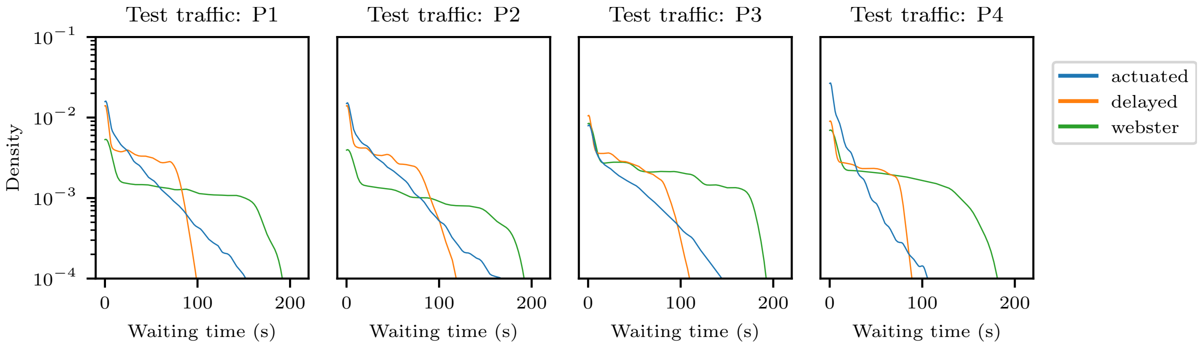


Figure 8: Vehicle waiting time distribution for 25 episodes across 4 traffic distributions for 3 benchmarks: Webster, delay and actuated.

Figure 9 shows the waiting time distribution of the DQN RL agent for all 16 combination of train-test traffic pairs. The waiting time distribution appears to be invariant to the traffic which the DQN agent is trained on, except for a slight benefit when training and testing on the same traffic for P3 and P4 respectively. This is expected for the more varying traffic as the agent has seen both regimes of traffic during training.

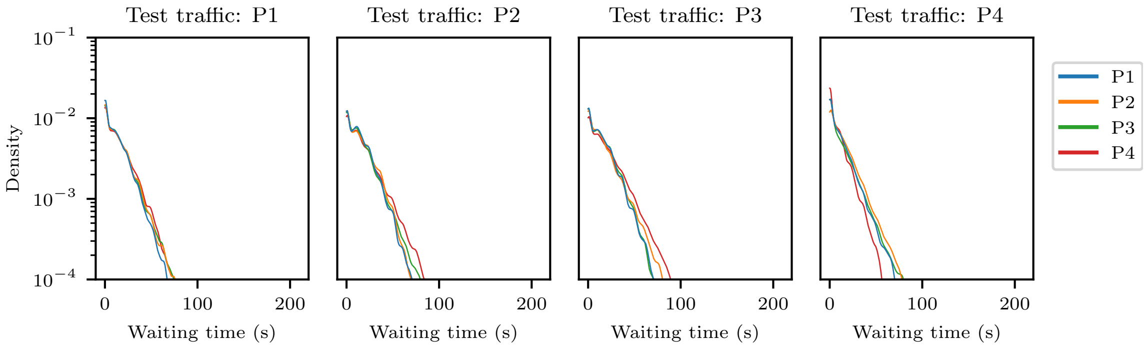


Figure 9: Vehicle waiting time distribution for 25 episodes for 4 test traffic patterns and 4 train traffic patterns for DQN.

Figure 10 shows the waiting time distribution of the PPO RL agent and tells a similar story to the DQN RL agent with the agents tested on the same network they were trained on performing slightly better compared to the other trained traffic patterns, but overall, all agents perform well and no cars have a waiting time greater than 100 seconds.

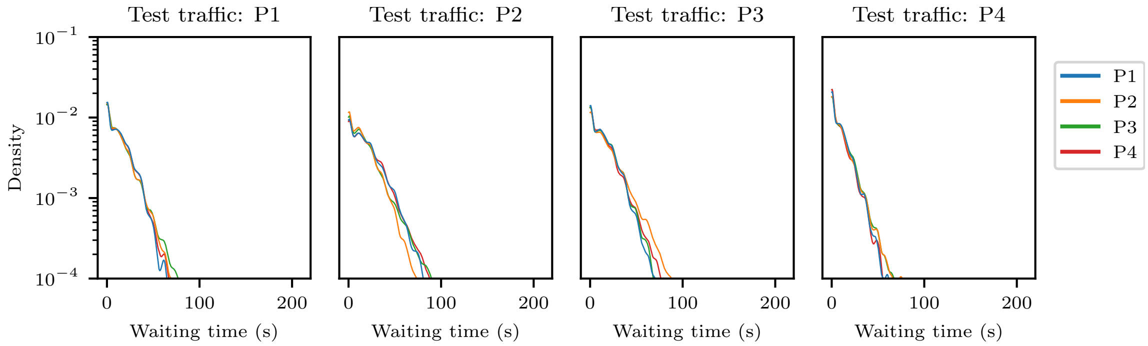


Figure 10: Vehicle waiting time distribution for 25 episodes for 4 test traffic patterns and 4 train traffic patterns for PPO.

Figure 11 compares the 3 benchmarks to the DQN and PPO agents trained and tested on the same traffic pattern. From the graph it is apparent the benefit of using the RL agents. They are adapting to the traffic demands, bringing the tail end of the waiting time distribution in to around 50 seconds. In other words, if the RL agents are used, there are no cars waiting at the intersection for longer than 50 seconds. According to Figure 11, the RL agents perform very similarly.

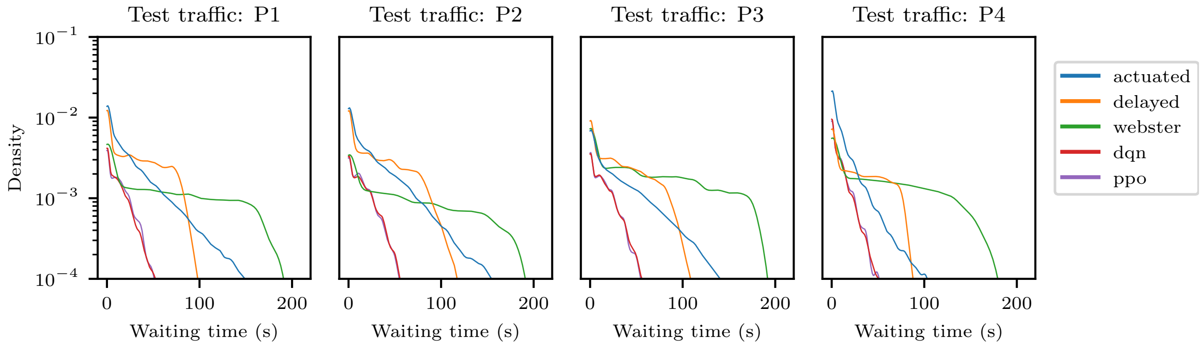


Figure 11: Vehicle waiting time distribution for 25 episodes across four test traffic patterns comparing benchmarks with the two RL agents.

The speed and queue lengths distributions show similar results to the waiting time distribution with the RL agents outperforming all the benchmarks. Figure 12 and Figure 13 show the two best RL agents (trained and tested on the same traffic) against the three benchmarks.

Figure 12 shows that for the benchmark agents there are more vehicles reaching higher speeds compared to the RL agents. However, there is a significant amount of cars that have speed less than 3 m.s^{-1}, whereas the RL agents' speed distributions are more uniformly distributed serving all cars more equally.

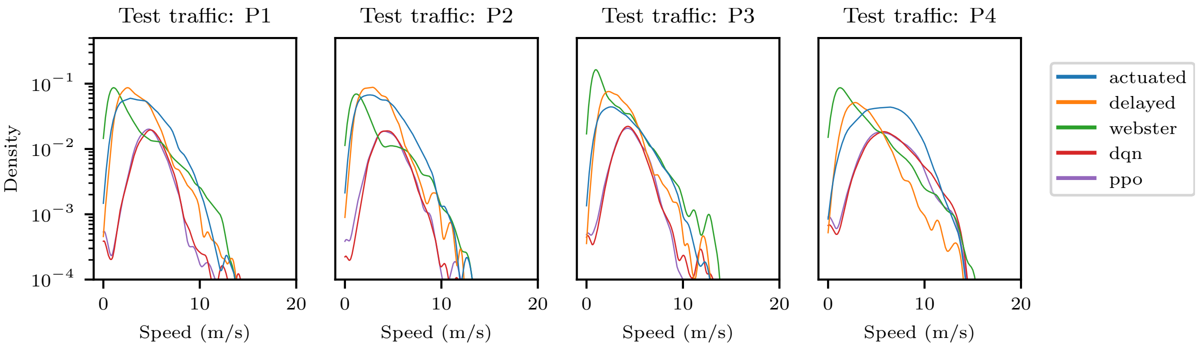


Figure 12: Vehicle speed distribution for 25 episodes across 4 test traffic patterns comparing benchmarks with RL agents.

Figure 13 provides another lens on the performance of the RL agents, showing the distribution of queue lengths for the three benchmarks and the two RL agents (trained and tested on the same traffic). From the graph it is clear that the RL agents are outperforming the benchmark agents as the RL agents are able to clear the lanes well before they fill up to their maximal capacity of 150m.

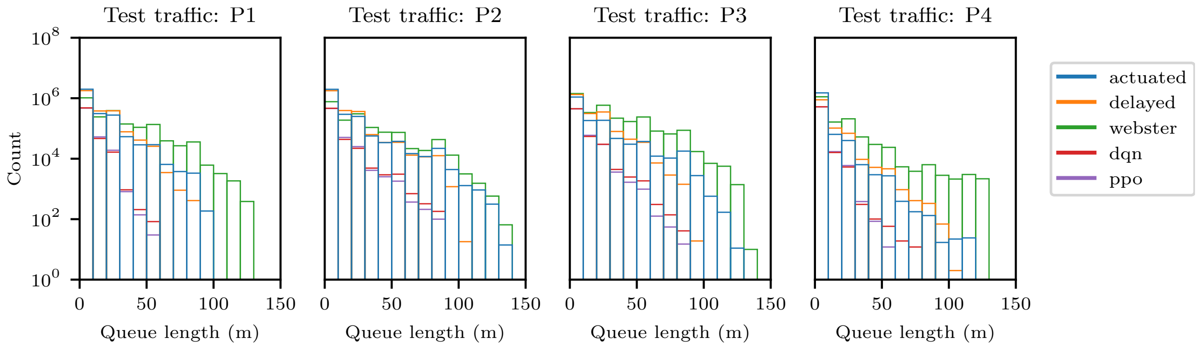


Figure 13: Lane queue length distribution for 25 episodes across four test traffic patterns comparing benchmarks to RL agents.

5.3 Trained policies

In this section we attempt to analyse the policy each of the benchmark and RL agents followed when being presented with a particular traffic pattern. In the case of the RL agents, how the policies are affected when trained on the various traffic patterns P1 through P4. We gain insights into the trained policies by looking at two different metrics namely the percentage time spent in a phase over the 25 episodes and the number of times an agent switched from one phase to another over the 25 episodes (phase transition matrix).

5.3.1 Percentage time spent in a phase

In order to condense the eight dimensional action space of the agents into a graph we us polar plots shown in Figure 15 with their legend shown in Figure 14. The polar plots compare the five agents against each other using the defined polar plot legend. The percentage time an agent spends in the different phases during testing determines the shape of the polygon. The further the vertex is from the centre, the longer the agent spent in that respective phase. This allows us to see insights such as if certain phases were favoured or ignored in the policy.

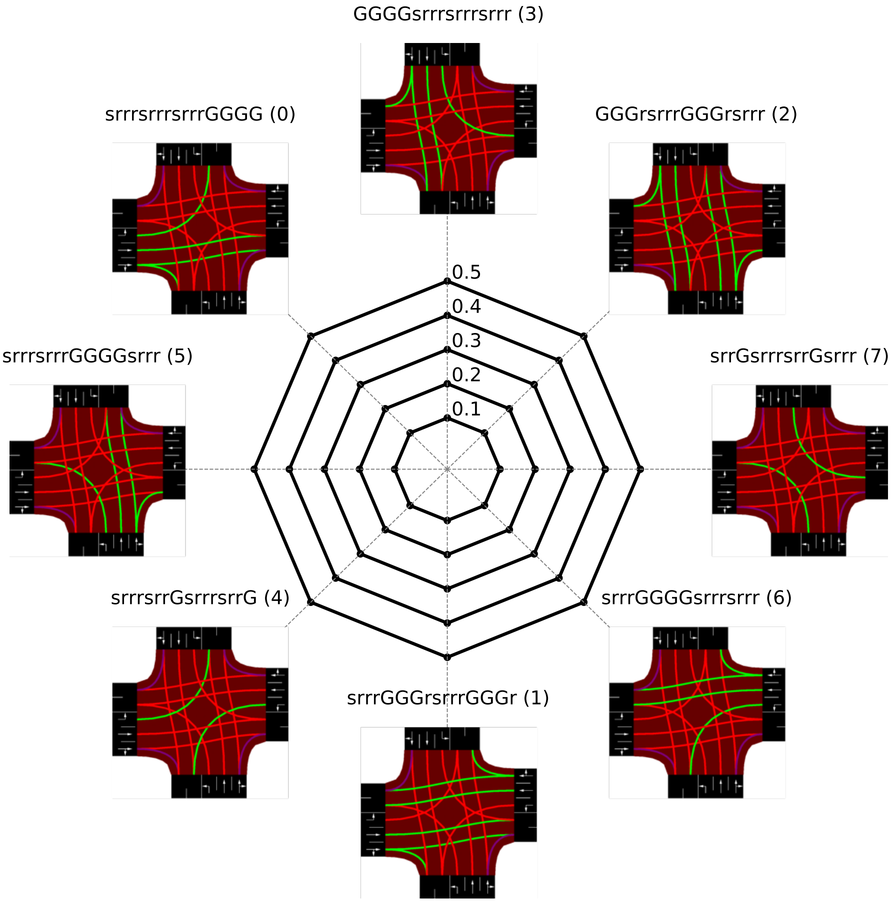


Figure 14: Policy polar plot legend.

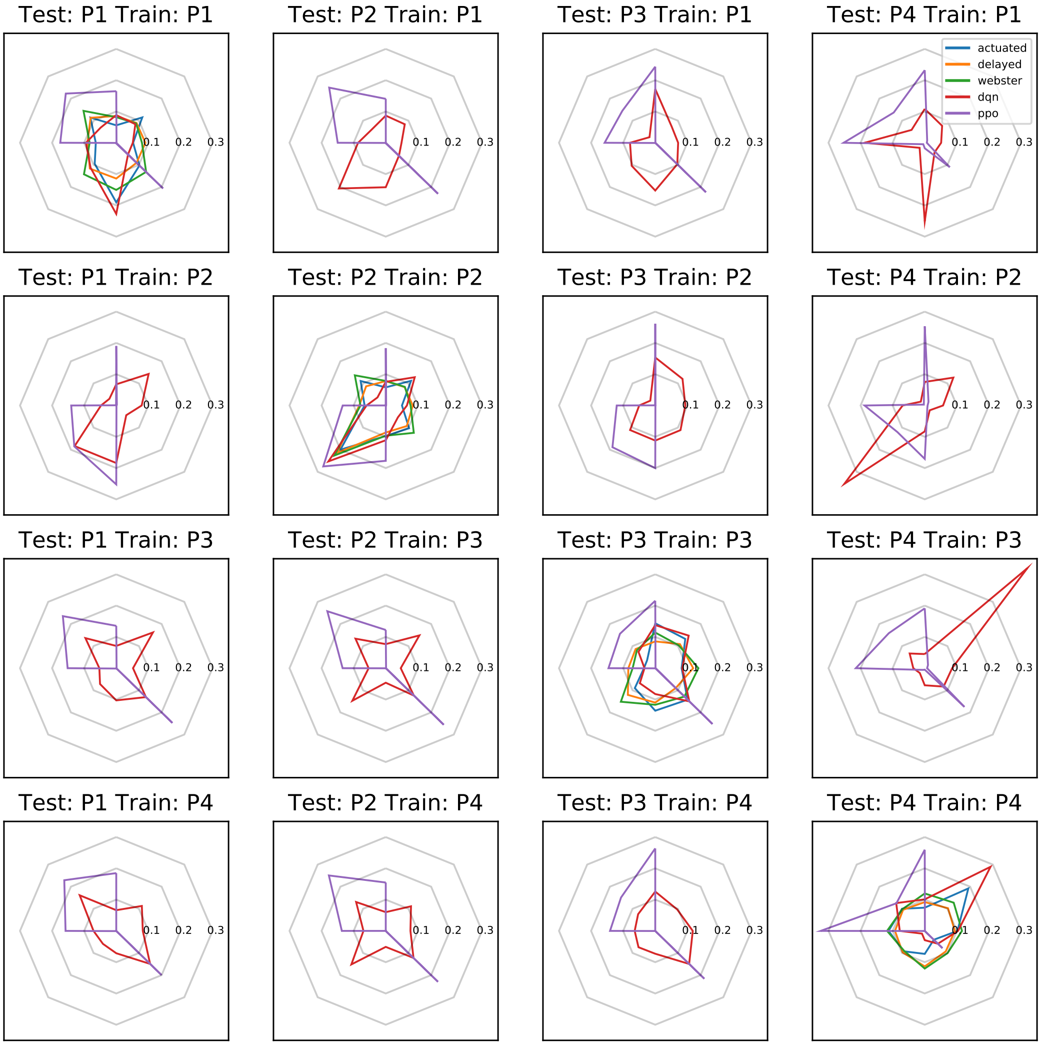


Figure 15: Percentage time spent in each state per agent per test-train traffic over 25 testing episodes.

From the emergent patterns in Figure 15 it is clear that the DQN and PPO agents have very different policies when it comes to optimising a given traffic set. This is evident by the different shaped polygons indicating the different policies favoured different states. A few note worthy patterns from Figure 15 are:

- The bird-like pattern the DQN policy has when trained on traffic P3 and P4 and tested on traffic P2.
- The kite-like pattern the PPO policy has when trained on traffic P1, P3 and P4 and tested on traffic P2, as well as when trained on traffic P3 and P4 and tested on traffic P1.

- The similarities in policies between the DQN agent and the benchmarks when the test and train traffic is P2.

All three of these cases are tested on the P2 traffic pattern which includes higher demand turning traffic. This highlights that increased turning traffic demands creates a significant change in the policy.

The bird-like pattern the DQN has when trained on traffic P3 and P4 and tested on traffic P2 can be explained by looking at the four phases the DQN agent visited most, namely: 0, 2, 4, and 6. Referencing Figure 14 we can see that these four phases allow the agent to switch for traffic on the main N-S route (phase 2), the E-W route (phase 6), the W-E route (phase 0) as well as having a dedicated turning phase (phase 4) for the E-W/W-E routes. Objectively, the bird-like policy of the DQN makes sense, as favour using these phases should allow an agent to address most traffic patterns.

The bird-like policy of the DQN is in stark contrast to the kite-like policy the PPO agent learnt. If we take a closer look at which phases the PPO agent favoured: 0, 3, 5 and 6, we can see that two of these overlap. Both agents require phase 0 and 6 to clear the E-W/W-E traffic. However, the difference between the PPO and DQN agents is evident in the way which they clear the N-S/S-N traffic, the PPO instead utilises phase 3 and 5 instead of phase 2 the DQN uses. Again, considering that both these RL agents performed equally well for our metrics and outperformed the benchmarks, it is clear that there are multiple optimum policies for a given intersection and traffic pattern.

Lastly, looking at the similarities between the DQN agent and benchmark agents when the train and test traffic both equal P2, we can see that the RL agents estimate the benchmark's policies under certain conditions. However, the DQN agent is able to put less of a dependency on phase 0 and 6 and clear the turning traffic demand more efficiently using phase 4. Whereas, the PPO agent resorts to using phase 3 and 5 for the N-S/S-N demand and in contrast to the DQN and benchmarks, uses phase 1 for the E-W/W-E demand instead of phase 0 and 6.

The waiting time, speed and queue length graphs from Section 5.2 showed comparable results across all policies. It is interesting to see that similar results can be achieved with such varied policies. It also highlights the significance of using such visualisations to evaluate the policies.

5.4 Phase transition matrices

The policy polar plots introduced in the previous subsection, are a useful insight into the policies in terms of where the agents spent their time. To gain further intuition into the policies we look into how the policies transition between the different phases. This can be visualised in Figures 16 through 20. These figures show the number of times each agent moved from one phase (on the x-axes) to another (y-axes) during the 25 episodes during testing.

We begin by introducing our baseline algorithms which are used as comparison for the RL agents. The phase transition matrices are shown for the three baseline methods namely Webster, delay-based and vehicle-actuated. The matrices can be visualised in Figures 16, 17 and 18 respectively.

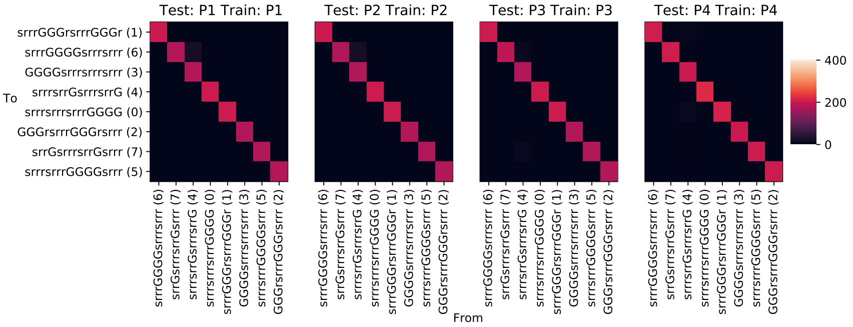


Figure 16: Number of times the traffic light switch from one phase (x-axis) to another (y-axis) over 25 episodes for the Webster algorithm.

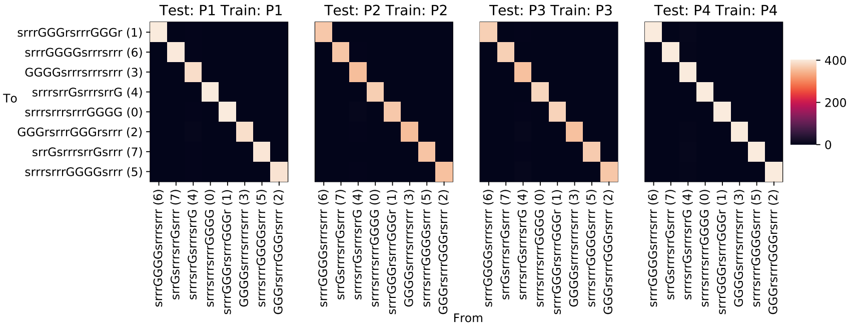


Figure 17: Number of times the traffic light switch from one phase (x-axis) to another (y-axis) over 25 episodes for the delay-based algorithm.

The first thing to note is the is the diagonals formed for the Webster (Figure 16) and delay-based (Figure 17) agents. These diagonals form as both these agents have a fixed sequence which they are allowed to use, so each phase is always followed by a particular other phase.

The delay-based method uses shorter minimum green times which can be extended dynamically as traffic patterns change. Since the Webster method has static phase timings not allowing for such extensions it's green time is set to a longer time period. This difference is illustrated in Figure 17 where the delay-based benchmark has higher counts (demonstrated by the brighter squares) than the Webster benchmark.

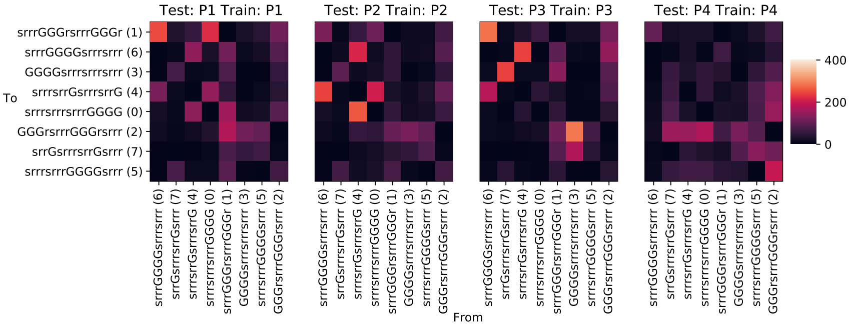


Figure 18: Number of times the traffic light switch from one phase (x-axis) to another (y-axis) over 25 episodes for the gap-based actuated algorithm.

The phase transition matrix for the gap-based actuated benchmark shown in Figure 18 shows that this benchmark is indeed able to dynamically select the next phase and in our configuration we've allowed it to choose any phase. There is little similarity between the the phase transition matrices of the gap-based actuated benchmark and the phase transition matrices of the DQN and PPO agents. This implies that the RL agents learned their own policies and did not learn a strategy similar to the conventional gap-based benchmark.

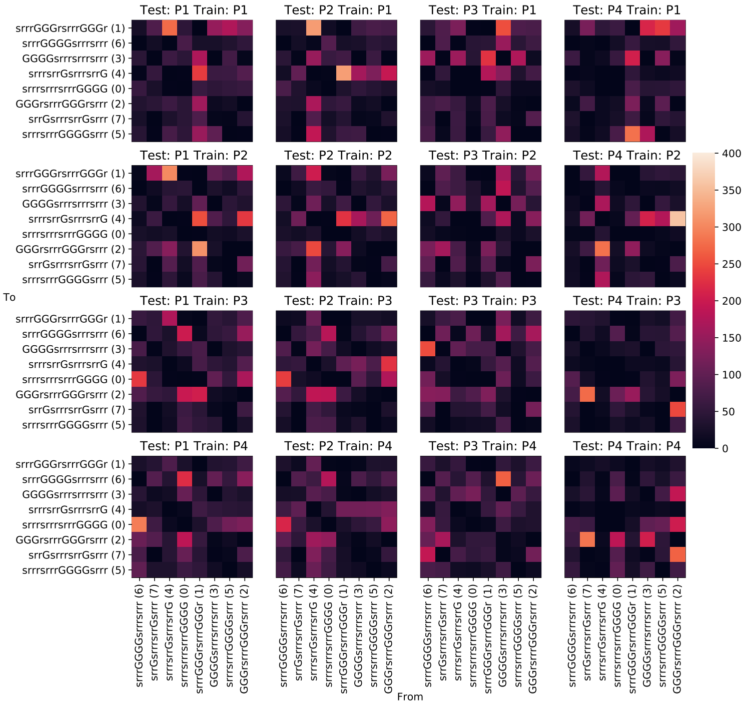


Figure 19: Number of times the traffic light switch from one phase (x-axis) to another (y-axis) over 25 episodes for the DQN agent.

Finally, we compare the phase transition matrices of the DQN and PPO agents, shown in Figure 19 and Figure 20 respectively. We can see that these agents approach the task at hand in vastly different ways. The DQN agent is much more "soft" with its transition, utilising all phases, whereas the PPO agent transitions between a subset of all phases and completely ignores some phases. This comes as a surprise as the PPO agent has the more probabilistic architecture, sampling from its sample space instead of making a hard prediction like the DQN. However, it appears from the phase transition matrices that these sampling distributions of the PPO quickly converge with very low variability around the sampling distribution.

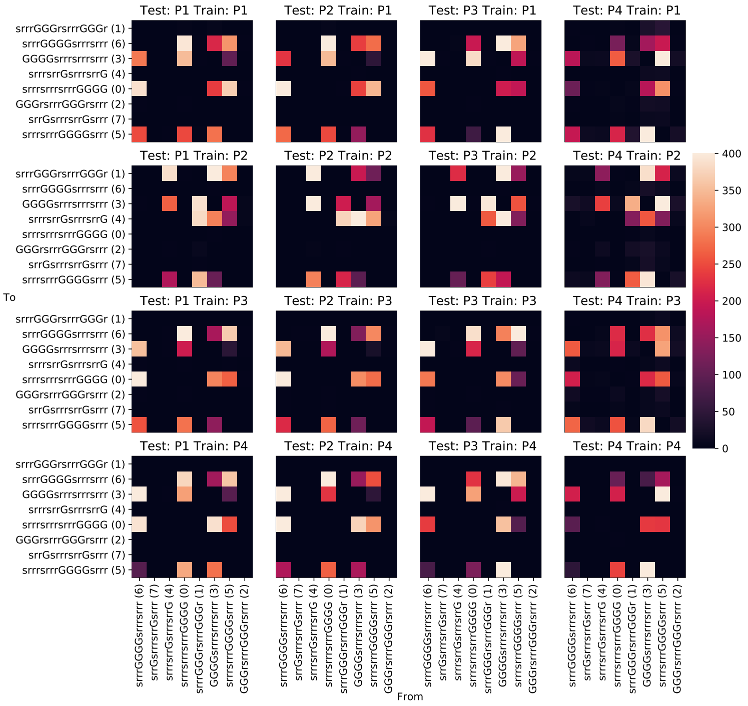


Figure 20: Number of times the traffic light switch from one phase (x-axis) to another (y-axis) over 25 episodes for the PPO agent.

It is also worth noting the patterns that emerge from the PPO's phase transition matrices across trained traffic patterns (same rows have same patterns). They suggest that the PPO agents overfit on the traffic pattern it has seen in training and struggles to generalise well to unseen traffic, only utilising the transitions it learnt during training. Whereas the DQN agent appears to generalise better across both training and testing, not favouring any transitions, but rather using all transitions to achieve the desired phase. For example when agents trained on traffic pattern P2 (which included a larger volume of left turning traffic) were tested on other traffic patterns, the agents tended to favour phases which allowed for the flow of left turning traffic. Even when tested on other traffic patterns, this agent is trained to optimise flow for this specific traffic pattern and tends to favour the phases that serve the left turning vehicles. The policies learnt by these agents approach the problem in vastly different ways, and when considering a real-world implementation of these agents, additional work would be required to ensure that the policies these agents learn transfer to the real world.

6 Conclusion

The aim of this paper was to provide an in depth comparison between Deep Q-learning and *proximal policy optimisation* (PPO) on a traffic light optimisation problem. Both the PPO and DQN methods drastically improved upon traditionally implemented methods namely, gap-based actuated, delay-based actuated and a Webster fixed-time approach. On a high level comparison of the agents, we see performance improve as they maximise their return during training.

Training time and computational efficiency can be a concern, since *deep reinforcement learning* (DRL) is known to require large amounts of data and many training iterations to converge to optimal policies. The experiments conducted in this study show that training times for this specific intersection are extremely short. In future research, it will be interesting to see how training times increase with more complex networks and intersections. Furthermore, no dataset is required to train an RL agent because the data required for training is generated by the simulation environment (SUMO). However, the simulation must be accurately parameterised, which usually necessitates the use of real-world data.

In addition to demonstrating that the policies are efficiently and effectively trained, the goal was to gain a deeper understanding of the strategies learned through the use of the various methodologies, namely PPO and DQN. To visualise the difference between the agents' policies, we introduce several plots, including the polar policy plot and phase transition matrices. These visualisations suggest that the strategies learned by these agents differ significantly. PPO agents typically use only a subset of all phases, whereas DQN agents spread their policy across all phases.

Another finding was that the agents tend to overfit on the traffic used to train a policy. An agent who has been trained on traffic with a high volume of vehicles turning left learns a policy that accommodates this specific traffic flow. This training artefact can be seen when the agent is tested on other traffic patterns, as discussed in the previous section. A future research avenue could be to train a more general policy that mitigates these training artefacts.

Despite the agents overfitting, this study shows the promise about using RL as an approach to traffic light optimisation. The agents performed well on the traffic they are trained on showing their ability to learn different types of traffic distributions and with a more encompassing training set can become a robust and efficient solution.

References

[1] Baher Abdulhai, Rob Pringle, and Grigoris J Karakoulas. Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering*, 129(3):278–285, 2003.

[2] Richard E Allsop. Delay-minimizing settings for fixed-time traffic signals at a single road junction. *IMA Journal of Applied Mathematics*, 8(2):164–185, 1971.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[4] Alvaro Cabrejas Egea and Colm Connaughton. Assessment of reward functions in reinforcement learning for multi-modal urban traffic control under real-world limitations. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 2095–2102. IEEE, 2021.

[5] Mengyu Guo, Pin Wang, Ching-Yao Chan, and Sid Askary. A reinforcement learning approach for intelligent traffic signal control at urban intersections. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 4242–4247. IEEE, 2019.

[6] PB Hunt, DI Robertson, RD Bretherton, and RI Winton. Scoot-a traffic responsive method of coordinating signals. Technical report, 1981.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[8] Li Li, Yisheng Lv, and Fei-Yue Wang. Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, 3(3):247–254, 2016.

[9] 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 *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[10] PR Lowrie. Scats, sydney co-ordinated adaptive traffic system: A traffic responsive method of controlling urban traffic. 1990.

[11] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[12] Sajad Mousavi, Michael Schukat, Peter Corcoran, and Enda Howley. Traffic light control using deep policy-gradient and value-function based reinforcement learning. *IET Intelligent Transport Systems*, 11, 04 2017.

[13] Robert Oertel and Peter Wagner. Delay-time actuated traffic signal control for an isolated intersection. In *Proceedings 90th Annual Meeting Transportation Research Board (TRB)*, 2011.

[14] Syed Shah Sultan Mohiuddin Qadri, Mahmut Ali Gökçe, and Erdinç Öner. State-of-art review of traffic signal control methods: challenges and opportunities. *European transport research review*, 12(1):1–23, 2020.

[15] Sampson, van As, Joubert, Dazeley, Labuschagne, and Swanepoel. South african road traffic signs manual. *Civil Engineering= Siviele Ingenieurswese*, 3(3):101, 2012.

[16] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[17] Li Song, Wei Fan, and Pengfei Liu. Exploring the effects of connected and automated vehicles at fixed and actuated signalized intersections with different market penetration rates. *Transportation Planning and Technology*, 44(6):577–593, 2021.

[18] Aleksandar Stevanovic, Cameron Kergaye, and Peter Martin. Scoot and scats: A closer look into their operations. 01 2009.

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

[20] Tomer Toledo, Tamir Balasha, and Mahmud Keblawi. Optimization of actuated traffic signal plans using a mesoscopic traffic simulation. *Journal of Transportation Engineering, Part A: Systems*, 146(6):04020041, 2020.

[21] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[22] F.V. Webster. Traffic signal settings. Technical report, 1958.

[23] Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. Recent advances in reinforcement learning for traffic signal control: A survey of models and evaluation. *ACM SIGKDD Explorations Newsletter*, 22(2):12–18, 2021.

[24] Hua Wei, Guanjie Zheng, Huaxiu Yao, and Zhenhui Li. Intellilight: A reinforcement learning approach for intelligent traffic light control. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2496–2505, 2018.

A Simulation configuration

Parameter	**Value**
Lane length	150 meters
Vehicle length	5 meters
Minimal gap between vehicles	2.5 meters
Car-following model	Krauss following model [9]
Max vehicle speed	13.42 m.s^{-1}
Acceleration ability of vehicles	2.6 m.s^{-2}
Deceleration ability of vehicles	4.5 m.s^{-2}
Duration of yellow signal	3 seconds
Minimum green time	10 seconds
Number of episodes	200
Episode length	1800 seconds

Table 3: Simulation parameters [24]

Traffic pattern	Direction	Arrival rate		
		Through	Left-turn	Right-turn
P1:Major/Minor road	N-S	0.05	0.025	0.01
	S-N	0.05	0.025	0.01
	E-W	0.1	0.05	0.01
	W-E	0.1	0.05	0.01
P2:Through/left-turn lane	N-S	0.05	0.025	0.01
	S-N	0.05	0.025	0.01
	E-W	0.05	0.1	0.01
	W-E	0.05	0.1	0.01
P3:Tidal traffic	N-S	0.1	0.08	0.01
	S-N	0.05	0.025	0.01
	E-W	0.1	0.08	0.01
	W-E	0.05	0.025	0.01
P4:Varying demand traffic	N-S	0.05	0.025	0.01
	S-N	0.05	0.025	0.01
	E-W	0.05 (0-1200 steps) 0.15 (1200-1800 steps)	0.025	0.01
	W-E	0.15 (0-600 steps) 0.05 (600-1800 steps)	0.025	0.01

Table 4: Arrival rates for different traffic patterns as outlined by Wei et al. [24]

B Benchmark configuration

Parameter	**Value**
Lost time	12
Max cycle	240
Min cycle	20
Sat headway	3.2

Table 5: Configuration used for Webster

P1	P2	P3	P4	State
32	50	34	23	srrrsrrGsrrrsrrG (4)
32	30	19	21	srrrsrrrsrrrGGGG (0)
34	22	27	26	srrrGGGrsrrrGGGr (1)
33	30	29	22	srrrGGGGsrrrsrrr (6)
21	21	33	25	srrGsrrrsrrGsrrr (7)
21	21	18	24	srrrsrrrGGGGsrrr (5)
21	21	26	26	GGGrsrrrGGGrsrrr (2)
20	21	29	26	GGGGsrrrsrrrsrrr (3)

Table 6: Webster phase timings for each traffic pattern

Parameter	**Value**
Detector gap	1
Minimum duration (all phases)	10
Maximum duration (all phases)	60
Frequency	300
Max gap	3
Nexts	All phases included current
Passing time	10

Table 7: Configuration for gap-based actuated

Parameter	**Value**
Detector range	100
Minimum duration (all phases)	10
Maximum duration (all phases)	60
Frequency	300
Minimum time loss	1

Table 8: Configuration for time-loss actuated

C Additional results

C.1 Speed

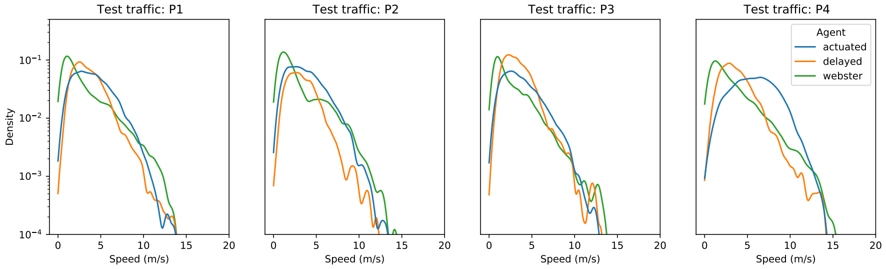


Figure 21: Benchmarks

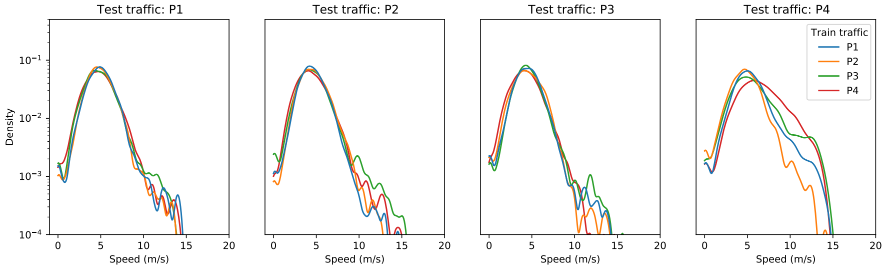


Figure 22: DQN

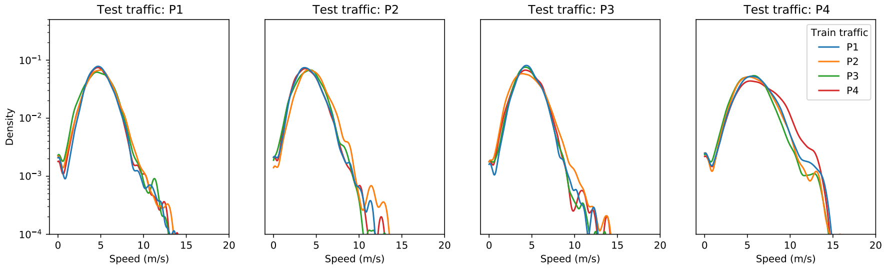


Figure 23: PPO

C.2 Queue lengths

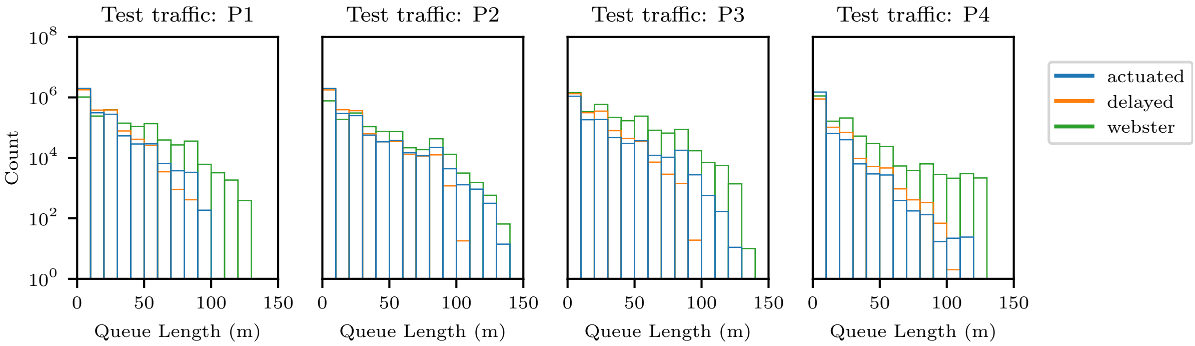


Figure 24: Benchmarks

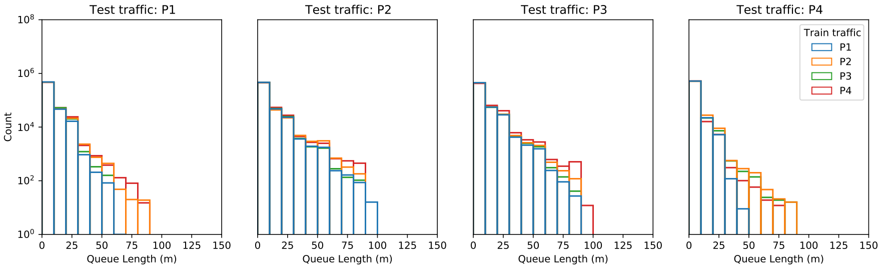


Figure 25: DQN

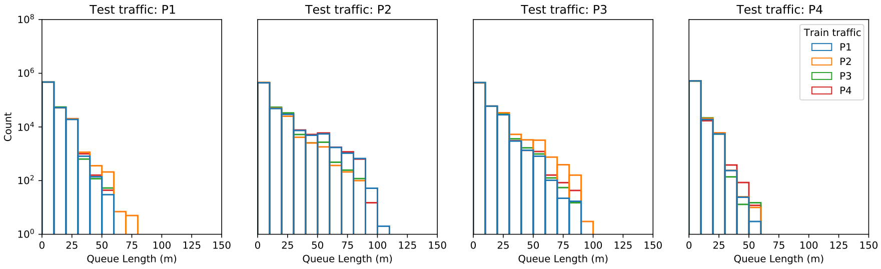


Figure 26: PPO

C.3 Percentage time spent in each phase

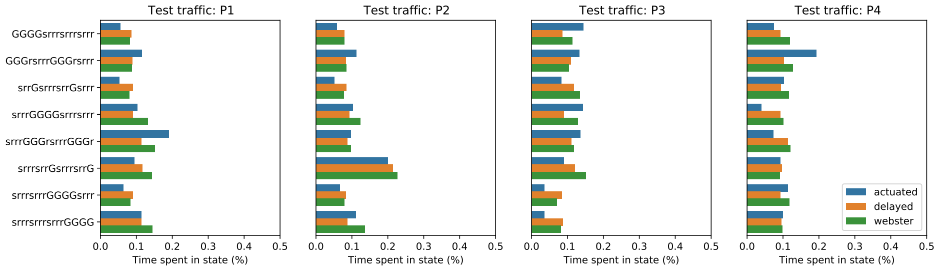


Figure 27: Percentage time spent in phase for the benchmarks agent for each test traffic .

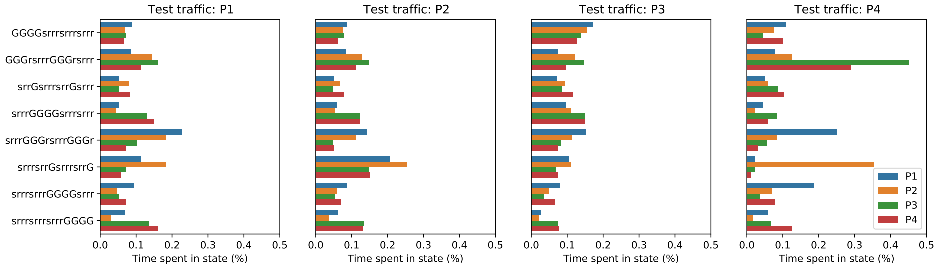


Figure 28: Percentage time spent in phase for the DQN agent for each test train traffic pair.

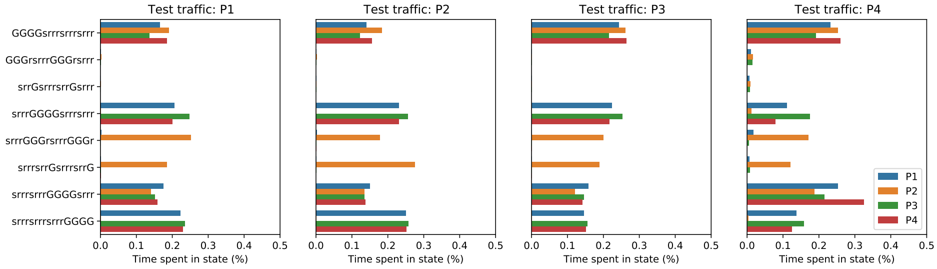


Figure 29: Percentage time spent in phase for the PPO agent for each test train traffic pair.

SUMO User Conference 2022
https://doi.org/10.52825/scp.v3i.114

Published: 29 Sept. 2022

High fidelity modelling of traffic light control with XML logic representation

Maik Halbach (1) and Jakob Erdmann (2),
(1) German Aerospace Center (DLR), Braunschweig, Germany
maik.halbach@dlr.de
(2) German Aerospace Center (DLR), Berlin, Germany
jakob.Erdmann@dlr.de

Abstract

The paper describes a novel approach to modelling traffic light control in SUMO with control logic configured in xml inputs. It shows also, that this approach allows for high fidelity replication of a real-world traffic controller by comparing original and simulated switching behavior when confronted with the same traffic situations. The approach could process from a traffic engineer with limited programming knowledge.

Contents

1 Motivation

Traffic lights are an important element of traffic simulation and their accurate representation is necessary to achieve good agreements between real world traffic and simulation traffic. Even when new control concepts are to be tested, it is necessary to represent existing systems for a fair assessment of gains.

The traffic simulation SUMO supports default traffic algorithms for fixed-timing control as well as adaptive traffic control based on detected time gaps. It also permits coupling with an external process for fine-grained control of traffic light switching via the TraCI API.[1] When using TraCI, it is possible to model the traffic light logic within different programming Languages such as Python, C++, Matlab, and Java[2]. Both native and coupled approaches have their advantages and disadvantages. In this paper we introduce a new configuration approach that has advantages over the previous approaches:

- High fidelity representation of control algorithms which differ from the default algorithms
- No need for process coupling (simpler and faster)
- Simplified and standardized algorithm description mirroring existing standards (compared to unconstrained TraCI code)
- Graphical analysis of operation and internal controller state with sumo-gui

Keywords: Simulation, traffic lights, adaptive control

2 Introduction

The paper describes a novel approach to modelling traffic light control in SUMO with control logic configured in xml inputs. We show that this approach allows for high fidelity replication of a real-world traffic controller by comparing original and simulated switching behavior when confronted with the same traffic situations.

In the following, we explain the syntax and semantics of the new configuration langue. For this purpose, an exemplary control algorithm is introduced and presented in the typical format used by traffic engineers in Germany. We then show how to translate this algorithm into an xml configuration usable by SUMO.

The figures C, 4, 7, 9, 10, 12, 15 represent a traffic light document (TLD). The translation of the TLD elements into a SUMO XML configuration are shown in figures

A, B, 1, 2, 3, 5, 6, 8, 11, 13, 14, 16, 18. The process of translating the TLD into the corresponding XML configuration is described in the following chapters.

2.1 Enhancements of the SUMO Software

To enable a high-fidelity modelling of traffic light control with xml logic representation, we implemented the new features in SUMO listed below. Further information about these features beyond the information in this paper, can be found in the SUMO documentation.[3]

- Coordination of actuated traffic lights
- Type 'actuated' with custom switching rules
- Overriding phase attributes with expressions
- Storing and modifying custom controller variables at runtime
- Extended signal plan visualization

In the researching project SAVeNoW[4] we used all listed functions to model the logic of a real traffic light controller in a SUMO xml-file. This paper uses an exemplary TLD which shows the most important controller design elements for a simplified intersection. The example controller is not completely efficient and realistic but it includes most of the functions typically needed for traffic light control modelling (phase logic and public transport prioritization).

2.2 Structure of this paper

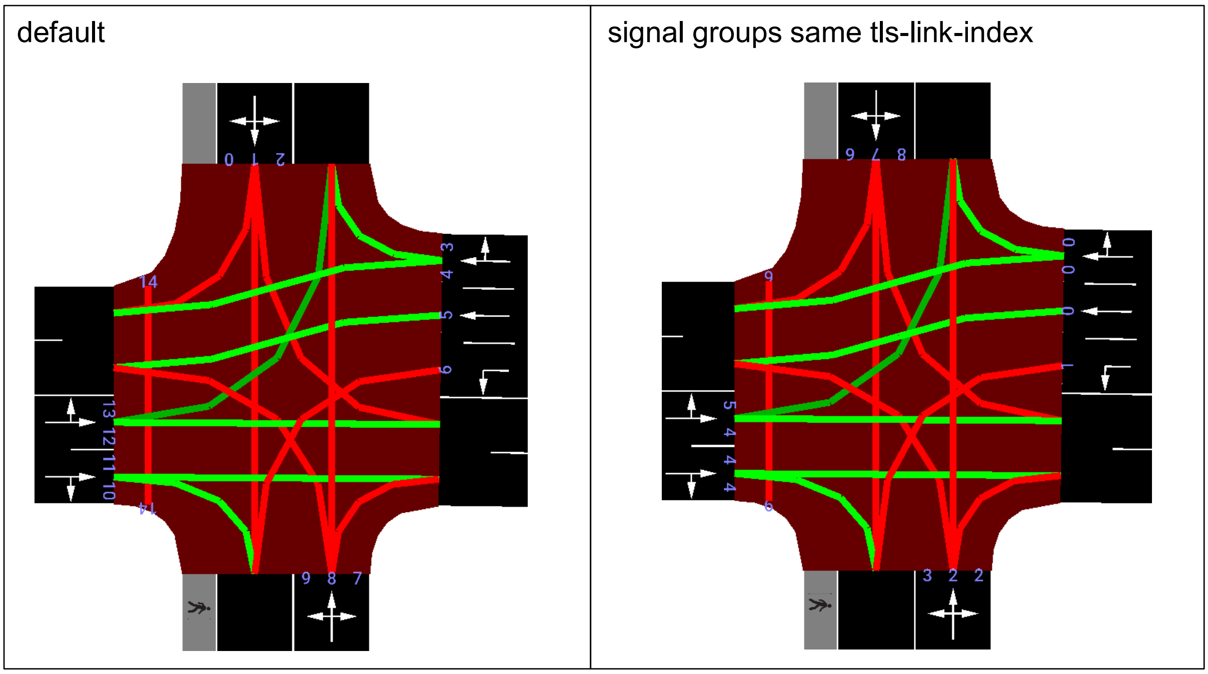
First of all, we describe the SUMO Net preparation and the preparation of the additional xml files for the detectors and the traffic light. Than we describe basic traffic light configurations. We follow by describing the modelling of traffic light logics in the traffic light xml file. First the definition of constants/parameters, then how to identify important logic blocs for the basic timing attributes of sumo and also the basic function of every expressions. Afterwards we describe the definition in detail and also discuss alternative definition styles.

3 XML modelling description

3.1 SUMO Net preparation

First of all, the SUMO Net and traffic light of the examined intersection has to be created. By default, every lane-to-lane connection is assigned its own “tls link index” to define its position within the signal state description string. To remove redundancy in the phase description and visualization, we define signal groups by assigning the same tls-link-index to all connections with the same signal behavior (in this, permissive green and protected green are counted as different signal behavior). (Figure 1, example link0, link2, link4; right frame), besides/apart from conditionally compatible the signal state description changes, so the link number has to be count up. By default, SUMO counts the link number clockwise. It is also possible to change the order of the link indices. An example can be seen in Figure 1.

Figure 1: SUMO intersection tls-link-index



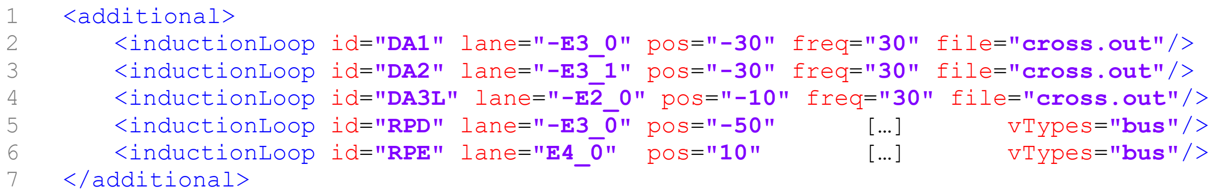
The traffic light program in SUMO is defined as a list of phases numbered from 0 to n. We distinguish these phases by calling them “stages” and “interstages” depending on their role during the operation of the traffic light. Stages permit vehicle movements and their length may be adjusted in response to traffic conditions. The state of every signal stays constant while a stage is active. Interstages form the transition between stages and must ensure that traffic flowing in one stage has left the intersection space before a new stage begins. So, the states of signals could change from one interstage phase to the next.

In the following program we assign to each sumo phase a name according to their stage number or the interstage (stage-to-stage transition) it belongs to (i.e. 1,2,3, 1.2, 2.3, 3.1). (TLD: Figure C, SUMO-xml: Figure A, Figure B) These names serve as a visual aid when following the sequence of phases in the tracking diagram (chapter: 3.6 Tracking Phases).

A signal plan can be generated for SUMO with the aid of an OCIT-file and the tool ocit2SUMO[5]. A signal plan may also be created visually in netedit (i.e. when the traffic light specification is only available as a pdf document).

An additional preparation for the SUMO simulation is the definition of the detectors (induction loops). They are defined with the same name and also the same position (Figure 2, Figure 3) at each lane according to the TLD signal location plan (Figure 4).

Figure 2: detector definition – SUMO det.add xml



```
1   <additional>
2       <inductionLoop id="DA1" lane="-E3_0" pos="-30" freq="30" file="cross.out"/>
3       <inductionLoop id="DA2" lane="-E3_1" pos="-30" freq="30" file="cross.out"/>
4       <inductionLoop id="DA3L" lane="-E2_0" pos="-10" freq="30" file="cross.out"/>
5       <inductionLoop id="RPD" lane="-E3_0" pos="-50"      […]       vTypes="bus"/>
6       <inductionLoop id="RPE" lane="E4_0"  pos="10"       […]       vTypes="bus"/>
7   </additional>
```

Figure 3: net - SUMO GUI

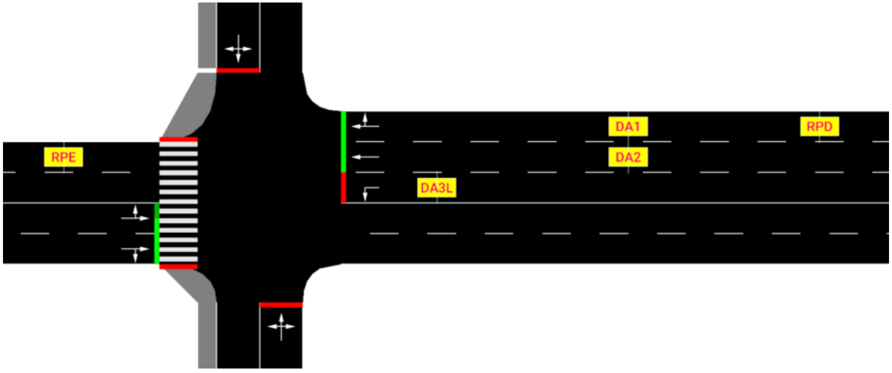
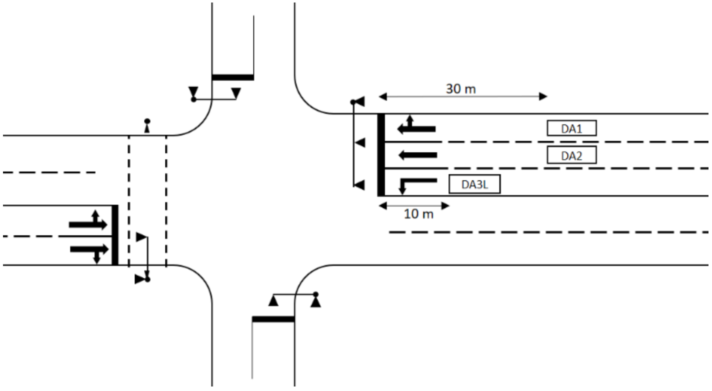


Figure 4: signal location plan - TLD



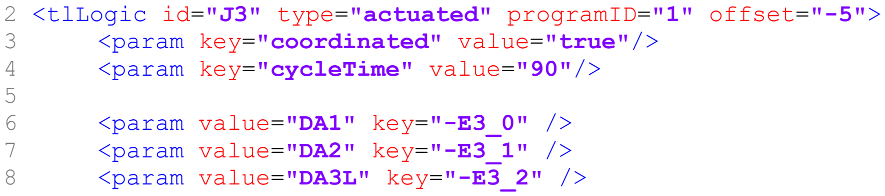
For the public transport prioritization, mostly "telegrams" (sent upon passing a specific location) are used. Here it is possible to use induction loops in sumo that are placed at the corresponding location and which are only triggered by specific vehicle types (i.e. bus) (Figure 2, line 5,6). All detectors are defined in an additional file. (Figure 2)

3.2 Basic traffic light configurations

The next step is to declare the traffic light type of the example intersection as “actuated” to activate the core functionality of switching in response to traffic detectors. (Figure 5, line 2; type = “actuated”). The parameter coordinated has to be set to “true” in order to align the cycle second counter of the traffic light with the simulation time (Figure 5, line 3). In coordinated mode, the cycle-second time range configured by the attributes *earliestEnd* and *latestEnd* will be aligned with all other traffic lights of the same *cycleTime*. The *cycleTime* attribute (Figure 5, line 4) denotes the duration of one switching cycle. The *offset* attribute (Figure 5, line 2) defines where within the cycle, the signal program will start, effectively shifting the initial cycle second.

An optional next step is to assign the detectors related to the signal plan and the intersection (Figure 5, line 6-8). This permits using custom detector placement instead of detectors that would otherwise be generated in default locations on each incoming lane during initialization. To define a custom detector, a lane that is incoming to the traffic light is used as the key and the id of a custom detector is used as the value. Both custom and automatic detectors can later be observed when tracking operations visually (chapter: 3.6 Tracking Phases)

Figure 5: basic traffic light configurations – SUMO xml



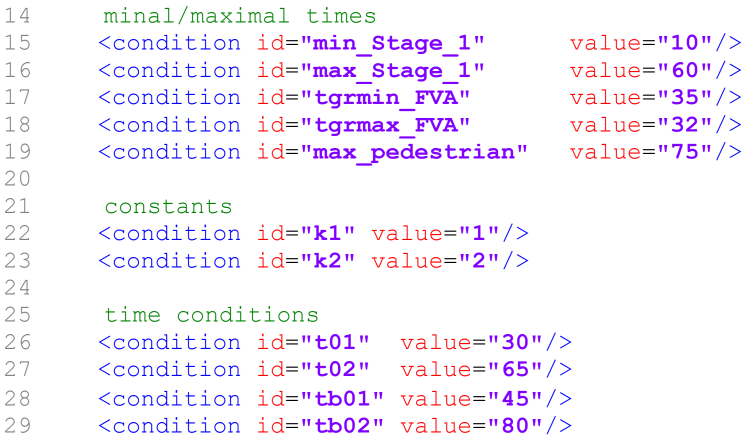
```
2 <tlLogic id="J3" type="actuated" programID="1" offset="-5">
3      <param key="coordinated" value="true"/>
4      <param key="cycleTime" value="90"/>
5
6      <param value="DA1" key="-E3_0" />
7      <param value="DA2" key="-E3_1" />
8      <param value="DA3L" key="-E3_2" />
```

(abstract of Figure A)

3.3 Define constants/parameters out of the TLD

The parameter setting out of the TLD are defined in conditions with fixed values in the SUMO XML (Figure 6). This corresponds directly to a table of values as in TLD (Figure 7). This simplifies re-use of an existing algorithm by only modifying its parameters to create another signal plan with different parameters for a different time of the day.

Figure 6: Table of algorithm configuration constants – SUMO xml



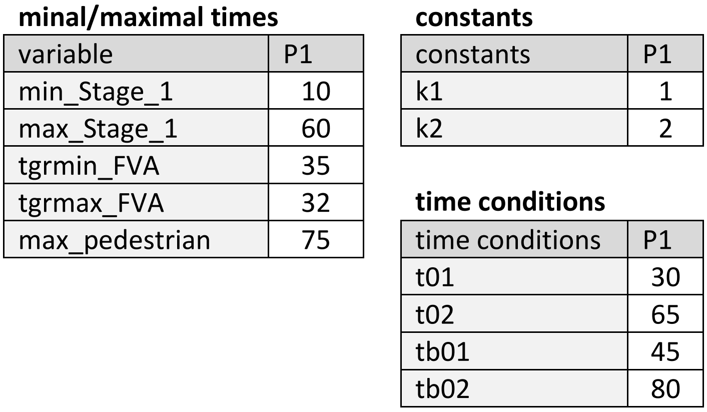
```
      minal/maximal times
     <condition id="min_Stage_1"      value="10"/>
     <condition id="max_Stage_1"      value="60"/>
     <condition id="tgrmin_FVA"       value="35"/>
     <condition id="tgrmax_FVA"       value="32"/>
     <condition id="max_pedestrian"   value="75"/>

      constants
     <condition id="k1" value="1"/>
     <condition id="k2" value="2"/>

      time conditions
     <condition id="t01"  value="30"/>
     <condition id="t02"  value="65"/>
     <condition id="tb01" value="45"/>
     <condition id="tb02" value="80"/>
```

(abstract of Figure A)

Figure 7: Table of algorithm configuration constants – TLD



minal/maximal times

variable	P1
min_Stage_1	10
max_Stage_1	60
tgrmin_FVA	35
tgrmax_FVA	32
max_pedestrian	75

constants

constants	P1
k1	1
k2	2

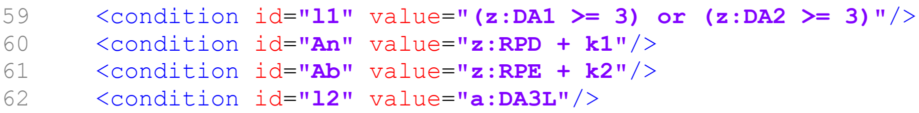
time conditions

time conditions	P1
t01	30
t02	65
tb01	45
tb02	80

3.4 Define logical condition

The logical condition out of the TLD (Figure 9) are defined in conditions (Figure 8).

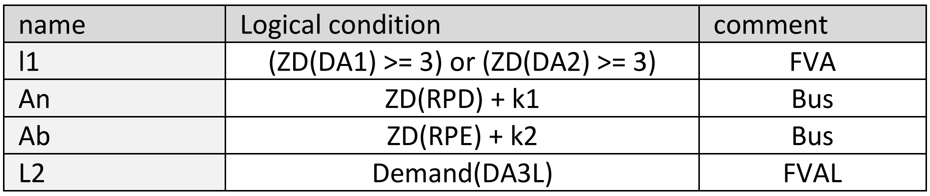
Figure 8: logical conditions – SUMO xml



```
    <condition id="l1" value="(z:DA1 >= 3) or (z:DA2 >= 3)"/>
    <condition id="An" value="z:RPD + k1"/>
    <condition id="Ab" value="z:RPE + k2"/>
    <condition id="l2" value="a:DA3L"/>
```

(abstract of Figure A)

Figure 9: logical conditions – TLD



name	Logical condition	comment
l1	(ZD(DA1) >= 3) or (ZD(DA2) >= 3)	FVA
An	ZD(RPD) + k1	Bus
Ab	ZD(RPE) + k2	Bus
L2	Demand(DA3L)	FVAL

All logic conditions here used functions to retrieve information from detectors previously assigned to the traffic light controller, but it is also possible to retrieve information from every detector loaded into the simulation. The function z:DETID retrieves the time (in seconds) since the last vehicle detection at the detector with id DETID. The function a:DETID returns a value of 1 if the given detector is occupied and 0 otherwise.

3.4 Phase logic modelling

We show two possible ways to model a phase logic.

	TLD	SUMO-XML modelling
Constant time bounds	Figure10	Figure A
Variable time bounds	Figure13	Figure B

The approach of the first example can be used when all attributes that describe the time boundaries of a phase *minDur*, *maxDur*, *earliestEnd* and *latestEnd* have fixed values.

```
<phase […] minDur="0" maxDur="40" earliestEnd="70" latestEnd="84"/>
```

(abstract of Figure A)

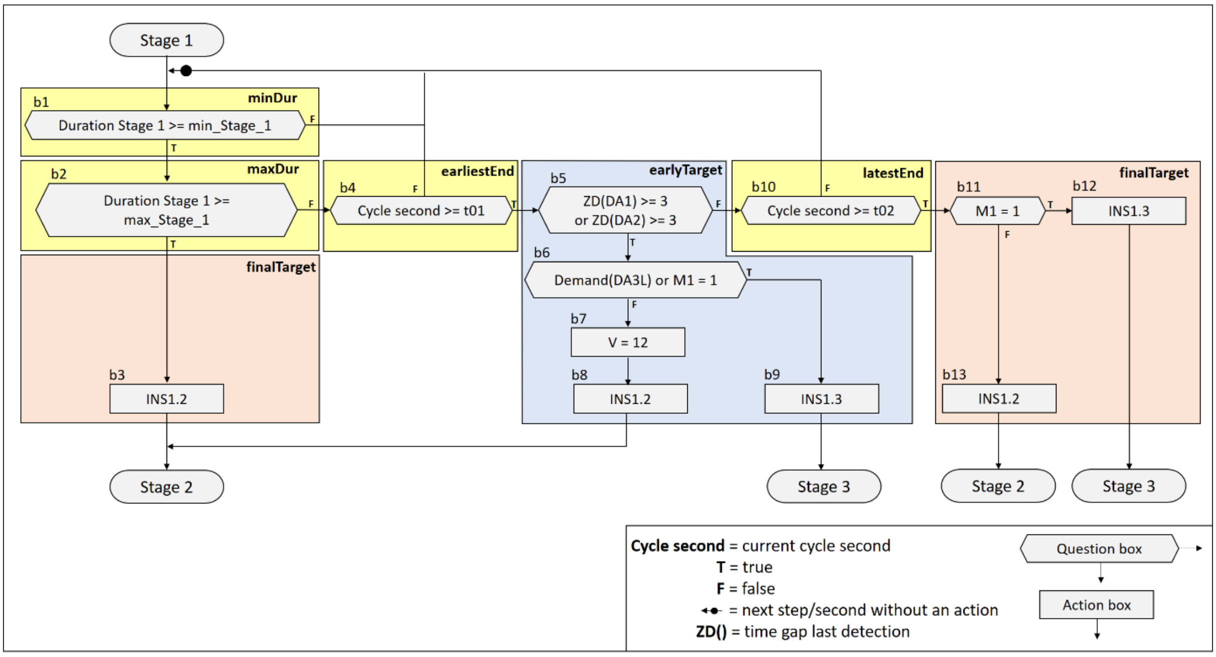
The advantage of this approach is the brevity of the XML definition. However, some TLD descriptions may be too complex to use constant values to describe time boundaries or it may be too hard to restructure the flow diagrams of the TLD and extract these constants.

In this case it can be simpler to “blindly” transcribe all logic elements from the TLD and forgo the “simpler” XML definition. Hence, we also describe a second approach where the attributes *minDur*, *maxDur*, *earliestEnd* and *latestEnd* are replaced by complex conditions. This more general approach can be used to transcribe any TLD logic but the resulting XML configuration is lengthier and somewhat harder to understand. Good names for the employed conditions help to keep the descriptions readable and aids in debugging with the phase tracking dialog.

3.4.1 Example with constant time bounds

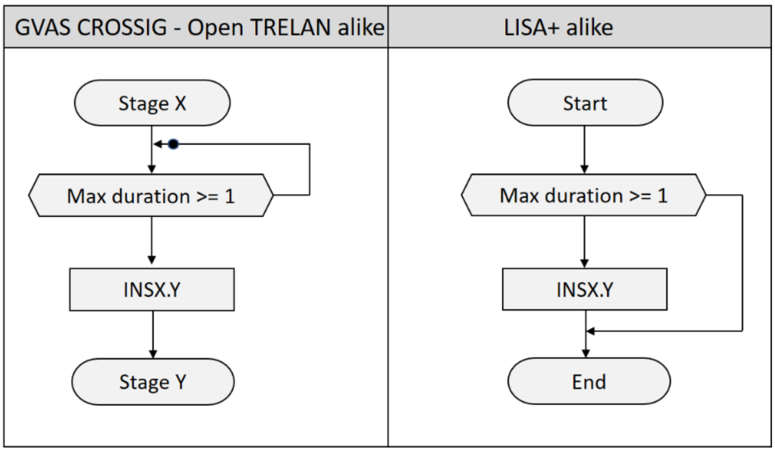
Figure 10 shows the phase logic of the first example as expressed in the TLD.

Figure 10: Example 1 - TLD logic



All shown TLD phase logics in this paper are similar to the CROSSIC Software notation (Open TRELAN). For readers familiar with the LISA+ notation, we show the main difference in flow diagram style in Figure 11.

Figure 11: Phase x logic example



3.4.1.1 Sumo phase definition

Each phase (stage or interstage) can either be of fixed or variable duration. Whereas fixed phases may be fully described by the *duration* attribute, the *attributes minDur, maxDur, earliestEnd, latestEnd* are used to describe the time bounds for variable-length phase. The attributes *next, earlyTarget, finalTarget* are used to define successor relationships among phase and the conditions for switching between them. All these attributes will be described in the following.

3.4.1.2 Phase logic modelling

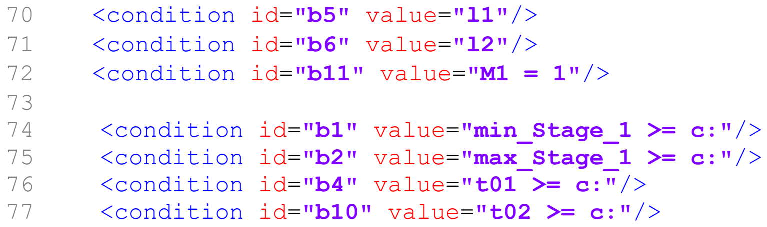
The switching time for a phase of variable length may be restrained by upper and lower bounds with regard to its running duration and also with regard to a time window for coordination. For this purpose, the 'phase' element provides the following attributes:

- **minDur**: minimum running duration (mandatory)
- **maxDur**: maximum running duration (optional)
- **earliestEnd**: earliest time within a cycle for ending phase (optional)
- **latestEnd**: latest end within a cycle for ending phase (optional)

These attributes correspond to standard control parameters in a typical TLD and in our example they take on the values of b1, b2, b4 and b10 (Figure 10).

The possible reachable interstages are here in the blocs b3, b8, b9, b12 and b13 (Figure 10). These blocs represent which interstages are used to skip in another phase and also define in which following phase the phase0/stage1 is able to switch. Attribute *next* defines in which possible following phases the phase0/stage1 could switch. *EarlyTarget* defines conditions which are checked upon entering the lower time bounds of *minDur* and *earliestEnd*. *FinalTarget* defines conditions which are checked when reaching the upper time bounds of *maxDur* or *latestEnd*. In Figure 10, the yellow blocks (b1, b2, b4, b10) define these time bounds. The remaining blocks from the *earlyTarget* area *(blue)* and *finalTarget* area *(red)* in Figure 10 we have to define for possible paths through the logic. So, every question bloc in the *earliestTarget* area (Figure 10, blue) and *finalTarget* area (Figure 10, red) are defined as a condition with the same name like the bloc named in TLD Logic. (Figure 10) These conditions are necessary for later modelling. For later defining of b10 (in chapter 3.4.1.6) we have also to model the blocs b1, b2, b4, b10. (Figure 12) The function c: access the current cycle second of the operating signal plan.

Figure 12: Example 1 bloc conditions – SUMO xml



```
    <condition id="b5" value="l1"/>
    <condition id="b6" value="l2"/>
    <condition id="b11" value="M1 = 1"/>

     <condition id="b1" value="min_Stage_1 >= c:"/>
     <condition id="b2" value="max_Stage_1 >= c:"/>
     <condition id="b4" value="t01 >= c:"/>
     <condition id="b10" value="t02 >= c:"/>
```

(abstract of Figure A)

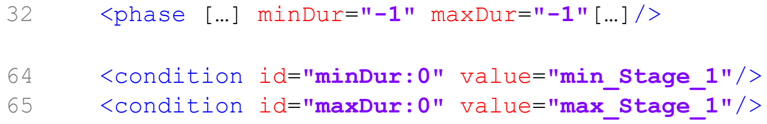
For *earlyTarget* the starting point is in general *minDur* and/or *earliestEnd***.** In our example it is *earliestEnd***,** so b4. In the example *earlyTarget* is checked when *minDur* and *earliestEnd* is reached. For *finalTarget* the starting point is in general *maxDur* or *latestEnd***.** In our example it is *maxDur* or *latestEnd* which has to be reached**,** so b2 or b10. Now every frame condition is described in general. In the following the detailed modelling of each frame condition will be described.

3.4.1.3 MinDur maxDur

The first values are the *minDur* and *maxDur*. In SUMO we could implement this in two ways. With a fixed value

```
<phase […] minDur="10"[…] maxDur="60"[…]/>
```

or we override the attributes with an expression



```
<phase […] minDur="-1" maxDur="-1"[…]/>

<condition id="minDur:0" value="min_Stage_1"/>
<condition id="maxDur:0" value="max_Stage_1"/>
```

(abstract of Figure A)

Even if the minimum duration of a phase is constant, it may be advantageous to override the phase attribute in order to collect all configuration parameters in a single place within the XML configuration. This makes it easier to copy and re-use a configuration for another controller program.

3.4.1.4 EarliestEnd latestEnd

For the *earliestEnd* and *latestEnd* the same override approach as described for *minDur* and *maxDur* may be used. But also, a fixed value defined directly in the phase definition works.

```
    <phase […] earliestEnd="-1" latestEnd="-1" […]/>

    <condition id="earliestEnd:0"    value="t01"/>
    <condition id="latestEnd:0"      value="t02"/>
```

(abstract of Figure A)

EarliestEnd and *latestEnd* will be always compare with currently cycle second.

3.4.1.5 Next

Attribute *Next* defines possible successor phases for a given phase. It is typically used in stages to declare the first phase for each possible interstage sequences that target a successor stage. However, in some controllers it is also possible to branch of into different interstages from within an interstage. It is possible to switch when any of the basic conditions ((*minDur* and/or *earliestEnd*) or *maxDur* or *latestEnd*) is reached. In the example we defined in *next* with phase indices 1 and 6 because these are the interstages/phases which transition after a switch to the stage 2 and stage 3.

```
    <phase name="stage1" […] next="1 6"/>
```

(abstract of Figure A)

These transitions are shown in the stage-sequence-diagram (Figure C). If a phase does not use attribute *next*, the signal plan switches to the subsequent phase in the phase list after reaching max duration (with a wraparound to 0 at the end). To define a signal plan with a single phase that is permanent green phase one could write the index of the phase itself in the *next* attribute or specify only a *minDur* but no *maxDur*.

3.4.1.6 EarlyTarget and finalTarget

In our controller there are two ways to switch out of the phase0/stage1. The first way is by adapting to traffic measurement (elapsed time since detection). Effectively, the phase is ended when a defined condition evaluates to 'true' (or non-0). In the example we could exit the phase earlier as soon as *minDur* and *erliestEnd* are reached, then Sumo will check by order of the values of *next* the listed phases. The attributes *earliestEnd* check the including conditions of the mentioned phases. If a condition is true the signal plan switches in this phase. If none of the conditions are true, the signal plan remain in the phase 0/stage1. For *earliestEnd* here are two possible paths.

path b5-b6-b9

```
   <phase […] name="INS1.2" earlyTarget="b5 and !b6 " […]/>
```

(abstract of Figure A)

and path b5-b6(f)-b7-b8

```
    <phase […] name="INS1.3" earlyTarget="b5 and b6" […]/>
```

(abstract of Figure A)

We have only to model the questions blocks from the paths because they will change something on the operating decision. Here the logic conditions (Figure 8) and the conditions about the blocs (Figure 12) are used to define the paths. The possible paths must be written in the attribute *earliestEnd* for each phase which is possible to reach for the signal program (INS1.2/phase 1 and INS1.3/phase 6), as shown above.
The path b5-b6(f)-b7-b8 has a special bloc b7 (Figure 10), in this bloc a variable is be written. It is not possible do it directly in the path. But when the path b5-b6(f)-b7-b8 is reached it is a unique path within the how logic, so we define an assignment to modelling that. If all blocs of the path get the right state, V will be assigned with the integer 12 and the path b5-b6(f)-b7-b8 is modelled completely.

```
    <condition id="V" value="0"/>

    <assignment id="V" check="b1 and b2 and b4 and b5 and !b6" value="12"/>
```

(abstract of Figure A)

It is also possible to model it when there are more than two paths for the *earliestTarget*.

When defining *latestEnd = cycle time - duration_interstage from the last phase*, then each cycle will last for the give cycle duration.

The second diagram path for leaving the state is via the *finalTarget* bloc. It is used when *maxDur* or *latestEnd* are reached. Here we have two possible pathways. The path begins at the starting element *maxDur* b2 and *latestEnd* b10. Sumo will check by order of the values of *next* Phase the attributes *finalTarget* and check the including conditions of the mentioned phases. If a condition is true the signal program switch to this phase. Here we have to define two paths to the interstage1.2:

path 1: b2-b3 path 2: b10-b11(f)-b13

```
    <phase […] name="INS1.2" finalTarget="b2 or !b11"/>
```

(abstract of Figure A)

and one path to the Interstage1.3:

path: b10-b11-b12

```
    <phase […] name="INS1.3" finalTarget="b10 and b11"/>
```

(abstract of Figure A)

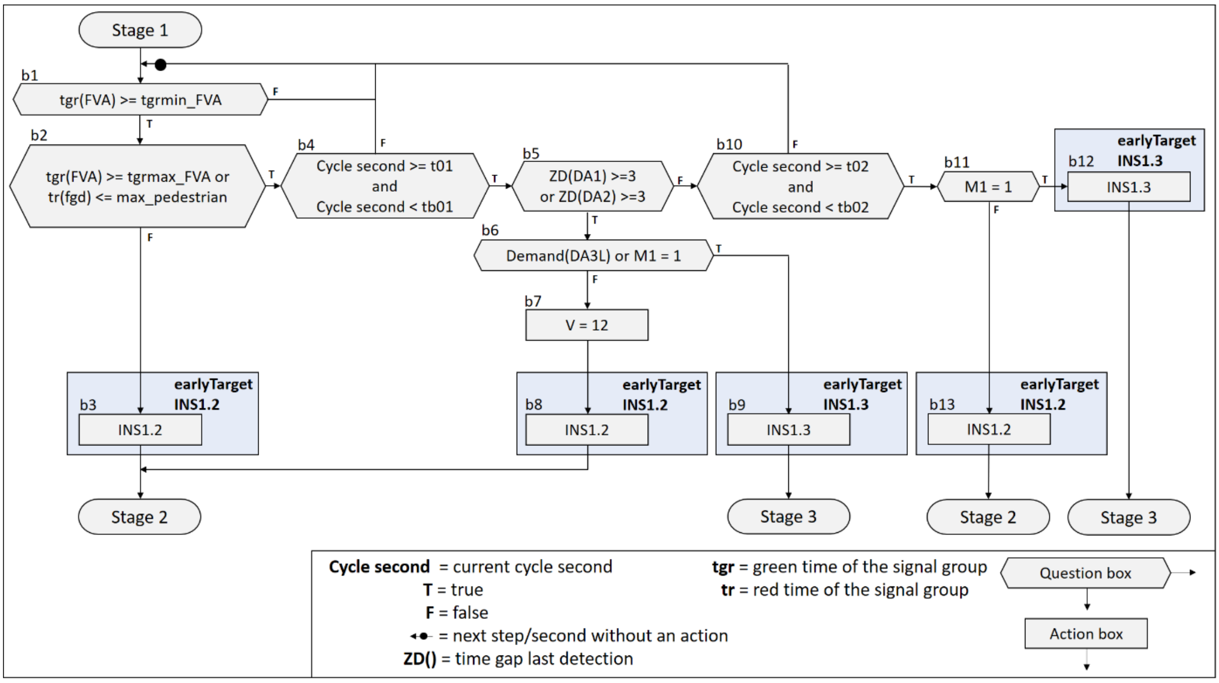
Here we also only model the question blocs for the same reason as mentioned. If none of the condition evaluates to true, SUMO switches to the last value of the next attribute as a fallback. If next is not defined, it switches to the next phase in definition order. In a typical TLD this should not occur and at least one of the path conditions should evaluate to true.

With this, every possible path in the logic is modeled and the SUMO controller can determine the switching conditions as described in the TLD.

3.4.2 Example with variable time bounds

For the second example we transcribe a TLD that does not express the time boundaries *minDur* (b1), *maxDur* (b2), *earliestEnd* (b4) and *latestEnd* (b10) with numerical constants (Figure 13). This means the XML definition cannot use attributes *minDur*, *maxDur*, *earliestEnd* and *latestEnd* as shown in the first example. In the following text we describe how to create the appropriate XML definitions.

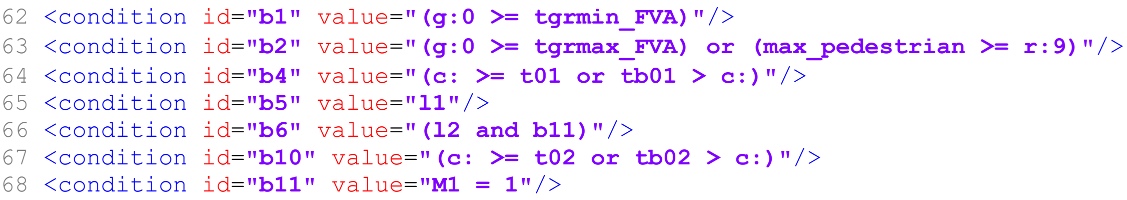
Figure 13: Example 2 - TLD logic


Stage 1
b1
tgr(FVA) >= tgrmin_FVA
F
T
b2
tgr(FVA) >= tgrmax_FVA or tr(fgd) <= max_pedestrian
T
F
b4
Cycle second >= t01 and Cycle second < tb01
F
T
b5
ZD(DA1) >=3 or ZD(DA2) >=3
F
T
b6
Demand(DA3L) or M1 = 1
T
F
b7
V = 12
b10
Cycle second >= t02 and Cycle second < tb02
F
T
b11
M1 = 1
T
F
b12
earlyTarget INS1.3
INS1.3
b3
earlyTarget INS1.2
INS1.2
b8
earlyTarget INS1.2
INS1.2
b9
earlyTarget INS1.3
INS1.3
b13
earlyTarget INS1.2
INS1.2
Stage 2
Stage 3
Stage 2
Stage 3
Cycle second = current cycle second
T = true
F = false
= next step/second without an action
ZD() = time gap last detection
tgr = green time of the signal group
tr = red time of the signal group
Question box
Action box

First of all, we define a condition for every question bloc with the same bloc name as in the TLD logic. (Figure 14)

Here we used two SUMO function which we want to describe shortly. Function **g:** accesses the current running green time of a link. function **r:** accesses the current running red time of a link. This could be used for example to define that a maximum waiting time for pedestrians is not exceeded.

Figure 14: Example 2 bloc conditions – SUMO xml



```
<condition id="b1" value="(g:0 >= tgrmin_FVA)"/>
<condition id="b2" value="(g:0 >= tgrmax_FVA) or (max_pedestrian >= r:9)"/>
<condition id="b4" value="(c: >= t01 or tb01 > c:)"/>
<condition id="b5" value="l1"/>
<condition id="b6" value="(l2 and b11)"/>
<condition id="b10" value="(c: >= t02 or tb02 > c:)"/>
<condition id="b11" value="M1 = 1"/>
```

(abstract of Figure B)

In our example in bloc b2 (Figure 13) expresses that max_pedestrian has to be smaller or equal than the red time of fgd/link 10, here we have to swap both expressions (from: tr(fgd) <= max_pedestrian to max_pedestrian >= tr(fgd)), because the traffic light definition is processed in a xml file, for that reason we have to pay attention with the possible syntax of xml. The XML standard prohibits use of the ‘<’ character within an attribute value. The simplest solution is to reverse the inequality and use the permitted

'>' character. A less readable alternative would be to use the xml code '&lt;' to encode the '<' character. These two options are possible ways to implement the specific condition:

```
<condition id="b2" value="(g:0 >= tgrmax_FVA) or (max_pedestrian >= r:9)"/>
```

(abstract of Figure B)

or

```
<condition id="b2" value="(g:0 >= tgrmax_FVA) or (r:9 &lt;= max_pedestrian)"/>
```

Then we define the pathways for reachable interstage.

```
<condition id="b3" value="(b1 and !b2)"/>
<condition id="b8" value="(b1 and b2 and b4 and b5 and !b6)"/>
<condition id="b9" value="(b1 and b2 and b4 and b5 and b6)"/>
<condition id="b13" value="(b1 and b2 and b4 and !b5 and b10 and b11)"/>
<condition id="b12" value="(b1 and b2 and b4 and !b5 and b10 and !b11)"/>
```

(abstract of Figure B)

The conditions were named like the interstage action bloc. Then we define conditions which will be used for the attribute *earlyTarget* of every reachable interstage

```
<condition id="earlyTarget_INS_1_2" value="b3 or b8 or b13"/>
<condition id="earlyTarget_INS_1_3" value="b9 or b12"/>
```

(abstract of Figure B)

Then every *earlyTarget* attribute is assigned the condition (defined above) which collects all paths by which this transition may be reached.

```
<phase […] name="INS1.2" earlyTarget="earlyTarget_INS_1_2"/>

<phase […] name="INS1.3" earlyTarget="earlyTarget_INS_1_3"/>
```

(abstract of Figure B)

While in stage 1, the conditions for each path are checked in every simulation step. If any of them evaluates to true (non-zero) the phase will switch. It works because every path is unique.

Effectively, all checks that were modelled by attributes *minDur, maxDur, earliestEnd, latestEnd* and *finalTarget* in the first example, were moved into the *earlyTarget* check. for that reason, minDur was set "0" and maxDur to a very high value (maxDur may also be omitted for the same effect).

```
<phase duration="99"[…] name="stage1" minDur="0" maxDur="1000000" […]/>
```

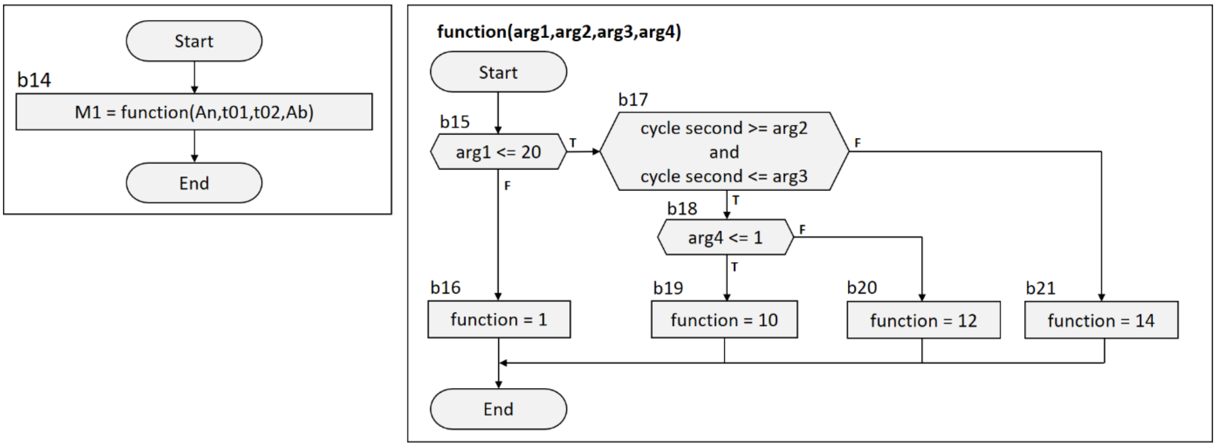
(abstract of Figure B)

In the example 1, the logic uses a special function which we describe below.

3.5 Function definition

In the algorithm description within the TLD logic, function definitions are often use to structure repeating code. In our example we use a function for data preparation of the public transport prioritization.

Figure 15: function – TLD logic



A function definition requires a name (*id*) and the number of input arguments (*nArgs*). (Figure 16, line 83) This is followed by assignments which model all possible paths through the logic diagram of the TLD function. In our example we named the assignments/path like the last action bloc in each path (Figure b16, b19, b20, b21). In the value description of every assignments the input arguments are defined with a **$** number argument. Then assignments for the output argument of the function are defined (Figure 16, line 88-91). The function now checks if a unique path is reached, depending on this a corresponding value is assigned to the function output **$0**. To call the function the expression *ID:arg1,arg2,…,argN* is used in a condition expression. (Figure 16, line 94) (Figure 15, b14). Note that there may be no spaces after a comma.

Figure 16: function – SUMO-xml

```
83   <function id="function" nArgs="4">
84     <assignment id="function_b16" check="1" value="!(20 >= $1)"/>
85     <assignment id="function_b20" check="1"
             value="(20 >= $1) and ((c: >= $2) and ($3 >= c:)) and !(1 >= $4)"/>
86     <assignment id="function_b21" check="1"
             value="(20 >= $1) and !((c: >= $2) and ($3 >= c:))"/>
87     <assignment id="function_b19" check="1"
            value="(20 >= $1) and ((c: >= $2) and ($3 >= c:)) and (1 >= $4)"/>
88     <assignment id="$0" check="function_b16" value="1"/>
89     <assignment id="$0" check="function_b20" value="10"/>
90     <assignment id="$0" check="function_b21" value="12"/>
91     <assignment id="$0" check="function_b19" value="14"/>
92   </function>
93
94   <condition id="M1" value="function:An,t01,t02,Ab"/>
```

(abstract of Figure B)

Assignment definitions placed outside a function definition are evaluated and executed in every simulation step where switching is possible. They are executed in the order in which they are defined. Likewise, in a function all assignments are executed in definition order every time the function is evaluated.

3.6 Tracking Phases

A useful mode for observing and analyzing the switching behavior of a traffic light is a diagram that shows signal and detector states over time. Such functionality is often part of the software suite used by commercial traffic light design software. The sumo-gui application which is part of the SUMO software package, provides the ‘Phase Tracker’ (Figure 17) dialog to display such a diagram. It was recently extended to optionally show internal controller variables along with detector states to aid in debugging controller operation. The dialog is accessed by right-clicking on a traffic signal and selecting the ‘track phases’ menu entry.

With the parameter key show-conditions, the list of observed expression can be customized.

```
    <param key="show-conditions" value="b5 b6"/>
```

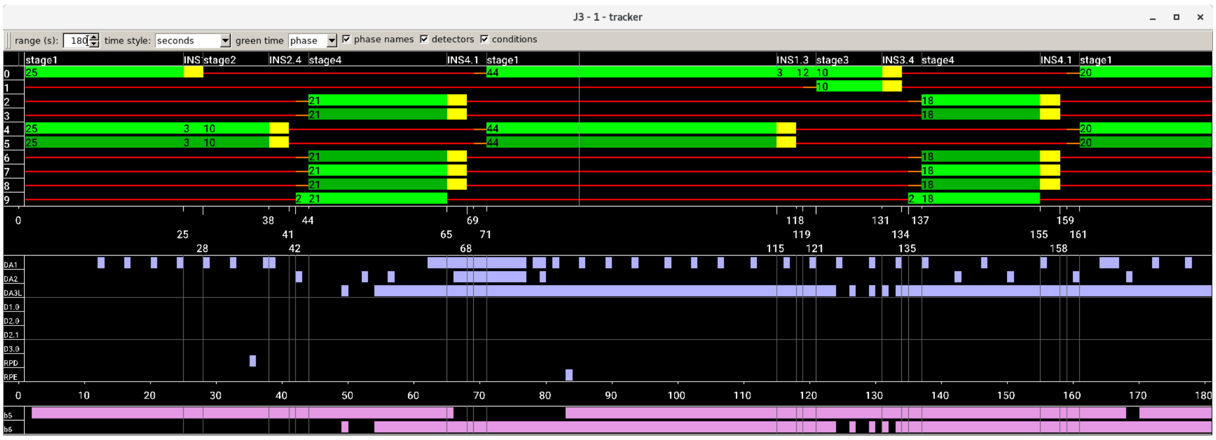
(abstract of Figure A / Figure B)

With the parameter key extra-detectors, it is possible to visualize in the tracking mode any additional detectors within the sumo simulation. In our example, this is used to track the state of additional induction loops used for the bus prioritization. It works for induction loops as well as laneAreaDetecors.

```
    <param key="extra-detectors" value="RPD RPE"/>
```

(abstract of Figure A / Figure B)

Figure 17: Tracking Phases:



J3 - 1 - tracker

range (s): 180 time style: seconds green time phase ☑ phase names ☑ detectors ☑ conditions

4 Replication study

4.1 Traffic light protocol

Real-world traffic lights may supply a second-by-second record of their signal states, detector occupation, and public transport telegrams.

Figure 18: Traffic light record:

```
2021-07-31-11-19-57 secOfDay:40796 lza:15 konr:156 diff:1836 values: 303 0 0 189 145 273 0 0 0 0 0 729 175 0 0 0 40 3630 3970 40
2021-07-31-11-19-58 secOfDay:40797 lza:15 konr:156 diff:1857 values: 403 0 0 190 145 273 0 0 0 0 0 729 175 0 0 0 40 3620 3960 40
2021-07-31-11-19-59 secOfDay:40798 lza:15 konr:156 diff:1775 values: 503 0 0 191 145 273 0 0 0 0 0 729 175 0 0 0 40 3610 3950 40
```

This protocol is specific to a particular controller software version and also to the control algorithm itself. With the aid of controller software documentation and the TLD it is possible to interpret all elements of the record and to write a semantically identical record based on the outputs of a SUMO simulation.

By preparing a simulation that replays the detector-states from a real-world controller record, and writing a corresponding simulation record, we can compare whether the simulated controller has the same switching behavior as the real-world controller for the recorded traffic situation.

4.2 TraCI Simulation

To simplify the replay-simulation, we have used the TraCI client functionality for setting artificial detector states. This avoids the need for creating vehicles which would trigger the detectors at the recorded times.

The TraCI function `inductionloop.overrideTimeSinceDetection` was used to replicate the exact detection times from the real-world recording.

Figure 19 example trigger a SUMO detector with TraCI (python)

```
traci.inductionloop.overrideTimeSinceDetection("DA1",0)   #Demand
traci.inductionloop.overrideTimeSinceDetection("DA1",-1)  #no Demand
```

In our project, the real-world intersection (located in Ingolstadt) also participates in a network wide control scheme. This means it is supplied externally with integer values that may change over time and which are used in the controller logic.

To emulate access to these variables, we have added virtual laneAreaDetectors in SUMO and supplied the values (x) from the real-world record via the function `traci.lanearea.overrideVehicleNumber`.

Figure 20: example trigger a SUMO laneAreaDetectors with TraCI (python)

```
traci.lanearea.overrideVehicleNumber("T6", X) #Value X
traci.lanearea.overrideVehicleNumber("T6",-1) #No Value
```

This way we can use the expression “a:DETECTOR_ID” to retrieve the recorded numerical values.

Figure 21: implementation to retrieve the recorded numerical values in the traffic light control

```
<condition id="T6" value="a:T6"/>
<assignment id="t06" check="a:T6 != 0 " value="T6"/>
```

Since the operating of the network wide control scheme has its own control logic based on parameters and detector states, it would have also been possible to replicate this logic within the custom switching rules. This, would make it necessary to include all detectors that participate in network control within the simulation.

4.3 Results

We simulated 60 minutes with the detector record data and compared the second-by-second sequence of stages and interstages from the record with the corresponding sequence from the simulation. A matching sequence indicates that the simulation traffic light in SUMO has the same behavior as the recorded real-world traffic light.

In our experiment the stages and interstages were reproduced with an accuracy of 96%. While analyzing the real-world record we observed errors such as gaps and duplicate time steps and this is likely a source of the disagreement between both records. And some synchronizations situations from outside/external are not completely reproduced.

Due to time constraints, we did not translate the logic modules for initializing the program at daybreak or for switching it off at nightfall. We also excluded the code for emergency vehicle prioritization. In our tests we therefore used a record sequence that did not feature these events.

Unfortunately, we also could not test the code for bus prioritization because some of the functions referenced in the TLD were not made available by the vendor of the controller software in time for this publication.[i] For this reason, our tests also excluded bus approaches.

Nevertheless, we are convinced that the omitted logic modules can be reproduced in SUMO as long as their behavioral description is available in full.

5 Conclusion

In prior versions of SUMO, it was only possible to model detailed adaptive traffic light control with an external TraCI client process and this route was only open to users with considerable experience in computer programming. With the configuration language presented in this work, it is possible to achieve a high-fidelity simulation given familiarity with traffic signal design documents and very limited programming knowledge.

Nevertheless, replicating the described toolchain for replaying a controller record and generating a corresponding simulation record still requires programming experience.

[i] GEVAS Software GmbH has graciously provided control flow diagrams for some of its functions already and we expect to replicate all features of bus prioritization eventually.

6 Abbreviations

b	logic bloc
bx(f)	bx is false
bx-by-bz	it defines a path of logic blocs
by	by is true
cycle second	the time within the current cycle (0 <= cycle second < cycle time)
cycle time	the duration of a full cycle
F	false
Fg	Pedestrian
FV	Individual motorized Traffic
INS	interstage
P1	Signal program 1 / Signal plan 1
T	true
tgr	green time of the signal group
TLD	traffic light document
tr	red time of the signal group
TraCI	Traffic Control Interface (a SUMO API)

7 Appendix

Figure A: Example 1 – SUMO-xml

```
<additional>
   <tlLogic id="J3" type="actuated" programID="1" offset="-5">
       <param key="coordinated" value="true"/>
       <param key="cycleTime" value="90"/>

       <param value="DA1" key="-E3_0" />
       <param value="DA2" key="-E3_1" />
       <param value="DA3L" key="-E3_2" />


       <param key="show-conditions" value="b5 b6"/>
       <param key="extra-detectors" value="RPD RPE"/>

       <!-- minal/maximal times-->
       <condition id="min_Stage_1"          value="10"/>
       <condition id="max_Stage_1"          value="60"/>
       <condition id="tgrmin_FVA"           value="35"/>
       <condition id="tgrmax_FVA"           value="32"/>
       <condition id="max_pedestrian"       value="75"/>

       <!--constants-->
       <condition id="k1" value="1"/>
       <condition id="k2" value="2"/>

       <!--time conditions-->
       <condition id="t01"  value="30"/>
       <condition id="t02"  value="65"/>
       <condition id="tb01" value="45"/>
       <condition id="tb02" value="80"/>

      <!—link index:               0123456789      -->
       <phase duration="99" state="GrrrGgrrrr" name="stage1" minDur="-1" maxDur="-1"
              earliestEnd="-1" latestEnd="-1" next="1 6"/>                        <!--0-->
       <phase duration="3" state="yrrrGgrrrr" name="INS1.2"
              earlyTarget="b5 and !b6" finalTarget="b2 or !b11"/>                 <!--1-->

       <phase duration="10"state="rrrrGgrrrr" name="stage2"/>                     <!--2-->

       <phase duration="3" state="rrrryyrrrr" name="INS2.4"/>                     <!--3-->
       <phase duration="1" state="rrrrrrrrrr" name="INS2.4"/>                     <!--4-->
       <phase duration="2" state="rruurruuuG" name="INS2.4" next = "13"/>         <!--5-->

       <phase duration="3" state="Grrryyrrrr" name="INS1.3"
              earlyTarget="b5 and b6" finalTarget="(b10 and b11)"/>               <!--6-->
       <phase duration="1" state="Grrrrrrrrr" name="INS1.3"/>                     <!--7-->
       <phase duration="2" state="Gurrrrrrrr" name="INS1.3"/>                     <!--8-->

       <phase duration="10"state="GGrrrrrrrr" name="stage3"/>                     <!--9-->

       <phase duration="3" state="yyrrrrrrrr" name="INS3.4"/>                     <!--10-->
       <phase duration="1" state="rrrrrrrrrr" name="INS3.4"/>                     <!--11-->
       <phase duration="2" state="rrurrruuuG" name="INS3.4"/>                     <!--12-->

       <phase duration="40" state="rrGgrrgGgG" name="stage4" minDur="0" maxDur="40"
              earliestEnd="70" latestEnd="84"/>                                   <!--13-->

       <phase duration="3" state="rryyrryyyr" name="INS4.1"/>                     <!--14-->
       <phase duration="1" state="rrrrrrrrrr" name="INS4.1"/>                    <!--15-->
       <phase duration="2" state="urrruurrrr" name="INS4.1"/>                    <!--16-->

       <!--logical condtion definition-->
       <condition id="l1" value="(z:DA1 >= 3) or (z:DA2 >= 3)"/>
       <condition id="An" value="z:RPD + k1"/>
       <condition id="Ab" value="z:RPE + k2"/>
       <condition id="l2" value="a:DA3L"/>

       <condition id="minDur:0" value="min_Stage_1"/>
       <condition id="maxDur:0" value="max_Stage_1"/>

       <condition id="earliestEnd:0"        value="t01"/>
       <condition id="latestEnd:0"          value="t02"/>

       <condition id="b5" value="l1"/>
       <condition id="b6" value="l2"/>
       <condition id="b11" value="M1 = 1"/>

       <condition id="b1" value="min_Stage_1 >= c:"/>
       <condition id="b2" value="max_Stage_1 >= c:"/>
       <condition id="b4" value="t01 >= c:"/>
       <condition id="b10" value="t02 >= c:"/>

       <condition id="V" value="0"/>

       <assignment id="V" check="b1 and b2 and b4 and b5 and !b6" value="12"/>

       <function id="function" nArgs="4">
              <assignment id="function_b16" check="1" value="!(20 >= $1)" />
              <assignment id="function_b20" check="1"
                      value="(20 >= $1) and ((c: >= $2) and ($3 >= c:)) and !(1 >= $4)"/>
              <assignment id="function_b21" check="1"
                      value="(20 >= $1) and !((c: >= $2) and ($3 >= c:))"/>
              <assignment id="function_b19" check="1"
                      value="(20 >= $1) and ((c: >= $2) and ($3 >= c:)) and (1 >= $4)"/>
              <assignment id="$0" check="function_b16" value="1"/>
              <assignment id="$0" check="function_b20" value="10"/>
              <assignment id="$0" check="function_b21" value="12"/>
              <assignment id="$0" check="function_b19" value="14"/>
       </function>

       <condition id="M1" value="function:An,t01,t02,Ab"/>

    </tlLogic>
 </additional>
```

Figure B: Example 2 – SUMO-xml

```
<additional>
   <tlLogic id="J3" type="actuated" programID="1" offset="-5">
      <param key="coordinated" value="true"/>
      <param key="cycleTime" value="90"/>

      <param value="DA1" key="-E3_0" />
      <param value="DA2" key="-E3_1" />
      <param value="DA3L" key="-E3_2" />


      <param key="show-conditions" value="b5 b6"/>
      <param key="extra-detectors" value="RPD RPE"/>

      <!-- minal/maximal times-->
      <condition id="tgrmin_FVA"           value="35"/>
      <condition id="tgrmax_FVA"           value="32"/>
      <condition id="max_pedestrian"       value="75"/>

      <!--constants-->
      <condition id="k1" value="1"/>
      <condition id="k2" value="2"/>

      <!--time conditions-->
      <condition id="t01"  value="30"/>
      <condition id="t02"  value="65"/>
      <condition id="tb01" value="45"/>
      <condition id="tb02" value="80"/>

     <!-- link index:            0123456789 -->
      <phase duration="99" state="GrrrGgrrrr" name="stage1" minDur="0" maxDur="1000000"
             next="1 6" />                                                           <!--0-->

      <phase duration="3" state="yrrrGgrrrr" name="INS1.2"
             earlyTarget="earlyTarget_INS_1_2"/>                                     <!--1-->

      <phase duration="10"state="rrrrGgrrrr" name="stage2"/>                         <!--2-->

      <phase duration="3" state="rrrryyrrrr" name="INS2.4"/>                         <!--3-->
      <phase duration="1" state="rrrrrrrrrr" name="INS2.4"/>                         <!--4-->
```

```
    <phase duration="2" state="rruurruuuG" name="INS2.4" next= "13"/>              <!--5-->

    <phase duration="3" state="Grrryyrrrr" name="INS1.3"
           earlyTarget="earlyTarget_INS_1_3"/>                                     <!--6-->
    <phase duration="1" state="Grrrrrrrrr" name="INS1.3"/>                         <!--7-->
    <phase duration="2" state="Gurrrrrrrr" name="INS1.3"/>                         <!--8-->

    <phase duration="10"state="GGrrrrrrrr" name="stage3"/>                         <!--9-->

    <phase duration="3" state="yyrrrrrrrr" name="INS3.4"/>                        <!--10-->
    <phase duration="1" state="rrrrrrrrrr" name="INS3.4"/>                        <!--11-->
    <phase duration="2" state="rrurrruuuG" name="INS3.4"/>                        <!--12-->

    <phase duration="40" state="rrGgrrgGgG" name="stage4" minDur="0" maxDur="40"
           earliestEnd="70" latestEnd="84"/>                                      <!--13-->

    <phase duration="3" state="rryyrryyyr" name="INS4.1"/>                        <!--14-->
    <phase duration="1" state="rrrrrrrrrr" name="INS4.1"/>                        <!--15-->
    <phase duration="2" state="urrruurrrr" name="INS4.1"/>                        <!--16-->

    <!--logical condition definition-->
    <condition id="l1" value="(z:DA1 >= 3) or (z:DA2 >= 3)"/>
    <condition id="An" value="z:RPD + k1"/>
    <condition id="Ab" value="z:RPE + k2"/>
    <condition id="l2" value="a:DA3L"/>

    <condition id="b1" value="(g:0  >= tgrmin_FVA)"/>
    <condition id="b2" value="(g:0 >= tgrmax_FVA) or (max_pedestrian >= r:9)"/>
    <condition id="b4" value="(c: >= t01 and tb01 > c:)"/>
    <condition id="b5" value="l1"/>
    <condition id="b6" value="(l2 and b11)"/>
    <condition id="b10" value="(c: >= t02 and tb02 > c:)"/>
    <condition id="b11" value="M1 = 1"/>

    <condition id="b3" value="(b1 and !b2)"/>
    <condition id="b8" value="(b1 and b2 and b4 and b5 and !b6)"/>
    <condition id="b9" value="(b1 and b2 and b4 and b5 and b6)"/>
    <condition id="b13" value="(b1 and b2 and b4 and !b5 and b10 and b11)"/>
    <condition id="b12" value="(b1 and b2 and b4 and !b5 and b10 and !b11)"/>

    <condition id="earlyTarget_INS_1_2" value="b3 or b8 or b13"/>
    <condition id="earlyTarget_INS_1_3" value="b9 or b12"/>

    <condition id="V" value="0"/>

    <assignment id="V" check="b1 and b2 and b4 and b5 and !b6" value="12"/>

    <function id="function" nArgs="4">
           <assignment id="function_b16" check="1" value="!(20 >= $1)" />
           <assignment id="function_b20" check="1"
                  value="(20 >= $1) and ((c: >= $2) and ($3 >= c:))and !(1 >= $4)/>
           <assignment id="function_b21" check="1"
                  value="(20 >= $1) and !((c: >= $2) and ($3 >= c:))"/>
           <assignment id="function_b19" check="1"
                  value="(20 >= $1) and ((c: >= $2) and ($3 >= c:)) and (1 >= $4)"/>
           <assignment id="$0" check="function_b16" value="1"/>
           <assignment id="$0" check="function_b20" value="10"/>
           <assignment id="$0" check="function_b21" value="12"/>
           <assignment id="$0" check="function_b19" value="14"/>
    </function>

    <condition id="M1" value="function:An,t01,t02,Ab"/>

  </tlLogic>
</additional>
```

Figure C: Stage following diagram – TLD

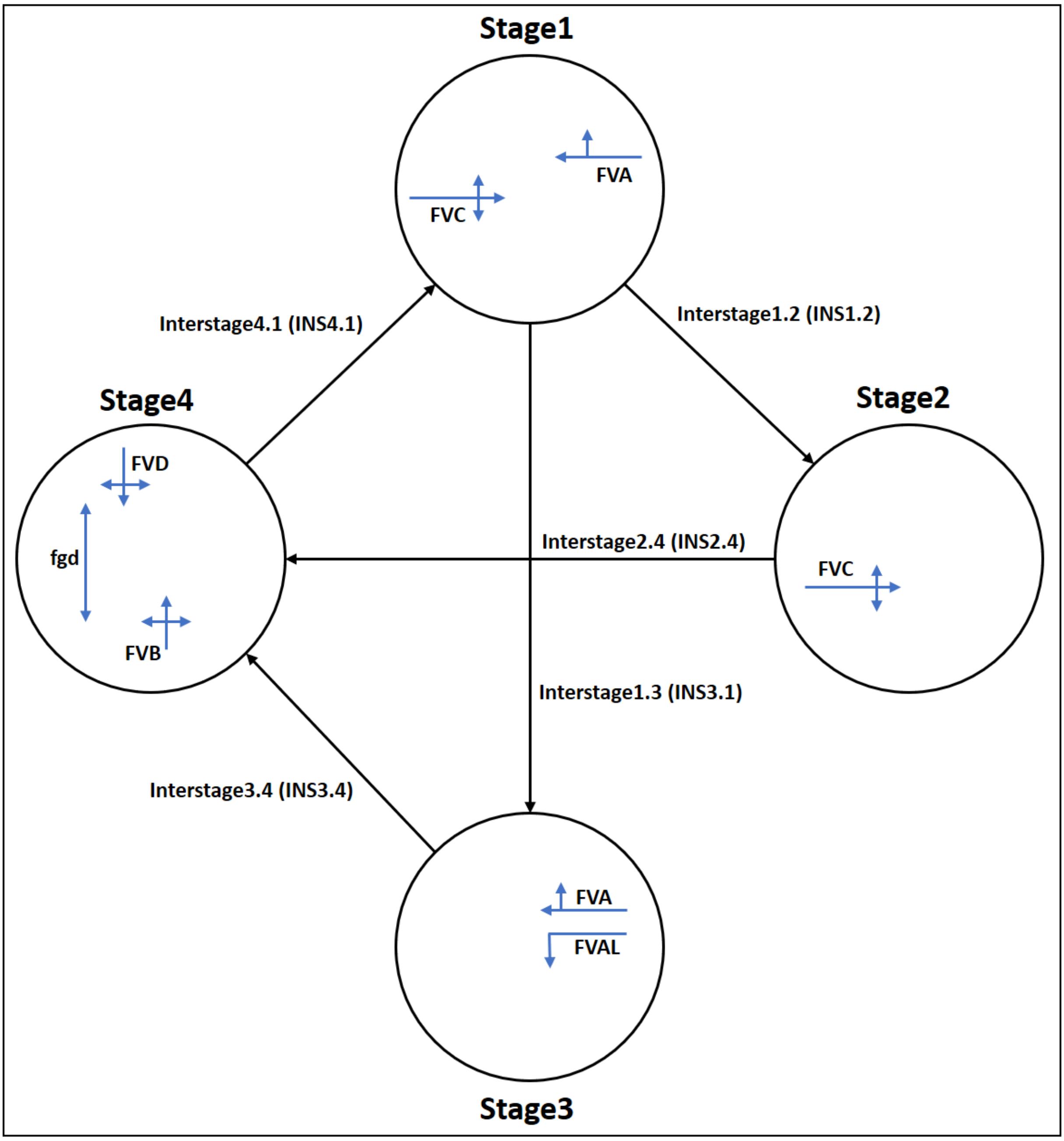


Figure D: confic.xml – SUMO

Note, that the detectors det.xml has always to be loaded bevor the traffic light configuration which references the detectors is loaded.

```
<configuration>

 <input>
   <net-file value="paper_net.net.xml"/>
   <route-files value="paper_routes.rou.xml"/>
   <additional-files value="paper_inductionloop.det.xml,paper_traffic_light_control.add.xml"/>
 </input>

</configuration>
```

8 References

[1] https://sumo.dlr.de/docs/TraCI.html#using_traci
Web page of German Aerospace Center (DLR), accessed 11.04.2022
[2] https://sumo.dlr.de/docs/TraCI.html#resources
Web page of German Aerospace Center (DLR), accessed 11.04.2022
[3] https://sumo.dlr.de/docs/Simulation/Traffic_Lights.html
Web page of German Aerospace Center (DLR), accessed 11.04.2022
[4] https://savenow.de/de/
Web page of SAVeNoW, accessed 11.04.2022
[5] https://github.com/DLR-TS/sumo-ocit
Source code repository for import of OCIT data files, accessed 11.04.2022

SUMO User Conference 2022
https://doi.org/10.52825/scp.v3i.165

Published: 29 Sept. 2022

Investigating the behaviors of cyclists and pedestrians under automated shuttle operation

Yun-Pang Flötteröd[1][https://orcid.org/0000-0003-3620-2715], Iman Pereira[2], Johan Olstam[2], and Laura Bieker-Walz[3]

[1] Institute of Transportation Systems, German Aerospace Center, Germany
[2] The Swedish National Road and Transport Research Institute, Sweden
[3] Dataport, Germany
Yun-Pang.Floetteroed@dlr.de, Iman.Pereira@vti.se, Johan.Olstam@vti.se,
Laura.Bieker-Walz@dataport.de

Abstract

To understand the influence of the automated shuttles on active modes as pedestrians and bicyclists, data was collected at the pilot site Linköping within the context of the European project SHOW, where AS provide regular transport service on the campus and run along a corridor restricted to bike and pedestrian traffic with pre-defined stops. Three types of data were collected, i.e. video data, shuttle data and traffic count with use of Telraam, while the first one was the main data source for analyzing VRU behaviors and the others were used for checking the validity of video data. The investigation mainly focused on VRU's space usage, speed, acceleration and lateral position and distance with and without AS presence. Bikes maneuvers, compatible with overtaking, were also examined. The analysis results can help for simulation model improvement.

1 Introduction

The aim to introduce automated shuttles (AS) into daily life is to extend and enhance mobility quality and services as well as improve user experiences. It is also expected to increase accessibility not only in the temporal and spatial respects, but also in the aspect of user groups. Accordingly, more and more demonstrations take place for examining the respective impacts for further improvement on AS planning and operation. In some of demonstrations, active modes as pedestrians and bicyclists share space with AS. Due to the current regulations in many European countries and requirements on safe operations on and/or from the manufactures most AS run at low speed. Under such condition, AS's performance and possible contribution on transport system are limited as well. Furthermore, the introduction of automated shuttles on bike paths might imply additional interactions and delays for bicycle traffic. To analyze the interactions and potential delays the movements can be recorded via camera systems with computer vision, generating trajectories. However, there is also a need to investigate how the effects depend on the penetration rates and possible operation speed of the AS.

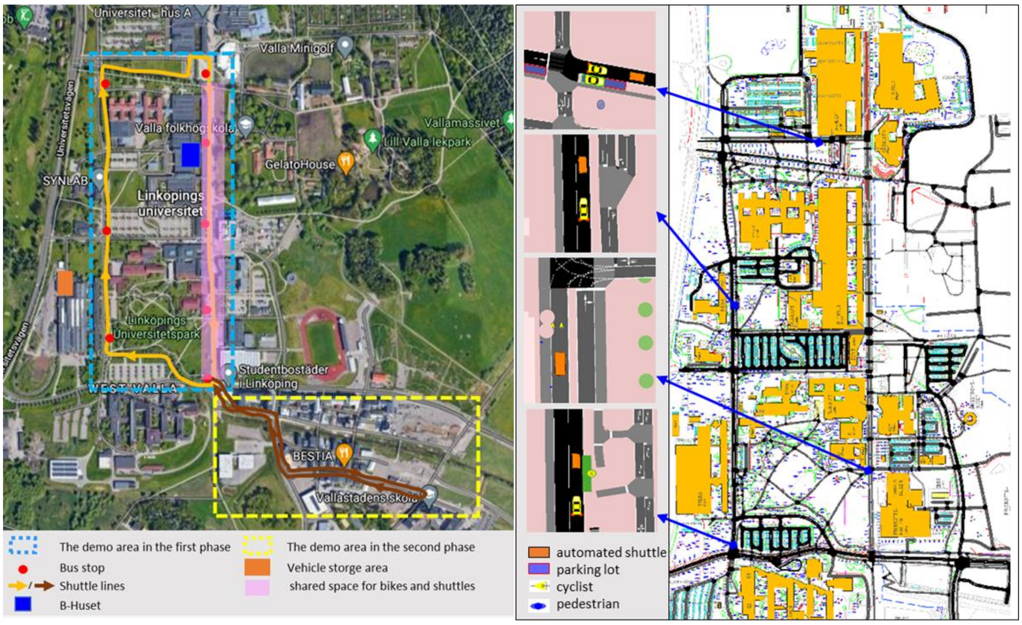
Traffic simulation is a suitable supplementary instrument for evaluating AS's impacts under different conditions given that models in traffic simulation can properly represent road users' behaviors. In this paper, extracted trajectory data from video data, collected in the shared space at the test site Linköping, was analyzed with the following aims

- to better understand the general influence of the AS introduction on VRU;
- to extract key pedestrian and bike related parameters/characters for enhancing simulation models, such as desired speed distributions for pedestrians and bikes, clearance distances between pedestrians, bikes and AS and maneuvers related to overtaking actions.

Apart from the video data, trajectory and operation data, collected by AS, and traffic count and speed data collected by Telraam [1] was also used for cross-checking the data validity and for gathering traffic demand for a longer time period for future upcoming traffic simulation experiments.

2 Test site Linköping

Test site Linköping is a part of the Swedish twin mega sites [2] within the European project SHOW [3]. The main objectives are to improve user experience and to provide a robust first and last mile solution to public transportation. The test site's overall layout is illustrated in Figure 1(a), and it is divided into two parts according to the demonstration phases. The first part is the university campus as surrounded by the blue dotted line. During the data collection period two AS ran in clockwise direction with maximum speed 14 km per hour, whilst there are three AS in operation currently. They serve 8 pre-defined locations, indicated as red dots. The planned AS schedule is 20 minutes. The second part is the adjacent residential area, as bounded by the yellow dotted line.



(a) Layout of the test site Linköping (b) Simulation network of the campus area

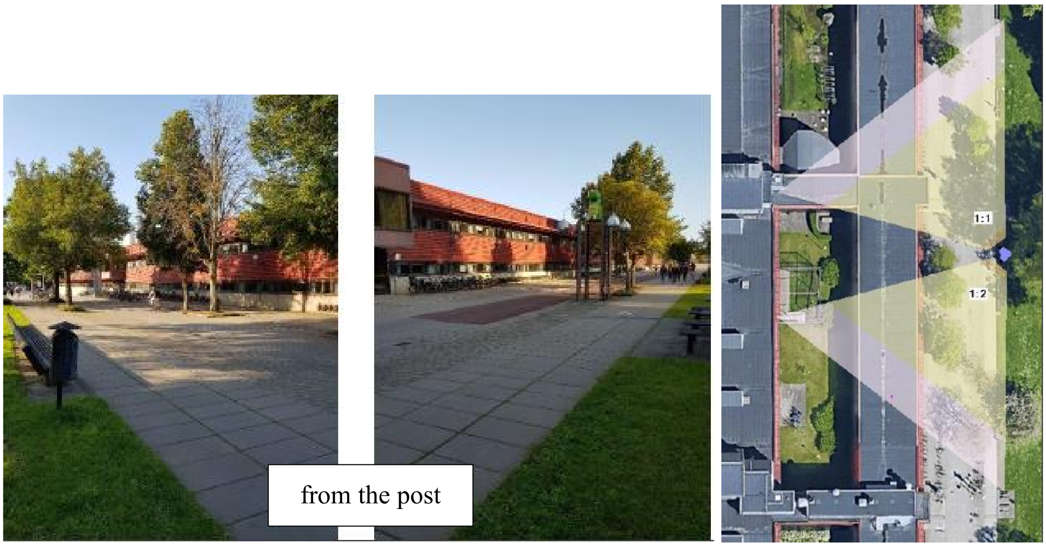
Figure 1: Overview of Test Site Linköping

The data analysis and the ongoing simulation work with SUMO [4] focus on the campus area (see Figure 1(b)) [5, 6, 7]. The analyzed video data was collected at the B-Huset in the shared space on the eastern side, marked in pink in Figure 1. In this area, cyclists and shuttles share the bike path, located in the middle of the space, and pedestrians can cross the bike path anytime if necessary. All intersections on campus are priority-controlled intersections.

3 Data processing and verification

Data was collected from three sources, i.e. video-camera based measurement system from Viscando, Telraam and log data from the AS, and is briefly explained below.

- Shuttle data: it consists of the AS trajectory information, i.e. timestamp, position expressed in longitude and latitude and speed, and the operational data, such as status of the vehicle door, battery level, load information, operation mode (manual or automatic), etc. The latter one is irrelevant to this study scope.
- Telraam data: it contains the number of passages per type over a cross-section at B-Huset in the shared space. In addition, the counts are given together with an estimate of the speed and the direction of which the object is moving.
- Viscando data: the measuring site was chosen in the middle of the shared space corridor with the consideration of the road infrastructure and the locations of lamp posts. Two OTUS3D cameras from Viscando were deployed. Figure 2 and Figure 3 show the space layout and the coverage overview from the video cameras.



Source: [8] and Google Map (right)

Figure 2: Space layout of the measuring site

Data from shuttles and the OTUS3D cameras were available for the entire study period from the 20th of September 10:00 AM to the 26th of September. As the Telraam counter was set up later, data from the counter is available from the 22nd of September 12:00 AM to the 26th of September. Since the shuttles did not operate during weekends and evenings, the study period is from the 22nd to the 24th of September between 8:00 and 18:00.

The OTUS3D cameras captured all road users in the area with the highest attention to detail. Thus, this is considered to be the main data source used for analysis. Both the shuttle data and the Telraam data were mainly used for verification purposes as both data sources were very limited for the purpose of studying variations in traffic performance of bicyclists with and without shuttle presence.

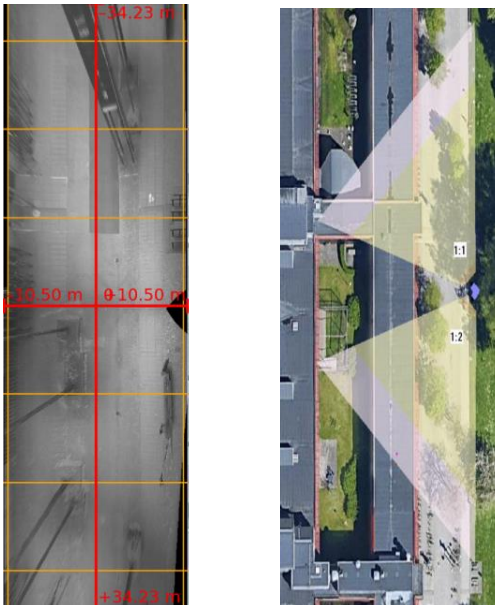
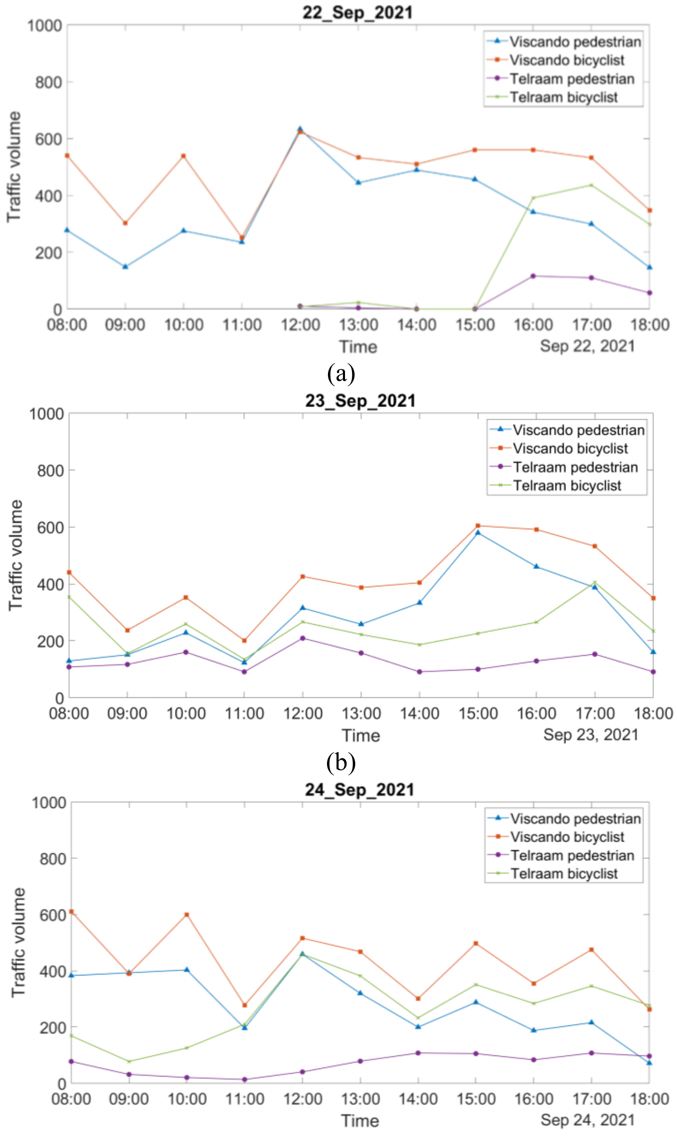


Figure 3: Coverage overview of the measuring site from the video cameras **[8]**

Furthermore, the OTUS3D camera data was processed and cleaned due to some misclassifications (only for AS), ghost trajectories, trajectory fragmentation with temporal/spatial jumps and short trajectories. All trajectories shorter than 9 meters were excluded. Pedestrian trajectories with space jump larger than 3 meters, and bicycle trajectories with space jump larger than 6 meters were also excluded. Lastly, trajectories with inconsistencies in time sampling rates were split if time jump exceeded 3 seconds. In the end, data was split into two groups: (1) Data set 1 – Trajectory data when a shuttle was present, and (2) Data set 2 – Trajectory data when no shuttle was present.

The time periods with shuttle presence were relatively short, causing an imbalance in the amount of data between the two datasets. Also, the shuttles were not always detected correctly, and the corresponding amount was underestimated (10% - 40% daily in the whole study period) when comparing with the ground truth shuttle data. In the end, the data set with shuttle presence is naturally smaller and constitutes 2% of the entire data set. Approximately 15% and 11% of the data in the data sets 1 and 2 were not used for analysis respectively.

In addition, the flow comparison between Viscando data and Telraam data was also carried out. Figure 4 (a), (b) and (c) show that the Telraam counter consistently underestimated the traffic volume in comparison to the Viscando system with regards to pedestrians and bicyclists. It implies that Telraam counter, which is developed to count passages over a cross section, seems to have difficulty to handle traffic counting in a shared space where objects move more freely than those on normal roads. However both systems show that traffic peaks appeared around 12:00 PM, and close to 17:00 PM, which coinsided with lunch break and the last lecture given at the university.



(a)

(b)

(c)

Figure 4: Hourly traffic volumes detected by Viscando and Telraam

4 Data analysis

4.1 Space usage

As mentioned in Section 2, the shared space was originally designed to be used by pedestrians and cyclists, and has then also been used by AS since 2020, where cyclists and AS share the bike path. Figure 5 shows the collected trajectories by type and direction when AS were present. The directions of pedestrian and cyclist flows were mainly from north to south and from south to north, whilst the AS ran only from north to south. It also shows that both pedestrian and cyclists mainly used their respective designated paths, but did deviate from these to some extent using each other's paths as well.

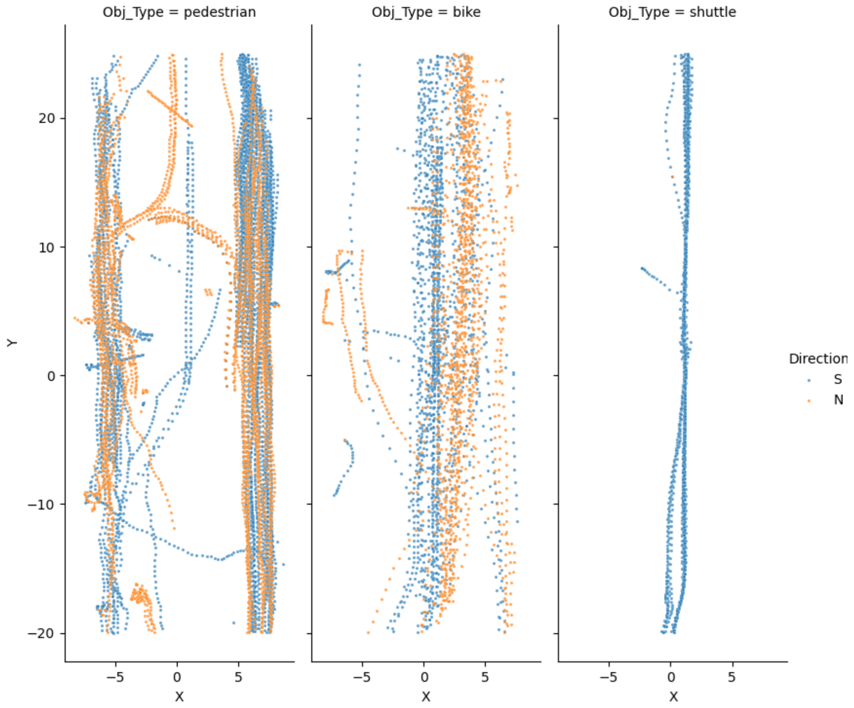


Figure 5: Object trajectories when shuttle was present

The usage of bike path with and without the AS presence was further analzed. The result in Table 1 shows that AS ran 99.5% within the bike path as expected, and their paths deviated from the pre-defined path sometimes due to unexpected events, which can also be oberseved from their trajectories in Figure 5. The pedestrians tended not to use the bike path even when the AS were not present. The percentage of pedestrains using the bike path slightly decreased from 7.2% to 5.3% when the AS were present. In comparison to that, the decreasing rate for cyclists reached around 20%, while 68% of cyclists used the bike path without AS presence. It indicates that cyclists were those directly affected by the AS introduction. Pedestrians were also affected due to that cyclists use then more often the sidewalks with the AS presence.

X position (m)	Type	with shuttle presence	without shuttle presence
-2 <= x <= 2 (within bike path)	AS	99.5%	-
	pedestrian	5.3%	7.2%
	bike	47.0%	68.2%

Table 1: Changes in space usage with and without shuttle presence

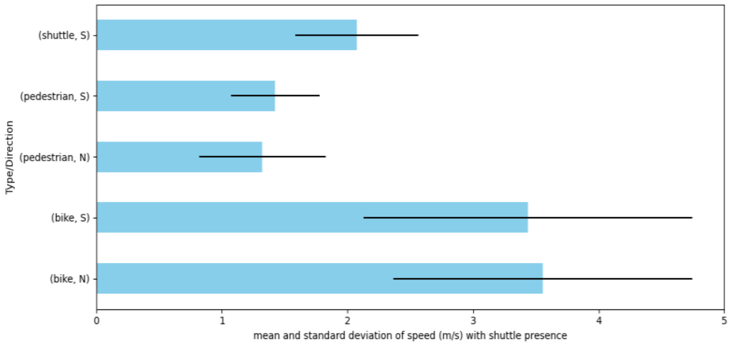
4.2 Speed and acceleration

To understand the influence of the AS introduction on the motions of pedestrians and bikes, mean speed, mean acceleration and the respective standard deviations for each object type were analyzed. The result is summarized in Table 2 and illustrated in Figure 6. It shows that the mean speed of bikes was slightly higher when traversing through the study area in the same direction as the shuttle in comparison to that when bikes were moving in the opposite direction of the shuttle. This is not entirely unexpected as there is a slight slope in the north part of the study area. Therefore, bikes entering the study area from the north might have gained a little speed. It also shows that bikes tended to slow down in both directions when the shuttles were present. However, such slow-down is not statistically significant according to the t-test result with a significance level of 0.05.

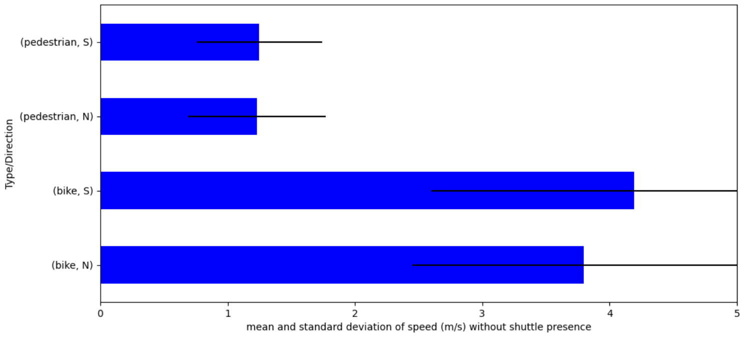
AS presence	type	Southbound (AS's running direction)				Northbound			
		mean speed	speed s.d.*	mean acc.*	acc s.d.*	mean speed	speed s.d.*	mean acc.*	acc s.d.*
Yes	ped.	1.4244	0.3526	-0.0003	0.3935	1.3231	0.5057	0.0139	0.4541
	bike	3.4367	1.3088	0.0149	0.8962	3.5557	1.1880	0.0624	0.8669
	AS	2.0756	0.4902	-0.0247	0.6269	-	-	-	-
No	ped.	1.2503	0.4911	0.0022	0.4044	1.2306	0.5394	-0.0016	0.6261
	bike	4.1919	1.5903	0.0077	0.9128	3.7995	1.3492	0.0386	0.7294

*: s.d.: standard deviation; acc: acceleration

Table 2: Mean speed and acceleration by type and direction with and without AS presence



(a) with shuttle presence



(b) without shuttle presence

Figure 6: Mean speed and standard deviation per type and direction with and without AS presence

Regarding pedestrians' speed it seems that there was a small tendency for pedestrians to walk slightly faster when the shuttle was present. According to the t-test the mean speed difference is statistically significant at a confidence level of 0.05. It could potentially mean that pedestrians got a sense of urgency when the shuttle was present and therefore walked slightly faster. Moreover, there were substantially fewer crossings while the shuttle was present. Thus, another possible reason could be that speeds during straight walking are faster than those during crossing. In any case it is hard to know exactly why the data shows this unexpected result and would need deeper investigations.

When looking at the accelerations both pedestrians' and bikes' mean accelerations were around 0 m/s^2 either with or without the AS presence, and no statistically significant difference exists between the mean accelerations with and without the AS presence for both object types.

The illustration in Figure 7 gives a clear overview about the speed-acceleration relationship of each object type when AS were present. AS's speed spectrum was similar to the pedestrians' speed spectrum (between 0 and 3 m/s), whilst bikes' speed spectrum was relatively wider. The acceleration spectrum was mainly between 1 and -1 m/s^2 for all object types. A few of pedestrians' accelerations were close to 2 or -2 m/s^2. It could be due to the misclassification or measurement errors.

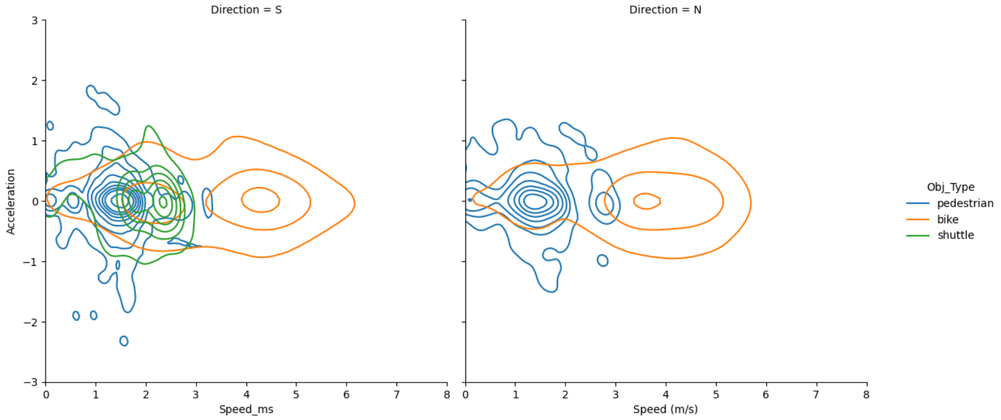
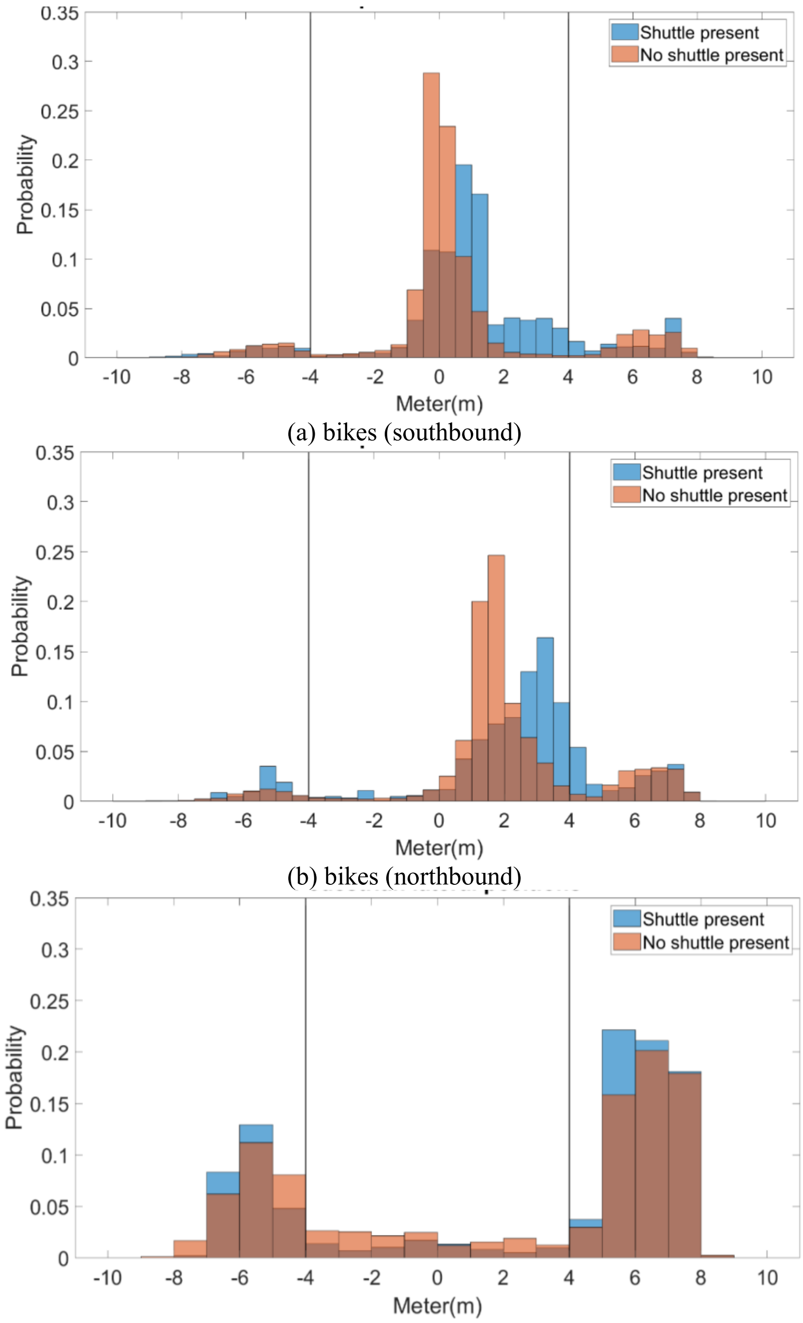


Figure 7: Relationship between speed and acceleration per type and direction with the AS presence

4.3 Lateral position and distance

Figure 8 (a) and (b) show lateral positions of bikes and pedestrians for the case when a shuttle is present and the case when no shuttle is present. Both figures show that bi-directional bicycle traffic shifted to the same direction, i.e. to the east of the shuttle, when the shuttle was present. It seems as the sidewalk shift was larger for the southbound bikes than for the northbound bikes. In addition, it seems that bikes travelling in the opposite direction of the shuttle tended to encroach the sidewalks more than those travelling in the shuttle's running direction. On the contrary, pedestrians still mostly walked or stood on the pedestrian paths although some pedestrian activities occurred on the bicycle paths as well. There is not much of a difference in the two studied situations although there seemed to be a slight increase in the probability to choose the pedestrian paths when the shuttle was present. These findings correspond to the result shown in Section 4.1 space usage.

Moreover, the lateral distances in relation to the longitudinal distances between objects and AS were also examined. Figure 9 depicts that pedestrians mostly kept at least a lateral distance 3 m away from



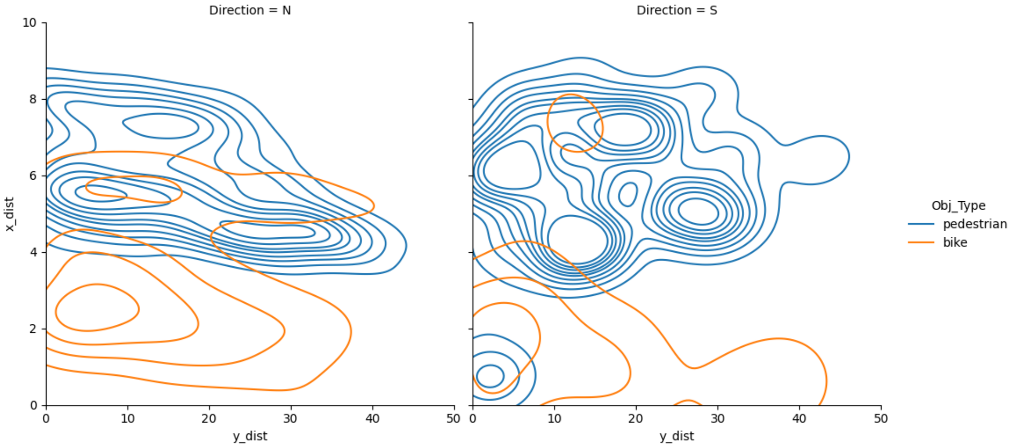
(a) bikes (southbound)

(b) bikes (northbound)

(c) pedestrians (both directions)

Figure 8: Bikes' and pedestrians' lateral positions with the AS presence

the shuttles' x positions in the shuttles' running direction. When the longitudinal distance between pedestrians and shuttles was less than 10 m, the respective lateral distance increased to mostly more than 4 m. In the opposite northbound direction, the minimum lateral distance was around 3.8 m, and it increased to more than 4 m with a longitudinal distance less than 10 m. In comparison to that, the lateral distances of the southbound bikes varied between 0.4 m to 4.3 m within 10 m longitudinal distance, and tended to decrease when the longitudinal distance increased, i.e. cyclists tended to move back to the bike path when they are getting far away from the shuttles. In addition, it seems that bikes' lateral distances to the shuttles' x positions varied quite large even when the respective longitudinal distance less than 10 m. The minimum lateral distance within 10 m longitudinal distance was around 0.8 m. In addition, there is a tendency for cyclists to decrease their lateral gaps when their longitudinal gaps increase in the southbound direction, but not in the northbound direction.



*: x_dist: lateral gap (m) between object's and shuttle's x-positions; y_dist: longitudinal gap (m) between object's and shuttle's y-positions

Figure 9: Lateral distances in relation to the longitudinal distances between the objects and the AS

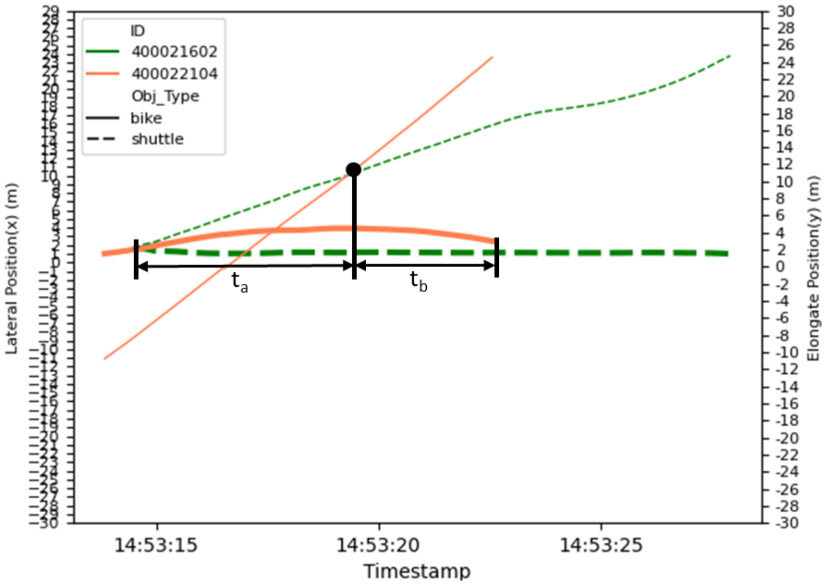
4.4 Interactions between objects and shuttles

Interactions between objects and shuttles cannot be observed from extracted trajectory data, only the maneuvers resulting from cyclists/pedestrians intending to take. Accordingly, only the corresponding action points, not decision points, can be discovered. Some interactive maneuvers, compatible with yielding, following, and overtaking, were observed according to the respective time-space diagrams. Such interactive maneuvers appeared quite rarely in the study area during the whole data collection period. Most of them were between bikes and shuttles, since pedestrians mainly used sidewalks. The mainly identified maneuvers were the maneuvers related to overtaking and conflict avoiding in the southbound and northbound directions respectively. Accordingly, the required time distribution for such maneuvers can be derived from the corresponding time-space diagrams.

A. Overtaking maneuvers

The concept to derive the afore mentioned duration distribution is to firstly identify the object candidates which fulfill the pre-defined criteria corresponding to an overtaking maneuver. The proposed criteria consist of (1) an object catches and passes the respective shuttle at a certain time point within the same time window, i.e. there is a cross point in the respective y-positions-based time-space diagram; (2) the x-position of the object in (1) gets closer and closer to the shuttle's x-position over time after catching the shuttle; and (3) the x-position of the object in (1) begins to get farther away from its previous x positions and the shuttle's x-position at some certain time point before catching the shuttle. After that, the time ($T_{overtake}$) spent for the whole overtaking-related maneuver is divided into two parts:

- t_a: duration between the point when a bike catches up with a shuttle, i.e. reaching the same y position, and the point when this bike begins to deviate its path from the shuttle path (see Figure 10).
- t_b: duration between the point when a bike catches up with a shuttle, i.e. reaching the same y position, and the point when this bike moves back to the bike path (see Figure 10).



*: fat dash line: shuttle's x positions; thin dash line: shuttle's y positions; fat solid line: bike's x positions; thin solid line: bike's y positions.

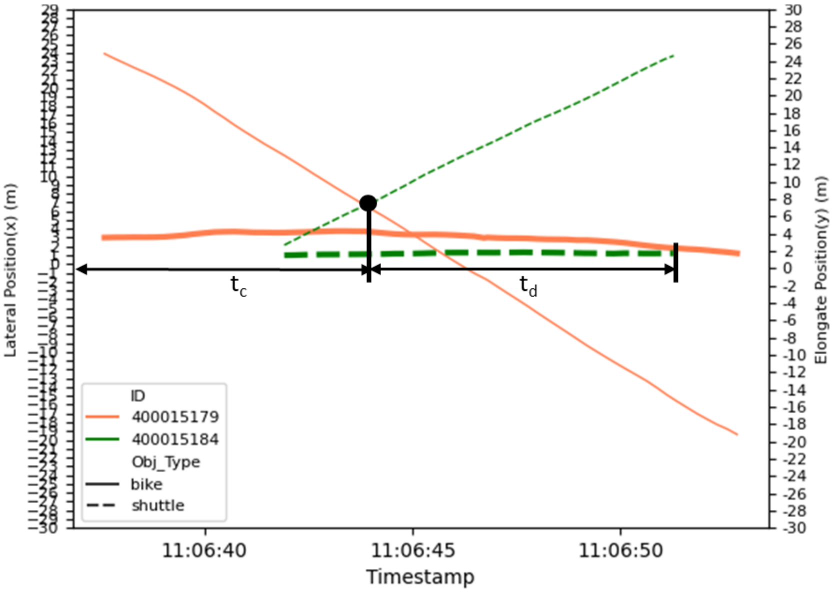
Figure 10: Time-space diagram of the exemplary objects with overtaking maneuver

B. Maneuvers to avoid conflicts

Following the similar concept in Section A objects are selected as candidates when their maneuvers correspond the following characters: (1) An object is coming towards the shuttle running in the opposite position; (2) The object in (1) deviates his/her path from the bike path before meeting the shuttle (move-out); and (3) The x-position of the object in (1) gets closer and closer back to the bike path, i.e. shuttle's x-position) over time after meeting the shuttle (move-back). The time (T_{avoid}) spent for the whole maneuver is also divided into two parts, just like $T_{overtake}$:

- t_c: duration between the point when a bike meets a shuttle at the same y position and the point when this bike begins to deviate it path from the shuttle path (see Figure 11).
- t_d: duration between the point when a bike meets a shuttle at the same y position and the point when this bike moves back to the bike path (see Figure 11).

The reason for the time separation is due to incomplete trajectory data or/and trajectory fragmentation, which can be seen in Figure 11 as example. Moreover, data interpolation will be applied as well in order to get more samples. Accordingly, the respective duration distributions can be derived. The implementation work is currently undertaken.



*: fat dash line: shuttle's x positions; thin dash line: shuttle's y positions; fat solid line: bike's x positions; thin solid line: bike's y positions.

Figure 11: Time-space diagram of the exemplary objects for avoiding conflicts

5 Conclusion and perspective

Video-camera based measurements has been used for traffic data collection since years due to its efficiency and effectiveness for temporally and spatially collecting large amount of road user movements. Various processing methods have been developed and commercialized. With the consideration of data protection respective image resolution is normally limited and it can result in some imprecision in data extraction especially when other conditions are not adequate, such as light, visibility, monitoring position and the complexity of movements. In this study, some imprecise data exists in the extracted trajectory data, and it could be resulted from several situations, e.g. (short) incomplete trajectories du to shuttles blocked the objects behind, double counting due to shadow effect, misclassification due to that pedestrians walked with bikes or they were too close to each other. According to the comparison result with the ground truth shuttle data approximately 70% of the shuttle passages were captured correctly. Moreover, the analysis of pedestrian and bicycle trajectories resulted in an error between approximately 11% and 14% depending on the dataset. Since there are only anonymized videos available for pedestrians and bicyclists, it is harder to correctly identify all the errors. It is assumed that uncertainties in the data still exist after data filtering. In addition, Telraam counter seems to have difficulty to properly count the passages of each object type in a shared area where pedestrians and bikes can move more freely than standard roads.

According to the analysis results bikes were the main objects directly affected by the AS introduction. They tended to encroach sidewalks when the shuttles were present. Pedestrians tended to continue using sidewalks and were possibly then affected by the changed movements of the bikes accordingly. The shuttle presence did not statistically significantly affect the acceleration behaviors of either pedestrians or bikes. However, the mean speeds of both road users were slightly affected by the shuttle presence. Bikes tended to traverse the area with a lower mean speed while the shuttle was present. The possible reasons could be that bicycles got hindered by the shuttles operating on the bike path or/and cyclists were not being able to travel at their desired speed due to the reduction in available space (with AS presence). In contrast to the bikes, there was an increase in pedestrians' mean speed with statistical significance when the shuttles were present. Despite of the result of statistical significance the amount of speed increase is very limited (0.17 m/s southbound and 0.09 m/s northbound). Together with the consideration of (1) no difference in acceleration behavior and (2) little difference in lateral positioning it would be difficult to observe or feel a real speed difference while walking throughout the area. Further investigation is then needed. Moreover, interactive maneuvers between objects and shuttles were examined with use of time-space diagrams. Only quite limited maneuvers, compatible with yielding, following and overtaking, were observed, while the latter one occurred more often.

In this paper, the focus puts on if there is any influence on the selected performance indicators (speed, acceleration, space usage) with the AS introduction. A concept to derive the duration distributions for the time spent for overtaking-related maneuvers is proposed. The respective implementation work is undertaken. In the next step, the interactions between pedestrians and bikes under the AS presence will be investigated. Moreover, several parameters, such as speed/acceleration distributions, durations for overtaking-related maneuvers will be derived and comparisons with simulated data will be carried out for examining and enhancing the respective microscopic traffic modelling. If the data is sufficient, the focus will be further put on the use of Bayesian inference to model the decision of the maneuver compatible with overtaking.

6 Acknowledgement

The research presented in this paper is funded by the SHOW project under the H2020 Research and Innovation Programme with Grant Agreement No. 875530.

References

[1] Telraam, "Telraam," [Online]. Available: https://telraam.net/.

[2] European project SHOW, "Mega sites - Sweden," [Online]. Available: https://show-project.eu/mega-sites-sweden/.

[3] European project SHOW, "SHared automation Operating models for Worldwide adoption," [Online]. Available: https://show-project.eu/.

[4] P. Alvarez 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," *IEEE Intelligent Transportation Systems Conference (ITSC),* 2018.

[5] Y.-P. Flötteröd, L. Bieker-Walz and J. Olstam, "Towards safe and efficient shared-space oriented DRT Service – some insights with real case study in Linköping," in *The proceedings of the ITS World Congress 2021*, Hamburg, 2021.

[6] L. Töttel, J. Hillebrand, M. Hartmann, C. Katrakazas, C. Rudloff and Y.-P. Flötteröd, "SHOW Deliverable 10.1: Simulation scenarios and tools," 2020.

[7] C. Katrakazas, J. Hillebrand, Y.-P. Flötteröd, D. Krajzewicz, L. Bieker-Walz, L. Xiao, M. van der Tuin, M. Hartmann, M. Schratter, M. Sekadakis, M. Oikonomou, C. Rudloff, L. Töttel, E. Mintsis, K. Touliou, K. Porfyri, D. Hauch, M. Berglund, J. Olstam, S. Oguz Kagan Capkin and S. Koskinen, "SHOW Deliverable 10.2: Pilot guiding simulation results," 2021.

[8] I. Pereira and J. Olstam, "Analysis of bicycle traffic performance when automated shuttles uses bike paths," Swedish National Road and Transport Research Institute (VTI), Working paper, VTI, Linköping, 2022.

SUMO User Conference 2022
https://doi.org/10.52825/scp.v3i.113

Published: 29 Sept. 2022

SUMO4AV: An Environment to Simulate Scenarios for Shared Autonomous Vehicle Fleets with SUMO Based on OpenStreetMap Data

Reichsöllner E.[1], Freymann A.[2], Sonntag M.[1], and Trautwein I.[2]

[1] University of Applied Sciences Esslingen, Germany
emanuel.reichsoellner@hs-esslingen.de,
mirko.sonntag@hs-esslingen.de
[2] Fraunhofer IAO, Stuttgart, Germany

andreas.freymann@iao.fraunhofer.de, ingo.trautwein@iao.fraunhofer.de

Abstract

In the past years the progress in the development of autonomous vehicles has increased tremendously. There are still technical, infrastructural and regulative obstacles which need to be overcome. However, there is a clear consent among experts that fully autonomous vehicles (level 5 of driving automation) will become reality in the coming years or at least in the coming decades. When fully autonomous vehicles are widely available for a fair trip price and when they can easily be utilized, a big shift from privately owned cars to carsharing will happen. On the one hand, this shift can bring a lot of chances for cities like the need of less parking space. But on the other hand, there is the risk of an increased traffic when walking or biking trips are substituted by trips with shared autonomous vehicle fleets. While the expected social, ecological and economical impact of widely used shared autonomous vehicle fleets is tremendous, there are hardly any sci-entific studies or data available for the effects on cities and municipalities. The research project *KI4ROBOFLEET* addressed this demand. A result of the project was *SUMO4AV*, a simulation environment for shared autonomous vehicle fleets, which we present in this paper. This simulation tool is based on *SUMO*, an open-source traffic simulation package. *SUMO4AV* can support city planners and carsharing companies to evaluate the chances and risks of running shared autonomous fleets in their local environment with their specific infrastructure. At its core it comprises the mapping of *OpenStreetMap*[1] entities into *SUMO* objects as well as a *Scenario Builder* to create different operation scenarios for autonomous driving. Additionally, the simulation tool offers a recursive execution with different fleet sizes and optimization strategies evaluated by economic and ecologic parameters. As eval-uation of the toolset a simulation of an ordinary scenario was performed. The workflow to simulate the scenario for shared autonomous vehicle fleets was successfully processed with the *SUMO4AV* environment.

1 Introduction

Within the last decades, the automotive industry has enhanced their development with many automation levels ending up in autonomous vehicles (AVs) [12]. Today that field has got a wide interest and is further expanding. The German parliament has recently passed a regulation that facilitates the operation of autonomous driving in February 2022 [3]. However, critical topics such as safety or trust in autonomous systems still needs to be further improved [12]. It is expected that in future shared autonomous vehicles (*SAVs*) will be one building block of

[1] Url: https://www.openstreetmap.org/

mobility in cities. Thus, the definition of autonomous systems and scenarios and how they can positively influence the mobility for individuals and the society need to be further investigated [12]. One challenge is the lack of scientific data about the operation of shared autonomous vehicle fleets (*SAV fleets*). The simulation of AVs is one method to obtain realistic data. It helps to get deeper insights into influence factors in a very complex setup, e.g., fleet sizing, pricing, or changes in the individual mobility behavior [13].

To address this issue, we developed a toolset for the simulation of SAV fleets, called *SUMO4AV*, that we describe in this paper and which was developed within the project *KI4ROBOFLEET*[2]. *SUMO4AV* extends the *Simulation of Urban Mobility* (SUMO) package [10]. We use the scenario of transportations between points of interest using SAV fleets (called *TPOI-Scenario*) as running example. To be precise, the *TPOI-Scenario* simulates people who need a ride after work from office or from home to other amenity locations such as restaurants, parks or cinemas.

Basically, *SUMO* enables the simulation of vehicles on roads by representing the roads as edges and the junctions as nodes in a network. The vehicles can move along the edges and turn to another edge/road at the junctions. However, this representation as a network alone does not provide a satisfactory implementation of SAV fleets and their behavior in distinct use cases. The implementation of the *TPOI-Scenario* and other complex scenarios requires the simulation model to reflect different points of interest, which cannot be modeled directly by the *SUMO* network. This is where *SUMO4AV* comes into play. It enables *SUMO* to simulate SAV fleets that operate according to defined scenarios and that reflect points of interest. *SUMO4AV* comprises several steps. In the first step, infrastructure data such as roads and points of interests are analyzed and processed. This data can be extracted from *OpenStreetMap* and will be transferred into a simulation model. The *OSMWebWizard* [6] tool from *SUMO* is used to perform the import. The second step enables to connect the map entities (e.g., *POIs*) with the *SUMO* objects like edges and parking areas. To enable interactions with the map entities, a projection of the map entities on the network is provided. With this realization, vehicles can now navigate to the map entities that are located on an edge. The third step allows to create customized scenarios. Finally, the fourth step offers a choice of predefined routing algorithms to simulate the customized scenarios.

The developed *SUMO4AV* environment is evaluated using map data from Mannheim, a city in Germany. Through an iterative process, the simulation was executed with multiple fleet sizes and optimization algorithms. The results show that the simulations give insights into the operation of SAV fleets. Using such a tool, cities or companies in the mobility sector are enabled to evaluate local opportunities and risks for the operation of SAV fleets reflecting local characteristics, like infrastructure, *POIs* or living and business areas [13].

The rest of the paper is organized as follows: Section 2 provides a review of relevant literature. Section 3 represents the workflow of how to generate the *SUMO* model to simulate the *TPOI-Scenario*. This comprises, firstly, how a static simulation model is created using *OpenStreetMap* data. Secondly, it includes how the required entities are modeled as *SUMO* objects for the *TPOI-Scenario*. Thirdly, it involves how the *TPOI-Scenario* is configured and which optimization algorithms are provided. The extended simulation environment is evaluated in Section 4 using a practical example, and the resulting findings and ideas for further work initiatives are finally presented in Section 5.

[2] Url: https://www.keim.iao.fraunhofer.de/de/projekte/KI4ROBOFLEET.html

2 Related Work

An observable recurring challenge is the development of reliable and usable mobility scenarios to evaluate new mobility concepts especially in the context of AVs. Several publications present simulation scenarios that deal with the simulation of fleet behavior patterns.

Simulations of autonomous and non-autonomous fleets: Wang et al. [18] extend an approach of Autonomous Intersection Management at an intersection with the focus on connected AVs by considering different challenges such as pedestrian road crossings. Vakayil et al. [17] describe a model for the allocation of user demand between an autonomous mobility-on-demand-system and mass transit. They developed a framework to support an operator's fleet management planning. Spieser et al. [15] use rebalancing strategies to enable a fleet operator to achieve a favorable balance between customer satisfaction and corporate goals. Different rebalancing strategies are evaluated in a simulation. The last two approaches do not reflect different types of *POIs* for the simulation of SAV fleets. From our point of view, this aspect is important for running more precise scenarios in the context of autonomous mobility and, hence, to show its potential. Although the information about the characteristics of *POIs* is extracted via the *OSMWebWizard*, this information is only used to display buildings and other map entities in the *SUMO GUI*. In contrast to that, our approach is to integrate information about the *POIs* into *SUMO* by a new component, so that the individual AVs can access them in a simulation.

Simulations of conventional fleets in SUMO: Malinverno et al. [11] developed framework to simulate vehicle-to-infrastructure. That framework supports different communication protocols. Codeca et al. [4] run mobility scenarios that support different kinds of traffic demands or free-flow patterns, scenario dimensions and road categories. Additionally, multi-modal traffic evaluations and traffic scenarios like rush hours are considered. The *SUMO* simulation also contains *POIs*, which are extracted from *OpenStreetMap*. As opposed to our approach, the *POIs* are differentiated binarily: into buildings and parking lots. In another work by Codeca et al. [5] the impact of vulnerable road users on road traffic is investigated. The goal is to optimize traffic and reduce traffic jam through Cooperative Intelligent Transport System applications. Bautista et al. [2] investigate the effects of dynamic route planning in vehicle simulations. They examine the impact of vehicle rerouting capabilities on vehicle mobility and vehicle network connectivity.

Simulations of SAV fleets in SUMO: S. Alazzawi et al. [1] performed a study that included the analysis of traffic count data and mobile phone data to investigate mobility demand and traffic jams. This data was combined with extensive simulations of conventional (classical) cars and self-driving robo-taxis in Milan, Italy. *SUMO* was chosen as the simulation environment and the map material was imported from *OpenStreetMap*. The self-driving robo-taxis are offered as on-demand mobility service and have the goal to transport people over a certain route and to pick up other users on the way having the same travel destination. The focus of Schweizer et al. [14] is on the development of travel demand generators that aim to create person-based plans for *SUMO*. This involves generating populations, activities and associated locations, travel plans, and determining travel time. Based on the calculated travel times, people can modify their travel plans. In Li et al. [9], the open-source simulator for autonomous driving research CARLA[3] is combined with *SUMO* to create a traffic environment for training AVs. Another approach is followed by Kusari et al. [8]. Here, the background traffic is approximated to reality by adjusting the parameter distribution of well-known car-following

[3] Url: https://carla.org/

models from driving databases. The second difference is that scenarios are abstracted by taking the strengths of the *SUMO* simulator and combining them with those of OpenAI Gym. The work of Gasper et al. [7] deals with the use of autonomous shuttles. Here, the autonomous shuttles move autonomously in an area shared with pedestrians. The simulation in *SUMO* is intended to derive further requirements for the development of AVs. The vehicles as well as the pedestrians both move on the streets and certain situations are simulated, such as a shuttle passing pedestrians without collisions.

3 Creating the SUMO Model

The workflow and its activities for extracting map entities and generating scenarios with *SUMO4AV* is presented in Figure 1. The figure also sketches the topics of the following subsections. In the first activity, *Import Map*, the *OSMWebWizard* extracts the map material from *OpenStreetMap* and stores it in individual XML files. The file *osm.poly.xml* contains information about the *POIs*, which is further processed in the self-developed component *SUMO4AV* (grey box) that also comprises the other activities of the workflow. Basically, converting data from *OpenStreetMap* using the *OSMWebWizard* provides a convenient, reliable and fast way to create a running *SUMO* model from scratch. To be able to access map entities like *POIs* and buildings in the *SUMO* model created with the *OSMWebWizard* a workaround must be performed, which is implemented in the second activity, *Process Map Entities* (Section 3.2). The *SUMO4AV* environment provides *GUI* based functions to process *POIs* and make them accessible for AVs. In the next activity, *Generate Scenarios* (in Section 3.3), the *Scenario Builder* is used to generate the considered *TPOI-Scenario*. The last activity, *Run Simulation* (3.4), describes the flow of the simulation, which includes on the one hand the input file (*osm.sumocfg*) and static parameters, which go directly into the *SUMO* simulation and on the other hand the dynamic routing of the selected strategies of the AVs, which are transmitted to the simulation via the interface *TraCI*[4].

3.1 Import Map

The first activity aims to create a *SUMO* model from *OpenStreetMap* by selecting a map area within the *OSMWebWizard*. For the *TPOI-Scenario* we used map data from Mannheim, Germany. The output comprises several files, which are shown in Figure 1. They are necessary for an executable *SUMO* model and are listed in the following:

- *osm.net.xml* contains all essential map entities for the simulation like roads, lanes, rails and traffic lights.
- *osm.poly.xml* contains polygons and points representing specific areas, buildings and *POIs*. This file is not required to run the simulation but improves the map appearance in the *SUMO GUI* by displaying buildings, *POIs* and other map entities in different colors.
- *osm.view.xml* contains the *SUMO GUI* settings.
- osm.[x]trips.xml: contains a random base traffic specification (see Section 3.4).
- routes.xml: contains base traffic specification from traffic counting data (see Section 3.4).
- *osm.sumocfg*: is the *SUMO* master file which includes the files above.

[4] Url: https://sumo.dlr.de/docs/TraCI.html

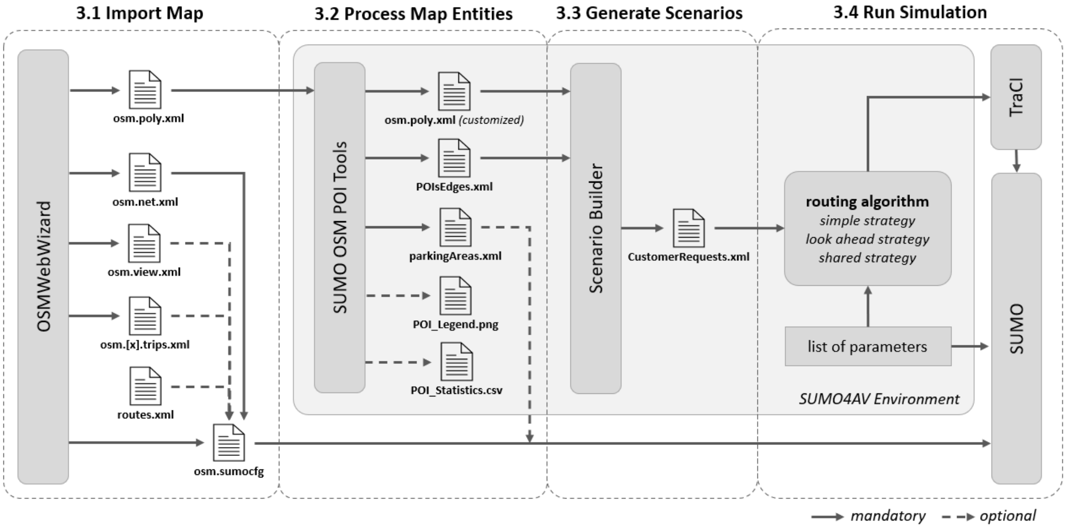


Figure 1: Workflow diagram for map entity extraction and scenario generation for *AVs*

3.2 Process Map Entities

As described in Section 3.1, the *osm.poly.xml* file is one of the output files of the first activity and serves as input file for the second activity, which contains the mapping of *POIs* to *SUMO* accessible objects.

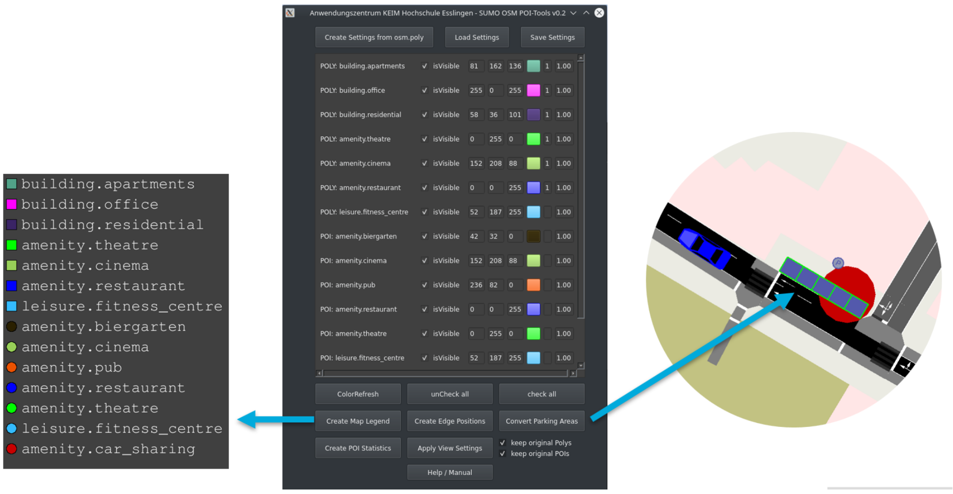
Every entity in the file contains a unique Id and further information (e.g., the *POI* type like *amenity.restaurant*), which was transferred from the *OpenStreetMap* data. In the *SUMO GUI* these entities can be inspected, and their view settings can be changed, but there is no interaction possible between these map entities and the simulation. For closing this gap, we developed the *SUMO OSM POI-Tools*, which are part of *SUMO4AV* (Figure 2). They provide a set of functions to create a workaround that meets the aforementioned demands of interacting with map entities for more detailed and realistic simulations and scenario analysis. Furthermore, they comprise functions to analyze, process, map, and display *POIs* and other map entities.

Figure 2 shows how different *OpenStreetMap* entity types (e.g., *building.office*) are imported and colored in a customizable way. Each entity type consists of a main type or primary feature (e.g., building) and a sub type (e.g., office). The usage of types and subtypes follow a certain principle described in the *OpenStreetMap* documentation[5]. The first step in the *SUMO OSM POI-Tools* is to select the map entity types that should be considered for the simulation. Optionally, the colors for these map entities on the map can be set.

After the selection, the following functions (scripts) for further processing are available:

- **Create Edge Positions**: With this script each entity instance of the selected entity types gets a position in the *SUMO* model. The entity instances are mapped on a *SUMO* edge (by the edge Id) and given an edge position to make them accessible for the *SUMO* routing algorithm. The edge position is a float value between zero and the length of the edge. To identify which end of the edge equals to the zero position the edge definition has to be considered. The result is the file *POIsEdges.xml*.

[5] Url: https://wiki.openstreetmap.org/wiki/Map_features



🟩 building.apartments
🟪 building.office
🟪 building.residential
🟩 amenity.theatre
🟩 amenity.cinema
🟦 amenity.restaurant
🟦 leisure.fitness_centre
● amenity.biergarten
● amenity.cinema
● amenity.pub
● amenity.restaurant
● amenity.theatre
● leisure.fitness_centre
● amenity.car_sharing

POLY: building.apartments	✓	isVisible	81	162	136	🟩	1	1.00
POLY: building.office	✓	isVisible	255	0	255	🟪	1	1.00
POLY: building.residential	✓	isVisible	58	36	101	🟪	1	1.00
POLY: amenity.theatre	✓	isVisible	0	255	0	🟩	1	1.00
POLY: amenity.cinema	✓	isVisible	152	208	88	🟩	1	1.00
POLY: amenity.restaurant	✓	isVisible	0	0	255	🟦	1	1.00
POLY: leisure.fitness_centre	✓	isVisible	52	187	255	🟦		1.00
POI: amenity.biergarten	✓	isVisible	42	32	0	🟫		1.00
POI: amenity.cinema	✓	isVisible	152	208	88	🟩		1.00
POI: amenity.pub	✓	isVisible	236	82	0	🟧		1.00
POI: amenity.restaurant	✓	isVisible	0	0	255	🟦		1.00
POI: amenity.theatre	✓	isVisible	0	255	0	🟩		1.00
POI: leisure.fitness_centre	✓	isVisible	52	187	255	🟦		1.00

✓ keep original Polys
✓ keep original POIs

Figure 2: Extract of the SUMO OSM POI-Tools

- **Convert Parking Areas**: With this script the entity instances with type *amenity.parking* are converted to parking areas (with several parking lots) which can be utilized by the *SUMO* routing algorithm. The routing strategies, described in Section (3.4), use these parking areas to park AVs that have no trip request at hand. In some *amenity.parking* entities the number of parking lots are specified. We use this data to create the correct number of parking lots for the simulation. If the number is not available, we take a default value. With this script, also the type *amenity.car_sharing* can be converted to parking areas, because it is considered that today's carsharing companies will be operators of SAV fleets in the future. The output file *parkingAreas.xml* is created and can be included into the *SUMO* model, i.e., the *osm.sumocfg* file.

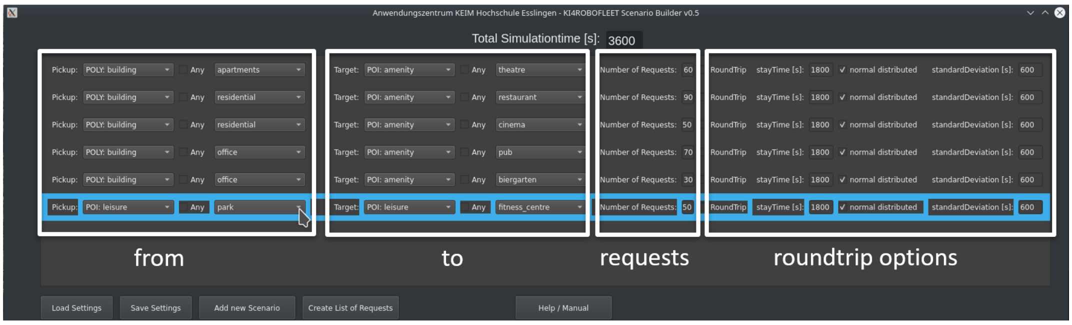
- **Apply View Settings**: This script modifies the *osm.poly.xml* input file according to the entity type selection and color settings in the *SUMO4AV GUI* to apply them in the *SUMO GUI*.

- **Create POI Statistics**: This script creates the *POI_Statistics.csv* file with statistics of the occurrence for each map entity type (e.g., building.school) of the selected map area.

- **Create Map Legend**: This script creates the *POI_Legend.png* file providing a map legend with all selected entity types and their current color settings. The polygons are represented as colored squares and the *POIs* as colored points.

3.3 Generate Scenario

The *Scenario Builder* module of *SUMO4AV*, shown in Figure 3, is a tool to create customized lists of random requests which define certain simulation scenarios. For each scenario a list of pickup and target map entity types has to be specified as input data. A use case of a common leisure (after work) scenario could be, for example, to pick up customers at an entity of type *building.office* and to bring them to an entity of type *amenity.restaurant* or *leisure.fitness_centre*. A scenario usually comprises a list of several transportation use cases with separate pickup and target map entity types. The *Scenario Builder* randomly picks map entities of the specified pickup and target map entity types and creates a list of customer requests. The submission time is randomly distributed, so the requests arise randomly within the total simulation time specified in the *SUMO4AV GUI*. For each use case in the list, a certain number of requests is specified. An additional feature is the roundtrip option. Here, for each request a return ride is scheduled after a certain stay time, which can optionally be normal distributed with a given standard deviation to model the customer behavior more realistically. The output of the *Scenario Builder* is the *CustomerRequests.xml* file, which contains a list of *SUMO* readable edge Ids and edge positions for the pickup and target map entities. This file contains also additional meta data for the considered map entities (e.g., restaurant type, opening times and the url), but this data is not used yet by our toolset.



Total Simulationtime [s]: 3600

Pickup:	POLY: building	□ Any	apartments	Target:	POI: amenity	□ Any	theatre	Number of Requests:	60	RoundTrip	stayTime [s]:	1800	✓ normal distributed	standardDeviation [s]:	600
Pickup:	POLY: building	□ Any	residential	Target:	POI: amenity	□ Any	restaurant	Number of Requests:	90	RoundTrip	stayTime [s]:	1800	✓ normal distributed	standardDeviation [s]:	600
Pickup:	POLY: building	□ Any	residential	Target:	POI: amenity	□ Any	cinema	Number of Requests:	50	RoundTrip	stayTime [s]:	1800	✓ normal distributed	standardDeviation [s]:	600
Pickup:	POLY: building	□ Any	office	Target:	POI: amenity	□ Any	pub	Number of Requests:	70	RoundTrip	stayTime [s]:	1800	✓ normal distributed	standardDeviation [s]:	600
Pickup:	POLY: building	□ Any	office	Target:	POI: amenity	□ Any	biergarten	Number of Requests:	30	RoundTrip	stayTime [s]:	1800	✓ normal distributed	standardDeviation [s]:	600
Pickup:	POI: leisure	□ Any	park	Target:	POI: leisure	□ Any	fitness_centre	Number of Requests:	50	RoundTrip	stayTime [s]:	1800	✓ normal distributed	standardDeviation [s]:	600

Figure 3: The SUMO4AV Scenario Builder module

3.4 Run Simulation

After creating the *SUMO* model and the *CustomerRequests.xml* file, the simulation can be started with three different routing strategies, which define how the customer requests from the *CustomerRequests.xml* file are processed. In the following Table 1, the developed routing strategies as well as the related conditions and parameters are described. The two parameters LF (lateness factor) and RT (realistic time) for the shared strategy are not self-evident. They are used to determine the length of the detour to pick up a second customer on a similar route. Following calculations are performed in this context: travel_time= sumo_estimate * RT and expected_finish = travel_time * LF where sumo_estimate is the estimated travel time from the *SUMO* routing algorithm.

The interaction of the running *SUMO* traffic simulation with AVs and customers is implemented in Python, using the *SUMO traffic control interface (TraCI)*[6]. *TraCI* provides a large set of functions to access and modify entities of a *SUMO* traffic simulation during the runtime. In the current implementation of the fleet management algorithm and the routing strategies, *TraCI* was used to push the data of the Python objects for AVs and customers to the running simulation and to fetch the current state. When a customer request from the *CustomerRequests.xml* file is submitted, the waiting customer is placed via *TraCI* at the specified pickup edge id and edge position. Then an AV needs to be dispatched to this customer. *SUMO* provides a configurable Taxi dispatch algorithm to simulate this case of demand responsive

[6] Url: https://sumo.dlr.de/docs/TraCI.html

transport (DRT)[7]. All strategies described in Table 1 use the *SUMO* Taxi dispatch algorithm, but they differ in the way how the requests are processed.

Table 1: Overview of the routing strategies

Strategy	Description	Conditions	Parameters
simple	After submitting a customer request, the closest free AV is assigned to this request.	only one customer per AV, fleet size is fixed	fleet size
look ahead	The fleet is informed about the predicted location of an upcoming customer request in advance, e.g., 10 min (look ahead time) and sends an AV to this location. The idea is to make request predictions based on AI techniques.	only one customer per AV, fleet size is fixed	fleet size, look ahead time [sec]
shared	The AV can make a detour to pick up a second customer with a similar route.	One or two customers per AV, flexible fleet size	LF (lateness factor)*, RT (realistic time)*

** The length of the detour to pick up a second customer on a similar route can be specified by two parameters called lateness factor (LF) and realistic time (RT).*

Optionally, it should be considered to include daily base traffic in order to get a more realistic simulation model. A straightforward way to create a random base traffic is provided by the *OSMWebWizard*, where different types and flow-rates of traffic (i.e., cars, trucks, buses, motorcycles, bicycles, pedestrians, trams, trains and ships) can be included. For each type of traffic entity, a separate file is created, e.g., *osm .bus.trips.xml*. However, in order to consider a more realistic base traffic, a more complex approach can be performed by using traffic counting data, which can be converted to XML route files (*routes.xml*) by using the *SUMO* routeSampler[8] script.

4 Evaluation

For the evaluation of *SUMO4AV* and the complete workflow we performed the *TPOI-Scenario* at the city center of Mannheim (Figure 4). The scenario represents a typical and realistic leisure scenario, which can be observed in cities. Pickup locations are, for example, apartments, offices or residential areas and target locations are restaurants, cinemas, pubs, theatres, fitness centers or beer gardens.

This *TPOI-Scenario* has been applied to the workflow described in Section 3 . Firstly, we imported the map area from the *OSMWebWizard* as described in the first activity in the workflow (Figure 1). Secondly, we applied the next steps by using *SUMO4AV*. They comprised the processing of the map entities (activity two in the workflow). For that we used the *SUMO OSM POI-Tools* to select and to colorize the map entities which were relevant to the *TPOI-Scenario*. This is shown in Figure 2. Moreover, we generated the *TPOI-Scenario* with the

[7] Url: https://sumo.dlr.de/docs/Simulation/Taxi.html
[8] Url: https://sumo.dlr.de/docs/Tools/Turns.html#routesampler.py

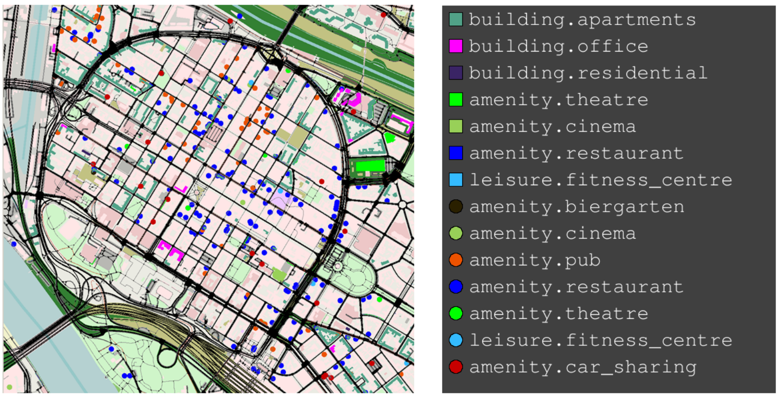


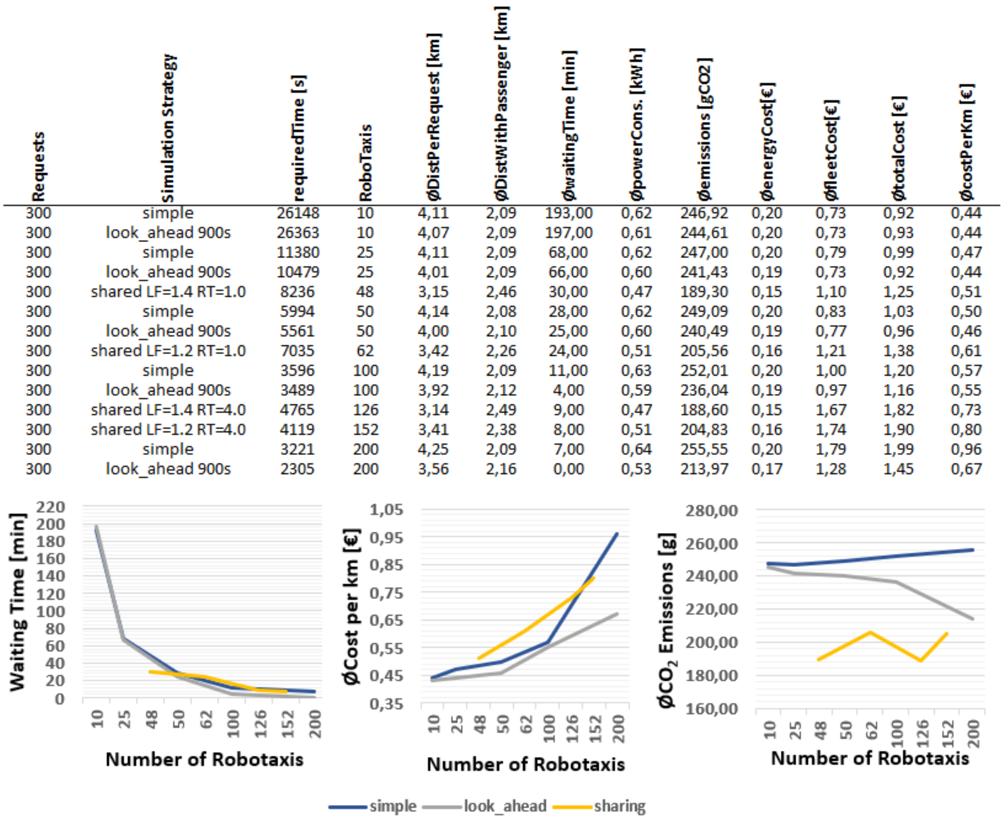
Figure 4: TPOI scenario: City center of Mannheim in the evening

Scenario Builder as shown in Figure 3 including several pickup and target map entity types. The output file *CustomerRequests.xml* of the *Scenario Builder* contains 300 requests and was processed 14 times with three different routing strategies, each with different parameter sets. With respect to the routing strategies as described in Table 1, the simulations with the simple strategy and with the look ahead strategy were each performed with a fleet size parameter of 10, 25, 50, 100 and 200. Additionally, for the look ahead strategy, the parameter for the look ahead time was set to 900 seconds (15 minutes). The shared strategy was applied with four different pairs of values for the lateness factor (LF) and realistic time (RT): LF=1.2 and RT=1.0, LF=1.4 and RT=1.0, LF=1.2 and RT=4.0, LF=1.4 and RT=4.0. All 300 requests were submitted within the first 30 minutes, but the initial simulation time was set to a much higher value of 10 hours (36000 seconds) to ensure, that even simulation runs with small fleet sizes can complete all 300 requests and subsequently to make the simulation results comparable. When all 300 requests are fulfilled, the current simulation run is aborted and the result file is written. We originally planned to use base traffic derived on traffic counting data. Therefore the city of Mannheim provided traffic counting data of 89 counters in the considered map area, which were converted to XML route files (*routes.xml*) by using the *SUMO* routeSampler[9] script . After we faced severe stability problems by including the (*routes.xml*) file, we decided to perform the proof-of-concept simulations without base traffic.

The AVs were considered as fully electric vehicles with following boundary conditions for the simulation and the subsequent calculation of key figures:

- Energy consumption per vehicle: 15kWh/100km
- Emissions: 401g/kWh (according to the German energy mix for generating electricity 2019 [16])
- Energy costs: 0,32€/kWh
- Fleet base costs per vehicle: 3€/h

[9] Ibid.



Requests	Simulation Strategy	requiredTime [s]	RoboTaxis	ØDistPerRequest [km]	ØDistWithPassenger [km]	ØwaitingTime [min]	ØpowerCons. [kWh]	Øemissions [gCO2]	ØenergyCost[€]	ØfleetCost[€]	ØtotalCost [€]	ØcostPerKm [€]
300	simple	26148	10	4,11	2,09	193,00	0,62	246,92	0,20	0,73	0,92	0,44
300	look_ahead 900s	26363	10	4,07	2,09	197,00	0,61	244,61	0,20	0,73	0,93	0,44
300	simple	11380	25	4,11	2,09	68,00	0,62	247,00	0,20	0,79	0,99	0,47
300	look_ahead 900s	10479	25	4,01	2,09	66,00	0,60	241,43	0,19	0,73	0,92	0,44
300	shared LF=1.4 RT=1.0	8236	48	3,15	2,46	30,00	0,47	189,30	0,15	1,10	1,25	0,51
300	simple	5994	50	4,14	2,08	28,00	0,62	249,09	0,20	0,83	1,03	0,50
300	look_ahead 900s	5561	50	4,00	2,10	25,00	0,60	240,49	0,19	0,77	0,96	0,46
300	shared LF=1.2 RT=1.0	7035	62	3,42	2,26	24,00	0,51	205,56	0,16	1,21	1,38	0,61
300	simple	3596	100	4,19	2,09	11,00	0,63	252,01	0,20	1,00	1,20	0,57
300	look_ahead 900s	3489	100	3,92	2,12	4,00	0,59	236,04	0,19	0,97	1,16	0,55
300	shared LF=1.4 RT=4.0	4765	126	3,14	2,49	9,00	0,47	188,60	0,15	1,67	1,82	0,73
300	shared LF=1.2 RT=4.0	4119	152	3,41	2,38	8,00	0,51	204,83	0,16	1,74	1,90	0,80
300	simple	3221	200	4,25	2,09	7,00	0,64	255,55	0,20	1,79	1,99	0,96
300	look_ahead 900s	2305	200	3,56	2,16	0,00	0,53	213,97	0,17	1,28	1,45	0,67

Figure 5: Simulation results of the SUMO4AV (TPOI-Scenario)

Figure 5 shows that the runs for the *TPOI-scenario* in the SUMO4AV environment were a successful proof-of-concept simulation with comprehensible results, even when the negligence of the base traffic might gloss over the results. The required simulation time depends strongly on the fleet size and varies between 2305 seconds and 26363 seconds. Further simulation results are ecological and economical key figures, e.g., CO_2 emissions, total costs, or waiting times, calculated as average values per request to make these values more descriptive. The results show, that for the given boundary conditions with 300 requests, a minimum fleet size of 50 is necessary to achieve an acceptable waiting time of less than 30 minutes. A fleet size of 25 leads to waiting times of more than one hour and a fleet size of 10 leads to waiting times of more than 3 hours, but with such small fleet sizes the costs per kilometer are less than 0.5€ which can be attractive for some specific use cases. The look ahead strategy seems to be the most efficient strategy concerning waiting time and cost per kilometer. As expected, by transporting more than one passenger at once, the shared strategy causes the lowest emissions, because the average driving distance per request is significantly lower compared to the other strategies. Surprisingly the costs per kilometer are slightly higher, which can be explained by the fact, that the parameter set which specifies the length of the allowed detour to pickup further passengers is not optimized yet. This leads to idle time for some vehicles, causing higher fleet base costs.

5 Conclusions

In this paper we presented *SUMO4AV*, a simulation environment that facilitates the simulation of SAV fleets in cities. It extends the *SUMO* package by making *OpenStreetMap* entities like *POIs* accessible during simulation runs and by the possibility to define different use case scenarios. We described in detail the complete workflow to create, customize, run and analyze the simulations. *SUMO4AV* is currently in a prototypical state available on GitHub[10].

As proof-of-concept, we simulated a leisure scenario where an SAV fleet transports people in the evening after work from office or from home to locations like restaurants or cinemas. The scenario was successfully performed by importing *OpenStreetMap* data of the city center of Mannheim with the *OSMWebWizard*, by extracting map entities and creating the scenario with the *SUMO4AV* environment, by running the simulation with *SUMO* and by attaining first simulation results. We used three different routing strategies for the SAV fleet and several different fleet sizes. Due to stability problems during runtime the proof-of-concept had to be performed without base traffic.

Future work is planned to improve the usage of the environment and to perform larger studies on the routing strategies, especially for the shared strategy in order to optimize the parameter sets. A further improvement can be attained by implementing a combination of the shared and the look ahead strategy to unite the advantages of both strategies. Also, the stability problems by using base traffic needs to be investigated and solutions must be found. When these problems are solved, we want to apply *SUMO4AV* for other municipalities, in the ideal case also using further data from mobility studies. It is also planned to implement an additional feature to consider charging states and charging cycles to give SAV fleet operators more accurate results regarding the maximum degree of capacity utilization of an SAV fleet.

References

[1] Sabina Alazzawi, Mathias Hummel, Pascal Kordt, Thorsten Sickenberger, Christian Wieseotte, and Oliver Wohak. Simulating the Impact of Shared, Autonomous Vehicles on Urban Mobility – a Case Study of Milan. In Evamarie Wießner, Leonhard Lücken, Robert Hilbrich, Yun-Pang Flötteröd, Jakob Erdmann, Laura Bieker-Walz, and Michael Behrisch, editors, *SUMO 2018- Simulating Autonomous and Intermodal Transport Systems*, volume 2 of *EPiC Series in Engineering*, pages 94–110. EasyChair, 2018.

[2] Pablo Barbecho Bautista, Luis Urquiza-Aguiar, and Mónica Aguilar Igartua. Evaluation of Dynamic Route Planning Impact on Vehicular Communications with SUMO. pages 27–35, 2020.

[3] Deutscher Bundestag. Bundestag nimmt Gesetz zum autonomen Fahren an (text in German), 2022. URL: https://www.bundestag.de/dokumente/textarchiv/2021/kw20-de-autonomes-fahren-840196, accessed 2022-02-25.

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

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

[6] German Aerospace Center (DLR) and others. OSMWebWizard, 2022. URL: https://sumo.dlr.de/docs/Tutorials/OSMWebWizard.html, accessed 2022-02-25.

[7] Rainer Gasper, Stephan Beutelschieß, Mario Krumnow, Levente Simon, Zoltan Baksa, and Jochen Schwarzer. Simulation of Autonomous RoboShuttles in Shared Space. In *SUMO 2018- Simulating*

[10] Url: https://github.com/keim-hs-esslingen/ki4robofleet

Autonomous and Intermodal Transport Systems, volume 2 of *EPiC Series in Engineering*, pages 183–193. EasyChair, 2018.

[8] Arpan Kusari, Pei Li, Hanzhi Yang, Nikhil Punshi, Mich Rasulis, Scott Bogard, and David J. LeBlanc. Enhancing SUMO simulator for simulation based testing and validation of autonomous vehicles, 2021.

[9] Pei Li, Arpan Kusari, and David J. LeBlanc. A Novel Traffic Simulation Framework for Testing Autonomous Vehicles Using SUMO and CARLA, 2021.

[10] 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, 2018.

[11] Marco Malinverno, Francesco Raviglione, Claudio Ettore Casetti, Carla-Fabiana Chiasserini, Josep Mangues-Bafalluy, and Manuel Requena-Esteso. A Multi-stack Simulation Framework for Vehicular Applications Testing. 2020.

[12] Daniel Omeiza, Helena Webb, Marina Jirotka, and Lars Kunze. Explanations in Autonomous Driving: A Survey. IEEE Transactions on Intelligent Transportation Systems, 2021.

[13] Kristian Schaefer, Konrad Sagert, Emanuel Reichsöllner, Andreas Rößler, Johannes Feifel, Claudia Wizgall-Jambor, Rüdiger Kladt, Harald Zielstorff, Lutz Nolte, Volker Durchholz, Claudia Kempka, and Marco Gergel. KI für autonome Fahrzeugflotten (text in German). Fraunhofer-Institut für Arbeitswirtschaft und Organisation IAO, 2021.

[14] Joerg Schweizer, Federico Rupi, Francesco Filippi, and Cristian Poliziani. Generating activity based, multi-modal travel demand for SUMO. In Evamarie Wießner, Leonhard Lücken, Robert Hilbrich, Yun-Pang Flötteröd, Jakob Erdmann, Laura Bieker-Walz, and Michael Behrisch, editors, *SUMO 2018- Simulating Autonomous and Intermodal Transport Systems*, volume 2 of *EPiC Series in Engineering*, pages 118–133. EasyChair, 2018.

[15] Kevin Spieser, Samitha Samaranayake, Wolfgang Gruel, and Emilio Frazzoli. Shared-Vehicle Mobility-on-Demand Systems: A Fleet Operator's Guide to Rebalancing Empty Vehicles. Transportation Research Board 95th Annual Meeting, Washington DC, United States, 2017.

[16] Umweltbundesamt. Bilanz 2019: CO2-Emissionen pro Kilowattstunde Strom sinken weiter (text in German), 2020. URL: https://www.umweltbundesamt.de/presse/pressemitteilungen/bilanz-2019-co2-emissionen-pro-kilowattstunde-strom, accessed 2022-02-28.

[17] Akhil Vakayil, Wolfgang Gruel, and Samitha Samaranayake. Integrating Shared-Vehicle Mobility-on-Demand Systems with Public Transit, 2017.

[18] Michael I.-C. Wang, Charles H.-P. Wen, and H. Jonathan Chao. Roadrunner+: An Autonomous Intersection Management Cooperating with Connected Autonomous Vehicles and Pedestrians with Spillback Considered. volume 6, pages 1–29, 2022.

SUMO User Conference 2022
https://doi.org/10.52825/scp.v3i.110

Published: 29 Sept. 2022

Extending SUMO for Lane-Free Microscopic Simulation of Connected and Automated Vehicles

Dimitrios Troullinos[1][https://orcid.org/0000-0003-3228-3888],
Georgios Chalkiadakis[1][https://orcid.org/0000-0002-0716-2972],
Diamantis Manolis[1][https://orcid.org/0000-0003-4569-8732],
Ioannis Papamichail[1][https://orcid.org/0000-0002-6332-9327], and
Markos Papageorgiou[1][https://orcid.org/0000-0001-5821-4982]

[1] Technical University of Crete, Chania, Greece
{dtroullinos, dmanolis, ipapa, markos}@dssl.tuc.gr
gehalk@intelligence.tuc.gr

Abstract: This paper presents some new developments related to TrafficFluid-Sim, a lane-free microscopic simulator that extends the SUMO simulation infrastructure to model lane-free traffic environments, allowing vehicles to be located at any lateral position, disregarding standard notions of car-following and lane-change maneuvers that are typically embedded within a (lane-based) simulator. A dynamic library has been designed for traffic monitoring and lane-free vehicle movement control, one that does not impose any inter-tool “communication” delays that standard practices with the TraCI module introduce; and enables the emulation of vehicle-to-vehicle and vehicle-to-infrastructure communication. We first summarize the various core components that constitute our simulator, and then discuss the new capability to utilize the bicycle kinematic model, additionally to the usual double-integrator model, as a more realistic model of vehicle movement dynamics, particularly for a lane-free traffic environment. Finally, we developed the necessary components so that the bicycle model can alternatively be combined with the use of global coordinates for more realistic simulation in road networks with curvature, such as roundabouts.
Keywords: lane-free traffic, microscopic modelling and simulation, connected and automated vehicles

Introduction

Technological advancements in the automotive industry reinforce the promise of Connected and Automated Vehicles (CAVs) [1] featuring superb perception and automated driving capabilities, fostered by vehicle-to-vehicle and vehicle-to-infrastructure communications. In consequence, novel traffic flow paradigms appear that consider current or projected capabilities of CAVs, such as the TrafficFluid paradigm [2] that investigates novel traffic environments featuring two novel vehicle characteristics, namely: (i) lane-free traffic, meaning that vehicles' lateral placement can be arbitrary within the road boundaries; and (ii) vehicles may use their automated driving and connectivity capabilities to apply “vehicle nudging” caused by other neighboring vehicles. Such a pushing force may lead vehicles to adjust their lateral position appropriately to accommodate faster vehicles upstream to pass. In addition, nudging is found to lead to improved characteristics of the emerging traffic flow, e.g., in terms of stability and capacity [2].

The use of traffic simulation software is central in the design and testing of vehicle movement strategies and related applications and can significantly facilitate research in emergent

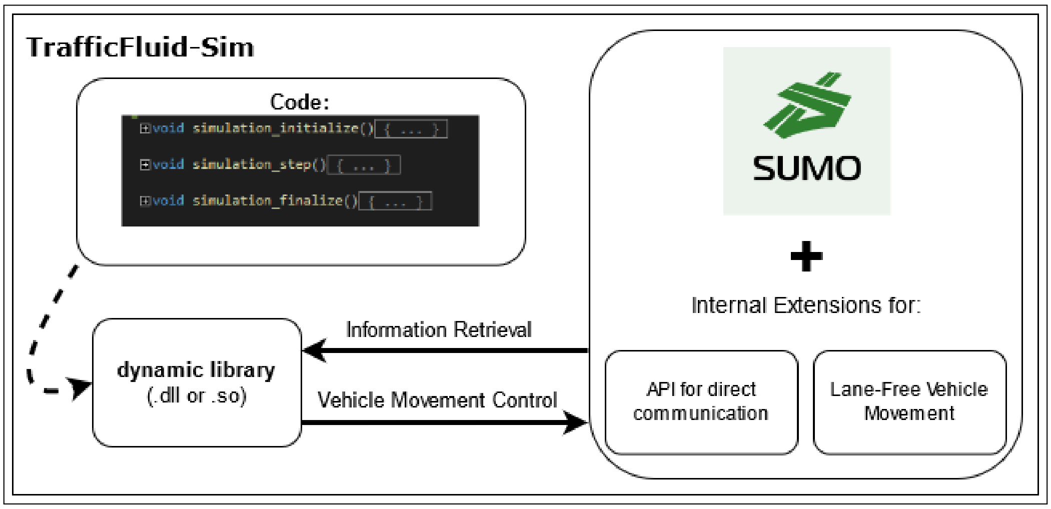

TrafficFluid-Sim
Code:
void simulation_initialize() { ... }
void simulation_step() { ... }
void simulation_finalize() { ... }
SUMO
+
Internal Extensions for:
API for direct communication
Lane-Free Vehicle Movement
Information Retrieval
Vehicle Movement Control
dynamic library
(.dll or .so)

Figure 1. A high-level overview of the different parts that constitute our lane-free microscopic simulation tool.

traffic paradigms. As such, it is important to have an appropriate simulator for lane-free traffic environments, one that allows for large-scale simulation scenarios, and its use can be easily generalized for different types of vehicles, connectivity schemes and road network structures.

To this end, we provide an extension of the SUMO simulator infrastructure, rendering it appropriate for the design and assessment of several vehicle movement strategies, including vehicle nudging, under the lane-free paradigm, considering the existence of CAVs. In what follows, we first summarize the most important aspects of our simulator, which are presented in more detail in [3]. Then, we present our newer development, the adoption of the bicycle model as an alternative to the existing double-integrator model, and the use of global coordinates for certain applications. Finally, we discuss imminent future work and conclusions.

Simulator Overview

SUMO [4] (Simulation of Urban MObility) is the prevalent platform to work with since it is an open-source project and therefore appropriate for the adjustments and extensions needed. While the use of TraCI for CAV related endeavors is quite popular in other existing tools, such as VEINS [5], "iTETRIS Control System" (iCS) [6] and MOSAIC (formerly known as VSim-RTI) [7], TraCI was not considered due to limited efficiency and lack of customizability. Note that we are interested in designing and testing novel vehicle movement strategies in lane-free environments, and that our simulator should support large-scale experiments. Simulations with a large number of controllable vehicles would require a substantial amount of communication, when using TraCI, since each vehicle would request relevant information through the TraCI API; and also provide a custom control input through it. Therefore, in simulations containing, e.g., thousands of vehicles, these delays would impose a significant bottleneck. We note that, while the Libsumo tool of SUMO addresses the communication overhead of TraCI, its use is still quite limited and it does not operate with the GUI application of SUMO. Moreover, TraCI (and SUMO in general) is designed for lane-based traffic, meaning that it relies on notions tied to lane-based traffic, such as car-following and lane-changing behaviors. Only through internal modifications and extensions in the codebase we could provide a simulator that is appropriate for simulation in lane-free traffic environments, and one that allows for further customization and extensions for imminent and future requirements.

Therefore, we have opted to construct a new dynamic library and an API that are tied to lane-free traffic environments. In Figure 1 we show a high-level overview of our application, which

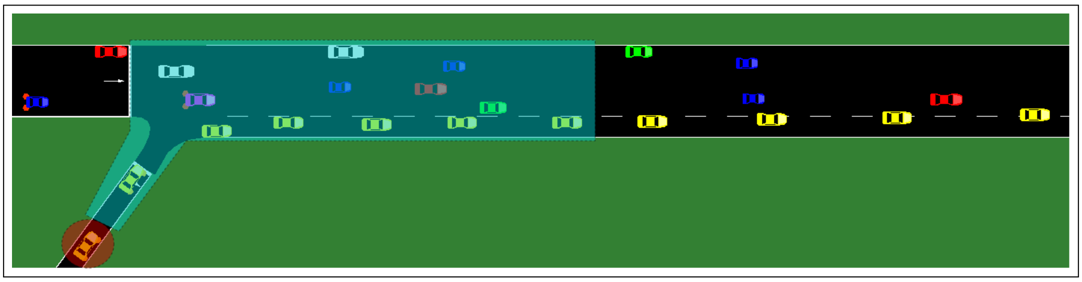


Figure 2. The ego vehicle on the on-ramp indicated with red, has direct access to the vehicles downstream located in the next segments.

contains the dynamic library that the user has access to and can develop code in C/C++ for any vehicle movement strategy, provided that the code yields the necessary control input of the vehicles in every discrete time-step. This is in contrast to the standard operation of SUMO, where the user has to select among a specific range of car-following and lane-changing models, and can only tune the associated set of parameters.

The code developed by the user is compiled into a .dll or .so file (depending on the operating system, windows or linux based), and is connected with our extension of the SUMO application, that contains the implementation of our API and the enabling of lane-free vehicle movement. With this API, the user has access to information regarding the traffic environment through the use of unique IDs (similar to TraCI). One can monitor online each vehicle's current status (position and speed), and request information regarding certain vehicle properties (e.g., length, width, followed route). Road Networks in SUMO have a graph structure, with road segments corresponding to the graph's edges. According to the unique ID of each segment, the user has access to relevant information, such as the IDs of the vehicles currently within that segment (ordered according to longitudinal position), and information regarding densities of vehicles within user-defined adjustable subregions of the segment. Dimensions of each road segment (length and width) are also available. Moreover, simple loop detectors placed through the road network can be monitored through the API as well.

We should emphasize that we mostly rely on the local coordinate system that vehicles employ through SUMO, meaning that each vehicle's position is w.r.t. the current road segment. This lifts the task of manually performing turning operations when needed and allows us to design vehicle movement strategies considering that each ego vehicle always operates on a straight highway. Essentially, a vehicle only needs to be placed appropriately laterally so that it follows the requested route (this is further discussed in future work, see Section 4). Each vehicle can observe downstream and upstream traffic, w.r.t. its own position and routing, and obtain an ordered (according to longitudinal distance) set of the neighbor IDs downstream and upstream. The important aspect of this feature is that the ego vehicle can automatically obtain information about the vehicle in the next (or previous, for upstream requests) road segments, depending on the specified range. Figure 2 showcases an example with an ego vehicle on an on-ramp that is about to enter the acceleration lane of the highway. It can request access to vehicles downstream by simply providing a longitudinal observation distance. The ego vehicle observes the highway as an unfolded straight road due to geometry of the road being handled by SUMO, and we performed the necessary developments so that information regarding longitudinal and lateral distances of neighboring vehicles (from the ego vehicle) is calculated accordingly.

Lane-Free Vehicle Movement with the Bicycle Model

In this section, we first briefly address how we consider the positioning of the vehicles in SUMO for our lane-free settings, and discuss the standard use of the double-integrator model for the movement dynamics. Then, we present the new option for movement dynamics with the bicycle kinematic model, along with the alternative to operate under global coordinates alongside this model.

Longitudinal and Lateral Positioning in SUMO

Regarding the positioning of the vehicles, the *local* coordinates (x, y) of the rectangular-shaped vehicles that we consider are as follows: The longitudinal position x of the vehicle is the distance of its center point from the starting point of the road segment. On the other hand, the lateral position y measures the distance from the right road boundary to the vehicle's center point. Hence, the vehicles observe a single lateral position, without knowledge about lateral placement w.r.t. to lane centers, as this information is not meaningful anymore for designing a vehicle movement control strategy under lane-free settings.

Double-Integrator Model

The 'Ballistic-Update' option of SUMO for the longitudinal movement, and the incorporation of lateral dynamics in a similar double-integrator manner, constitute a double-integrator model for the lane-free vehicle movement dynamics, in which longitudinal and lateral movements are disjointed and are controlled by respective independent (longitudinal and lateral) accelerations. For the double-integrator model, the orientation of the vehicle is essentially according to the ratio of longitudinal versus lateral speed, but it is always considered in parallel with the road boundaries for simplicity. This approximation is good enough for highways, where we typically have vehicles moving with high longitudinal speeds, while lateral speeds are much smaller. In the following subsection, we present an alternative approach, namely the bicycle kinematic model, which provides a more realistic depiction of movement dynamics, since it incorporates explicitly the orientation of the vehicle.

The associated controller for vehicles employing the double-integrator model should provide the control inputs, i.e., a longitudinal and a lateral acceleration (in m/s^2) value in every time-step. Also, under the lane-free paradigm, where vehicles can be placed anywhere laterally within the road boundaries, a collision between two vehicles is reported when their rectangular shapes overlap. This is a straightforward check, given the vehicles' information regarding positioning and their dimensions (length, width), since the vehicles' orientation is assumed parallel with the road boundaries.

Bicycle Kinematic Model

An important development, necessary to enable microscopic simulation of complex urban network applications such as [8], is the incorporation of the *bicycle kinematic model* [9] into the simulator. In Figure 3, we show a snapshot from a roundabout scenario containing vehicles that use the approach of [8] and utilize the bicycle kinematic model and global coordinates. In this model, the vehicle's front (and back) wheels are abstracted to a unique front (and back) wheel located at the respective axle middle points, whereby the front wheel is steerable, controlling the vehicle's orientation, and there is a unique forward acceleration (control input) and a unique forward speed, both in the direction of the current vehicle orientation. We refer the interested reader to the relevant paper [9] for more details regarding this kinematic model. In this model, longitudinal and lateral dynamics are interconnected and nonlinear, and, more generally, the bicycle model is more accurate in describing vehicle movement than the linear double-integrator model and is particularly interesting in a lane-free environment, when vehicles are driving in curves at relatively low speeds, such as in urban networks and roundabouts. In such cases, the user has the option to select an alternative method for vehicle movement dynamics in lane-free traffic. The bicycle model's state variables are, beyond the longitudinal and lateral vehicle

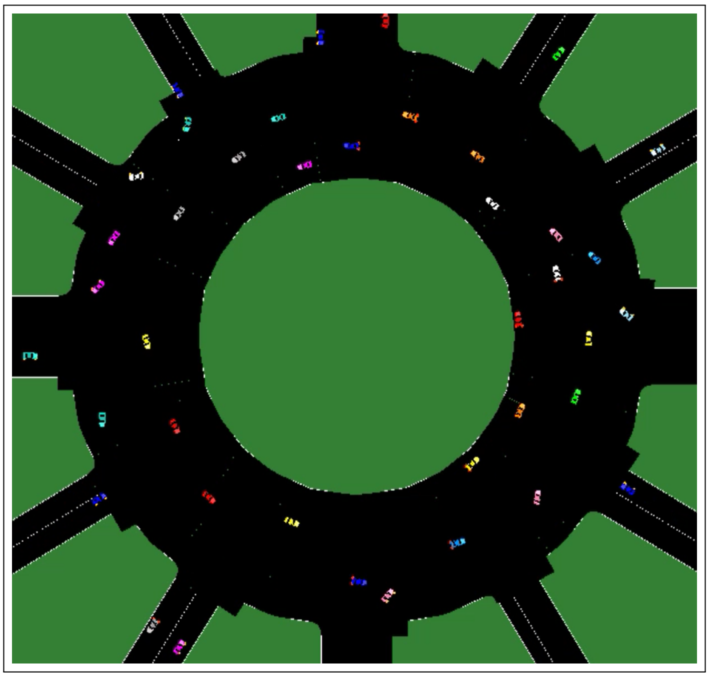


Figure 3. Vehicles employing the bicycle kinematic model with global coordinates in a roundabout scenario.

position (x, y), the vehicle orientation θ and the forward speed v; while the control variables are the forward acceleration and the steering angle. Figure 4 illustrates the state variables of the bicycle model, where we do not have two separate dynamics for longitudinal and lateral movements, but a unified and more realistic movement, involving the actual orientation resulting from steering the vehicle.

In terms of technical developments for the simulation, this involved an internal implementation regarding the position (x, y) and speed v update process of the vehicles, according to the bicycle kinematic model. Information regarding the orientation θ of the vehicle and the ability to directly change it is already available through SUMO, so we can adjust its value through internal extensions within the source code. The local coordinates (x, y) of the vehicle again refer to its center, as discussed earlier, and as depicted in Figure 4. However, the bicycle model utilizes the position corresponding to (x_b, y_b) in Figure 4, i.e., the vehicle's orientation changes with respect to this point. This is considered within our implementation, and this point is also directly available to the user who wishes to design a movement strategy using the bicycle model. User access to the orientation of the vehicles is granted through the API, providing either global (with respect to the global coordinate system of SUMO) or local (with respect to the longitudinal axis of the current road vehicles reside) coordinates. The associated lane-free controller should simply provide the value for the forward acceleration F (in m/s^2) and the steering angle δ (in rad) as control inputs for every time-step (instead of the longitudinal and lateral accelerations of the double-integrator model).

As of now, we have a simplified way that incorporates the orientation of the vehicles for the collision check. Essentially, we draw a rectangle that is in parallel to the road and contains the vehicle (red rectangular in Figure 4), and report a collision if these rectangular regions of two vehicles overlap. Of course, this procedure may report a collision even if two vehicles do not actually collide. However, it also guarantees that no collision will be disregarded. An exact collision identification method is in the course of implementation and will be reported when ready and tested.

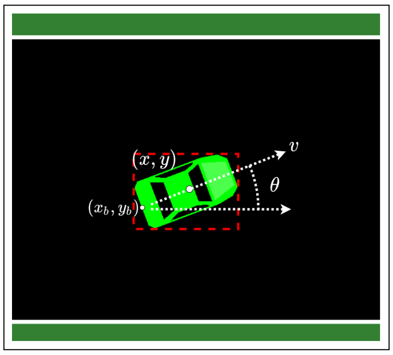


Figure 4. Illustration of a vehicle utilizing the bicycle model.

Bicycle Kinematic Model with Global Coordinates

The local coordinate system of SUMO is utilized for the bicycle model, and, as such, controllers do not need to take into account the geometry of the residing road. Hence, the operating orientation of the vehicles is local, i.e., with respect to the road segment the vehicle is currently located at, so turning operations (e.g., at junction points) are handled through SUMO. However, for certain applications, we may be interested in controlling turning operations instead of letting SUMO automatically handle such procedures, something that is also needed for more realistic depiction in road structures with continuous curvatures, e.g., roundabouts. We also provide the option to do so, in a way that does not affect or cause regression to existing functionalities. One can utilize the bicycle kinematics along with global coordinate control for more realistic behavior, that does not succumb to the road structure, but is rather based on the global Cartesian coordinates (x, y). This feature serves to facilitate an impending application that involves lane-free movement in a roundabout, where we wish to work with junctions for vehicles entering and exiting, and utilize polar coordinates instead of Cartesian ones for vehicles operating within the roundabout. This provides a more realistic and more convenient depiction of the actual vehicle and emerging traffic behavior compared to the use of the standard local coordinate system SUMO provides. Of course, in such a global environment, we need to properly design the above-mentioned behaviors.

For this feature, the use of internal mapping from global to local coordinates (and vice-versa) was crucial, given that the existing infrastructure and our extensions rely on the local coordinate system that SUMO provides. If we would completely neglect the local coordinates in favor of simplicity in the development process, then certain features would not function properly, e.g., the observation capabilities on surrounding vehicles, and as such, essential information for the design and real-time operation of vehicle movement strategies would not be available. Global coordinate control is tied only with the bicycle kinematics, since it would be quite restrictive for the double-integrator model, due to the absence of orientation control.

Future Extensions

A forthcoming extension is the incorporation of lateral boundaries based on the desired path for vehicles to follow, along with the capability to control them at execution time through the API for generalizing the vehicles' behavior in more complex road networks with vehicles following different routing schemes. In lane-based environments, vehicles can follow any (feasible) path by simply choosing lanes appropriately. For instance, a vehicle entering a highway from an on-ramp will typically need to perform a lane-change in order to merge on the highway. SUMO provides information regarding the availability of the road downstream through information on available lanes downstream. In our case, a vehicle adhering to the lane-free paradigm, wishing to enter a highway through an on-ramp, will again need to merge on the highway appropriately.

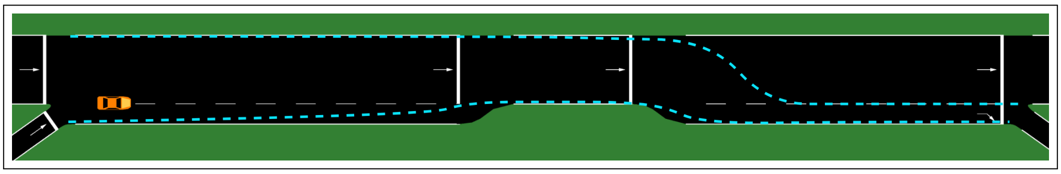


Figure 5. Left and right lateral boundaries for vehicles entering or scheduled to exit from the off-ramp.

Then, it is important to introduce an equivalent notion in a lane-free domain, since providing information on available lanes downstream is not appropriate in lane-free environments.

As of now, such maneuvers are performed with ad-hoc techniques, tied to specific traffic scenarios. Yet, there is a need to establish a formal way for vehicles to follow left and right boundaries on the road, guiding them to always operate within an admissible lateral region, so as to comply with their routing. This is illustrated in Figure 5, where we observe a vehicle with a route initiating from the on-ramp and leading to the off-ramp. The blue lines indicate the left and right lateral bounds, relevant to the path the vehicle needs to follow. As such, the vehicle will be able to request information regarding these two lateral bounds through the API, at any longitudinal distance corresponding to its position. Therefore, the vehicle will have the necessary time to adjust its behavior in order to be located within the bounds, and, as a consequence, to follow its desired routing scheme. Besides the application for on-ramps and off-ramps, this feature is crucial to enable microscopic simulation on emerging applications such as internal boundary control in lane-free traffic for two-way streams, as introduced in [10]. This will also involve an online update process for the boundaries' lateral position through the API.

Conclusions

In this work, we presented some advancements of TrafficFluid-Sim, an extension of SUMO appropriate for lane-free traffic environments, that is developed for the research project *TrafficFluid* [2]. Specifically, we discussed on the new capability to employ a more realistic depiction for movement dynamics with the bicycle model, which is more appropriate in certain applications that consider the lane-free paradigm. This tool is already being utilized for the design and evaluation of various vehicle movement strategies [2], [11]–[14].

Data Availability Statement

The associated code is developed for the Trafficfluid project [2] and cannot be shared as of now.

Underlying and related material

The interested reader may refer to https://bit.ly/3tf53jD, https://bit.ly/3K4azwz, https://bit.ly/3AtBNJR, and https://bit.ly/3K3olzc for work that utilizes TrafficFluid-Sim, and to https://trafficfluid.tuc.gr for more information regarding TrafficFluid.

Author contributions

The authors confirm contribution to the paper as follows: study conception and design: D. Troullinos, G. Chalkiadakis, I. Papamichail, M. Papageorgiou; data collection: D. Troullinos, D. Manolis; analysis and interpretation of results: D. Troullinos, I. Papamichail, M. Papageorgiou; draft manuscript preparation: D. Troullinos, G. Chalkiadakis, I. Papamichail, M. Papageorgiou. All authors reviewed the results and approved the final version of the manuscript.

Competing interests

The authors declare no competing interests.

Funding

The research leading to these results has received funding from the European Research Council under the European Union's Horizon 2020 Research and Innovation programme/ ERC Grant Agreement n. [833915], project TrafficFluid.

References

[1] D. Elliott, W. Keen, and L. Miao, "Recent advances in connected and automated vehicles," *Journal of Traffic and Transportation Engineering (English Edition)*, vol. 6, no. 2, pp. 109–131, 2019, ISSN: 2095-7564. DOI: https://doi.org/10.1016/j.jtte.2018.09.005.

[2] M. Papageorgiou, K.-S. Mountakis, I. Karafyllis, I. Papamichail, and Y. Wang, "Lane-free artificial-fluid concept for vehicular traffic," *Proceedings of the IEEE*, vol. 109, no. 2, pp. 114–121, 2021. DOI: 10.1109/JPROC.2020.3042681.

[3] D. Troullinos, G. Chalkiadakis, D. Manolis, I. Papamichail, and M. Papageorgiou, "Lane-free microscopic simulation for connected and automated vehicles," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 3292–3299. DOI: 10.1109/ITSC48978.2021.9564637.

[4] P. A. Lopez, M. Behrisch, L. Bieker-Walz, *et al.*, "Microscopic traffic simulation using sumo," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2575–2582. DOI: 10.1109/ITSC.2018.8569938.

[5] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Transactions on Mobile Computing (TMC)*, vol. 10, no. 1, pp. 3–15, Jan. 2011. DOI: 10.1109/TMC.2010.133.

[6] M. Rondinone, J. Maneros, D. Krajzewicz, *et al.*, "iTETRIS: A modular simulation platform for the large scale evaluation of cooperative its applications," *Simulation Modelling Practice and Theory*, vol. 34, pp. 99–125, 2013, ISSN: 1569-190X. DOI: https://doi.org/10.1016/j.simpat.2013.01.007.

[7] B. Schünemann, "V2X simulation runtime infrastructure VSimRTI: An assessment tool to design smart traffic management systems," *Computer Networks*, vol. 55, no. 14, pp. 3189–3198, 2011, ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2011.05.005.

[8] M. Naderi, M. Papageorgiou, I. Papamichail, and I. Karafyllis, "Automated vehicle driving on large lane-free roundabouts," in *2022 IEEE International Intelligent Transportation Systems Conference (ITSC), accepted*, 2022.

[9] P. Polack, F. Altché, B. d'Andréa-Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" In *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 812–818. DOI: 10.1109/IVS.2017.7995816.

[10] M. Malekzadeh, I. Papamichail, M. Papageorgiou, and K. Bogenberger, "Optimal internal boundary control of lane-free automated vehicle traffic," *Transportation Research Part C: Emerging Technologies*, vol. 126, p. 103 060, 2021, ISSN: 0968-090X. DOI: https://doi.org/10.1016/j.trc.2021.103060.

[11] D. Troullinos, G. Chalkiadakis, I. Papamichail, and M. Papageorgiou, “Collaborative multiagent decision making for lane-free autonomous driving,” in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, Virtual Event, United Kingdom, 2021, pp. 1335–1343. [Online]. Available: https://dl.acm.org/doi/10.5555/3463952.3464106.

[12] V. K. Yanumula, P. Typaldos, D. Troullinos, M. Malekzadeh, I. Papamichail, and M. Papageorgiou, “Optimal path planning for connected and automated vehicles in lane-free traffic,” in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 3545–3552. DOI: 10.1109/ITSC48978.2021.9564698.

[13] D. Troullinos, G. Chalkiadakis, V. Samoladas, and M. Papageorgiou, “Max-sum with quadtrees for decentralized coordination in continuous domains,” in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, International Joint Conferences on Artificial Intelligence Organization, 2022, pp. 518–526. DOI: 10.24963/ijcai.2022/74.

[14] M. Malekzadeh, D. Manolis, I. Papamichail, and M. Papageorgiou, “Empirical investigation of properties of lane-free automated vehicle traffic,” in *2022 IEEE International Intelligent Transportation Systems Conference (ITSC), accepted*, 2022.

SUMO User Conference 2022
https://doi.org/10.52825/scp.v3i.167

Published: 29 Sept. 2022

Multi-Modal Traffic Simulation Calibration and Integration with Real-Time Hardware in Loop Simulator

Vikhyat Kalra[1][https://orcid.org/0000-0002-5340-1719], Punit J Tulpule[1] and Jacob A. Isaman[1]

[1] The Ohio State University, USA
Kalra.50@buckleyemail.osu.edu, tulpule.3@osu.edu

Abstract

The ongoing research in intelligent transport systems and connected and automated vehicles, enabled by advancements in artificial intelligence, integrating traffic simulations has become an essential part of product/software development for the automotive industry. Nowadays, traffic simulations are used to mimic real-world environment scenarios for virtual testing of advanced transportation technologies. . With the increase in data collection methods for traffic flow, the calibration of the microscopic traffic simulations has emerged as an important research area. The underlying question in traffic modeling is how accurately simulations can mimic the real environment traffic flow conditions? This paper attempts to create a framework for microscopic traffic simulation calibration procedure which can be scaled for large networks. This paper makes the following major contributions. First, a calibration framework is proposed which harnesses the existing data set collected from The Ohio State University (OSU) campus bus service (CABS) busses using Global Positioning System (GPS) sensors to determine the traffic state in the real environment and create a microscopic traffic simulation. The traffic simulation is implemented for a section of the OSU campus ("Woody Hayes Drive") in an open-source traffic simulator – Simulation of Urban MObility (SUMO). The traffic flow generation is probabilistic to introduce variability between scenarios. The second contribution is the development of a communication interface between real-time dSpace ASM Hardware in Loop setup with SUMO to create a complete real-time simulation of urban environments for advanced driver assist systems (ADAS) virtual testing. Ademonstration scenario is the Ohio State University campus network with traffic demand generated using the calibrated model from the first part of the work.

1 Introduction

Traffic simulations are widely used by city planners and government agencies to understand the movement of people and goods in a road network environment. With increased interest in ADAS and connected and autonomous vehicles (CAV) technologies for intelligent transportation, traffic simulations have become an important part of automotive research. The development of ADAS and other connected vehicle fields requires millions of miles of testing to demonstrate improvements. Most often virtual simulation testing is a safer, more efficient and faster alternative to road testing[1]. For a realistic testing environment setup, traffic simulations play a very critical part. Traffic simulations consist of multiple modes of transportation (vehicles, pedestrians, scooters, etc.) interacting with a road network environment and infrastructure (e.g. traffic lights and signs). In a microscopic traffic simulation, each individual traffic participant is modeled as independent agent while ensuring the overall traffic flow matches the desired flow. The task of calibration of traffic simulation entitles the accurate vehicle flow information and vehicle interaction in a closed-loop road environment. Despite the wide research in the field of traffic simulations, the calibration approach used requires on-road traffic data which intern requires sensors infrastructure [2], such as, cameras, induction loop detectors, etc. This data may not be readily available due to cost and time.

A university campus is a unique environment as it consists of cars, buses, utility vehicles, and foot traffic. The traffic patterns tend to follow a specific pattern that depends on class schedule and vacations. Besides, the OSU campus offers unique opportunities for further research related to smart cities. Hence, this study focuses on modeling multi-modal traffic on the OSU campus. The key challenge is that vehicle counts data is very limited due to lack of sensors. The calibration approach used in this paper is based on GPS locations available from CABS busses. This approach does not require vehicle counts data from infrastructure sensors. In this approach we first compute the average travel time at intersections using the GPS data and calibrate the model to match this travel time. The underlying assumption is that the travel time is an indicator of traffic flow rate.

Calibrated traffic models alone are not sufficient to test ADS and ADAS safety. The vehicle under test (a.k.a. ego vehicle) should be controlled using the algorithm being tested. The fidelity of the ego vehicle models, and traffic simulator models are decided based on the validation level. However, establishing a seamless communication between vehicle simulators and traffic simulators is not always trivial as the models have different accuracies and time steps. In this paper we consider one such application of traffic models to Hardware in the Loop (HIL) testing of ADAS algorithms. A lot of research has been done in scenario-based testing of ADAS and CAVS [3]. The scenario-based testing consists of few vehicles, shorter time horizon and smaller distances. There are lot of ADAS testing software packages like MATLAB® ADT, dSpace ASM, CARLA, Roadrunner, etc. But no commercial or open-source available software has linked the traffic simulation with a vehicle simulator in a HIL setup [1]. In this paper the traffic simulation model calibrated for a section of the OSU campus is linked with the ADAS equipped vehicle model running on a HIL setup to demonstrate a closed loop virtual testing environment. The challenge is due to difference in the vehicle dynamics model fidelities and differences in time steps between SUMO and dSpace ASM. The SUMO model can run at maximum of 10Hz, but the HIL setup must run at-least at 100Hz. Besides, the yaw rate of SUMO vehicle dynamics is not accurate enough to feed directly to the high Degree of Freedom (DoF) vehicle dynamics model of ASM. Hence a synchronization interface is developed to approximate vehicle positions. This paper is structured as follows. Section 2 of the paper describes the traffic model calibration process and results of calibration for a section of the OSU campus. In Section 3 we explain the process to integrate traffic simulation models with HIL simulator. Finally, in Section 4 we make concluding remarks.

2 Multi-Modal Traffic Calibration using Probe Data

The simulation structure in SUMO is represented by Figure 1. The three primary components in a sumo simulation configuration file are: vehicle demand, additional files (virtual detectors and other road infrastructure) and network (road network with links and junctions etc.). The additional files and the network components of the simulation are not the variables for traffic flow calibration. The calibration parameters in this paper are the vehicle flow(count) in the network. The car following model used is Intelligent Driver Model (IDM). The selection of IDM as the candidate for car following model is based on current literature which depicts its better performance in high density junction traffic scenario to match the real-world driving behavior. [4]

The road network selected is Woody Hayes Drive in the OSU campus. The selected road network map is shown in *Figure 1*. The network consists of six intersections, out of which, five intersections are controlled by traffic signals. There are total twelve entry points for the vehicles in the road network. The primary reasons for selecting this section of the network are:

- Woody Hayes connect the major routes of CABS on OSU campus
- Woody Hayes Drive connects major department and student classes buildings thus it has the most footfall during office hours 8:00 a.m – 9:00 a.m (time considered in this simulation for traffic data)

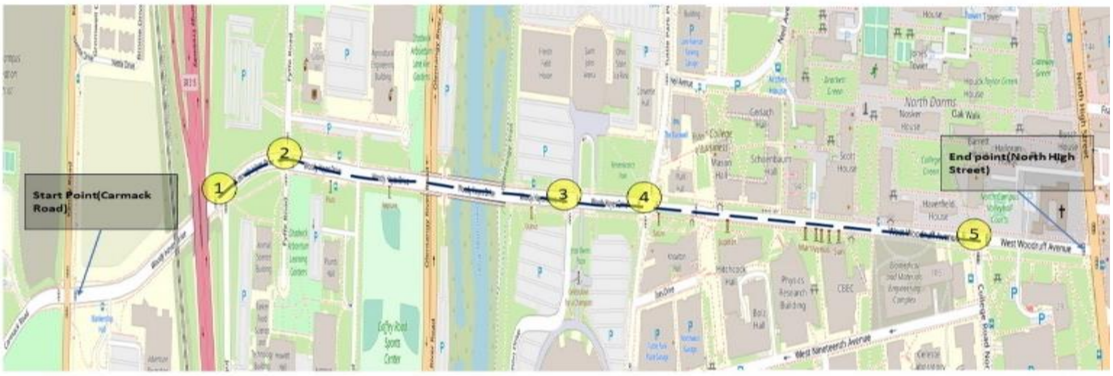


Figure 1: Woody Hayes Drive Network

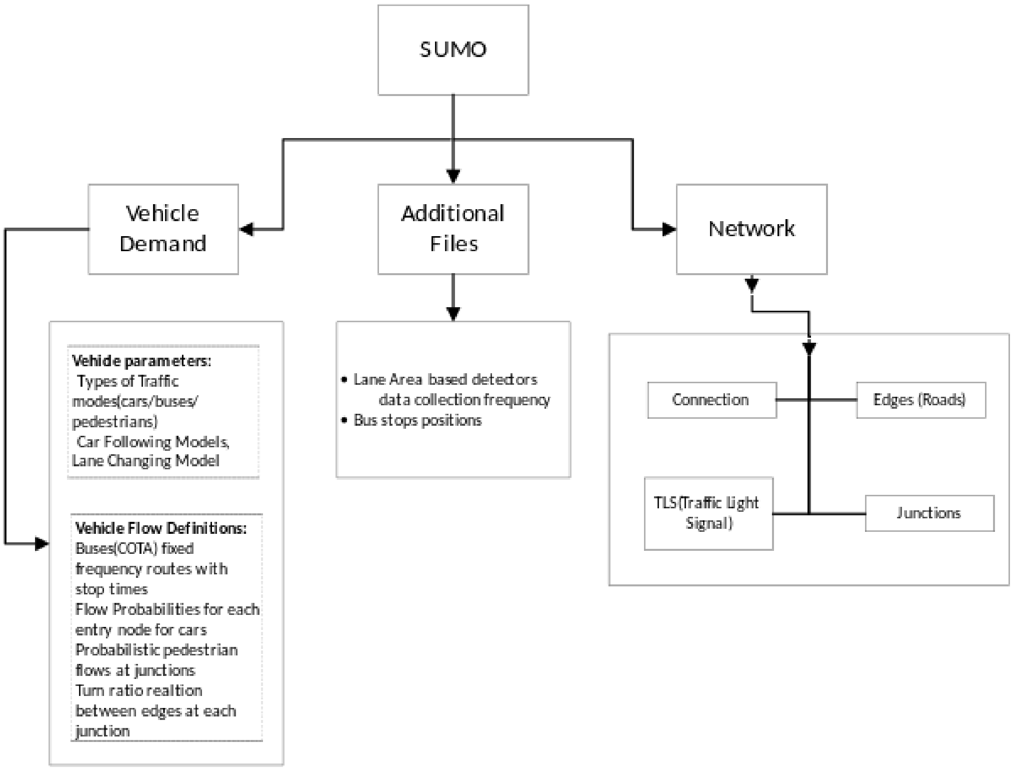

SUMO
Vehicle Demand
Additional Files
Network
Vehicle parameters:
Types of Traffic modes(cars/buses/ pedestrians)
Car Following Models,
Lane Changing Model
Vehicle Flow Definitions:
Buses(COTA) fixed frequency routes with stop times
Flow Probabilities for each entry node for cars
Probabilistic pedestrian flows at junctions
Turn ratio realtion between edges at each junction
• Lane Area based detectors data collection frequency
• Bus stops positions
Connection
Edges (Roads)
TLS(Traffic Light Signal)
Junctions

Figure 2: SUMO Traffic Simulation File Structure

The traffic simulation is multimodal with three vehicle classes - pedestrian, passenger cars and buses (CABS). The parameters used for these three modes are given in Table 1.

Table 1 Traffic Agents Parameters

Parameter	Pedestrian	Car	Bus
Length	0.215	4.3	12
Width	0.478	1.8	2.5
Height	1.719	1.5	3.4
Minimum Headway Gap (m)	0.25	2.5	2.5
Maximum Acceleration (m/s2)	1.5	2.9	1.2
Maximum Deceleration (m/s2)	2	7.5	4
Emergency Braking Deceleration (m/s2)	5	9	7
Maximum Speed (km/h)	5.4	180	85

2.1 Traffic Data

Any traffic model calibration process has two key steps – 1) road network calibration, and 2) traffic flow calibration. Road network calibration requires information about road segments and junctions including number of lanes, road length, speed limits, stop signs etc. Open Street Maps (OSM) is an open-source data consisting of all the necessary information regarding roads and junctions. However, the map imported from the OSM may have errors, particularly at junctions the number of lanes may not be correct or turn only lanes may not be correctly assigned. These errors are corrected using the process depicted in *Figure 4*. The corrections were validated by manually confirming against google maps street view. The other part of road network is traffic signal timings. The traffic signals assignment is done in SUMO using the traffic controller phase timings provided by the OSU Traffic and Transportation Management (TTM) Department and the phase timings are then inspected at the actual location to validate the controller data.

The vehicle flow data for three different modes are decided as per the following: -

- The CABS busses frequency and schedule are used to generate the flow(count) data. All the CABS busses are equipped with GPS logging devices. The data is available from 2011 to 2019. This data consists of GPS coordinates (latitude and longitude) and corresponding GPS timestamp. The data was filtered to extract this data of the busses on routes that pass through the selected region on campus. The data is not sampled consistently, meaning the sampling time varies depending on the number of available satellites and other uncertainties. Hence, another filter was implemented to discard inconsistently logged data. *Figure 3* shows the variability in successive data logs from GPS.

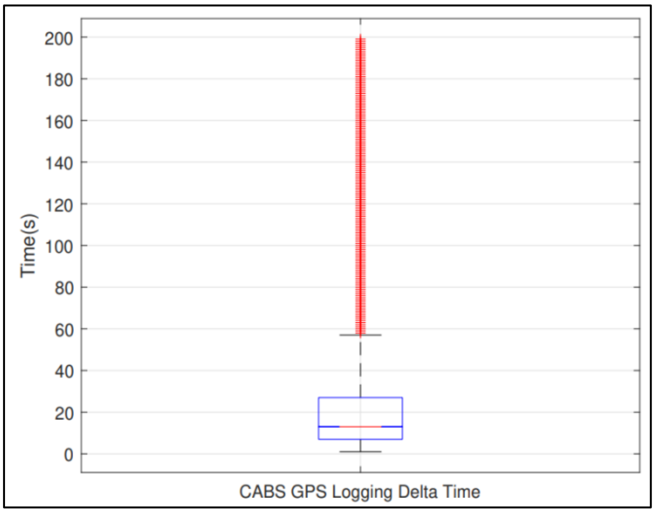


Figure 3: CABS GPS Timestamp delta distribution

- The traffic flow data was separately collected for three days which gives the number of vehicles for the vehicles at all five junctions in the network. The count data is used to generate turning ratios at each junction. *Figure 5* shows an example for turning ratios from the count data. The turning ratios are used as probability of turning in the SUMO simulation. For example, a vehicle coming from west direction in *Figure 5* as 30% probability of a left turn.
- Due to lack of pedestrian count data, the pedestrian flow is assumed to be constant at one pedestrian every 60 seconds at each crossing lane at the junctions. This frequency is used since the pedestrian walk green signal timing is fixed and it is activated upon pedestrians' request. The chosen frequency ensures that the pedestrian walk signal is requested in each cycle.

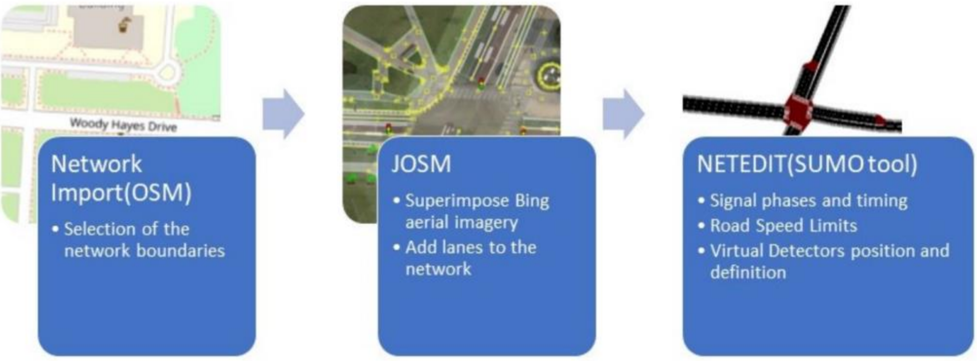

Woody Hayes Drive
Network Import(OSM)
• Selection of the network boundaries
JOSM
• Superimpose Bing aerial imagery
• Add lanes to the network
NETEDIT(SUMO tool)
• Signal phases and timing
• Road Speed Limits
• Virtual Detectors position and definition

Figure 4: Network Correction Process

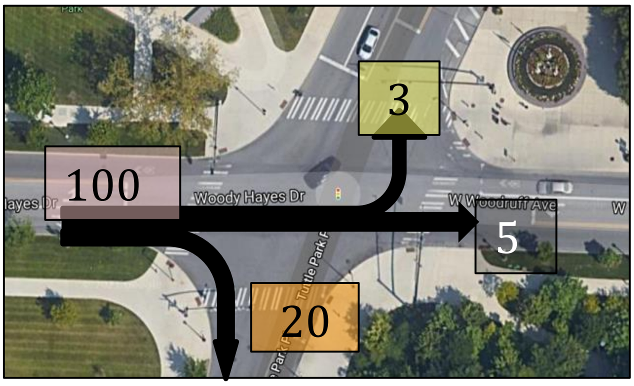


Figure 5: Turn Ratio Example

2.2 Vehicle Generation

SUMO has three packages to build the traffic in a network environment. The three packages are DFROUTER, JTRROUTER and DUAROUTER. In this work JTRROUTER is used to generate the vehicles. JTRROUTER requires turning ratios and vehicle flow probabilities from entry points in the network. The turning ratios are calculated from turn count data and doesn't change during a simulation run. The flow probabilities are used as a calibration parameter by the optimization algorithm.

2.3 Objective function and calibration parameters

The CABS GPS data is used as an indicator for the traffic density in the network. The entire network is divided into twelve sections. Six in West-East direction and six in East-West direction. A simple algorithm is developed which calculated the travel time in the network section as shown in *Figure 6*.

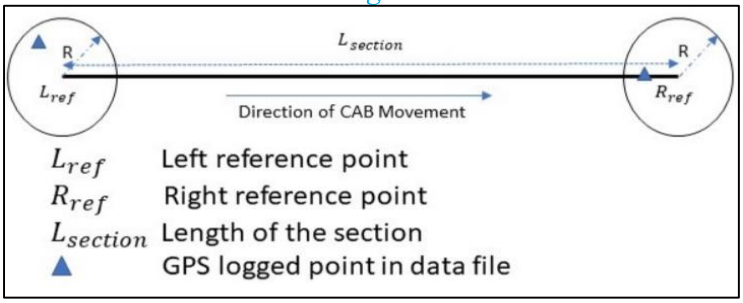

$L_{section}$
R
L_{ref}
R
R_{ref}
Direction of CAB Movement
L_{ref} Left reference point
R_{ref} Right reference point
$L_{section}$ Length of the section
▲ GPS logged point in data file

Figure 6: Network Section CABS Travel Time

GPS data has uncertainty in the position log. In this research we did not have access to the sensor information to quantify the uncertainty. Hence, as shown in *Figure 6*, a circle with radius R centered at the point of interest on the road is used as a region to find a GPS data point. The R used in the algorithm is 30m. Each section has two reference GPS coordinates, the start reference point (L_{ref}) and end section reference point (R_{ref}). The algorithm searches the GPS data file and if a data point is found within the circle around L_{ref} then the timestamp ($T_{L_{ref}}$) and position coordinates ($X_{L_{ref}}$) are stored in a memory. Then the algorithm finds the next timestamp ($T_{R_{ref}}$) when the GPS coordinates are in the circle centered at R_{ref}. The $T_{R_{ref}}$ and GPS position coordinates ($X_{R_{ref}}$) are stored in a separate memory buffer. To compute average travel time and speed across a junction, L_{ref} and R_{ref} are chosen on the road such that R_{ref} is upstream of the junction and L_{ref} immediately downstream of the junction. Timestamps of the CABS busses passing through these two points are logged using the algorithm described above. The time-stamps and corresponding GPS coordinates are used to compute average speed and average travel time.

$$\bar{V} = \frac{D_{GPS}}{\Delta T}$$

$$\bar{T}_{obs} = \frac{L_{section}}{\bar{V}}$$

Where $\Delta T = T_{R_{ref}} - T_{L_{ref}}$, $\mathrm{D_{GPS}} = f\left(\left|X_{R_{ref}} - X_{L_{ref}}\right|\right)$, $\bar{V}$ is the average velocity within the captured GPS points and $\overline{T}_{obs}$ is average travel time in the predefined section length. f is function to convert geolocations to linear distance. The SUMO simulation environment is also divided into twelve segments and the travel time in simulation for vehicles is compared with the travel time from CABS GPS data analysis.

The objective function used is a point mean relative error between measured travel time $T_{obs,j}$ and simulated travel time $T_{sim,j}$ for a section j. and the objective is to minimize this error. The genetic algorithm in the MATLAB optimization toolbox is used to minimize the function.

$$\underset{P_k}{\text{minimize}} \qquad \sqrt{\sum_{j=1}^{12} \left(\frac{T_{obs,j} - T_{sim,j}}{T_{obs,j}}\right)^2}$$

$$\text{Such that} \qquad T_{sim_j} = f_{SUMO}(P_k)$$

$$P_k \in [0.05\,,0.20] \quad \forall k = 1,2,\dots,12$$

where, k are the entry points of the network, j is the section number, P_k is the probability of vehicle generation at every one second, and f_{SUMO} is the SUMO traffic simulation model. The results of this simulation model are the of travel times for each section obtained from E3 detectors. The assumption here is that

$T_{sim,j} \propto \frac{1}{P_k}$. The flowchart in *Figure 7* shows the process for the optimization using GA.

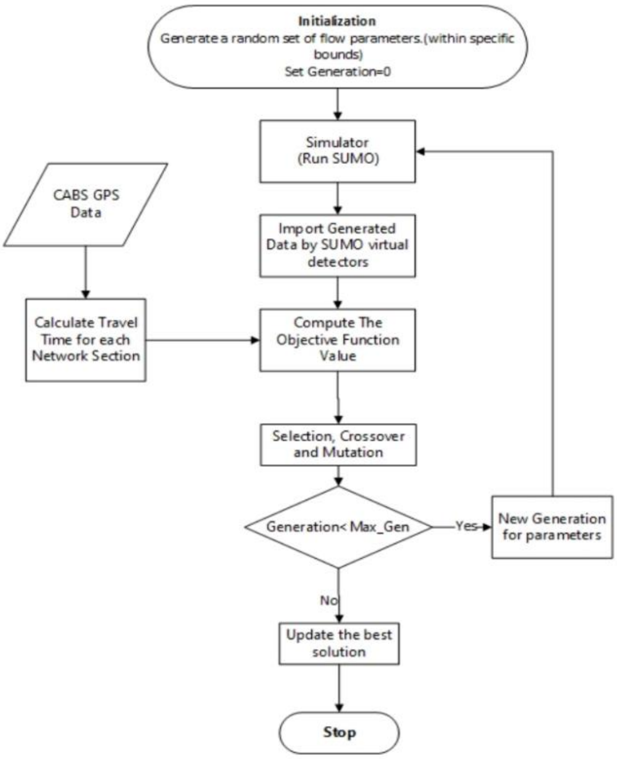

Initialization
Generate a random set of flow parameters.(within specific bounds)
Set Generation=0
Simulator
(Run SUMO)
CABS GPS Data
Import Generated Data by SUMO virtual detectors
Calculate Travel Time for each Network Section
Compute The Objective Function Value
Selection, Crossover and Mutation
Generation< Max_Gen
Yes
New Generation for parameters
No
Update the best solution
Stop

Figure 7: Genetic Algorithm Optimization Process

2.4 Calibration Results and Discussion

The maximum number of generations used for GA are ten. For each generation a population size of 150 is generated. Here each population refers to the calibration parameter P_k which is a vector of 12 entry probabilities in the network. From the *Figure 8* it can be observed that the objective function fitness value reaches steady state in the 4th generation and does not change from next generation. The *Figure 9* represents the average travel time comparison between travel time obtained from E3 detectors (T_{sim}), and the observed travel time from the CABS GPS data (T_{obs}) for each road section. In some of the sections the error is large and the primary reason for that is the inconsistency in the logged GPS data for CABS. If the delta time between CABS GPS timestamps can be reduced the uncertainty in the average travel time from calibration data set will reduce. The simulation average travel time doesn't have uncertainty because the virtual detectors can track the vehicle trajectory with a delta timestamp of 1 s which is very accurate compared to actual data with mean time of 15 s.

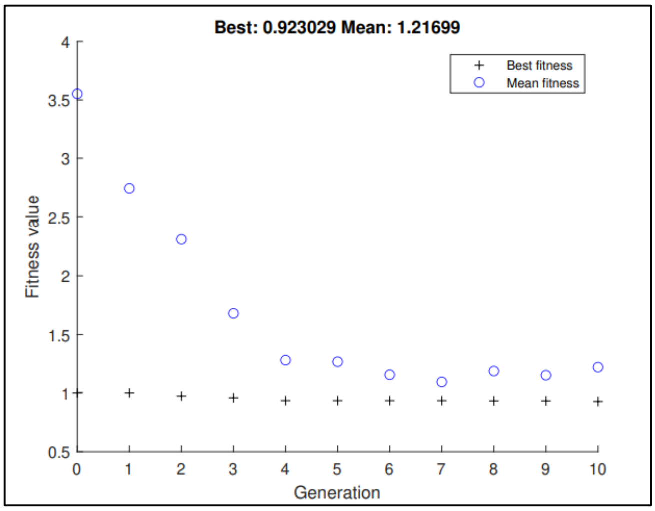


Figure 8: PMRE (Objective Function) Value Evolution

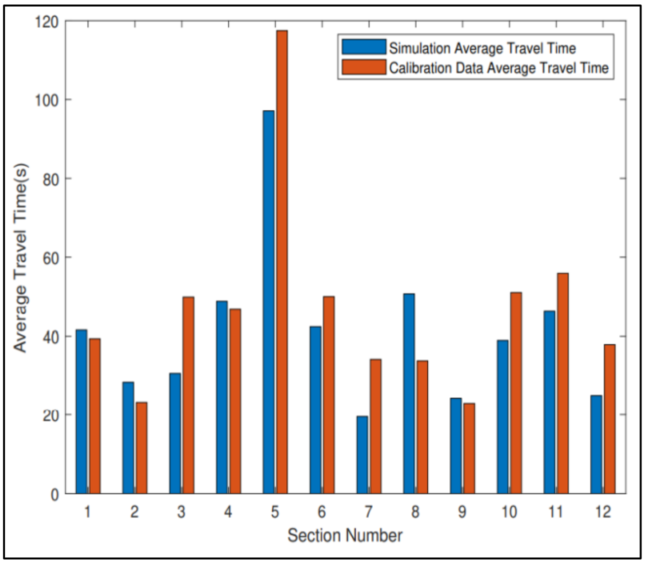


Figure 9: Average Travel Time Comparison

3 Hardware in Loop Simulator

In automotive research and development vehicle testing is done virtually using different x-In the Loop validation procedures. HIL setup is typically used for real time implementation of various electronic control units (ECU) along with communication channels. Although HIL testing is not new in automotive applications, the implementation of HIL for ADS testing is not trivial due to their extremely complex Opearational Design Domains (ODDs). Testing CAVs can be performed either in the real world with all the necessary hardware or parts of the environment could be simulated using virtual representation. The simulation-based approach is a much faster, safer, and efficient alternative compared to actual road testing[5]. The simulation for emulating the real environment involves a critical component of replicating the real traffic conditions in terms of road network and nearby vehicle behavior with respect to ego vehicle. In this paper a framework is created that integrates real time HIL setup(dSPACE) with large scale traffic simulator. The components for this framework are a high-fidelity simulation for ego vehicle dynamics compared to traffic vehicles and a communication interface with extrapolation algorithm to transfer the traffic information between traffic simulator and the HIL setup (dSPACE ASM)

3.1 Ego Vehicle Information Interface (EVI)

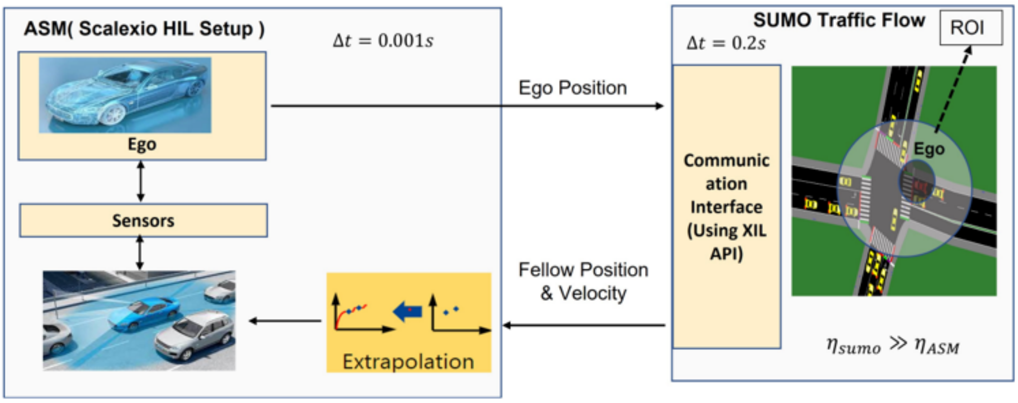

ASM(Scalexio HIL Setup)
$\Delta t = 0.001s$
Ego
Sensors
Extrapolation
Ego Position
Fellow Position & Velocity
SUMO Traffic Flow
$\Delta t = 0.2s$
ROI
Communication Interface (Using XIL API)
Ego
$\eta_{sumo} \gg \eta_{ASM}$

Figure 10: SUMO and dSpace ASM Co-Simulation Framework

The *Figure 10* shows how the information from the SUMO is passed to dSPACE ASM regarding the traffic vehicles and the feedback of ego vehicle speed and position from ASM to SUMO. The challenge in real time implementation is proper clock time synchronization between the two simulators. In SUMO TraCI is used with the command mentioned in *Table* . The SUMO simulator calculates the new position for traffic vehicles at every 0.2(5 Hz) and the ego vehicle dynamics model and ADS control is running on HIL at 0.001s (1000 Hz). The difference between the time update is handled using an extrapolation algorithm in the HIL setup which updates the position of traffic vehicles every 1ms till the nest update from SUMO at every 0.2s is available. Besides, rendering visualization and using high fidelity models for all the agents on the road is computationally inefficient and un-necessary. From ADS control perspective only fewer other significant traffic agents affect the perception and control of the ego vehicle. On the other hand, traffic simulation models simulate all the agents movements Hence, a region of interest (ROI) is calculated around the ego vehicle and the vehicles in this ROI are considered as nearby traffic vehicles. In this paper ROI with nearest 10 vehicles have been considered. Simultaneous to traffic vehicle information the traffic signal SPaT is also sent to dSPACE ASM. For SPaT only the ego vehicle nearest junction is considered as all the traffic vehicles are controlled by SUMO and do not need control command in the HIL simulator.

Table 2: Traci Commands for the EVI

TraCI Command	**Information**
traci.start()	Start the SUMO simulation
traci.simulationStep()	Progress simulation by one time step
traci.vehicle.getIDList()	IDs for all vehicle in SUMO scenario
traci.vehicle.getNextTLS('ego')	Get State Information for next traffic light in front of Ego
traci.vehicle.moveToXY()	Move Ego Vehicle to position specified by ASM
traci.vehicle.getPosition()	Get XY position for the fellow traffic in SUMO network
traci.vehicle.getLateralSpeed()	Get lateral speed in m/s t
traci.close()	Close the SUMO simulation
traci.vehicle.getAngle()	Get Yaw angle in degrees for traffic vehicles

3.2 Co-Simulation Scenario & Results Discussion

The scenario used in this paper is the calibrated traffic on Woody Hayes drive as shown in *Figure 12*. This figure shows the ego vehicle position in ASM network and its corresponding position in SUMO is passed. The ego vehicle travelled from West to East crossing six intersections. The *Figure 11* is the representation of the real time implementation of the communication interface. At t = 0 simulation is started and ego position is transferred to SUMO. Traffic vehicles positions are calculated and then transferred to dSpace ASM using an Ethernet interface. The SUMO and EVI execution time is around 150ms for the network used in this paper. The sumo step is paused till the clock time is 200ms and then next time step calculation starts in SUMO. As caching data to HIL takes time the writing of information about the fellow vehicles starts at 20ms before the indented clock time.

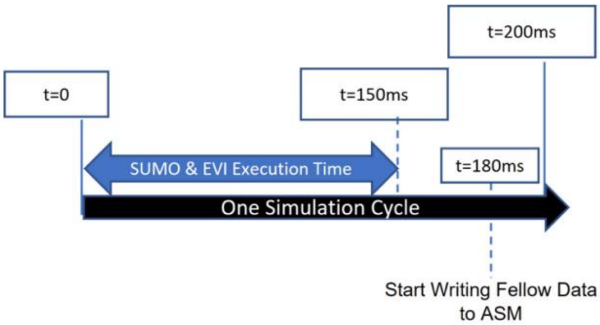


Figure 11: One Cycle of Simulation Synchronization: Repeats every 200ms

Figure 13 shows the extrapolation trajectory vs SUMO output for a fellow vehicle in the region of interest near ego vehicle. It can be observed that the extrapolated drifts from the actual path. The maximum deviation observed was 40 cm which can be reduced by reducing the 200ms SUMO time step to 100ms or lesser.

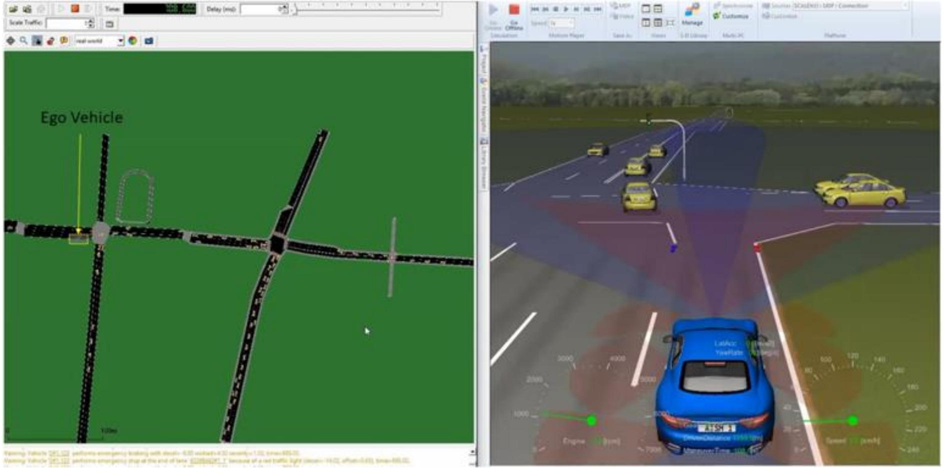


Figure 12: Co-Simulation Snapshot: ASM & SUMO

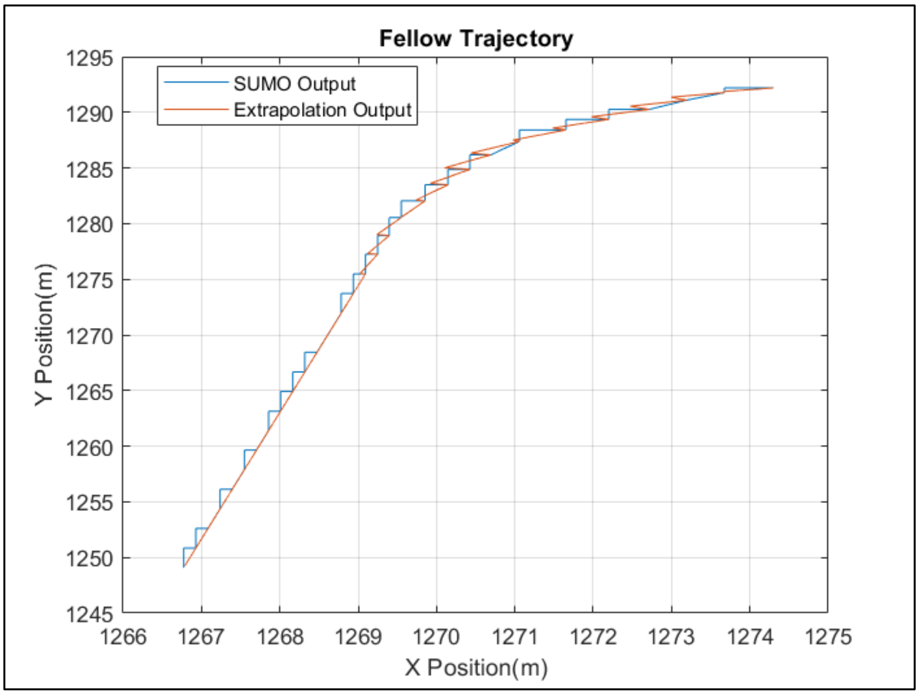


Figure 13: SUMO & Extrapolation Algorithm Fellow Trajectory Example

4 Conclusions

In this paper we presented an approach to calibrate the microscopic traffic simulation using limited dataset with genetic algorithm optimization approach. It is shown that the travel time across all the junctions could be approximately captured in the simulation using this approach. The benefit of the this approach is that it does not require traffic count data from infrastructure sensors. To demonstrate a use case of traffic simulation models in testing ADS features, the calibrated traffic simulation model in SUMO is then combined in a framework with the dSpace ASM HIL simulation. The framework uses region of interest concept to transfer only the relevant fellow traffic information to HIL simulator. This gives us an opportunity to create large network of roads in SUMO and link them with high fidelity vehicle powertrain and ADAS simulations. The time synchronization problem is solved by using the extrapolation between the larger timestep of SUMO.

The presented work can be extended to including pedestrian vehicle class in the HIL simulation. Also, the framework can be extended to other xIL simulation testing platforms such as camera in the loop. The traffic calibration will be improved using the vehicle counts from the camera detection installed on OSU campus traffic signals.

5 Definitions/Abbreviations

ASM	Automotive Simulation Models(dSpace)
SUMO	Simulation for Urban Mobility
SPaT	Signal Phase and Timing
GA	Genetic Algorithm
HIL	Hardware in Loop

References

[1] P. Tulpule, S. Midlam-Mohler, A. Karumanchi, and Y. Jin, “A Simulation Tool for Virtual Validation and Verification of Advanced Driver Assistance Systems,” *SAE Tech. Pap.*, no. 2021, pp. 1–9, 2021, doi: 10.4271/2021-01-0865.

[2] R. Dowling, A. Skabardonis, J. Halkias, G. McHale, and G. Zammit, “Guidelines for Calibration of Microsimulation Models: Framework and Applications:,” *https://doi.org/10.3141/1876-01*, no. 1876, pp. 1–9, Jan. 2004, doi: 10.3141/1876-01.

[3] H. Singh, S. Midlam-Mohler, and P. Tulpule, “Simulation Based Virtual Testing for Safety of ADAS Algorithms - Case Studies,” *SAE Tech. Pap.*, no. 2021, pp. 1–15, 2021, doi: 10.4271/2021-01-0114.

[4] L. Bieker-Walz, M. Behrisch, M. Junghans, and K. Gimm, “Evaluation of car-following-models at controlled intersections,” *31st Annu. Eur. Simul. Model. Conf. 2017, ESM 2017*, no. Krauss, pp. 238–243, 2017.

[5]. Hallerbach, Sven, et al. "Simulation-based identification of critical scenarios for cooperative and automated vehicles." *SAE International Journal of Connected and Automated Vehicles* 1.2018-01-1066 (2018): 93-106.

SUMO User Conference 2022
https://doi.org/10.52825/scp.v3i.119

Published: 29 Sept. 2022

Proposing a Simulation-Based Dynamic System Optimal Traffic Assignment Algorithm for SUMO: An Approximation of Marginal Travel Time

Behzad Bamdad Mehrabani[1,*][https://orcid.org/0000-0001-8585-7879], Jakob Erdmann[2][https://orcid.org/0000-0002-4195-4535], Luca Sgambi[1] and Maaike Snelder[3, 4][https://orcid.org/0000-0001-7766-2174]

[1] Université Catholique de Louvain, Louvain Research Institute for Landscape, Architecture, Built Environment (LAB), Louvain-la-Neuve, Belgium
[2] German Aerospace Center (DLR), Berlin, Germany
[3] Delft University of Technology, Civil Engineering Faculty, Delft, Netherlands
[4] Netherlands Organization for Applied Scientific Research (TNO), Hague, Netherlands

Abstract

User equilibrium (UE) and system optimal (SO) are among the essential principles for solving the traffic assignment problem. Many studies have been performed on solving the UE and SO traffic assignment problem; however, the majority of them are either static (which can lead to inaccurate predictions due to long aggregation intervals) or analytical (which is computationally expensive for large-scale networks). Besides, most of the well-known micro/ meso traffic simulators, do not provide a SO solution of the traffic assignment problem. To this end, this study proposes a new simulation-based dynamic system optimal (SB-DSO) traffic assignment algorithm for the SUMO simulator, which can be applied on large-scale networks. A new swapping/convergence algorithm, which is based on the logit route choice model, is presented in this study. This swapping algorithm is compared with the Method of Successive Average (MSA) which is very common in the literature. Also, a surrogate model of marginal travel time was implemented in the proposed algorithm, which was tested on real and abstract road networks (both on micro and meso scales). The results indicate that the proposed swapping algorithm has better performance than the classical swapping algorithms (e.g. MSA). Furthermore, a comparison was made between the proposed SB-DSO and the current simulation-based dynamic user equilibrium (SB-DUE) traffic assignment algorithm in SUMO. This proposed algorithm helps researchers to better understand the

[*] Corresponding author: behzad.bamdad@uclouvain.be

impacts of vehicles that may follow SO routines in future (e.g., Connected and Autonomous Vehicles (CAVs)).

1 Introduction

One of the most critical factors in the transportation planning process is solving the traffic assignment problem (Bamdad Mehrabani et al., 2021). Traffic assignment determines the routes that are used by vehicles based on certain behavioral principles, such as, for example, each vehicle seeks to minimize its own travel time. Many of the current traffic assignment algorithms are based on the two behavioral principles of Wardrop (Wardrop, 1952): (1) Wardrop's first principle: under user equilibrium (UE) conditions, no vehicle can unilaterally reduce its travel time by shifting to another route; (2) Wardrop's second principle: under system optimal (SO) conditions, traffic should be arranged in congested networks such that the average (or total) travel time is minimized. The first principle (UE) assumes that each vehicle attempts to minimize its own travel time (selfish routing). In contrast, in the second principle (SO), it is considered that each vehicle selects a route that minimizes not only its own travel time but also the entire network's travel time.

Traffic assignment methods can be broadly classified into two categories: 1) static traffic assignment (STA) and 2) dynamic traffic assignment (DTA) (Saw et al., 2015; Tsanakas, 2019). In STA, the traffic demand is static with respect to time and is typically used for strategic transportation planning. In DTA, the traffic demand is not static and varies over time, and the arrival time at a link is different from the departure time. Although the computational expenses of DTA are higher than those of STA, DTA attracts researchers owing to the several limitations of STA, for example, 1) limitations of static models because of the use of volume delay functions (such as no overtaking effects and no representation of the phenomenon of congestion spillback), 2) limitations in modeling of signal synchronization, 3) limitations in modeling of lane-based effects (such as high-occupancy vehicle lanes), and 4) limitations in modeling intelligent transportation system-related applications, such as traveler information systems (Chiu et al., 2011).

Different models exist in the literature to solve the DTA. The most important models are listed in Table 1.

Table 1: Different approaches of DTA

Model	Approach
Analytical Based Model	Mathematical Programming
	Optimal Control Formulations
	Variational Inequality-Based
Simulation-based Model	Micro Simulation
	Meso Simulation

Analytical models of DTA use analytical formulations to predict the propagation of traffic in a network (network loading). Traffic propagation models, which are used in analytical models, are typically based on extensions of the Lighthill–Whitham–Richards (LWR) model (Lighthill & Whitham, 1955; Richards, 1956). LWR is a macroscopic one-dimensional traffic model that uses traffic density and speed for traffic flow propagation (Li, 2016). The traffic flow propagation during dynamic network modeling can be based on the cell transmission model (Ziliaskopoulos, 2000) or link transmission model (Yperman, 2007). Although the mathematical closed-form is available for the analytical solution algorithm (thus, they are highly accurate), in practice, they cannot model certain phenomena (such as individual vehicles and vehicle interaction) in detail due to their macro-scale nature. Also, applying analytical assignment problems to large-scale networks may be highly time-consuming and complex to solve (Gawron, 1998). Simulation-based traffic assignment models use a traffic simulator to replicate the traffic flow dynamics (propagation) and spatio-temporal interactions (e.g., vehicle movements), which are based on micro/meso traffic flow simulation models (Saw et al., 2015). In addition, a traffic simulator is used as part of the search process to determine the optimal solution (Peeta & Ziliaskopoulos, 2001). They typically conduct several iterations to obtain optimum values (no closed-form is available). Compared to the analytical model, the simulation-based model appears to be more practical because of its ability to explain traffic flow propagation in more detail.

In the near future, several road users (e.g., connected and autonomous vehicles (CAVs)) are expected to follow Wardrop's second principle (SO) routines (Bagloee et al., 2017; Mansourianfar et al., 2021; J. Wang et al., 2019). It is important to provide a simulation-based dynamic system optimal (SB-DSO) traffic assignment model as a powerful tool to evaluate the impacts of such users. The dynamic system optimal (DSO) traffic assignment problem has been thoroughly studied in the literature. However, most of these works are based on an analytical model (e.g., (J. Liu et al., 2020; Ngoduy et al., 2021; Shen et al., 2006; Shen & Zhang, 2009; Tajtehranifard et al., 2018; Wie et al., 1990)) and very few studies utilize a simulation-based model which either are only microscopic or only mesoscopic (Ameli et al., 2020a; Hu et al., 2018; Mansourianfar et al., 2021; Peeta & Mahmassani, 1995; Sbayti et al., 2007; Yang & Jayakrishnan, 2012). Nevertheless, previous studies have demonstrated the superior performance of the simulation-based dynamic traffic assignment (SB-DTA) model (Ameli et al., 2020a), but many open-source (e.g., simulation of urban mobility (SUMO)) and commercial (e.g., Aimsun) traffic simulation software products do not provide SB-DSO traffic assignment algorithms. Therefore, this study contributes to the literature by proposing a SB-DSO traffic assignment algorithm based on a new swapping/convergence method that implements a logit route choice model for SUMO. The proposed algorithm can be applied on both micro and meso models of traffic flow which replaces the travel time of a link with a surrogate model of marginal travel times (MTT) to shift from DUE to DSO. To better understand the performance of the proposed algorithm, three case studies were conducted. A comparison was made between the proposed SB-DSO and the current simulation-based dynamic user equilibrium (SB-DUE) traffic assignment in SUMO. The remainder of this paper is organized as follows. The notations and abbreviations used in this paper are presented in section 2. In section 3, the literature is reviewed, followed by the research methodology (SB-DSO framework) in section 4. The proposed algorithm is applied to three case studies in section 5, and the conclusions are presented in Section 6.

2 Notations and Abbreviations

The list of all abbreviations used in this paper is included in Table 2. The notations used to present the proposed SB-DSO solution algorithm are listed in Table 3.

Table 2: The list of abbreviations in the paper

Abbreviation	Meaning	Abbreviation	Meaning
ADT	Average Distance Travelled	MSWA	Method of Successive Weighted Average
AS	Average Speed	MSAR	Method of Successive Average Ranking
ATL	Average Time Loss	MTT	Marginal Travel Time
CAV	Connected and Autonomous Vehicles	O-D	Origin-Destination pairs
DSO	Dynamic System Optimal	SA	Simulated Annealing
DTA	Dynamic Traffic Assignment	SB-DSO	Simulation-Based Dynamic System Optimal
DUE	Dynamic User Equilibrium	SB-DUE	Simulation-Based Dynamic User Equilibrium
DNL	Dynamic Network Loading	STA	Static Traffic Assignment
GA	Genetic Algorithm	SO	System Optimal
LWR	Lighthill–Whitham–Richards	TTT	Total Travel Time
MSA	Method of Successive Average	UE	User Equilibrium

Table 3: Notations

Indices	
c	Index for travel times (cost)
f	Index for traffic flow
i	Index for iteration steps
k	Index for path
Sets	
$G(V, A)$	traffic network
A	set of links ($a \in A$)
V	set of nodes ($v \in V$)
$J(R, S)$	set of vehicles ($j \in J$)
R	set of origin nodes ($r \in R$)
S	set of all destination nodes ($s \in S$)
I	set of simulation iterations ($i \in I$)
$P^{r-s}_{j,i}$	set of alternative paths for vehicle j in iteration i, travel from origin r to destination s
Variables, parameters, and elements	
c'_a	empty network travel time
c^i_a	travel time of link a in iteration i
$\bar{c}^i_a$	marginal travel time of link a in iteration i
C^i_k	travel time of path k in iteration i
$p^{r-s}_{j,i}$	selected path for vehicle j in iteration i, travel from origin r to destination s
$p^{*,r-s}_{j,i}$	adjusted selected path for vehicle j in iteration i, travel from origin r to destination s
$pr^i_{k,j}$	probability of selecting path k by vehicle j in iteration i
RSD^i_n	relative standard deviation of average travel time in the last n elements of i^{th} iteration
$av_{i'}$	the average travel time of the entire network in iteration i'

3 Literature Review

The Simulation-based DTA problem is split into two parts: 1- a simulation-based dynamic network loading (DNL) model and 2- an algorithm for finding the equilibrium solution (Ameli et al., 2020b). The DNL process explains the traffic flow dynamics and determines "how flows propagate with time through the network along the selected paths" (Jaume Barceló, 2010). A traffic simulator is used as the dynamic network-loading model in the simulation-based solution of the DTA problem. Whereas the second part determines the path used by the vehicles and the proportion of demand at each instant in time allocated to this determined path.

To find the equilibrium solution in simulation-based methods, usually, an iterative scheme is employed. These iterative methods start from an initial solution and update the path flow distribution for each iteration based on a path swapping algorithm. The reassignment process of vehicles in each iteration confirms whether the algorithm is in a descent direction or not. In other words, the algorithm forces vehicles at each iteration to follow a more efficient path than the previous iteration. Also, at the end of each iteration, a convergence criterion (or error) is calculated to check the algorithm's termination.

This approach was initially developed by Mahmassani and Peeta (Mahmassani & Peeta, 1993, 1995) and Peeta and Mahmassani (Peeta & Mahmassani, 1995). They incorporated a mesoscopic traffic simulator, DYNASMART (Jayakrishnan et al., 1994) (as the DNL model), in an iterative search solution framework to calculate the (marginal) travel times under the assumption of information availability for advanced traveler information system operations. The Method of Successive Averages (MSA) is used as the path swapping algorithm, and the convergence criterion is based on the differences in the number of vehicles assigned to various paths over successive iteration. The local approximation of MTT is evaluated by summing the link MTTs along the path according to the time-dependent link traversal times.

There are many studies in the literature based on the solution algorithm of Peeta and Mahmassani (Peeta & Mahmassani, 1995). For instance, Sbayti et al. (Sbayti et al., 2007) used MSA to solve the SB-DTA (with both DUE and DSO) in large-scale networks. They presented two new implementation techniques to address the disadvantages of MSA. Similar to Peeta and Mahmassani, the DYNASMART simulator was used to calculate the time-dependent link travel times, turn penalties, and link marginals. In addition, Yang and Jayakrishnan (Yang & Jayakrishnan, 2012) attempted to address the disadvantages of MSA in SB-DTA problems by implementing a gradient projection method. This study used the PARAMICS software in the DNL process. However, the travel times (generated by PARAMICS) are not directly fed into the route assignment procedure (the proposed gradient projection algorithm). As the proposed gradient projection algorithm requires an analytical function that represents link costs as traffic loads, link performance functions are used to calculate the path travel times in the path assignment process. An SB-DTA procedure was developed by Hu et al. (Hu et al., 2018) using a dynamic traffic simulator called DynaTAIWAN. The dynamic traffic simulator is used to simulate traffic flow distributions based on vehicle properties and routes (calculating the link travel time in each iteration). Four different vehicle class types (car, bus, motorcycle, and truck) and four different behavioral rules, including the pre-specified-path driver, UE driver, SO driver, and real-time information driver, are considered in the solution procedure. The MSA is applied to update the vehicles' path in each iteration. In a similar study, Mansourianfar et al. (Mansourianfar et al., 2021) developed an SB-DTA algorithm for mixed UE and SO users. They used the Aimsun traffic simulator instead of DynaTAIWAN to calculate the travel times. To address the shortages of MSA, they examined the method of successive weighted average (MSWA), which gives higher weights to later auxiliary flow patterns. This study proposes a new hybrid convergence criterion to find the mixed equilibrium solution. Another study that tried to overcome MSA's drawbacks is Ameli et al. (Ameli et al., 2020b). They study two new solution methods for the SB-DUE problem: a new extension of simulated annealing (SA) and an adapted genetic algorithm (GA). A comparison is made between the proposed meta-heuristic algorithms (SA and GA) and the classic methods (MSA, MSA ranking (MSAR)), and a gap-based algorithm). The results show that meta-heuristic algorithms dominate classical methods. However, it should be pointed out that all of the proposed meta-heuristic algorithm uses MSA as part of their solution. Also, this study implements a microscopic traffic simulator named Symuvia, which does not have meso modeling features for a trip-based dynamic simulation. Adopting the MATSIM software, Lämmel and Flötteröd (Lämmel & Flötteröd, 2009) developed an agent-based microsimulation DTA model. They replaced the travel time (based on which agents evaluate their routes) with the MTT to achieve SO. The results indicate that a simulation-based system leads to an acceptable approximation of the SO mathematical solution. There is also another group of studies on sustainable optimal DTAs (Chen et al., 2021; Lu et al., 2016). For instance, Lu et al. (Lu et al., 2016) solved an eco-system optimal DTA problem based on analytical and simulation-based models. Their study aimed to determine the SO ecological routes that minimize total vehicular emissions. The proposed simulation-based model combines macroscopic and microscopic traffic descriptions (mesoscopic) based on Newell's (Newell, 2002) simplified kinematic wave model and a simplified car-following model. In addition, this study introduces a novel approximation for path marginal emissions based on path MTT. Although the numerical examples of this study demonstrate the effectiveness of the model, it adopted a simplified car-following model (Newell) and not the commonly used car-following models, such as Krauß et al. (Krauß et al., 1997) and Gipps (Gipps, 1981).

As mentioned earlier, the traffic simulator software can be regarded as DNL of DTA (W. Wang et al., 2018); therefore, the available DTA solution methods in the most widely used and well-known micro/meso traffic simulator packages are presented in Table 4.

Table 4: Micro/Meso traffic simulation packages and their available DTA solution methods

Simulator	Developer	Scale	Available DTA Methods
Aimsun	J Barceló & Ferrer, 1997	Micro/Meso	Stochastic Route Choice
CONTRAM	Taylor, 2003	Meso	DUE
CORSIM	US-DOT, 1995	Micro	
DRACULA	R. Liu, 2010	Micro	
DTALite	Zhou and Taylor, 2014	Agent-based	
Dynameq	Mahut, 2001	Micro	
DynaMIT	Ben-Akiva et al., 1997	Meso	Converge to observed flows
DynaSMART	Jayakrishnan et al., 1994	Meso	Instantaneous information Predictive information DSO
DynusT	Y. C. Chiu et al., 2011	Meso	DUE
INTEGRATION	Van Aerde et al., 1996	Micro/Meso	
MATSIM	Dobler and Nagel, 2016	Agent-based	
PARAMICS	Smith et al., 1995	Micro	
PTV Vissim	Fellendorf, 1996	Micro/Meso	Stochastic Assignment
SUMO	Lopez et al., 2018	Micro/Meso	(Stochastic) DUE

Table 4 and the literature review revealed that, thus far, no study has proposed an SB-DSO algorithm using common traffic simulators (that can address the traffic flow propagation during dynamic network modeling with high accuracy in both micro and mesoscale simulations). Therefore, this study developed a new SB-DSO algorithm by implementing the SUMO traffic simulator, in which a new swapping algorithm (based on logit route choice model) and a new convergence criterion are incorporated.

4 Methodology

The simulation-based solution of the DTA problem does not include any closed-form analytical solution and typically relies on an iterative procedure. SUMO traffic simulator provides several tools and options for solving traffic assignment and route choice problem of vehicles (simulation-based approach). *duaIterate.py* (DLR, 2021) is the solution tool for the SB-DUE problem for micro and meso levels in SUMO. This study proposes a new solution framework for the SB-DSO problem based on *duaIterate.py*. The main difference between the proposed algorithm and the current algorithm of *duaIterate.py* is that the proposed algorithm replicates the travel time by a surrogate model of MTT. Also, a new swapping algorithm and convergence criterion are presented and tested against classical methods. Figure 1 illustrates the proposed solution framework for the SB-DSO problem. As shown in Figure 1, the framework consists of two parts: path selection procedure and DNL. For the path selection procedure, *duarouter*, which is an available algorithm in SUMO for the calculation of the shortest path, is used. For the DNL procedure, the SUMO traffic simulator is used.

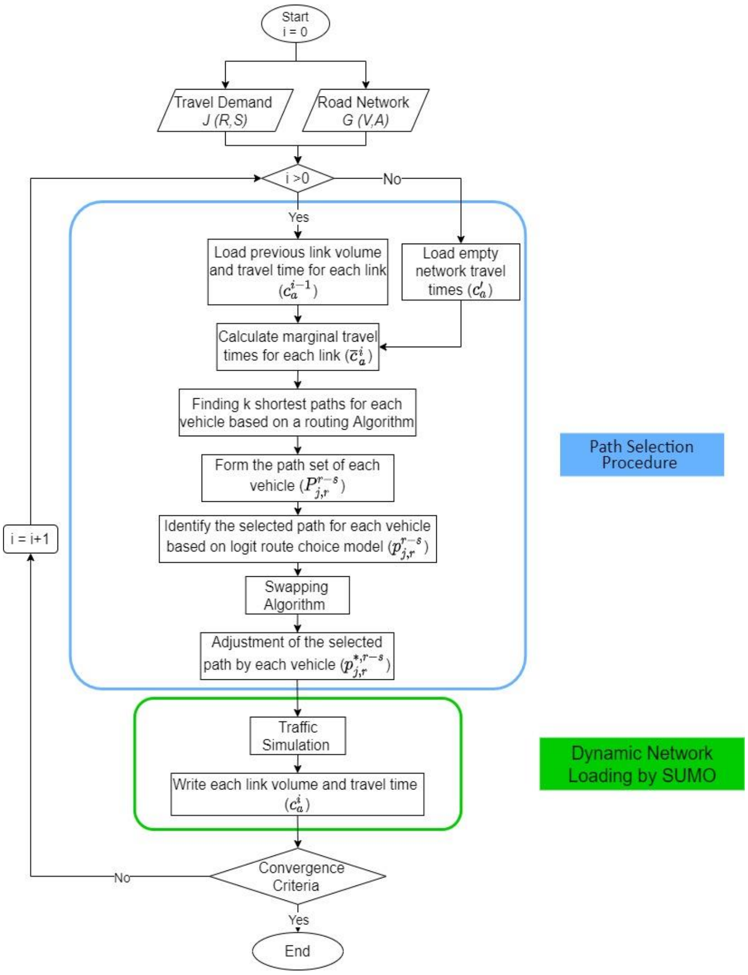

Start
i = 0
Travel Demand
J (R,S)
Road Network
G (V,A)
i >0
No
Yes
Load previous link volume and travel time for each link (c_a^{i-1})
Load empty network travel times (c_a')
Calculate marginal travel times for each link ($\overline{c}_a^i$)
Finding k shortest paths for each vehicle based on a routing Algorithm
Form the path set of each vehicle ($P_{j,r}^{r-s}$)
Identify the selected path for each vehicle based on logit route choice model ($p_{j,r}^{r-s}$)
Swapping Algorithm
Adjustment of the selected path by each vehicle ($p_{j,r}^{*,r-s}$)
Path Selection Procedure
i = i+1
Traffic Simulation
Write each link volume and travel time (c_a^i)
Dynamic Network Loading by SUMO
Convergence Criteria
No
Yes
End

Figure 1: Framework of the SB-DSO traffic assignment

Consider $G(V, A)$ as the directed traffic network, which includes a set of links A $(a \in A)$ and a set of nodes V $(v \in V)$. $J(R, S)$ represents the set of vehicles (demand file: usually imported to sumo by $Trips.XML$ file) between the origin and destination pairs where R $(r \in R)$ and S $(s \in S)$ denote the set of all origin nodes and the set of all destination notes, respectively. Hence, j^{r-s} is a vehicle that travels from origin r to destination s. The problem is stated as the assignment of $J(R, S)$ to $G(V, A)$. The equilibrium condition is computed by iteratively calculating the shortest routes and travel times. At each simulation iteration $i \in I$, first, a routing algorithm (Dijkstra, astar, Contraction Hierarchies (CH), or CHWrapper) is applied by *duarouter* to the road network to determine the set of alternatives paths, $P_{j,i}^{r-s}$, for each vehicle j^{r-s} (at each iteration, a new alternative path set is generated for each vehicle). The k-shortest paths are calculated using the previous simulation corresponding link MTT, $\bar{c}_a{}^{i-1}$. Next, a route choice model (Gawron, Logit, or Lohse) is applied to the set of alternative paths, $P_{j,i}^{r-s}$, to select a path, $p_{j,i}^{r-s}$ $(p_{j,i}^{r-s} \in P_{j,i}^{r-s})$. Then, a swapping algorithm is implemented to reassign a fraction of vehicles at each iteration (not all vehicles change their route necessary during successive iterations), ensuring the improvement of the selected path over iterations. Finally, the adjusted selected path of each vehicle, $p_{j,i}^{*,r-s}$ (known as *trips.rou.XML* file in SUMO), is sent to SUMO to perform the traffic simulation and consequently calculates the current travel time of each link, c_a^i (known as *edgedata* output in SUMO). The travel times written in this step, c_a^i, are used as an input in the next iteration step. By performing such a process iteratively, the total travel time (TTT) is minimized (SO condition).

4.1 Route Choice Model

In SUMO, it is possible to choose different route choice models among available alternatives, which are Gawron, Logit, or Lohse. In this study, the logit model is selected as the route choice model. Thus, the proposed algorithm computes the stochastic SB-DSO solution. The logit model is applied to each vehicle's set of alternative routes, $P_{j,i}^{r-s}$, in which the k-shortest paths for the subject vehicle are available. The travel times are considered as the cost for each alternative path. The travel time of each path is equal to the sum of the travel times of the corresponding links from the previous simulation. The logit model formulation is as follows

$$pr_{k,j}^i = \frac{\exp(-\theta C_k^i)}{\sum_1^k \exp(-\theta C_k^i)} \tag{1}$$

$$C_k^i = \sum\nolimits_{a \in A} \delta_{a,k}^i \, c_a^i \tag{2}$$

$$\delta_{a,p}^i = \begin{cases} 1 \; if \; link \; a \; is \; on \; path \; k \\ \quad 0 \; otherwise \end{cases} \tag{3}$$

where $Pr_{k,j}^i$ is the probability of selecting path k by vehicle j in iteration i ; C_k^i is the travel time (cost) of path k in iteration i; and θ is the logit model scale parameter. Given the multiple alternative routes with slightly different travel times, it may be reasonable to select a route other than the strictly shortest route (to avoid congestion on that route). Hence, the scale parameter θ assigns a probability for each route alternative. With a high value of theta, logit always selects the route with the least travel time, whereas with a low value of theta, logit selects all the routes with almost equal probability.

It should be mentioned that although this study works on the stochastic solution of the traffic assignment problem, it is possible to reach the deterministic solution by the same proposed algorithm (by replacing the route choice model with All-or-Nothing assignment) in SUMO.

4.2 Calculation of Marginal Travel Times

Previous studies have proven that the SO condition can be achieved by replacing the path travel time with the path MTT (Hu et al., 2018; Mansourianfar et al., 2021; Patriksson, 2015; Rahman et al., 2015). There are two ways to calculate the path MTT (Ameli et al., 2020a; Mansourianfar et al., 2021): 1) global approximation, which represents the changes in the total system travel time caused by an additional vehicle that is added to the path at a certain time interval, and 2) local approximation (Ghali & Smith, 1995; Peeta & Mahmassani, 1995), which represents the changes in the path travel time caused by an additional vehicle that is added to the route at a certain time interval. This approach considers the path MTT as a summation of the corresponding link MTTs. Although it has been proven that such a local approximation may lead to overestimation of the path MTT (Qian et al., 2012; Shen et al., 2007) it is a practical approximation in large-scale networks (Mansourianfar et al., 2021). On the other hand, because the global approximation of MTT is computationally expensive and is not practical for large-scale DTA, this study implements the local approximation of MTT. To achieve the local approximation of path MTT, first, the MTT of each link should be calculated; then, the MTTs of the corresponding links in each path are summed up. Numerous formulations exist in the literature to approximate the MTT of the links; however, it may be inappropriate to compare these formulations numerically as they use their own traffic flow propagation method and assumptions (Doan & Ukkusuri, 2015; Qian et al., 2012; Zhang & Qian, 2020). In addition, the traffic simulators (as the DNL model) used in each study are different. Sheffi (Sheffi, 1985) formulated the link MTT as follows and defined it as the "marginal contribution of an additional traveler on the a^{th} link to the TTT on this link." In other words, MTT is the derivative of the travel time with respect to flow:

$$\bar{c}_a(f_a) = c_a(f_a) + f_a \frac{dc_a(f_a)}{df_a} \tag{4}$$

This formulation is the sum of the two components. The first component, $c_a(f_a)$, is the travel time experienced by the additional traveler when the total link flow is f_a. This component can be explained by the average travel time on link a. The second component is the multiplication of $\frac{dc_a(f_a)}{df_a}$, the additional travel time burden that the additional traveler inflicts on each of the other travelers, by the number of travelers that already exist on the link (f_a). Therefore, the effect of one additional user on all the other travelers is considered by the second component.

Given that SUMO provides the average travel time of each link, it is not possible to calculate the additional travel time that one vehicle inflicts on the link. An alternative approach to compute the link MTT is to calculate the average travel time in successive iterations (with a different number of vehicles assigned to each link in each iteration) and compute the difference in link average travel time. Using this method, the average inflicted additional travel time on the link can be calculated. Therefore, in this study, we developed a surrogate model of MTT to achieve SO as follows:

$$\bar{c}_a^i = c_a^{i-1} + f_a^{i-1} \frac{c_a^{i-1} - c_a^{i-2}}{f_a^{i-1} - f_a^{i-2}} \tag{5}$$

where $\bar{c}_a^i$ is the surrogate MTT of link a at simulation step i; c_a^{i-1} and c_a^{i-2} are, respectively, the travel time (cost) of link a at simulation steps $i-1$ and $i-2$; and f_a^{i-1} and f_a^{i-2} are, respectively, the traffic flow of link a at simulation steps $i-1$ and $i-2$. The first term of this equation represents the average travel time of link a and the second term represents the average inflicted additional travel time on the link. In other words, the second term of this model indicates the extent to which adding one vehicle to a link leads to an increase in the travel time of the vehicles that are already in the link. In this way, instead of feeding only the average travel time that each vehicle would experience along a route, the additional cost that it imposes on the total travel time by selecting the route is added. Hence, the additional travel time that other vehicles must "pay" (on average) is addressed if the subject vehicle selects that route.

4.3 Swapping Algorithm

Most studies that implement simulation-based traffic assignment methods employ a swapping algorithm to reach the optimum value and avoid oscillating. The core idea of swapping algorithms is that not all vehicles should necessarily change their path in each iteration; instead, only a fraction of vehicles is in the reassignment process. Thus, a proper direction for the next iteration is obtained. Most of the previous studies apply the MSA as their swapping algorithm. The conventional MSA is calculated as:

$$f^i = \left(\frac{i}{i+1}\right) f^{i-1} + \left(\frac{1}{i+1}\right) y^i \tag{6}$$

In which f^i is the path flow distribution of iteration i, y^i is the auxiliary path assignments obtained by all-or-nothing assignment, and i is the number of iterations. In the primary iterations, the value of step size is too large, thus the travel time of vehicles does not reduce after several iterations. While in the last iterations the value of step size is too small, which leads to slow convergence speed. Therefore, previous studies provided several heuristic algorithms (Ameli et al., 2020b) or extensions of MSA (like Method of Successive Weighted Averages (MSWA) (H. X. Liu et al., 2009) or MSA Ranking (MSAR) (Sbayti et al., 2007)) to address the disadvantages of MSA. However, most of the previous studies implement these extensions in combination with deterministic traffic assignment (all-or-nothing assignment which is highly sensitive to the small changes in traffic flow).

This study proposes a new swapping algorithm for stochastic traffic assignment problems which is based on the logit route choice model. The new swapping algorithm is as follows:

$$p_{j,i}^{*,r-s} = \begin{cases} p_{j,i}^{r-s} & if\ x \geq \rho_i \\ p_{j,i-1}^{*,r-s} & if\ x < \rho_i \end{cases} \tag{7}$$

Where $p_{j,i}^{*,r-s}$ is the adjusted selected path by vehicle j in iteration i; $p_{j,i}^{r-s}$ is the selected path by vehicle j in iteration i from current logit model; $p_{j,i-1}^{*,r-s}$ the adjusted selected path by vehicle j in iteration $i-1$; x is a random variable between 0 and 1; and ρ_i is the sequence of step size in each iteration which can be considered as the probability of keeping the previous adjusted selected path. In this study ρ_i is predetermined $\rho_i = \frac{i}{\gamma}$; where i is the iteration number, and γ is a scale parameter. γ is a real number that determines the speed of convergence. With a low value of γ, the speed of the convergence is fast, but few alternative paths are tested by each vehicle. On the other hand, with a high value of γ, the convergence speed is slow, while several alternative paths (which are available in the path set) will be tested by each vehicle. Therefore, for stochastic assignments, it can be argued that higher values of γ are preferable. However, for large and medium scale networks, it is computationally expensive to wait for high number of iterations. In this study the value of γ is set to 100 and 50 for small scale and medium/large scale networks, respectively. This swapping algorithm prevents some vehicles, in successive iterations, from changing their routes, allowing them to follow the path they have chosen in the previous iteration. This ensures that the algorithm leads to the improvement of the selected path by each vehicle. For convenience of description, we name this swapping algorithm as “PSwap” (Probabilistic Swapping). PSwap is tested against the revised version of MSA in which the auxiliary path is obtained by the logit route choice model. In order to better understand the differences between two swapping algorithms, figure 2 shows the (maximum) fraction of vehicles that could change their routes per iteration.

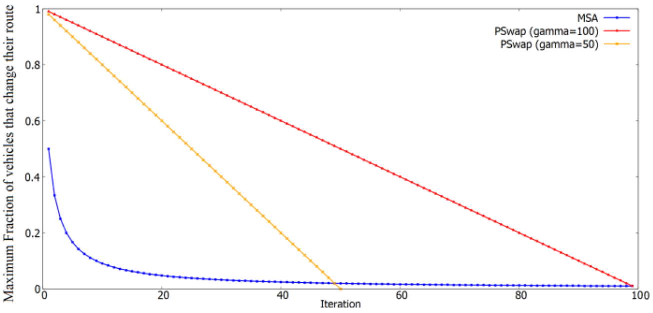


Figure 2: Maximum Fraction of vehicles that change their route per iteration

4.4 Convergence Criterion

Many studies in the past have provided different convergence criteria for the SB-DTA algorithm termination. As no closed-form is available for the simulation-based solutions, it is impossible to mathematically prove the algorithm's convergence. Therefore, all of the convergence criteria only provide some point where the algorithm can be terminated. A common approach in previous studies for the calculation of convergence criterion (error) is to calculate the maximum difference between the route flows of two iterations (e.g. (Peeta & Mahmassani, 1995)). However, previous studies pointed out that, this criterion does not guarantee the equilibrium condition since applying any swapping algorithm yield smaller route flow changes in successive iterations by default (Taale & Pel, 2015). Another common approach presented by previous studies is to calculate the relative gap between the current travel times of vehicles in each iteration and the least experienced travel times (Ameli et al., 2020b; Mansourianfar et al., 2021). However, the value to which the gap converges is not known beforehand, thus it is difficult to determine if convergence is reached (Taale & Pel, 2015).

As in equilibrium condition, no driver can unilaterally reduce his/her travel time by shifting to another route, when the standard deviation of average travel times between successive iterations is low, it implies that all remaining alternatives have "almost" the same travel time so the actual DUE criterion (or it's DSO equivalent) are basically fulfilled. Therefore, this study considers the relative standard deviation of average travel time (in the entire network) as convergence criteria:

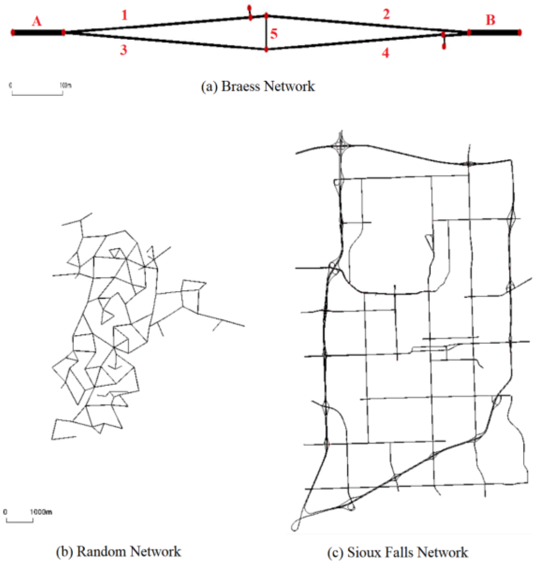
$$RSD_n^i = \frac{\sqrt{\frac{1}{n}\sum_{i'=(i-n)+1}^{i}(av_{i'} - \overline{av}_n^i)^2}}{\overline{av}_n} \tag{8}$$

$$\overline{av}_n = \frac{\sum_{i'}^{i} av_{i'}}{n} \tag{9}$$

Where RSD_n^i is the relative standard deviation of average travel time in the last n elements of i^{th} iteration; $av_{i'}$ is the average travel time of the entire network in iteration i'; and $\overline{av}_n$ is the mean of $av_{i'}$ in the last n iterations. This criterion evaluates the dispersion of average travel times in last n iterations. Low values of RSD_n^i represent that average travel time does not vary over successive iterations; thus, a decent point for termination is found. The proposed algorithm is considered being converged (after the minimum number of 10 iterations) if the value of RSD_n^i, become constant and less than ε (fixed at 0.05 for micro simulations and 0.005 for meso simulations).

5 Test Networks

The proposed algorithm was studied in three different networks (Figure 3): (a) small-size Braess like network, (b) medium-size abstract Random network, and (c) large size Sioux Falls network.



(a) Braess Network

(b) Random Network

(c) Sioux Falls Network

Figure 3: Test Networks

The Braess and Sioux Falls networks are commonly used as benchmarks in the literature. A random network was implemented to evaluate the performance of the algorithm in random unknown cases. In the following sections, the results of traffic simulations for the test networks are presented. For each network, four scenarios are simulated: (1) SB-DSO-PSwap, (2) SB-DSO-MSA, (3) SB-DUE-PSwap, (4) SB-DUE-MSA. The TTT in different iterations of each scenario is given first. The TTT is considered as the sum of travel times of all vehicles in the network and the waiting times of the vehicles which cannot insert into the network during the simulation. Then, traffic-related measures (including TTT (s), average speed (AS) (m/s), the average distance traveled (ADT) (m), and average time loss (ATL) (s)) of each scenario are evaluated. Finally, to have a better understanding of the differences between the proposed SB-DSO and the current SB-DUE, the traffic volume in the best iterations is illustrated.

5.1 Braess Network

In this study, a network similar to the Braess network (Figure 2 (a)) was simulated in four scenarios to evaluate the changes in TTT when the vehicles follow the proposed SB-DSO and DB-DUE. The microsimulation is performed with the demand of 1000 vehicles/h, departs from node A to destination B. Three routes are available for each vehicle. Each route includes a high-speed link (link 1 or 4), a low-speed link (link 2 or 3), and/or a high-speed shortcut (link 5). To analyze the behavior of the proposed algorithm (SB-DSO), the convergence patterns of the algorithms are presented in Figure 3 (value of TTT in successive iterations), while figure 4 shows the convergence pattern of the SB-DUE algorithm. For the sake of comparison, the fraction of vehicles that change their route per iteration is illustrated in figure 6.

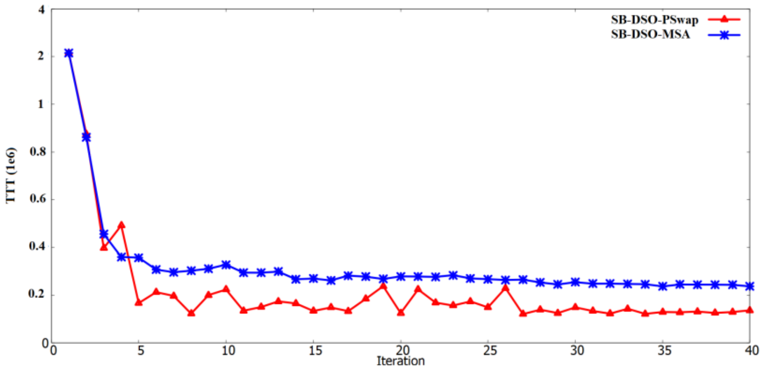


Figure 4: Convergence patterns for Braess network (SB-DSO)

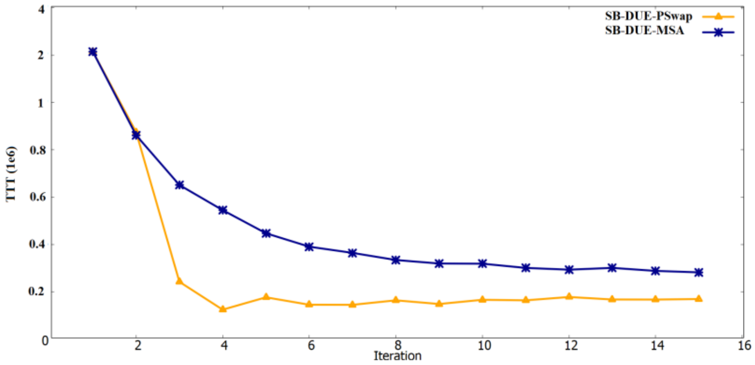


Figure 5: Convergence patterns for Braess network (SB-DUE)

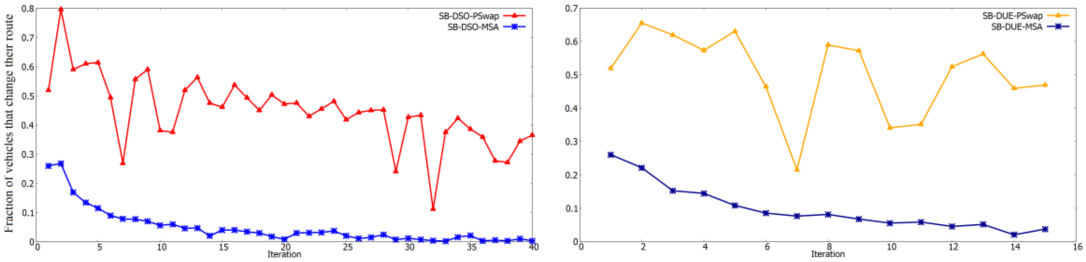


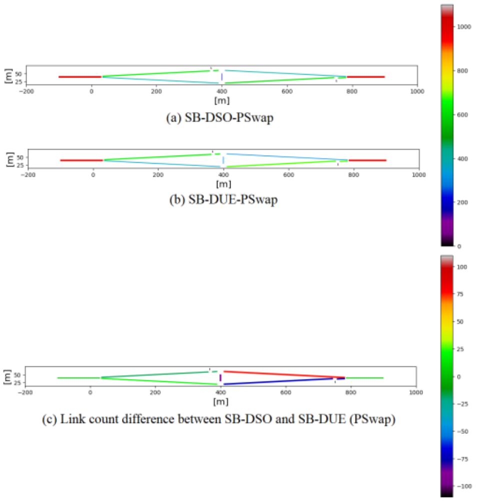
Figure 6: Fraction of vehicles that change their route per iteration (Braess Network)

Figures 4 and 5 show that MSA is dominated by PSwap. The TTT percentage difference (for the best iteration) between PSwap and MSA for the SB-DSO algorithm and SB-DUE algorithm are equal to 49.19% and 55.74%, respectively, which suggests the superior performance of the PSwap. The summary statistics for the system performance under different scenarios (best iterations) are reported in Table 5.

Table 5: Simulation Results for Braess Network

Scenario	TTT: Total Travel Time (s)	AS: Average Speed (m/s)	ADT: Average Distance Travelled (m)	ATL: Average Time Loss (s)
SB-DSO-PSwap	120130	8.69	1002.99	49.30
SB-DSO-MSA	236430	5.40	1007.10	160.82
SB-DUE-PSwap	125030	8.36	1006.01	56.74
SB-DUE-MSA	282490	4.85	1009.28	184.00
Percentage difference between PSwap and MSA (DSO) = 49.19%				
Percentage difference between PSwap and MSA (DUE) = 55.74%				
Percentage difference between DSO and DUE (PSwap) = 3.91%				
Percentage difference between DSO and DUE (MSA) = 16.3%				

As expected, for both swapping algorithms, the SB-DSO has lower TTT than SB-DUE. The Percentage TTT saving of SB-DSO over SB-DUE varies from 3.91% (for PSwap) and 16.30% (for MSA). In addition, an analysis of the data in Table 5 suggests that vehicle compliance with DSO routines increases vehicle AS and decreases vehicle ATL.



(a) SB-DSO-PSwap

(b) SB-DUE-PSwap

(c) Link count difference between SB-DSO and SB-DUE (PSwap)

Figure 7: Volume of the Braess network

To examine the superior performance of SB-DSO in more detail, the volume of the Braess network in SB-DSO-PSwap and SB-DUE-PSwap scenarios are illustrated in Figure 7. This figure also shows the traffic volume difference between these two scenarios. Figure 7 (c) demonstrates that when vehicles follow the SB-DSO, they not only use high-speed links (links 1, 5, and 4) to reach their destination, but they also use both high-speed and low-speed links simultaneously. In contrast, in the SB-DUE scenario, most vehicles tend to use high-speed links.

5.2 Random Network

A Random network was generated in SUMO (Figure 2 (b)) using *netgenerate*. This network consists of 278 edges and 100 junctions. Each edge has a minimum length of 200 and a maximum length of 1000 meters. The number of lanes is either one or two for each edge. A random traffic demand of 7200 vehicles was generated for a one-hour simulation. These vehicles were randomly distributed to the network. Similar to the previous test network, four scenarios were evaluated for the Random network: (1) SB-DSO-PSwap, (2) SB-DSO-MSA, (3) SB-DUE-PSwap, and (4) SB-DUE-MSA. The microsimulation is performed for the scenarios, all of which converged after 17 iterations. The convergence pattern of the SB-DSO and SB-DUE algorithms are displayed in Figure 8 and Figure 9, respectively. Besides, the fraction of re-routing vehicles per iteration for each scenario is given in figure 10.

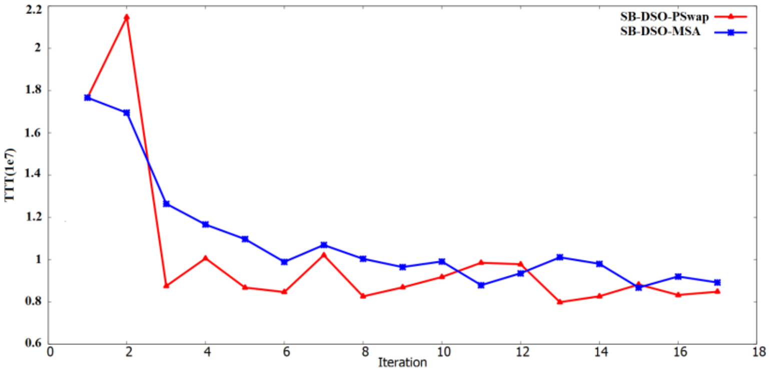


Figure 8: Convergence patterns for Random network (SB-DSO)

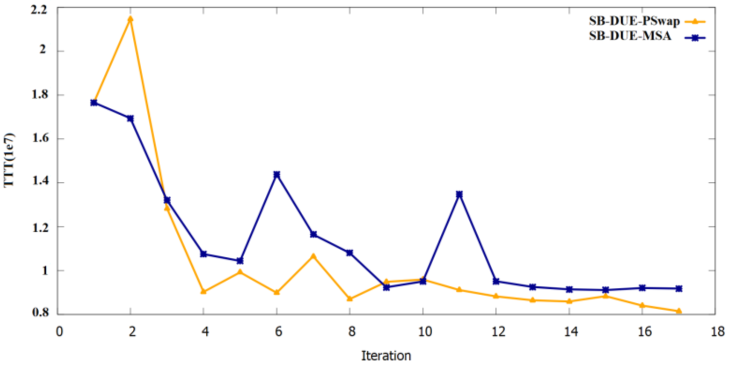


Figure 9: Convergence patterns for Random network (SB-DUE)

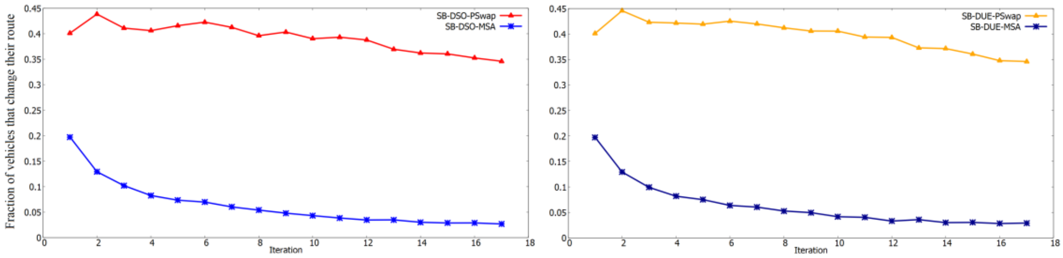


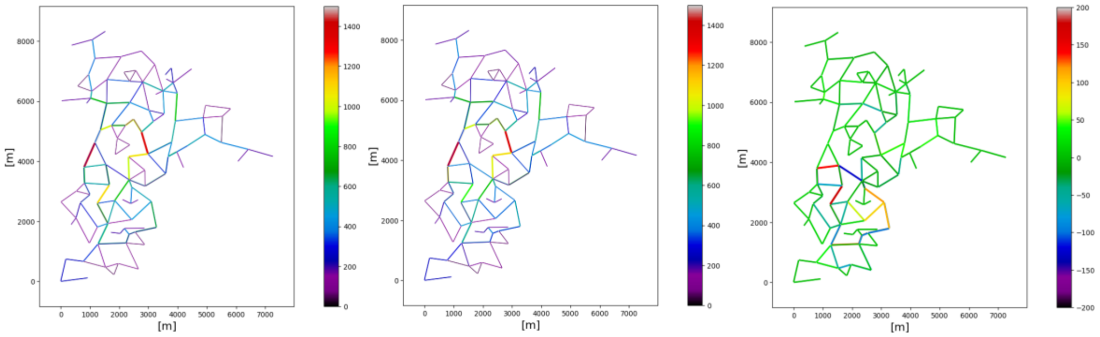
Figure 10: Fraction of vehicles that change their route per iteration (Random Network)

For both SB-DSO and SB-DUE, the value at convergence for PSwap is lower than MSA. Also, PSwap has better performance in terms of stability of the convergence and the TTT average value. The TTT percentage difference between PSwap and MSA for SB-DSO and SB-DUE are equal to 7.95% and 10.53%, respectively, which supports that PSwap has superior acting. The results of the optimum iteration for each scenario are presented in Table 6.

Table 6: Simulation Results for Random Network

Scenario	TTT: Total Travel Time (s)	AS: Average Speed (m/s)	ADT: Average Distance Travelled (m)	ATL: Average Time Loss (s)
SB-DSO-PSwap	7981776	8.77	6850.82	578.09
SB-DSO-MSA	8671176	8.52	6571.42	677.58
SB-DUE-PSwap	8157456	8.70	6787.33	599.06
SB-DUE-MSA	9118296	8.51	6581.33	761.72
Percentage difference between PSwap and MSA (DSO) = 7.95% Percentage difference between PSwap and MSA (DUE) = 10.53% Percentage difference between DSO and DUE (PSwap) = 2.15% Percentage difference between DSO and DUE (MSA) = 4.90%				

As expected, the TTT value decreased by 2.15% (PSwap) and 4.90% (MSA) in the SB-DSO scenario compared to that in the SB-DUE scenario. In addition to the reduction in TTT, we observed a decrease of 3.5% (PSwap) and 11% (MSA) in ATL. However, the ADT by vehicles in the SB-DSO scenario shows a slight increment over the SB-DUE scenario. This suggests that vehicles do not necessarily select routes that have the shortest distance in the SB-DSO scenario but rather select those that reduce the travel time of the entire network (TTT).



(a) SB-DSO-PSwap

(b) SB-DUE-PSwap

(c) Link count difference between SB-DSO and SB-DUE (PSwap)

Figure 11: Volume of the Random network

Figure 11 shows traffic volume in the Random network in different scenarios. The comparison of Figures 11 (a) and Figure 11 (b) reveals that in the SB-DSO scenario, the number of medium-volume links is slightly more than that of the SB-DUE scenario. This indicates that in the SB-DSO scenario, the traffic volume is distributed throughout the entire network. In contrast, in the SB-DUE scenario, only a limited number of short links are used. A similar result can be obtained by observing Figure 8 (c). In Figure 8 (c), the red highlighted links represent the links used by vehicles in the SB-DSO scenario and not in the SB-DUE scenario, while the green highlight indicates the links in which there is no difference in traffic volume between the two scenarios.

5.3 Sioux Falls Network

The latest case study in this article is the Sioux Falls network (Figure 3 (c)). The total number of simulated vehicles was 36000 which were distributed on different origins and destinations based on the demand pattern of LeBlanc's study (LeBlanc et al., 1975). In order to check the performance of the proposed algorithm on the mesoscale, the meso simulation feature of SUMO is implemented for this test network. As in the previous test networks, four scenarios were analyzed. The SB-DSO-PSwap and SB-SO-MSA scenarios converged after 35 iterations (Figure 12), while the SB-DUE-PSwap and SB-DUE-MSA scenarios converged after 25 iterations (Figure 13). The fraction of re-routing vehicles per iteration is also given in figure 14.

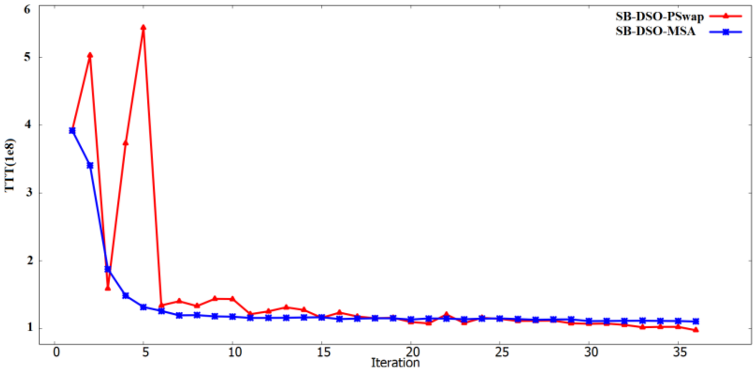


Figure 12: Convergence patterns for Sioux Falls network (SB-DSO)

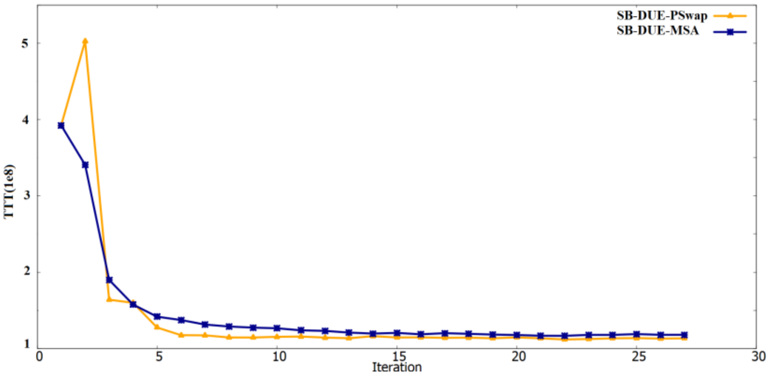


Figure 13: Convergence patterns for Sioux Falls network (SB-DUE)

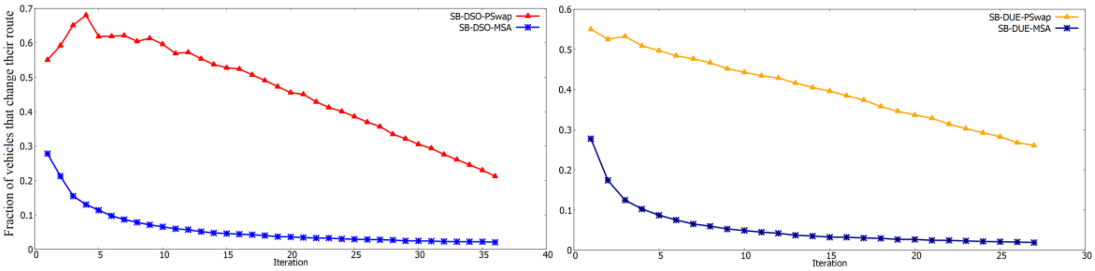


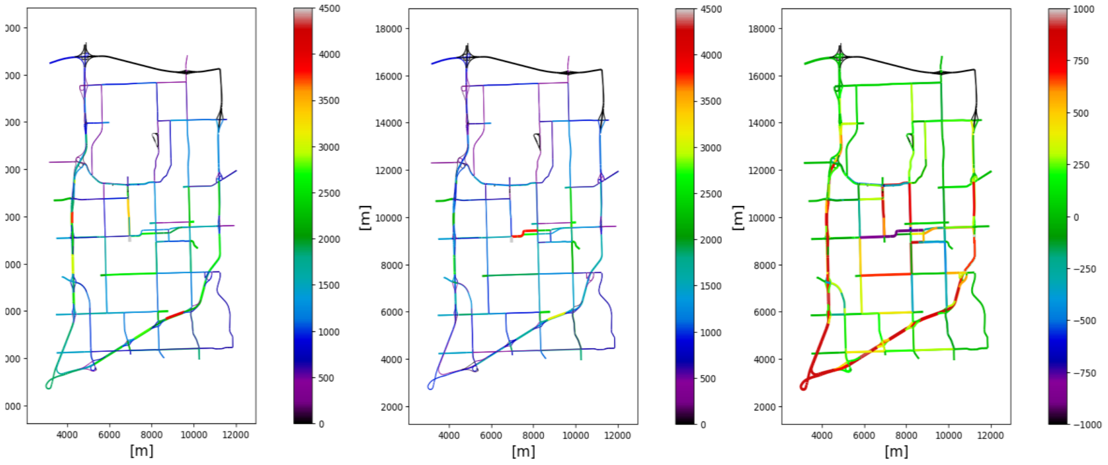
Figure 14: Fraction of vehicles that change their route per iteration (Sioux Falls Network)

Similar to previous test networks, PSwap has a better value at convergence than MSA. For the SB-DSO-PSwap scenario, the value at convergence is 11.64% lower than that of SB-DSO-MSA. For the SB-DUE algorithm, we can see a similar behavior where the TTT percentage difference between SB-DUE-PSwap and SB-DUE-MSA is 4.1%. The results of the meso simulations are presented in Table 7, which provides traffic-related measures for the Sioux Falls network.

Table 7: Simulation Results for Sioux Falls Network

Scenario	TTT: Total Travel Time (s)	AS: Average Speed (m/s)	ADT: Average Distance Travelled (m)	ATL: Average Time Loss (s)
SB-DSO-PSwap	97488027.9	12.75	9726.57	479.54
SB-DSO-MSA	110334105.4	12.75	9833.79	505.72
SB-DUE-PSwap	111853402.2	11.98	8164.91	526.21
SB-DUE-MSA	116637655.1	11.71	8384.31	595.16
Percentage difference between PSwap and MSA (DSO) = 11.64%				
Percentage difference between PSwap and MSA (DUE) = 4.1%				
Percentage difference between DSO and DUE (PSwap) = 12.84%				
Percentage difference between DSO and DUE (MSA) = 5.4%				

The simulation results indicate that the proposed SB-DSO traffic assignment leads to a reduction of 12.84% (PSwap) and 5.4% (MSA) in TTT and 8.86% (PSwap) and 15% (MSA) in ATL. In addition, the AS in SB-DSO was improved compared to that in SB-DUE (6% and 8.15% increase in AS for PSwap and MSA, respectively). However, ADT has increased in the DSO condition (16% and 14.7% increment of ADT for PSwap and MSA, respectively).



(a) SB-DSO-PSwap

(b) SB-DUE-PSwap

(c) Link count difference between SB-DSO and SB-DUE (PSwap)

Figure 15: Volume of the Sioux Falls network

Figure 15 shows the volume on the Sioux Falls network for the proposed SB-DSO-PSwap and the current SB-DUE-PSwap, categorized based on both colors and width, where thicker links indicate higher volume. Focusing on the difference between the two traffic assignment methods, Figure 15 (c) shows that vehicles are distributed among the entire network in SB-DSO scenarios and do not intend to use specific short links that minimize their own travel time (avoiding selfish routing). Instead, they select unused links, minimizing the travel time in the entire network. In other words, the presence of short minor roads may have a significant impact on SB-DSO versus SB-DUE. This is not because the network can handle more load but because shorter roads provide opportunities for selfish shortcuts that are prone to jamming.

6 Conclusion

SO and UE traffic assignments are among the most important traffic assignment methods and have been extensively investigated for several years. Although recent studies suggest that several road users (e.g., CAVs) are soon expected to follow the SO principle, most traffic simulation packages do not provide it. In addition, the majority of available SO solutions for CAVs are either static or analytical which have their own drawbacks. On the other hand, the available simulation-based methods are based on the MSA (or its extensions) algorithm, which has its own drawbacks. Therefore, this study proposes a new SB-DSO traffic assignment algorithm that replaces the travel times of links with a surrogate model of the MTT. The logit route choice model is incorporated in the solution algorithm. At each iteration, the route choice model is applied to the path set of each vehicle. A new swapping algorithm (Called PSwap) is presented, which is based on the logit route choice model to address the disadvantages of MSA (that uses an all-or-nothing assignment). The swapping algorithm prevents all vehicles from changing their routes in successive iterations. The proposed algorithm is tested on two classical case studies (Braess and Sioux Falls network) and a random network to assess its performance (both micro and meso scale). For each test network, four scenarios have been simulated: (1) SB-DSO-PSwap, (2) SB-DSO-MSA, (3) SB-DUE-PSwap, and (4) SB-DUE-MSA. The results of the simulations show that MSA is dominated by PSwap in all of the scenarios. Also, a comparison of the proposed SB-DSO (scenarios 1 and 2) and current SB-DUE (scenarios 3 and 4) traffic assignment algorithm is provided. We observed remarkable decreases in the TTT when vehicles followed the SB-DSO. The maximum percentage of TTT reduction was for the Braess network (16.3%), followed by 12.84% and 4.9% for Sioux Falls and Random networks, respectively. In summary, the proposed SB-DSO-PSwap has the least amount of TTT. These results indicate that if road users (such as CAVs) follow SO routines in the future, a significant reduction in travel time and pollution of the entire network can be obtained, which, accordingly, reduces several costs.

One of the most critical factors in the superiority of SB-DSO over SB-DUE is the presence of short minor roads, as these routes provide opportunities for selfish routing. Therefore, the proposed algorithm may not necessarily improve the TTT in networks where short minor roads (or alternative routes) are not present. Also, it should be pointed out that the number of teleporting vehicles has a considerable impact on the results of the TTT. Therefore, it is not appropriate to evaluate the performance of the algorithm if the number of teleporting vehicles is very high.

The proposed algorithm is freely available under the EPLv2 license on GitHub (Eclipse, 2022) by setting *--marginal-cost*, *--marginal-cost.exp*, and *--convergence-steps* options in *duaIterate.py*. As the MTTs in the proposed algorithm are calculated based on a local approximation, it may lead to its overestimation. Therefore, it is recommended to remove the second term of the MTT equation in case of inappropriate results (by removing *--marginal-cost.exp* option). This tool helps researchers and decision-makers in evaluating the effect of SO-seeking users (e.g., CAVs) on the road network in terms of traffic and environment-related issues. Beyond solving the SB-DTA problem by a new swapping algorithm, proposing a surrogate model for MTT is helpful in many road network management applications, such as providing marginal cost-based tolls.

Future studies should estimate the global approximation of MTTs. In this study, the examined scenarios were scenarios in which all vehicles followed either DSO or DUE rules; therefore, it is suggested that a mixed traffic algorithm should be developed in future research. In addition, owing to the importance of the demand level in traffic assignment, researchers are encouraged to assess the impact of different demand levels on the proposed SB-DSO traffic assignment algorithm. Another direction for future research is the comparison of environmental-related measures in DSO and DUE conditions.

7 Contribution and Disclosure statement

Behzad Bamdad Mehrabani: Conceptualization, Methodology, Software, Investigation, Data Curation, Writing – original draft, Visualization. **Jakob Erdmann**: Methodology, Software, Validation, Formal Analysis, Resources, Data Curation, Writing – review & editing. **Luca Sgambi**:, Validation, Writing – review & editing, Supervision, Project Administration, Funding Acquisition. **Maaike Snelder**: Conceptualization, Methodology, Validation, Writing – review & editing.

No potential conflict of interest was reported by the author(s).

References

Ameli, M., Lebacque, J.-P., & Leclercq, L. (2020a). Improving traffic network performance with road banning strategy: A simulation approach comparing user equilibrium and system optimum. *Simulation Modelling Practice and Theory*, *99*, 101995. https://doi.org/https://doi.org/10.1016/j.simpat.2019.101995

Ameli, M., Lebacque, J., & Leclercq, L. (2020b). Simulation-based dynamic traffic assignment: Meta-heuristic solution methods with parallel computing. *Computer-Aided Civil and Infrastructure Engineering*, *35*(10), 1047–1062.

Bagloee, S. A., Sarvi, M., Patriksson, M., & Rajabifard, A. (2017). A mixed user-equilibrium and system-optimal traffic flow for connected vehicles stated as a complementarity problem. *Computer-Aided Civil and Infrastructure Engineering*, *32*(7), 562–580. https://doi.org/https://doi.org/10.1111/mice.12261

Bamdad Mehrabani, B., Sgambi, L., Garavaglia, E., & Madani, N. (2021). Modeling Methods for the Assessment of the Ecological Impacts of Road Maintenance Sites. In P. Singh (Ed.), *Environmental Sustainability and Economy 1st Edition*. Elsevier. https://www.elsevier.com/books/environmental-sustainability-and-economy/singh/978-0-12-822188-4

Barceló, J, & Ferrer, J. L. (1997). Advanced Interactive Microscopic Simulator for Urban Network. *Departement d'Estadística i Investigació Operativa, Universitat Politèctina de Catalunya, User's Manual Edn*.

Barceló, Jaume. (2010). *Fundamentals of traffic simulation* (Vol. 145). Springer. https://doi.org/10.1007/978-1-4419-6142-6_1

Ben-Akiva, M. E., Koutsopoulos, H. N., Mishalani, R. G., & Yang, Q. (1997). Simulation laboratory for evaluating dynamic traffic management systems. *Journal of Transportation Engineering*, *123*(4), 283–289. https://doi.org/https://doi.org/10.1061/(ASCE)0733-947X(1997)123:4(283)

Chen, R., Becarie, C., & Leclercq, L. (2021). Learning link marginals from dynamic simulation to calculate sustainable system optimum. *HEART 2020, 9th Symposium of the European Association for Research in Transportation-Virtual Conference*, 13p. https://hal.archives-ouvertes.fr/hal-03154849

Chiu, Y. C., Nava, E., Zheng, H., & Bustillos, B. (2011). DynusT user's manual. In *Department of Engineering, University of Arizona, Tucson*.

DLR. (2021). *SUMO User Documentation*. https://sumo.dlr.de/docs/index.html

Doan, K., & Ukkusuri, S. V. (2015). Dynamic system optimal model for multi-OD traffic networks with an advanced spatial queuing model. *Transportation Research Part C: Emerging Technologies*, *51*, 41–65. https://doi.org/https://doi.org/10.1016/j.trc.2014.10.011

Dobler, C., & Nagel, K. (2016). Within-day replanning. In *The Multi-Agent Transport Simulation MATSim* (pp. 187–200). Ubiquity Press. https://doi.org/http:// dx.doi.org/10.5334/baw.30

Eclipse. (2022). *Eclipse SUMO*. https://github.com/eclipse/sumo/blob/master/tools/assign/duaIterate.py

Fellendorf, M. (1996). VISSIM for traffic signal optimisation. *Traffic Technology International'96. Annual Review Issue*. http://worldcat.org/issn/13569252

Gawron, C. (1998). *Simulation-Based Traffic Assignment. Computing user equilibria in large street networks* [Universität zu Köln]. http://kups.ub.uni-koeln.de/id/eprint/9257

Ghali, M. O., & Smith, M. J. (1995). A model for the dynamic system optimum traffic assignment problem. *Transportation Research Part B: Methodological*, *29*(3), 155–170. https://doi.org/https://doi.org/10.1016/0191-2615(94)00024-T

Gipps, P. G. (1981). A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, *15*(2), 105–111. https://doi.org/https://doi.org/10.1016/0191-2615(81)90037-0

Hu, T.-Y., Tong, C.-C., Liao, T.-Y., & Chen, L.-W. (2018). Dynamic route choice behaviour and simulation-based dynamic traffic assignment model for mixed traffic flows. *KSCE Journal of Civil Engineering*, *22*(2), 813–822. https://doi.org/10.1007/s12205-017-1025-8

Jayakrishnan, R., Mahmassani, H. S., & Hu, T.-Y. (1994). An evaluation tool for advanced traffic information and management systems in urban networks. *Transportation Research Part C: Emerging Technologies*, *2*(3), 129–147. https://doi.org/https://doi.org/10.1016/0968-090X(94)90005-1

Krauß, S., Wagner, P., & Gawron, C. (1997). Metastable states in a microscopic model of traffic flow. *Physical Review E*, *55*(5), 5597. https://doi.org/https://doi.org/10.1103/PhysRevE.55.5597

Lämmel, G., & Flötteröd, G. (2009). Towards system optimum: Finding optimal routing strategies in time-dependent networks for large-scale evacuation problems. *Annual Conference on Artificial Intelligence*, 532–539. https://doi.org/https://doi.org/10.1007/978-3-642-04617-9_67

LeBlanc, L. J., Morlok, E. K., & Pierskalla, W. P. (1975). An efficient approach to solving the road network equilibrium traffic assignment problem. *Transportation Research*, *9*(5), 309–318.

Li, C. (2016). *Multi-agent dynamic traffic assignment incorporating GPS diary and personalized supernetwork approach* [Eindhoven University of Technology]. https://research.tue.nl/en/studentTheses/multi-agent-dynamic-traffic-assignment-incorporating-gps-diary-an

Lighthill, M. J., & Whitham, G. B. (1955). On kinematic waves II. A theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, *229*(1178), 317–345. https://doi.org/https://doi.org/10.1098/rspa.1955.0089

Liu, H. X., He, X., & He, B. (2009). Method of successive weighted averages (MSWA) and self-regulated averaging schemes for solving stochastic user equilibrium problem. *Networks and Spatial Economics*, *9*(4), 485–503.

Liu, J., Mirchandani, P., & Zhou, X. (2020). Integrated vehicle assignment and routing for system-optimal shared mobility planning with endogenous road congestion. *Transportation Research Part C: Emerging Technologies*, *117*, 102675. https://doi.org/https://doi.org/10.1016/j.trc.2020.102675

Liu, R. (2010). Traffic simulation with DRACULA. In *Fundamentals of traffic simulation* (pp. 295–322). Springer. https://doi.org/https://doi.org/10.1007/978-1-4419-6142-6_8

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. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2575–2582. https://doi.org/10.1109/ITSC.2018.8569938

Lu, C.-C., Liu, J., Qu, Y., Peeta, S., Rouphail, N. M., & Zhou, X. (2016). Eco-system optimal time-dependent flow assignment in a congested network. *Transportation Research Part B: Methodological*, *94*, 217–239. https://doi.org/https://doi.org/10.1016/j.trb.2016.09.015

Mahmassani, H. S., & Peeta, S. (1993). Network performance under system optimal and user equilibrium dynamic assignments: implications for advanced traveler information systems. *Transportation Research Record*, *1408*, 83. http://onlinepubs.trb.org/Onlinepubs/trr/1993/1408/1408-011.pdf

Mahmassani, H. S., & Peeta, S. (1995). System optimal dynamic assignment for electronic route guidance in a congested traffic network. In *Urban Traffic Networks* (pp. 3–37). Springer. https://doi.org/https://doi.org/10.1007/978-3-642-79641-8_1

Mahut, M. (2001). *A DISCRETE FLOW MODEL FOR DYNAMIC NETWORK LOADING* [Universitk de Montreal]. https://trid.trb.org/view/660668

Mansourianfar, M. H., Gu, Z., Waller, S. T., & Saberi, M. (2021). Joint routing and pricing control in congested mixed autonomy networks. *Transportation Research Part C: Emerging Technologies*, *131*, 103338.

Newell, G. F. (2002). A simplified car-following theory: a lower order model. *Transportation Research Part B: Methodological*, *36*(3), 195–205. https://doi.org/https://doi.org/10.1016/S0191-2615(00)00044-8

Ngoduy, D., Hoang, N. H., Vu, H. L., & Watling, D. (2021). Multiclass dynamic system optimum solution for mixed traffic of human-driven and automated vehicles considering physical queues. *Transportation Research Part B: Methodological*, *145*, 56–79. https://doi.org/https://doi.org/10.1016/j.trb.2020.12.008

Patriksson, M. (2015). *The traffic assignment problem: models and methods*. Courier Dover Publications.

Peeta, S., & Mahmassani, H. S. (1995). System optimal and user equilibrium time-dependent traffic assignment in congested networks. *Annals of Operations Research*, *60*(1), 81–113. https://doi.org/https://doi.org/10.1007/BF02031941

Peeta, S., & Ziliaskopoulos, A. K. (2001). Foundations of dynamic traffic assignment: The past, the present and the future. *Networks and Spatial Economics*, *1*(3), 233–265. https://doi.org/https://doi.org/10.1023/A:1012827724856

Qian, Z. S., Shen, W., & Zhang, H. M. (2012). System-optimal dynamic traffic assignment with and without queue spillback: Its path-based formulation and solution via approximate path marginal cost. *Transportation Research Part B: Methodological*, *46*(7), 874–893. https://doi.org/https://doi.org/10.1016/j.trb.2012.02.008

Rahman, M. R., Siddique, A., & Shahid Mamun, M. (2015). Comparison of User Equilibrium (UE) and System Optimum (SO) Traffic Assignment Methods for Auto Trips. *International Conference on Recent Innovation in Civil Engineering for Sustainable Development (IICSD-2015*. https://doi.org/10.13140/RG.2.1.2169.3204

Richards, P. I. (1956). Shock waves on the highway. *Operations Research*, *4*(1), 42–51. https://doi.org/https://doi.org/10.1287/opre.4.1.42

Saw, K., Katti, B. K., & Joshi, G. (2015). Literature review of traffic assignment: static and dynamic.

International Journal of Transportation Engineering, *2*(4), 339–347. https://doi.org/10.22119/IJTE.2015.10447

Sbayti, H., Lu, C.-C., & Mahmassani, H. S. (2007). Efficient implementation of method of successive averages in simulation-based dynamic traffic assignment models for large-scale network applications. *Transportation Research Record*, *2029*(1), 22–30. https://doi.org/https://doi.org/10.3141/2029-03

Sheffi, Y. (1985). *Urban transportation networks* (Vol. 6). Prentice-Hall, Englewood Cliffs, NJ.

Shen, W., Nie, Y., & Zhang, H. M. (2007). On path marginal cost analysis and its relation to dynamic system-optimal traffic assignment. *Transportation and Traffic Theory 2007. Papers Selected for Presentation at ISTTT17Engineering and Physical Sciences Research Council (Great Britain) Rees Jeffreys Road FundTransport Research FoundationTMS ConsultancyOve Arup and Partners, Hong KongTransp*. http://worldcat.org/isbn/9780080453750

Shen, W., Nie, Y., & Zhang, H. M. (2006). Path-based system optimal dynamic traffic assignment models: formulations and solution methods. *2006 IEEE Intelligent Transportation Systems Conference*, 1298–1303. https://doi.org/10.1109/ITSC.2006.1707402

Shen, W., & Zhang, H. M. (2009). On the morning commute problem in a corridor network with multiple bottlenecks: Its system-optimal traffic flow patterns and the realizing tolling scheme. *Transportation Research Part B: Methodological*, *43*(3), 267–284. https://doi.org/https://doi.org/10.1016/j.trb.2008.07.004

Smith, M., Duncan, G., & Druitt, S. (1995). PARAMICS: microscopic traffic simulation for congestion management. *IEE Colloquium on Dynamic Control of Strategic Inter-Urban Road Networks*, 1–8. https://doi.org/10.1049/ic:19950249

Taale, H., & Pel, A. (2015). Better convergence for dynamic traffic assignment methods. *Transportation Research Procedia*, *10*, 197–206.

Tajtehranifard, H., Bhaskar, A., Nassir, N., Haque, M. M., & Chung, E. (2018). A path marginal cost approximation algorithm for system optimal quasi-dynamic traffic assignment. *Transportation Research Part C: Emerging Technologies*, *88*, 91–106. https://doi.org/https://doi.org/10.1016/j.trc.2018.01.002

Taylor, N. B. (2003). The CONTRAM dynamic traffic assignment model. *Networks and Spatial Economics*, *3*(3), 297–322. https://doi.org/https://doi.org/10.1023/A:1025394201651

Tsanakas, N. (2019). *Emission estimation based on traffic models and measurements* (Vol. 1835) [Linköping University Electronic Press]. https://doi.org/10.3384/lic.diva-155771

US-DOT. (1995). *TRAF User Reference Guide*. Federal Highway Administration.

Van Aerde, M., Hellinga, B., Baker, M., & Rakha, H. (1996). INTEGRATION: An overview of traffic simulation features. *Transportation Research Records*. http://www.civil.uwaterloo.ca/bhellinga/publications/Publications/TRB 1996 Integration Features.pdf

Wang, J., Peeta, S., & He, X. (2019). Multiclass traffic assignment model for mixed traffic flow of human-driven vehicles and connected and autonomous vehicles. *Transportation Research Part B: Methodological*, *126*, 139–168. https://doi.org/https://doi.org/10.1016/j.trb.2019.05.022

Wang, W., Uehara, M., & Ozaki, H. (2018). System optimum traffic assignment for connected cars. *2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)*, 370–375. https://doi.org/10.1109/CANDARW.2018.00074

Wardrop, J. G. (1952). Road paper. some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers*, *1*(3), 325–362. https://doi.org/https://doi.org/10.1680/ipeds.1952.11259

Wie, B.-W., Friesz, T. L., & Tobin, R. L. (1990). Dynamic user optimal traffic assignment on congested multidestination networks. *Transportation Research Part B: Methodological*, *24*(6), 431–442. https://doi.org/https://doi.org/10.1016/0191-2615(90)90038-Z

Yang, I., & Jayakrishnan, R. (2012). Gradient Projection Method for Simulation-Based Dynamic

Traffic Assignment. *Transportation Research Record*, *2284*(1), 70–80. https://doi.org/10.3141/2284-09

Yperman, I. (2007). *The link transmission model for dynamic network loading* [KU LEUVEN].

Zhang, P., & Qian, S. (2020). Path-based system optimal dynamic traffic assignment: A subgradient approach. *Transportation Research Part B: Methodological*, *134*, 41–63. https://doi.org/https://doi.org/10.1016/j.trb.2020.02.004

Zhou, X., & Taylor, J. (2014). DTALite: A queue-based mesoscopic traffic simulator for fast model evaluation and calibration. *Cogent Engineering*, *1*(1). https://doi.org/https://doi.org/10.1080/23311916.2014.961345

Ziliaskopoulos, A. K. (2000). A linear programming model for the single destination system optimum dynamic traffic assignment problem. *Transportation Science*, *34*(1), 37–49.

SUMO User Conference 2022
https://doi.org/10.52825/scp.v3i.112

Published: 29 Sept. 2022

Combining Operative Train Simulation with Logistics Simulation in SUMO

Jakob Geischberger [1*][https://orcid.org/0000-0001-6588-6728] and Norman Weik[1][https://orcid.org/0000-0002-5977-9958]

[1]German Aerospace Center (DLR), Institute of Transportation Systems (TS), Braunschweig, Germany
jakob.geischberger@dlr.de, norman.weik@dlr.de

Abstract

Rail freight logistics is usually planned and analyzed using a macroscopic aggregated view on railway networks and train operations. As a result, disjoint tools have developed for simulating train operations which requires a detailed representation of track assets as well as the signaling architecture and supply chain networks in logistics analyzing the flow of goods where mode-specific capacity and traffic situations are incorporated in an aggregated manner. However, integrating the two areas could help evaluating railway-specific operative implications (such as conflicts and consequent delays) on the level of transport chains and thus single transport units instead of trains or network areas. The simulation tool SUMO is identified to meet criteria from both disciplines. It is shown how a respective methodology can be realized in SUMO to create such a simulation model. A use case of northwestern Germany shows by the means of exemplary container trajectories that the two simulative approaches can be merged.

1 Introduction

In view of complex supply chain networks, just-in-time production and growing product individualization, the global flow of goods has reached unprecedented complexity. In order to understand, evaluate and predict these flows and associated processes respectively, logistics simulations can be used to model the flow of goods for specific - often larger - networks or regions. The underlying motivation is to identify bottlenecks and critical links within a distribution system or transportation network, as well as to analyze the effects of changes in the design of a system in order to support decision making.

Logistics networks, in general, may feature different means of transportation contributing to the transport of goods from origin to destination. This study focusses on rail-based transportation featuring comparably low emissions and large capacities. It is, however, bound to a certain infrastructure-related

[*] corresponding author

inflexibility and accompanied by complex routing and capacity planning problems. In contrast, conventional truck distribution is fast and flexible. It has, on the other side, limited capacity only and accounts for a large share of the CO2 emissions [1]. For this reason, significantly rising the share of goods transported by rail is formulated as a goal of the Rail Freight Forward Coalition [2].

These individual characteristics go along with different operational circumstances specific to certain means of transport. In this context, again, simulation tools are used to model the operational execution of logistics processes. In terms of railway transportation, these models often have a stronger focus on operative performance indicators such as speed, delay or occupancy rates and are usually built on a far higher (often microscopic) level of detail. However, simulation tools rarely combine the detailed microscopic perspective required to model delay propagation in train operations and the strategic planning horizon of goods transportation in supply networks. In the first case, railway specific characteristics such as the signaling system governing infrastructure access and timetable constraints need to be incorporated. In the latter case, a far more macroscopic, logistics perspective is adopted which allows to compare and assess different routes and transportation options. Rail-specific characteristics are considered in coarse-grain resolution and rail freight networks are modeled on a graph-theoretical node-edge representation.

The question arises, if a simulation tool can model the flow of goods in larger railway transportation networks on the basis of single units from a logistical perspective but at the same time consider detailed railway operations with its specifics. By integrating these two perspectives, a deeper understanding of freight railway operations on the level of goods would emerge, uncovering implications from train operations on transport chains. As of now, most train simulations consider trains as the smallest units to be observed. However, the reason for freight train operations is transporting goods from origin to destination in the first place. A combined model, featuring the described aspects, would shift the view from trains to goods as the units to be observed. Thus, the conventionally isolated analysis of either logistical processes or train operation would be integrated in a holistic approach. This can be compared to the change in perspective that has taken place in passenger train operations within the last years: important operational indicators such as delay are now often calculated and communicated on the level of single, specific passengers in addition to trains [3].

In this study, requirements, as well as a methodological framework for the combined simulation of train operations and logistical flow of goods are formulated. We show how the agent-based, microscopic simulation environment SUMO [4] can be used and adapted to integrate these two perspectives. As a result, detailed train operations based on integrated clock-face timetables can be merged with the agent-based flow of goods, given their demand.

The paper is arranged as follows: Section 2 gives an overview of related work and existing literature from the two different disciplines, as well as interfaces in between. Section 3 describes the proposed methodological approach from both an abstract and a technical perspective. In section 4 the proposed methodology is applied to a use case based on a freight rail network in northwestern Germany. A proof-of-concept for combined simulation of operative train simulation and container-based goods transportation is provided based on the presentation of trajectories of individual containers and their dependency on the underlying operative train simulation. Section 5 discusses the further potential of the proposed approach, with a special focus on further evaluation criteria and changes in design of the system.

2 Literature

Simulation is used for planning, realizing and operationalizing logistic systems. Simulation-based approaches contrast analytic and optimization-based methods [5]. The field of logistics simulations we are focusing on in this paper can be further grouped into different areas. From a methodological view, agent-based, discrete event and system dynamic simulations can be distinguished [6]. From another perspective, logistics simulation approaches can also be categorized by fields of application: One important context is to model Supply Chains with the aim of decision support and optimizing processes [7]. In this area, agent-based simulations provide detailed insights. They have a strong focus on individuals and their interaction within a system and can at the same time be used to model complex supply chains (cf. [8] for an overview of different approaches). However, Supply Chain simulation often has a rather macroscopic network perspective, frequently addressing network design issues. Other objectives of logistic simulation are modeling production systems with the aim of optimizing material flow [9]. Last, node-related distribution systems such as yards, terminals or ports are modeled and optimized by simulation [10]. In terms of transportation, especially network logistics simulation often has an abstract understanding of processes, sometimes not considering specifics of certain means of transport but rather systemically modeling transportation by parameters.

Mode-specific simulations of train operations in railway systems can be divided into microscopic and macroscopic approaches [11]. Microscopic simulation models have a strong link to real-life operations and focus on the detailed operational context (such as driving dynamics and the representation of the signaling system). Usually, a specific timetable is required as input and the effects of operational disturbances including the emergence and transfer of delay are studied. However, these detailed simulations are often bound to limited network sizes. Prominent examples of tools can be found in [12] and [13]. Meso- or macroscopic simulations, by contrast, investigate train operations on a station-by-station view. Here, train interactions and delay transfer are integrated and analyzed by means of aggregate train-following or headway constraints, making this class of simulation fast and capable of analyzing large-scale networks while delimiting the resolution locally [11].

Simulation applications range from microscopic timetable robustness analysis to capacity planning of railway lines, nodes and networks. In timetable assessment, different versions of timetables are simulated with the aim of evaluating the stability of timetables in case of disruptions. Their objective is to identify optimal timetables and their combination respectively [14]. Capacity assessment, by contrast, aims to provide insights into the dependency between traffic load and service quality [15], focusing on the available capacity of railway networks, single lines, nodes or a combination thereof [16]. With respect to freight logistics, capacity assessment often deals with the insertion of additional trains into existing timetable concepts [17 until 19]. While evaluation of traffic with respect to passenger trajectories and service experience is common (see, e.g. [20]), the underlying simulation remains based on trains as the elementary units, especially in freight transportation.

There has been some research in the thematic intersection of logistics simulation and railway simulation: [6] propose a Multi-Agent GIS Simulation approach for Railway Logistics Optimization with the objective of identifying the best possible program. Also, linear programming is frequently used to model railway operations in either yards or networks as part of intermodal transport chains, some of which model the context of harbors and hinterland traffic [21, 22]. These models, however, focus on analytical approaches rather than simulation in an operational sense. There are also agent-based simulation approaches in railway research: [23] propose a MATSim-based methodology to model railway operations with single-wagonload-units being the agents. While they do consider capacity constraints and present an approach suitable for large networks, they do not focus on operational aspects such as delay but rather interpret and model the respective network on a node-edge-basis. Their approach aims for optimizing production schemes.

3 Methodology

This section identifies requirements a simulation tool must fulfil in order to create a combined logistical and railway-specific simulation model and explains the subsequent choice of the simulation tool SUMO. It further proposes and illustrates a methodological framework to create a suitable model respectively.

3.1 Requirements of a combined simulation framework for railway operations and freight logistics

A combined simulation model of both a logistics and railway-specific level will have to fulfill certain criteria from both disciplines mentioned. Table 1 shows an overview of requirements to such a model and from which discipline they derive:

	discipline	
Requirement to a combined model	railway	logistics
Processing of timetables	x	
Signaling and train control	x	
Operative Modeling of rolling stock incl. driving dynamics	x	
Import of detailed, microscopic infrastructure	x	
Inclusion of capacity constraints	x	
Handling of large networks	(x)	x
Routing of vehicles (trains)	x	x
Modeling of single goods (e.g. containers) as agents		x
Merging goods and trains: routing of goods (transport chain)		x
Processing of OD demand on level of goods		x
Extensive output possibilities (e.g., punctuality, reliability, routes, network utilization…)	x	x
Easy adjustments on input criteria to evaluate systemic or individual effects	x	x
Acceptable calculation time	x	x

Table 1: requirements to a combined model

Railway-related requirements can be categorized as follows: first, a suitable simulation tool needs to model railway operation with its specifics: As signals govern the access to railway infrastructure, both their locations and functionality need to be implemented. Further, train-specific driving dynamic needs to be implemented in order to correctly model the occupation of infrastructure and how trains interact and propagate delay. Basis for both mentioned criteria is a detailed, microscopic infrastructure model. Second, the concept of timetabling needs to be implementable in order to model railway operations. Here, in addition to relation-wise departure and arrival times, the concept of minimal stopping times is of importance.

Further requirements derive from the logistical perspective, such as the implementation of the flow of goods on an individual, agent-based level to enable a detailed evaluation, e.g. of routes or transportation options. However, at the same time a network-wide analysis should still be possible by aggregating the agent-based output. Furthermore, the assignment of goods to a certain vehicle must be implementable as well. This allows the routing of goods, given transportation vehicles and their route. Coming from a demand perspective, OD matrices as an input on the level of simulative individuals must be processable. As logistical processes often are of complex and interconnected structure, the simulation tool must be capable of dealing with larger networks.

Last, some criteria have a more general simulative context, such as detailed output possibilities fitting the evaluation objective of the model: processable output in order to analyze delay on the basis of simulative agents as well as the correlated reliability can be mentioned here. Furthermore, network-related output (e.g. its utilization) is of importance, as well as routing-related output possibilities such as operationalized trajectories. Given all the different criteria from different perspectives, simulation and thus calculation time must still be acceptable.

It can be seen that railway operations planning goes along with detailed and specific operative processes (and thus requirements) whereas logistics simulation focusses on modeling and analyzing the flow of goods, given a demand and transportation network.

3.2 The SUMO software package

The simulation software package SUMO (Simulation of Urban Mobility) is an open-source microscopic simulation tool that uses an agent-based simulation approach to model mobility. It was originally designed as a digital twin of urban areas and had a focus on private cars. However, extensive development has been made in recent years in order to include non-urban use cases and other means of transports such as harbor processes [24] or railway operations [25, 26]. Even though operative train simulation in SUMO does not cover every single detail of train operation included in specialized railway simulation software tools (such as, e.g. RailSys [12], LUKS [27] or OpenTrack [13]), yet, it provides all functionalities required to analyze and simulate train operations on a microscopic switch-by-switch and signal-by-signal level of the infrastructure. In particular, it features several detailed train following models and respective driving dynamics. Moreover, standardized OSM imports including attribute-related selection is implemented. As normally much fewer train units than cars can be observed in a defined area, large railway networks can be simulated as the numbers of units/agents are comparable to dense mobility in an urban area. Finally, SUMO allows intermodal routing for passengers in public transport, which could possibly be adopted to container routing as well. While SUMO has standardized outputs, it does not yet have many agent-specific output possibilities in the sense of combined, multi-modal transport chains. Here, the existing output can be extended by individual approaches.

Given the fact that the SUMO package provides functionalities for both railway infrastructure modeling and train simulation, as well as agent-based approaches including cross-modal mobility patterns incorporating different mobility-related agents such as cars, trains, persons and containers amongst others, it already fulfills a wide span of the requirements defined in the previous package. We therefore choose SUMO as the fundamental building block of our combined rail freight logistics-simulation approach and show how it can be used and adapted to meet the criteria for simulating and analyzing the trajectories of individual freight units within the context of a rail freight logistics.

3.3 Methodological approach

Against the background of the two different disciplines explained in section 2 (logistics and railway simulation) and the choice of SUMO as simulation tool (cf. section 3.2), the methodological approach to generate a combined simulation model is explained in the following. Figure 1 shows different processes, which level they can be assigned to and how they interact to create the proposed model:

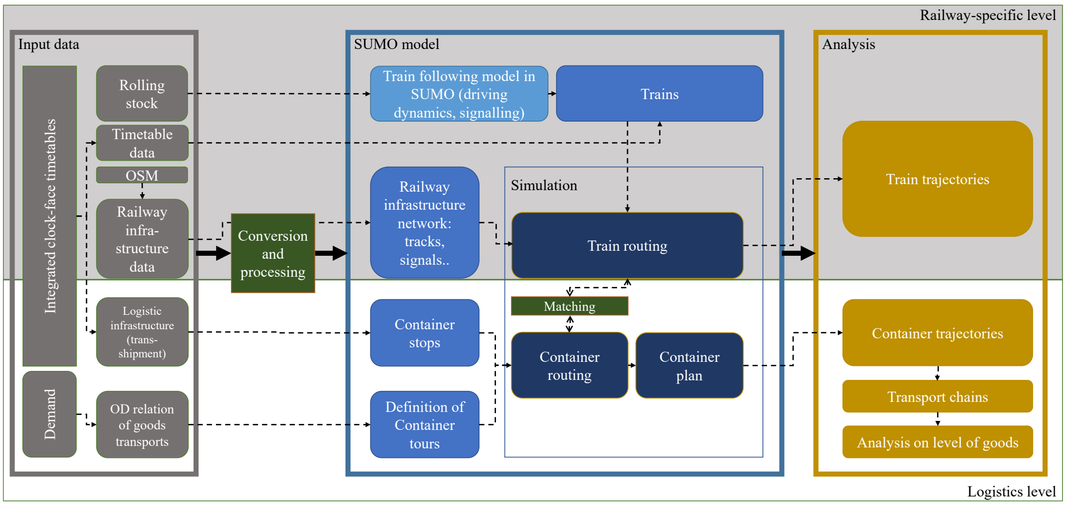

Railway-specific level
Input data
Integrated clock-face timetables
Rolling stock
Timetable data
OSM
Railway infra-structure data
Logistic infrastructure (trans-shipment)
Demand
OD relation of goods transports
Conversion and processing
SUMO model
Train following model in SUMO (driving dynamics, signalling)
Trains
Railway infrastructure network: tracks, signals..
Simulation
Train routing
Matching
Container stops
Container routing
Container plan
Definition of Container tours
Analysis
Train trajectories
Container trajectories
Transport chains
Analysis on level of goods
Logistics level

Figure 1: methodology to create a combined model

Train traffic representation

Various input data is used to create the model. In railway operations, timetables comprise information on departure and arrival times of trains as well as their route. Timetables, where trains move according to a regular pattern that cyclically repeats after a defined interval (e.g. one hour), are referred to as periodic or clock-face timetables. More specifically, integrated clock-face timetables are symmetric clock-face timetables where connections between trains in multiple directions are coordinated on the network-scale with the aim of reducing transfer times. They are also referred to as integrated fixed-interval timetables and often visualized as network timetables (see [28] for an example) [29]. These kind of timetables are of importance to both railway-specific and logistical input data, as they have deep correlations on a planning and infrastructural level: first, they do not only fix passenger transport, but also integrate fixed train paths for freight trains, so called "system paths" [30]. Second, they try to mathematically arrange train paths within a network so as to minimize transfer times in the nodes accordingly. In terms of input for a simulation model, these kinds of timetables do not only contain information on timetables of both passenger and freight trains, but also comprise details on logistical connections as follows: freight train paths are usually integrated into the timetable between larger shunting yards where freight train wagons are arranged to freight trains. Extracting logistical information from system paths and thus defining their stations of origin and destination as logistical distribution points means that no further modeling of logistical infrastructure or production systems (in the sense of which container is transshipped at which yard) is needed. System paths are rather interpreted as potentials for goods to find their way through the system. On a SUMO level, both container stops and timetables can be extracted from integrated clock-face timetable data. Information on rolling stock serves as parameters for the train following model applied in SUMO and thus the driving dynamic of the different trains (e.g. high speed, regional or freight trains).

In terms of demand, detailed data on the level of single units and their respective origin and destination have to be provided. As of now, the proposed approach interprets units in the sense of batches of goods or containers and models them as containers in SUMO.

Rail infrastructure data usage and conversion for usage in SUMO

Physical infrastructure is imported from Open Street Map via a common interface. For reasons of data size, only railway-specific objects are imported in the very first step, such as tracks and signals. The gathered data needs extensive conversion and processing in order to be converted to a SUMO model: OSM railway data is converted to a SUMO network by means of the tools SUMO netconvert. A major challenge when using publicly accessible data is that directed connections (in the sense of which OSM links are connected to each other in which direction) are not “straightened”, meaning that directions of tracks can at times lead to dead ends in the system. Two possible ways to handle this can be thought of: either to declare all edges as bidirectional ones, allowing trains to use them in both directions, or to erase dead ends. The first solution is chosen in the course of the proposed methodology for reasons of minimizing manual network correction work and thus human errors. While it might lead to a high number of available routes for a certain train, it ensures that a “realistic” route is amongst the possible route options. Therefore, depending on the quality of input data and the size of the network, this approach might need adjustments in track priorities to ensure that trains use preferred routes as well as operational directions and thus deadlocks are avoided. SUMO netconvert provides an algorithm to change railway edge priorities on the basis of manually declared priorities of single edges.

Container routing options

Timetable data is transferred to SUMO additional-files and route-files. Here, passenger train stops can be transferred to public transport stops and freight train stops to container stops. In terms of container tours, as of now, static tours are used: this means that every single container has a predefined order of container stops, its origin, transshipment stops if applicable and its destination. It is, however, not specifically defined which trains the respective containers are assigned to. This allows the flexibility to transport containers with any train on a specific route, given capacity and time constraints. Railway shunting yards are usually of very complex track topology and its processes subject of various research (cf. section 2). Here, transshipment stations are defined with container stops, not considering shunting processes in simulation but rather realizing them by process times respectively.

Simulation model

The steps described lead to a SUMO simulation model with realistic railway infrastructure, both passenger and freight trains running according to timetables and containers assigned to tours of container stops. When starting the simulation, trains will enter the network with a specific initial route automatically created by SUMO based on shortest path algorithms. This path can, however, be influenced by edge priorities as explained above. Freight trains will, if defined, load or unload containers at container stops, considering process times as described above. As agent-based microscopic simulations focus on individuals interacting, here trains will most probably interact and influence each other. This is due to various reasons: first, delays are inevitable in train operation e.g. for reasons of unexpected disruptions due to infrastructural or technical issues. Delayed trains have to “keep up” to their schedule, but might not have a free slot within the timetable construct anymore, especially on congested lines or networks. In this moment, trains will influence each other. This impact can be significantly at times, for example when a delayed high-speed train has to follow a slow freight train on a mixed-traffic line. SUMO can model this mutual impact (on the basis of driving dynamics, timetables and signaling) of trains and will at the same time transfer this information to the containers loaded. In this way, container transport can be modeled from a railway-specific perspective, with delay implications being transferred onto single units. Thus, a combined train and logistic simulation can be created.

SUMO offers different output functions to create data as input for analysis. In order to show that containers logistically behave according to their plan and at the same time are part of railway operations, a large output called fcd can be used to track trajectories of both vehicles and persons/containers. As this output can be fairly voluminous, it might be suitable to equip only those agents with a so called “fcd-device” that should also be part of the analysis. The trajectory information SUMO provides comprises coordinates of all/selected agents at every single timestep (1 second). Plotting the lateral speed of different container-specific trajectories vs. the simulation time elapsed shows how “smoothly” container run through the system and will show train-specific effects such as unplanned stops, delays or deviations. Moreover, container trajectories can be shown following time-distance-lines, a traditional way of displaying railway timetables.

4 Proof of Concept

The methodology proposed in section 3 to combine logistics and railway simulation in SUMO was applied to the network of northwestern Germany and shown on the basis of exemplary container trajectories. Figure 2 shows the according railway network in SUMO:

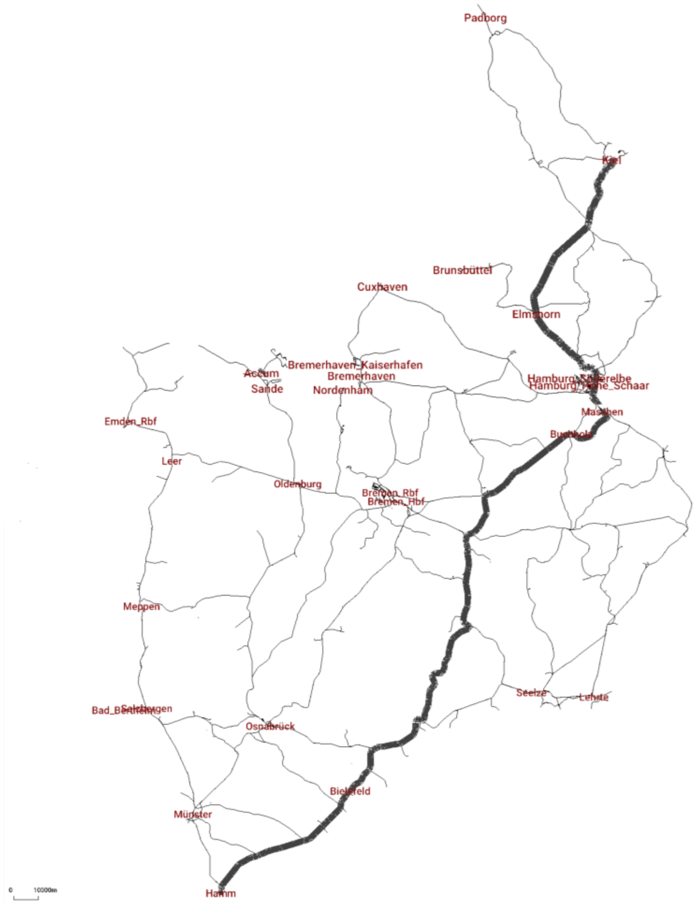


Figure 2: container route from Kiel to Hamm via Maschen yard (source: SUMO, edited)

The network was imported and processed according to the methodology described above. Stops and train timetable data (including system paths for freight trains) were imported from the so called “Deutschlandtakt”, a Germany-wide concept of an integrated clock-face timetable that is to be introduced in 2030 [28]. As explained in section 3.3, subsection “container routing options”, train stations and yards are defined as container/public transport stops in a SUMO additional-file. Moreover, train rides are transferred from the input data to SUMO route-files. Figure 2 shows the specific route of a container entering the System in the harbor city of Kiel and being transported by a first train to the large shunting yard of Maschen. This first train is to be seen as the operationalization of the freight train system path between Kiel and Maschen, operated every two hours at a specific departure minute. In Maschen, it is stored and/or transshipped. A yard-specific process time of 7 hours was assumed. The container is then loaded on a second train running on the system path Maschen-Hamm (3 paths in two hours) to be transported to its destination.

The functioning of the combined both logistical and railway-specific simulation can be shown when comparing the trajectory of different containers within the system. Figure 3 shows the container-specific speed profile of three different containers, all having the same initial route and plan (Figure 2):

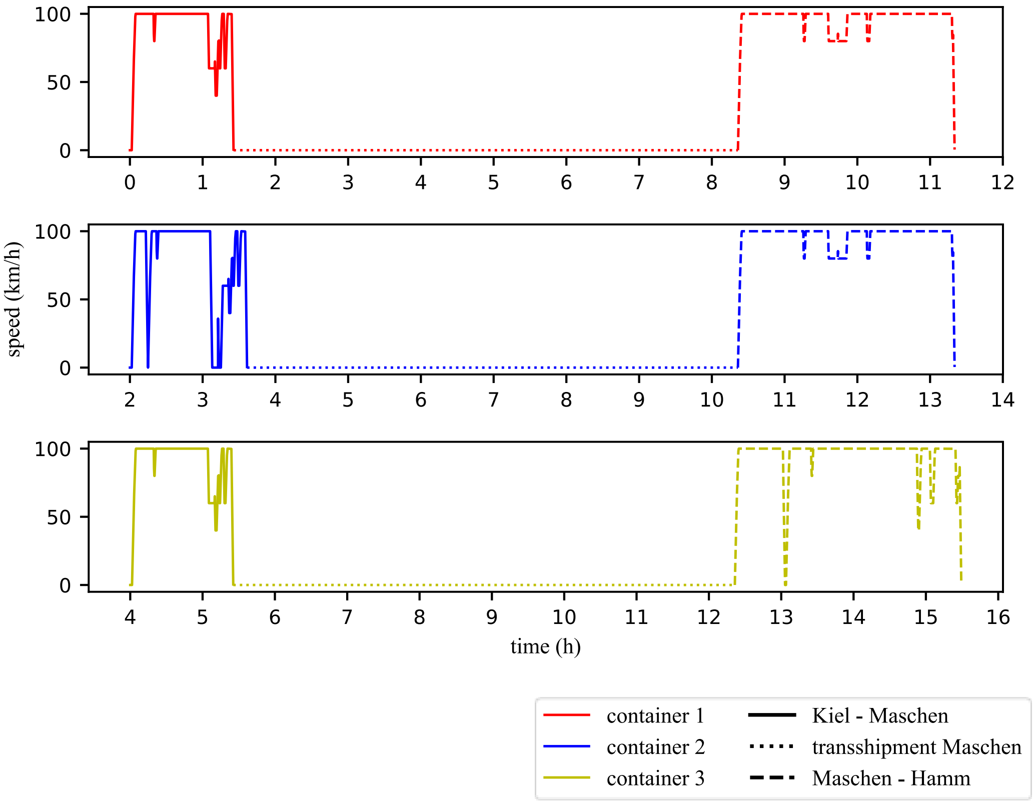


Figure 3: plot of speed over time for three different containers on the relation Kiel-Maschen-Hamm

The red curve of container 1 shows an undisturbed and executed-as-planned transportation chain. It can be seen, that most of the time the speed level ranges around the maximum speed level of the train, 100 km/h. At times, the level suddenly falls, for example when the train passes a densely intertwined node of the network, where allowed speed levels are lower. After arriving in the yard of Maschen, the speed of the container decreases to 0 and stays so for roughly 7 hours until the container is processed. This yard-intern process is not modeled explicitly, but in such a way that the container stays at the container stop until it is loaded again. The subsequent ride with a second train on the system path Maschen-Hamm via Bielefeld takes longer than the first one.

Container 2 has the same route as container 1, but is transported with the system path two hours later. The simulation output shows that the train transporting the container encounters some operational disruptions: at first, the freight train has to slow down and even fully stop for small moment, because of signaling and blocking restrictions: its path is conflicting with a delayed passenger commuter train that shares the route of the freight train of the container for a small segment. Shortly before Maschen, the train again encounters implications from other trains, as the very densely operated area around Hamburg forces the train to stop and wait until tracks are free: The figure shows, that for these reasons, the container arrives in Maschen later than container 1.

Container 3 encounters an undisturbed ride to Maschen comparably to container 1 (not identical, however) but its train is disrupted and has to use an alternative route in the second part of its transport chain from Maschen to Hamm. The deviation via Bremen and Osnabrück takes only a little bit longer, but has a quite different speed profile, as can be seen with the yellow curve.

The container trajectories can also be plotted as time-distance-lines, following the conventional way to visualize timetables. Figure 4 shows this information:

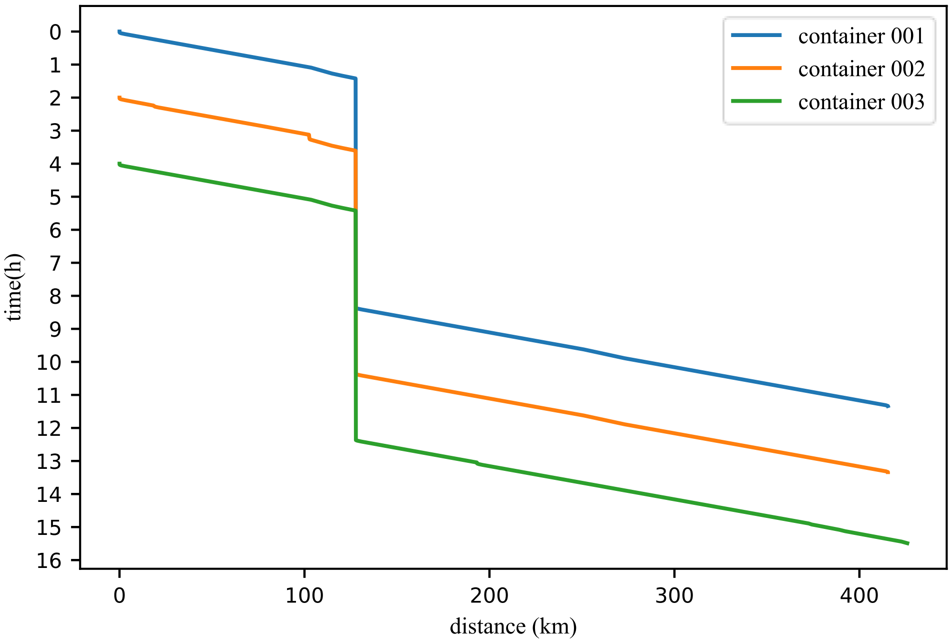


Figure 4: time-distance-line of containers on the relation Kiel-Maschen-Hamm

The figure shows the covered distance of a container (not a train) in correlation to elapsed time. At first sight, the curves appear comparably smooth and almost linear. This, however, can be traced back to the comparably long observation period of 14 hours here, which “flattens” the curve. Nevertheless, operational disruptions with container 2, as described above, can be seen here as well in form of a small bend. Also, the path of container 3 is longer in total because of the mentioned deviation of the train transporting it; this results in the longer distance covered by the container represented by the green curve in Figure 4.

Both Figure 3 and Figure 4 show that with the means of the simulation tool SUMO, detailed railway-specific operations and at the same time agent-based logistical transport chains can be modeled and analyzed. A combined simulation has large further potential as described in the following section.

5 Discussion and future research

Objective of this paper was to show that operational railway simulation and logistical simulation on the basis of single units (containers) can be combined and merged. This was confirmed by a proof-of-concept based on planned integrated clock-face timetables of northwest Germany. On the basis of three exemplary container trajectories with the same initially planned route it was shown that logistical processes on the level of individual agents (container chain) can be modeled while at the same time considering specifics of railway operation. This means that implications from train operations such as delays and trains conflicting with one another can be analyzed not only on the level of trains, but single agents transported by trains.

The shown merged approach can be extended by further research: container chains can be bundled and aggregated in order to enable a system-wide analysis. This can help identifying bottlenecks of a railway network not only in terms of trains, but also goods transported. By systematically analyzing all transport chains within the network, the delay of containers can be further evaluated (in contrast to the delay of trains) and both identify and quantify its dependencies to the design of the system. The reliability of a railway transportation network can thus be analyzed on the level of that goods which initially effected the transport in the first place. This is especially important against the background of supply chains and complex production processes, where reliability can be even more important than transportation time or speed. In addition to that, dependencies between container delay and transshipment process times can further be analyzed.

The model can also be a starting point for studies on both train- and container-routing. As of now, the model did not yet have dynamic intermodal routing implemented, as already possible for passengers in SUMO. By adopting this to containers, constant rerouting and changing the initial plan of containers can be examined. This could help to more dynamically assign free yard capacities. The model proposed might also help to deepen the understanding of sea harbor hinterland traffic in the sense of pondering harbor-, yard- and railway capacities against each other.

Lastly, the model could enable a feedback loop between operation and planning and thus converge two traditionally separate levels of realizing transportation directly on the level of larger networks.

Acknowledgements

The authors would like to thank Jakob Erdmann from the German Aerospace Center, Berlin, for his support and help with both railway-specific and general SUMO modeling.

Declarations

Conflict of interest: The authors declare that they have no conflict of interest.

Data availability: integrated clock-face timetable data for Germany is publicly available at https://www.mcloud.de/web/guest/suche/-/results/detail/1F36C20A-265A-4A4E-8DEC-863DDBC5C1DF

CRediT author statement: Jakob Geischberger: Study conception and design, Methodology, Conduction of analysis, Manuscript preparation and review **Norman Weik:** Advice on methodology, Manuscript preparation and review

References

[1] Methodology for GHG Efficiency of Transport Modes. Final Report, Fraunhofer-Institute for Systems and Innovation Research ISI, CE Delft 2020

[2] Rail Freight Forward coalition: 30 by 2030. Rail Freight strategy to boost modal shift. https://www.railfreightforward.eu/sites/default/files/usercontent/white_paper-30by2030-150dpi6.pdf, Accessed on: 24.02.2022

[3] Parbo, J., Nielsen, O. A. u. Prato, C. G.: Passenger Perspectives in Railway Timetabling: A Literature Review. Transport Reviews 36 (2016) 4, S. 500–526. doi: 10.1080/01441647.2015.1113574

[4] Lopez, P. A., Wiessner, E., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flotterod, Y.-P., Hilbrich, R., Lucken, L., Rummel, J. u. Wagner, P.: Microscopic Traffic Simulation using SUMO. 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE 2018, S. 2575–2582. doi: 10.1109/ITSC.2018.8569938

[5] Wenzel, S.: Simulation logistischer Systeme. In: Tempelmeier, H. (Hrsg.): Modellierung logistischer Systeme. Berlin, Heidelberg: Springer Berlin Heidelberg 2018, S. 1–34. doi: 10.1007/978-3-662-57771-4_1

[6] Gong, W., Zhou, L. u. Ye, F.: Multi-Agent GIS Simulation for Railway Logistics Optimization. 2019 4th International Conference on Intelligent Transportation Engineering (ICITE). IEEE 2019, S. 64–68. doi: 10.1109/ICITE.2019.8880169

[7] Oliveira, J. B., Lima, R. S. u. Montevechi, J. A. B.: Perspectives and relationships in Supply Chain Simulation: A systematic literature review. Simulation Modelling Practice and Theory 62 (2016), S. 166–191. doi: 10.1016/j.simpat.2016.02.001

[8] Clausen, U., Brueggenolte, M., Kirberg, M., Besenfelder, C., Poeting, M. u. Gueller, M.: Agent-Based Simulation in Logistics and Supply Chain Research: Literature Review and Analysis. In: Clausen, U., Langkau, S. u. Kreuz, F. (Hrsg.): Advances in Production, Logistics and Traffic. Lecture Notes in Logistics. Cham: Springer International Publishing 2019, S. 45–59. doi: 10.1007/978-3-030-13535-5_4

[9] Massow, S. von u. Afify, A.: Materialflusssimulation für die Prozessindustrie. Zeitschrift für wirtschaftlichen Fabrikbetrieb 105 (2010) 3, S. 216–221. doi: 10.3139/104.110272

[10] Dragović, B., Tzannatos, E. u. Park, N. K.: Simulation modelling in ports and container terminals: literature overview and analysis by research field, application area and tool. Flexible Services and Manufacturing Journal 29 (2017) 1, S. 4–34. doi: 10.1007/s10696-016-9239-5

[11] Johansson, I., Palmqvist, C.-W., Sipilä, H., Warg, J. u. Bohlin, M.: Microscopic and macroscopic simulation of early freight train departures. Journal of Rail Transport Planning & Management 21 (2022), S. 100295. doi: 10.1016/j.jrtpm.2022.100295

[12] Radtke, A. u. Hauptmann, D.: Automated planning of timetables in large railway networks using a microscopic data basis and railway simulation techniques. WIT Transactions on The Built Environment 74 (2004)

[13] Nash, A. u. Hürlimann, D.: Railroad simulation using OpenTrack. WIT Transactions on The Built Environment 2004 (74)

[14] Salido, M. A., Barber, F. u. Ingolotti, L.: Robustness for a single railway line: Analytical and simulation methods. Expert Systems with Applications 39 (2012) 18, S. 13305–13327. doi: 10.1016/j.eswa.2012.05.071

[15] Abril, M., Barber, F., Ingolotti, L., Salido, M. A., Tormos, P. u. Lova, A.: An assessment of railway capacity. Transportation Research Part E: Logistics and Transportation Review 44 (2008) 5, S. 774–806. doi: 10.1016/j.tre.2007.04.001

[16] Kianinejadoshah, A. u. Ricci, S.: Comparative Application of Analytical and Simulation Methods for the Combined Railway Nodes-Lines Capacity Assessment. Transportation Research Procedia 55 (2021), S. 103–109. doi: 10.1016/j.trpro.2021.06.011

[17] Borndörfer, R., Klug, T., Schlechte, T., Fügenschuh, A., Schang, T. u. Schülldorf, H.: The Freight Train Routing Problem for Congested Railway Networks with Mixed Traffic. Transportation Science 50 (2016) 2, S. 408–423. doi: 10.1287/trsc.2015.0656

[18] Cacchiani, V., Caprara, A. u. Toth, P.: Scheduling extra freight trains on railway networks. Transportation Research Part B: Methodological 44 (2010) 2, S. 215–231. doi: 10.1016/j.trb.2009.07.007

[19] Weik, N., Hemminki, E. u. Nießen, N.: The Effective Residual Capacity in Railway Networks with Predefined Train Services. In: Neufeld, J. S., Buscher, U., Lasch, R., Möst, D. u. Schönberger, J. (Hrsg.): Operations Research Proceedings 2019. Operations Research Proceedings. Cham: Springer International Publishing 2020, S. 725–731. doi: 10.1007/978-3-030-48439-2_88

[20] Caimi, G., Kroon, L. u. Liebchen, C.: Models for railway timetable optimization: Applicability and applications in practice. Journal of Rail Transport Planning & Management 6 (2017) 4, S. 285–312. doi: 10.1016/j.jrtpm.2016.11.002

[21] Hu, Q., Wiegmans, B., Corman, F. u. Lodewijks, G.: Integration of inter-terminal transport and hinterland rail transport. Flexible Services and Manufacturing Journal 31 (2019) 3, S. 807–831. doi: 10.1007/s10696-019-09345-8

[22] Caballini, C., Pasquale, C., Sacone, S. u. Siri, S.: An Event-Triggered Receding-Horizon Scheme for Planning Rail Operations in Maritime Terminals. IEEE Transactions on Intelligent Transportation Systems 15 (2014) 1, S. 365–375. doi: 10.1109/TITS.2013.2280493

[23] Mancera, A., Bruckman, D. u. Weidmann, U.: Single Wagonload Production Schemes Improvements Using GüterSim (Agent-based Simulation Tool). Transportation Research Procedia 10 (2015), S. 615–624. doi: 10.1016/j.trpro.2015.09.015

[24] Noyer, U., Rudolph, F., Rummel, J. u. Weber, M.: Digitaler Hafen: Moderne Hafenregelung durch Simulation. https://aci-mobility.de/fileadmin/user_upload/ACImobility/Papers/acimobility_2021_rudolph_dlr.pdf, Accessed on: 22.02.2022

[25] Merz, G.: AI-based Disposition using a Reinforcement Learning Approach. SUMO User Conference 2020. 2020

[26] Shankar, S., Schubert, L. A., Patil, A. J. u. Erdmann, J.: The use of big data and the SUMO transportation simulation in Rail2X. SIGNAL + DRAHT (2020) 10, S. 49–58

[27] Janecek, David, Weymann, F. u. Schaer, T.: LUKS-integriertes Werkzeug zur Leistungsuntersuchung von Eisenbahnknoten und-strecken. ETR Eisenbahntechnische Rundschau (2010) 1-2, S. 25–32

[28] Bundesministerium für Digitales und Verkehr: Deutschlandtakt. https://www.deutschlandtakt.de/, Accessed on: 23.02.2022

[29] Liebchen, C.: Periodic Timetable Optimization in Public Transport. In: Waldmann, K.-H. u. Stocker, U. M. (Hrsg.): Operations research proceedings 2006. Selected papers of the annual international conference of the German Operations Research Society (GOR), jointly organized with the Austrian Society of Operations Research (ÖGOR) and the Swiss Society of Operations Research (SVOR) : Karlesruhe, September 6-8, 2006. Operations Research Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg 2007, S. 29–36. doi: 10.1007/978-3-540-69995-8_5

[30] Bundesministerium für Digitales und Verkehr: Infrastruktur für einen Deutschland-Takt im Schienenverkehr. https://www.bmvi.de/SharedDocs/DE/Artikel/G/BVWP/bundesverkehrswegeplan-2030-deutschlandtakt.html, Accessed on: 28.02.2022

SUMO User Conference 2022
https://doi.org/10.52825/scp.v3i.158

Published: 29 Sept. 2022

Signal priority for improving fluidity and decreasing fuel consumption

Khaled Belhassine[1,2], Jacques Renaud[1,2], Leandro C. Coelho[1,2,3], and Vincent Turgeon[4]

[1] Université Laval, Québec, Canada
[2] Interuniversity research centre on enterprise networks, logistics and transportation – CIRRELT
[3] Canada research chair in integrated logistics
[4] City of Trois-Rivières, Department of Information Technology
{khaled.belhassine, jacques.renaud, leandro.coelho}@fsa.ulaval.ca, vturgeon@v3r.net

Abstract

Signal priority is a strategy to handle traffic that adjusts the timing of a traffic signal phases to achieve a mobility goal, typically to reduce congestion or waiting time. This strategy enables increased mobility and is usually used in public transport operations; it works by making small modifications to the phases of a cycle in order to delay the green phase or anticipating the end of the red phase. We build a detailed simulation of the city center where an industrial partner operated and introduce signal priority handling to improve operations and mobility for their heavy-duty vehicles. The idea is that when these vehicles stop at red signs, they incur high consumption to regain speed, and cause conges-tion on the traffic behind them due to their lower acceleration. Detailed computational experiments demonstrate that this strategy generates significant gains for the partner fleet and has a side benefit of also improving fluidity for the surrounding traffic.

1 Introduction

Traffic congestion is a major issue in many cities. Dense urban areas as well as those within city centers are prone to congestion, which leads to many adverse effects: noise, pollution, decreased mobility, increased risks of accidents, etc. This is particularly important when heavy-duty vehicles must traverse these areas. This is often the case when industrial parks are located within city limits. In this paper, we present a case study of such a city, and with access to large databases of terrain data, including the latest origin-destination survey of the geographical area, detailed LiDAR counters, and connected onboard devices on a partner's fleet, we design transit signal priority rules in order to alleviate many of these issues.

Signal priority is a strategy that modifies the schedule of traffic lights on signal-controlled intersections. The goal is to improve fluidity of the axis of interest, without any major disruption on the surrounding traffic [8]. This strategy is well-known in transit systems, typically for vehicles in service, allowing them to decrease their travel time as well as improving the adherence to their schedules [4]. Different strategies and parameters have been tested, mostly related to the operational performance of bus systems [6]. Using signal priority to alleviate the congestion caused by heavy-duty vehicles and to decrease their fuel consumption does not seem to have been largely tested.

In our case study, we collaborate with the city administration and with an industrial partner who allowed their heavy-duty vehicles to be equipped with GPS and real-time communication devices. The goal is to design signal preemption rules in order to prevent these heavy vehicles from stopping too often within the city center due to red lights, giving them priority if required to ensure congestion is avoided and that their fuel consumption is then optimized. In order to achieve these goals, we make use of four large datasets:

1. road network and traffic signal phases,
2. origin-destination matrices of people movement in the metropolitan area,
3. camera counters of traffic,
4. detailed GPS traces and fuel consumption data from the partner fleet.

These different sources of data and how we adjusted them to obtain a representative simulation in SUMO [1] are described in Section 2. In Section 3 we described how we managed the traffic lights. In Section 4 we present several relevant statistics of the simulation including total fuel consumed by the fleet of interest and by all the vehicles of the simulation, the time lost in traffic, among others. Section 5 concludes the paper and suggests future research directions.

2 Data description and adjustments

In this section we describe the data available to us, and how we have adjusted the simulation parameters to ensure that the simulation model is representative of the real life.

2.1 Road network

The road network of the area of interest was first imported from OpenStreetMaps [7]. This area covers the territory of the Origin-Destination survey provided to us as presented in the next section. After several rounds of corrections such as lane numbers, connections, traffic lights, speed limit, and edge geometry, the graph contains 17 194 segments for 4 643 km and covers 1 965 km^2, for a population of about 175 000 inhabitants. A snapshot of the complete region [7] is provided in Figure 1.

2.2 Detailed Origin-Destination information

We had access to detailed data from the 2011 origin-destination (O-D) survey on travel habits conducted by the Transport Systems Modeling Department from the *Ministère des transports du Québec* [5]. This survey covers the Trois-Rivières census metropolitan area which encompasses 18 municipalities. Each line of this survey represents a number of movements containing information regarding the origin, destination, departure and arrival times, mode of transportation, along with socio-demographic information. This large file was obtained from interviews, and was also treated by statisticians who determine that each one of these trips represents a number of trips actually happening. So for example, one trip from a given residential area (a street or a block) toward a commercial area (also as small as a street or a block) is representative of, as an example, 15 such trips over the same O-D pair, the same approximate time, and the same mode of transportation.

Overall, the 58 084 lines of this O-D survey represent a total of 471 530 movements (396 523 of them being vehicle movements, others may be bicycle or buses trips for example) per day in the regions of Ville de Trois-Rivières, Bécancour, Batiscan, Champlain, Grand-Saint-Esprit, Nicolet, Notre-Dame-du-Mont-Carmel, Saint-Barnabé, Saint-Célestin, Saint-Étienne-des-Grès, Sainte-Geneviève-de-Batiscan, Saint-Luc-de-Vincennes, Saint-Maurice, Saint-Narcisse, Saint-Sévère, Wôlinak and Yamachiche. The data from this O-D survey gives a general overview of the traffic sources and profiles.

We input this data into SUMO by using the attribute *fromLonLat* and *toLonLat* on a trip file. Thereby, each route was created from the latitudes and longitudes of origin and destination

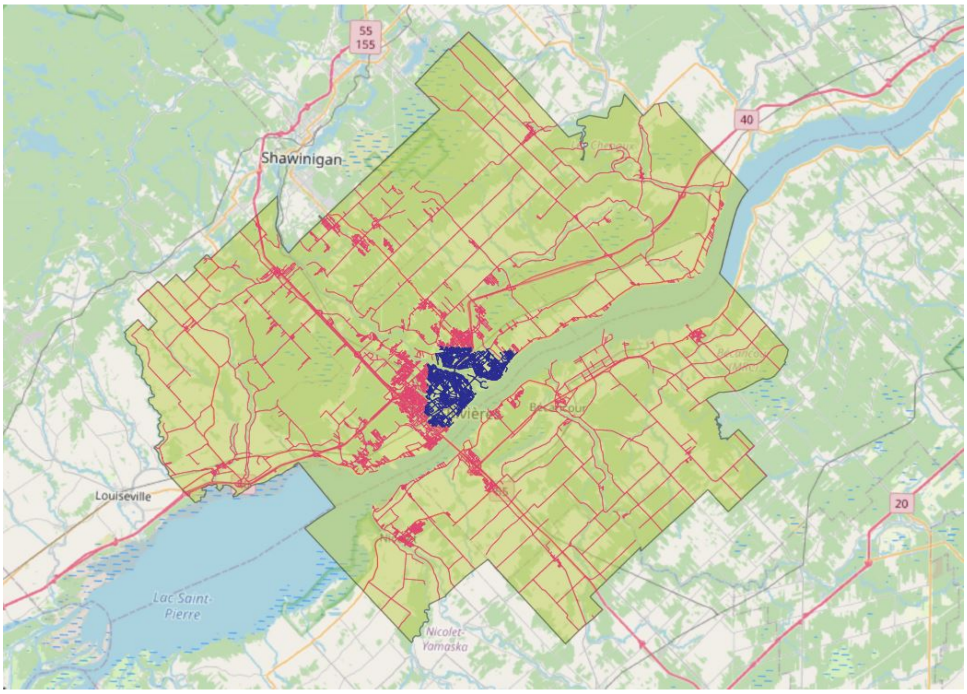


Figure 1: Interest zone

and we created a complete 24 hours model. To be representative of the types of vehicles, we used seven different types of vehicles (we generated 75% of passenger vehicles, 15% of sport utility vehicles, 3% of light vehicles, 1% of light delivery vehicles, 2% of buses, 1% of tractor trailers type 1, 1% of tractor trailers type 2, and the individual vehicle's trips of the fleet of our partner). These proportions were obtained from the city officials which use cameras with LiDAR technology. These cameras are able to identify the type of vehicles like regular cars, light trucks, heavy trucks, buses and tractor trailers. Data come from 18 different intersections which are representative of the heavy circulation in this area. Note that we did not model real bus lines. We carefully selected the best SUMO consumption model to be representative of the vehicle's consumption rates reported by the manufacturers.

By studying the O-D survey data, we observed that departure times were often rounded up to 15 minutes. To avoid a jagged generation, we took each observation and spread its departure time to more or less 5 minutes. In the O-D data, each individual observation is associated with a multiplication factor $Fact$. This factor means that each line in the sample represents $Fact$ similar trips in the population. Then we generate $Fact$ movements uniformly distributed around the departure time within an interval of 10 minutes (5 minutes before and 5 minutes after). In addition, in order to avoid that all trips depart and arrive from exactly the same coordinates, we also used a buffer of 500 meters around the departure and arrival locations, and generated $Fact$ different departure locations and $Fact$ different arrival locations within this radius.

At this step, the 24h simulation has 396 523 trips and takes 88 341 seconds to run without the Graphical User Interface, on a computer equipped with Intel® Core™ i9-9900K CPU @ 3.60GHz and 64GB of RAM. At this stage we keep all the information (location and time) on the movements of each original trip even if the simulation was not yet calibrated and adjusted, and it contained a number of teleports, traffic jams and waiting time.

2.3 Area of interest

We have then reduced the geographical area of O-D survey to concentrate on the Trois-Rivières city center. In this smaller graph, we have 5 480 edges for a length of 659.85 km and it covers 39 km^2. In order to have all vehicle trips passing thought the interest zone, we have analyzed each of the 396 523 original vehicle trips of the O-D survey to consider only the part of their path that travels through the area of interest. The function *duarouter* was used to facilitate the process. This is depicted in Figure 2, where the area of interest is shown as the white polygon within the complete map. This region is illustrated in blue in Figure 1. An example of a trip starting at O and ending at D outside the zone of interest is shown. For this trip, the departure time at O and the arrival time at D were obtained from original simulation over the complete O-D zone. When the trip enters the interest zone, we noted its exact location O' and entering time $t(O')$. The trip continues and exits the interest zone at location D'. Thus, in the smaller simulation we have a trip starting at $t(O')$ from O' to D'. For an improved visualization of this zone, we use satellite image tiles from Google Maps [3].

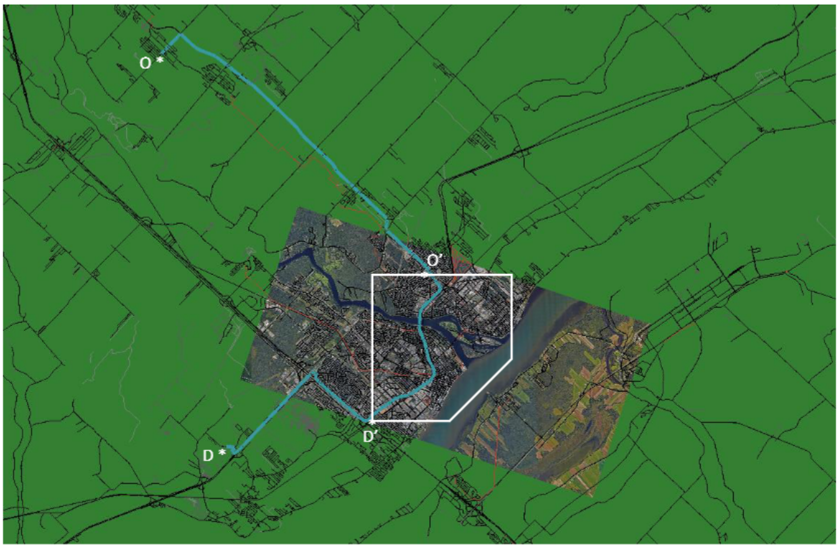


Figure 2: Zone of interest and trips interception

Note that trips originating and ending outside the zone of interest, for which no part of the trip travelled through this zone, are then not considered. Obviously, trips that are completely within the zone of interest remain unchanged. The function *cutRoutes.py* was used to generate the reduced trips files. These two procedures (decreasing the zone of interest, and intercepting the trips within this zone) decreased the size of the graph from 17 194 to 5 480 segments. Moreover, the number of vehicle trips now considered in the simulation decreased from 396 523 to 262 393.

Finally we ran the dynamic user assignment procedure *one-shot.py* on the 262 393 routes with a travel-time updating intervals of 30 seconds to better balance the users' routes.

2.4 Camera counters

To make sure that the simulation reproduces the city traffic, we used again up-to-date camera counters from LiDAR technology, thank to the city officials. These counters were available from nine intersections and cover 24 road segments on the city's main boulevard where our partner's trucks circulate. These city counters report a total of 88 529 vehicles. They indicate with very high precision the number and type of vehicles traveling, their speed, direction and cornering information, among others. At this point, our simulation reports 68 698 vehicles passing through the same roads, which is about 78% of the observed traffic. This gap can be explained by the fact that the O-D survey was conducted in 2011 and that the city camera counts were taken in 2021.

Based on these information, we have calibrated the simulation using the *calibrator* functions of SUMO which allow dynamic adaptation of traffic flows, speeds and vehicle parameters. The *calibrator* removes vehicles in excess of the specified flow and it inserts new ones to try and match the counts. We used hourly flows for the calibration. Several iterations were carried out in order to adjust the simulation counts. The routes of the partners trucks remained unchanged in the calibration simulations. At the end of the calibration process, we have 87 351 vehicles passing through the studied intersection, which represents 98.66% of 88 529 trips reported by the LiDAR counters. To obtain these values, 26 729 trips were added by the *calibrator* for a total of 289 122 trips which is now the basis of the initial simulation (see Table 1).

Figure 3 shows the complete traffic profile of the O-D study and the traffic profile over the simulated region. We can see that both profiles follow the same hourly profile. The simulated traffic is very close to the O-D one especially outside the peak hours. Both profiles differ in the peak hours because we do not consider the vehicle movements outside of our region, as discussed.

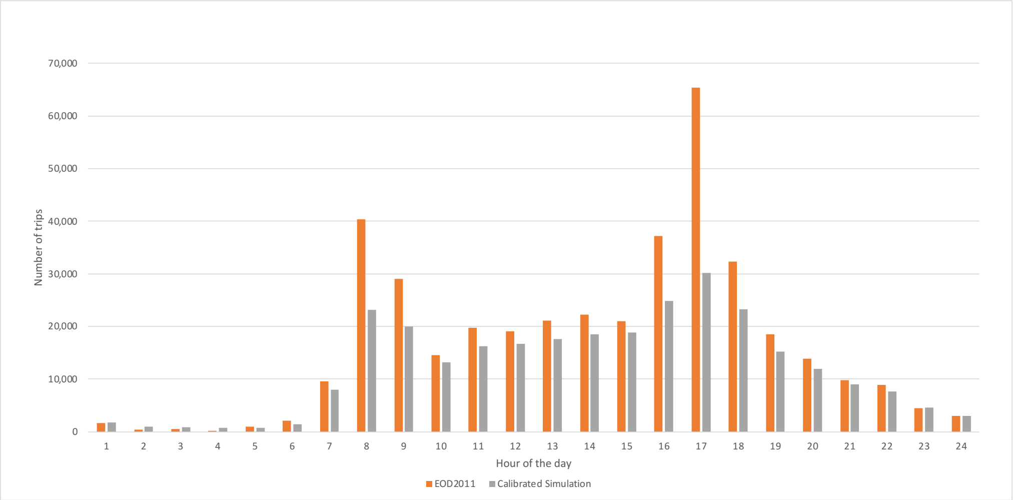


Figure 3: O-D and simulated traffic profiles

2.5 GPS traces and fuel consumption data

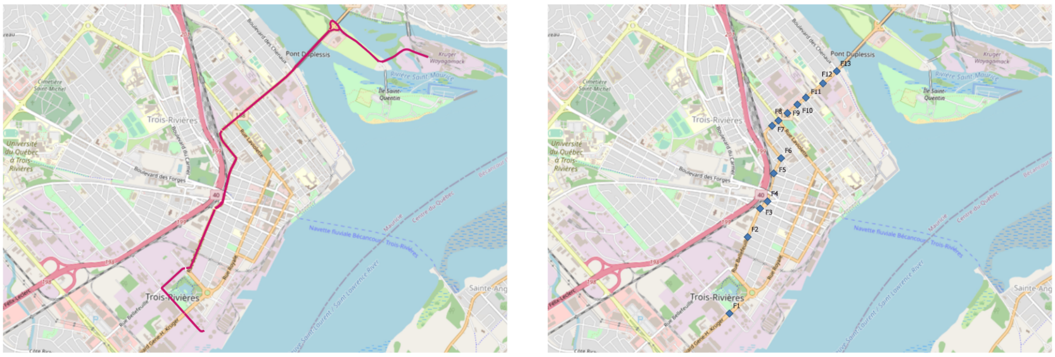
The trucks of our industrial partner are all equipped with high-frequency GPS devices that stores their location and instantaneous speeds, providing a detailed understanding of their trips, average speeds, acceleration and deceleration patterns, and how often they stop at red lights along the way. Moreover, these high-frequency devices are connected to the on-board computer of the vehicles and also provide us with detailed fuel consumption, acceleration rates, among other useful data.

The company's trucks perform many round trips, each for 11.74 km and facing a total of 26 traffic lights. Their trucks travel fully loaded, weighting around 50 tons, and return empty, with a curb weight of around 25 tons. To better model the vehicle consumption we use two different SUMO consumption classes (PHEMlight/HDV-TT-D-EU6 and PHEMlight/HDV-TT-D-EU4) for an average consumption of 119.4 l/100 km when loaded and of 113.8 l/100 km unloaded. These values are representative of the consumption levels reported by the company. Figure 4a depicts the path traveled by the company's vehicle.

2.6 Traffic signal phases

Finally we also obtained from the city the traffic signal phases of all traffic lights in the path of our partner's vehicle (see Figure 4b). These phases were carefully implemented in SUMO. These were corroborated by field observations and video recording of several trips. All others traffic lights are set to *actuated*.

At this step we have a simulation based on the data of the O-D survey, where the hourly profile matches the survey and LiDAR-obtained vehicle counts. Traffic on the main street of interest was validated with the city counters and the fuel consumption corresponds closely either to the manufacturer specifications or to our partner's consumption data. The final simulation takes 12660 seconds to run without the Graphical User Interface.



(a) Detailed vehicle trip

(b) Traffic lights location

Figure 4: Vehicle route and traffic lights (images from [2])

3 Managing the traffic lights

In order to obtain simulation results that can yield significant savings without affecting too much the surrounding traffic or changing too much the signal phases, we establish several signal priority strategies to be simulated. Our goal is to provide the city officials with strategies that can be used to improve fluidity in the city center, allow fuel economy for the partner and not cause any disruption on the surrounding traffic.

To dynamically manage the traffic lights, we positioned a number of sensors on the path followed by the trucks of our partner. To interact with these sensors we use the Traffic Control Interface (TraCI) which allows us to retrieve the state of the simulation and change the value of some simulation parameters according to different sets of rules. Despite the synchronization of the green wave, city officials confirm that traffic on this boulevard is recognized as being problematic. This is a very busy boulevard and the succession of very close traffic lights causes several problems.

We positioned 24 sensors on 13 locations on the forward path and 24 sensors on 13 locations of the truck's return path. As we will see, the good positioning of each sensor is crucial to enhance the performance of the simulated network. If the sensor is too close to the traffic light, the truck may not reach it if too many cars are waiting at the red light. If the sensor is too far, the light may have time to turn red before the truck has crossed the intersection. In the following we tested a number of strategies.

Strategy 0: This strategy is the *status quo* where the sensors are not used. These results are used as a basis for improvement.

Strategy 1: Under this strategy each sensor is located at 150 meters of the light that it controls. When a truck reaches the sensor, the light is automatically switched to green.

Strategy 2: In this strategy, the position of the sensor follows a function of the duration of the green light phases d_g and the allowed speed (in meters per second) on the segment towards it, s. The sensor is located at $\frac{d_g}{2s}$ meters of the traffic light. For a 20 seconds green light and a 50 km/h speed limit (13.88 m/s), the sensor will be located at 138 meters. The light turns green as soon as the truck reaches the sensor.

Strategy 3: Under strategy 3 we do not automatically give a green light to the truck but rather consider the current phase of the traffic light. When the truck reaches the sensor, we have three cases:

i) if the light is red, we change the lights of the other direction to yellow for α seconds and then the truck is given a green light;

ii) if the light is green but for less than β seconds, we extend the duration of the green such that it lasts β seconds; and,

iii) if the light is green for more than β seconds, or if the light is yellow, no action is taken.

These rules are applied to the sensor positioning of Scenario 1. Indeed, as the speed limit is the same all along the vehicle's path, the use of a fixed β implies that all the sensors are positioned at the same distance from the traffic lights. Results are presented in the next section.

4 Simulation results

At this step, the simulation provides a representative digital twin of the city, with a particular emphasis in the city center where the fleet of trucks of our industrial partner often affect traffic, and is impacted by the large number of traffic lights. Our goal is to evaluate how much can be saved in terms of time, congestion, and fuel, if traffic lights are operated in a coordinated way, considering the real-time location of the trucks.

Table 1 presents the results of the simulation under the initial Strategy 0. We present the number of trips, consumption in liters (column Cons.), the distance traveled in kilometers, and the consumption rate in liters/100 km. This table also presents a column Trips % which shows for each vehicle type, its percentage of trips with respect to the total number of trips. We can see that the proportion of generated vehicle of each type closely respects the percentages presented in Section 2.2. For all vehicle types, the generated percentages are within 0.05% except for passenger vehicles which is 0.17%. This deviation mainly comes from the calibration process where the added vehicles to respect the camera counters are of the *Passenger vehicle* type.

We detail the findings for each vehicle type: buses, articulated buses, tractor trailers, passenger vehicles, sport utility vehicles, light vehicles, light delivery vehicles, tractor trailers, and Partner (standing for our partner's vehicles). In this simulation, 289 590 vehicle trips were inserted and only 383 teleports were reported by the simulation (this is reported in Table 5). The simulation provides a consumption of 166 liters for the company's trucks and of 195 265 liters for the whole set of 289 122 trips. We can see that the simulation returns realistic consumption rates for all vehicles.

Table 1: Initial simulation results under Strategy 0

Vehicle types	**Nb. trips**	**Trips %**	**Cons. (liters)**	**Dist. (km)**	**Cons. (l/100 km)**
Buses	5 674	1.96%	19 191	25 259	76.0
Articulated buses	5 679	1.96%	19 374	25 232	76.8
Tractor trailers	2 758	0.95%	15 099	12 177	124.0
Passenger vehicle	217 320	75.17%	92 826	883 234	10.5
SUVs	43 260	14.96%	27 634	191 076	14.5
Light vehicle	8 697	3.01%	6 282	38 013	16.5
Light delivery vehicle	2 931	1.01%	4 523	12 789	35.4
Tractor trailers	2 792	0.97%	10 170	12 170	83.6
Partner	11	0.00%	166	129	128.7
Total	289 122	100.00%	195 265	1 200 078	16.27

Table 2 shows the results when the control of the traffic lights is activated under Strategy 1. Here, whenever one of the partner's vehicles passes through the sensors, it is granted the green phase. We can see that 289 540 trips were performed during the simulation, an increase of 418 completed trips (less than 0.14%), and the number of teleports also slightly increases to 387. However the distance traveled increases by 13 629 km (1.1%). As the distances traveled was not exactly the same, we perform some adjustments to set the consumption on the same basis. For example, in the second simulation, the buses traveled 420 km less. In order to compare the results of two tables we need to increase the consumption accordingly, thus we added 420 km at the average consumption rate of 75.4 l/100 km for 316.92 liters. Thus the adjusted consumption is 18 908 liters. We can see that our partner's vehicle reduces its consumption by 9 litters from 166 to 157. For this scenario, the consumption of the partner's truck is 119.4 l/100 km when loaded and 113.8 l/100 km when empty. To this data we add 2.2 liters while the truck is running idle for a global consumption of 121.7 l/100 km. This is very close to the annual consumption level of 125.6 l/100 km reported by the company. For the overall vehicles, the second simulation reduces the total consumption by 2 893 litters (-1.48%). We can conclude that not only our partner's truck benefits from the traffic control but also many of the vehicles running on this congested road can also benefit from the added green light period. A second-hand benefit is that the trucks do not stop at red lights having then to slowly regain their speed, avoiding causing congestion for the vehicles behind them.

Table 2: Results with traffic light control under Strategy 1

Vehicle	**Nb.**	**Cons.**	**Dist.**	**Cons.**	**Diff.**	**Diff.**	**Adjust.**	**Cons. red.**
type	**trips**	**(liters)**	**(km)**	**(l/100 km)**	**(km)**	**(liters)**	**cons.**	**(liters)**
Buses	5 665	18 591	24 838	74.8	-420	316.92	18 908	283
Articulated buses	5 702	18 812	24 886	75.6	-346	263.59	19 076	298
Tractor trailers	2 759	14 692	11 952	122.9	-225	277.85	14 969	130
Passenger vehicle	217 582	90 351	873 820	10.3	-9 414	981.38	91 332	1 494
SUVs	43 300	26 857	188 407	14.3	-2 669	383,27	27 241	393
Light vehicle	8 757	6 210	37 877	16.4	-136	22.38	6 232	50
Light delivery vehicle	2 948	4 411	12 565	35.1	-224	78.76	4 490	33
Tractor trailers	2 816	9 807	11 975	81.9	-195	160.99	9 968	202
Partner	11	157	129	121.7	0	0.00	157	9
Total	289 540	189 887	1 186 449	16.0	-13 629	2 485	192 372	**2 893**

Table 3 shows that further improvements can be obtained with an optimized positioning of the sensors. With these sensors the trucks of our partner reduce their consumption by 11 liters (6.6%) and for the whole vehicle fleet, the consumption reduces by 4 231 liters (2.17%).

Table 3: Results with traffic light control under Strategy 2

Vehicle	**Nb.**	**Cons.**	**Dist.**	**Cons.**	**Diff.**	**Diff.**	**Adjust.**	**Cons. red.**
type	**trips**	**(liters)**	**(km)**	**(l/100 km)**	**(km)**	**(liters)**	**cons.**	**(liters)**
Buses	5 664	18 561	24 892	74.6	-366	275.55	18 836	355
Articulated buses	5 714	18 872	25 062	75.3	-169	128.86	19 001	373
Tractor trailers	2 756	14 878	12 086	123.1	-91	112.31	14 990	109
Passenger vehicle	217 429	89 331	871 530	10.2	-11 704	1214.81	90 546	2 280
SUVs	43 201	26 409	187 165	14.1	-3 910	558.65	26 967	667
Light vehicle	8 728	6 056	37 351	16.2	-663	108.49	6 164	118
Light delivery vehicle	2 945	4 311	12 444	34.6	-344	120.51	4 432	91
Tractor trailers	2 801	9 798	11 995	81.7	-175	144.77	9 943	202
Partner	11	155	129	120.2	0	0.00	155	11
Total	289 249	188 370	1 182 655	15.93	-17 423	2 664	191 034	**4 231**

Table 4 shows the results when we add the traffic light management rule with $\alpha = 3$ and $\beta = 15$ seconds. Here we were able to obtain an improved consumption reduction of 4 464 liters (2.2%). This scenario is also the one producing the most important consumption reduction for our partner fleet which is now 153 l instead of 166 l under the original scenario.

Some relevant statistics of these simulations are presented in Table 5. We observe that for almost 300 thousand vehicles simulated, there were only 383 teleports in Scenario 0, for an average trip duration of 588 s. The time lost in traffic (waiting time) at speeds below 0.1 m/s was 186 s on average, and the total time loss due to driving below the ideal speeds, which is a measure of traffic incurred, averaged 313 s per trip. The same statistics for the simulation of

Table 4: Results with traffic light control and management rules, Strategy 3

Vehicle type	Nb. trips	Cons. (liters)	Dist. (km)	Cons. (l/100 km)	Diff. (km)	Diff. (liters)	Adjust. cons.	Cons. red. (liters)
Buses	5 626	18 079	24 547	74	-711	532.15	18 611	580
Articulated buses	5 690	18 581	24 755	75	-477	361.77	18 943	431
Tractor trailers	2 744	14 607	11 958	122	-219	269.61	14 876	223
Passenger vehicle	217 170	89 725	874 377	10	-8 857	919.82	90 645	2 181
SUVs	43 175	26 426	187 026	14	-4 050	578.91	27 005	629
Light vehicle	8 698	6 059	37 293	16	-721	118.12	6 177	105
Light delivery vehicle	2 953	4 320	12 460	35	-328	114.94	4 435	88
Tractor trailers	2 787	9 767	11 941	82	-229	189.45	9 956	214
Partner	11	153	129	118	0	0.00	153	13
Total	288 854	187 716	1 184 487	15.85	-15 591	3 085	190 801	**4 464**

Scenario 1 show a clear reduction on the time lost in traffic and on the waiting time. Scenario 2, with an optimized location for the sensors, demonstrated that a significant improvement is possible. Particularly, the number of teleports has decreased and the time loss and waiting time values have considerably decreased compared to the previous cases. Finally, the different timing control of the traffic lights as in Scenario 3 shows that the statistics of interest remain stable with respect to Scenario 2, indicating no signs of deterioration in the simulation as a result of the improved traffic signal handling.

Table 5: Simulation statistics

Statistic	Strategy 0	Strategy 1	Strategy 2	Strategy 3
Total vehicles loaded	292 965	293 475	292 708	292 593
Teleports	383	387	326	324
Duration (s)	588.3	563.01	551.41	553.45
Waiting time (s)	186	170	162	162
Time loss (s)	313.7	292.3	281.4	282.2

5 Conclusions

In this article we have proposed different strategies to manage traffic lights according to the movements of our partner's trucks while respecting a detailed simulation serving as a digital twin of a city. The objective was to improve the fluidity of the journeys of these heavily loaded trucks while not interfering with local traffic.

Using data from an Origin Destination survey and detailed counts from selected intersections, we have faithfully reproduced the traffic profile of the area studied and implemented the phasing of the city's traffic lights. The reference simulation generates 289 122 trips for a distance of 1 200 078 km and a consumption of 195 265 litres. The consumption of our partner's vehicle is 166 liters under this scenario.

Our results show that by carefully positioning the sensors it is possible to improve not only the consumption of our partner but also of all the vehicles. The best strategy proposed reduces the consumption of our partner's trucks by 13 liters per day and of all the simulated vehicles by 4 464 liters. These results show that the new method of traffic light management allows an overall improvement in improving the fluidity on the boulevard that the trucks traverse and where a significant part of the traffic of the city takes place.

The city is currently deploying cameras and sensors at the main intersections in its territory. On the city's main boulevard where our partner's trucks circulate, 9 cameras are already installed out of the 19 that would be needed. The other cameras will be installed shortly since they will also be used for fire vehicles and ambulances.

Acknowledgements

We thank three anonymous referees for their valuable suggestions on an earlier version of this paper.

References

[1] *Microscopic Traffic Simulation using SUMO*, 2018.

[2] Google Earth. Trois-Rivières metropolitan area, Quebec, Canada, 2021.

[3] Google Maps. Trois-Rivières metropolitan area, Quebec, Canada, 2021.

[4] W. Kim and L. R. Rilett. Improved transit signal priority system for networks with nearside bus stops. *Transportation Research Record*, 1925(1):205–214, 2005.

[5] Ministère des transports du Québec. Enquêtes origine-destination. https://www.transports.gouv.qc.ca/fr/ministere/Planification-transports/enquetes-origine-destination/Pages/enquetes-origine-destination.aspx, 2021.

[6] V. Ngan, T. Sayed, and A. Abdelfatah. Impacts of various parameters on transit signal priority effectiveness. *Journal of public Transportation*, 7(3):4, 2004.

[7] OpenStreetMap contributors. Trois-Rivière Metropolitan area extracted from http://extract.bbbike.org/, 2021.

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

SUMO User Conference 2022
https://doi.org/10.52825/scp.v3i.175

Published: 29 Sept. 2022

Simulating platooned connected autonomous vehicle in a large scale urban scenario

Joerg Schweizer[1][https://orcid.org/0000-0003-2289-6111], Cristian Poliziani[1], and Federico Rupi[1]

University of Bologna, Bologna
joerg.Schweizer@unibo.it

Abstract

This article is concerned with the performance evaluation of connected, autonomous vehicles (CAVs) in a realistic large-scale microsimulation scenario. In particular the ques-tion is: how much could a high diffusion of CAVs possibly change (1) the average travel speeds (2) the trip times of all traffic participants, including pedestrians, and (3) the en-ergy/fuel consumption? For this purpose, admittedly favourable assumptions are made: a 100% diffusion of platooning-capable CAVs as substitution for private cars as well as a high maximum speed of platooned vehicles in order to enable platoon formation. The morning rush hour scenario of the metropolitan area of Bologna, Italy has been selected for assessment. This scenario, which has been created and validated in previous works, represents an activity based demand model with travel plans for individual citizens, including all relevant transport modes. The microsimulation is performed by means of the SUMO simulator. The entire demand has been generated with the SUMOPy tool. For the platooning of CAVs, SUMO's SIMPLA module has been used, which controlls the vehicles via the interactive TRACI API.

Results show an increased speed and reduced travel time for CAV vehicles, with respect to human driven cars, in particular in the periphery and less in the center with a dense road network. However, the reason for improved speeds and travel times is predominantly the higher maximum speed allowed for vehicles trying to catch up and join a platoon. Furthermore these higher speed would also be resposible for an increase in fuel consumption of approximately 5%.

In conclusion, CAVs alone are unlikely to reduce congestion in an urban area. To make the platooning concept work, additional technology and infrastructure is required in order to merge platoons effectively at freeways and at traffic lights. The latter could be simulated with GLOSA

Contents

1 Introduction

Micro-simulating realistic, large scale urban areas is challenging, when considering all modes of transport, including pedestrians and public transport [9]. Nevertheless, the performance assessment of connected, autonomous vehicles (CAVs) necessitates a microscopic simulation approach, especially when it comes to platooning. As stated in [6]: "macroscopic studies only consider traffic average parameters, this micro approach is initially more logical to study cooperation between vehicles, to define optimal gaps and speeds, etc". With *platooning*, vehicles are bunched together in trains, where headways between vehicles inside a platoon are shorter than what is achievable with human driven cars. The first vehicle of a platoon is called the leader while all the others are follwers. Vehicles trying to join the platoon are called catchup-followers. It is obvious that the catchup-follower vehicle needs to drive faster than the platoon in order join it.

The simulation of the formation and dissolution of platoons requires a sufficiently detailed modeling of the position, speed and acceleration of vehicles on the one hand as well as the knowledge of the vehicles routes. The vehicle route, which is decided based on link travel times is used to decide whether vehicles form a platoon or not. This means that there is a strong inter-dependency between microscopic events and macroscopic quantities (routes, flows and link speeds), suggesting that a separation between local microscopic simulations and large-scale macroscopic models would give unrealistic results.

The previously cited article also motivates the use of platooning for CAVs: if autonomous vehicles were to run individually, then they would actually decrease capacity because autonomous vehicles drive less agressive and have therefore larger headways than human driven vehicles. In addition, new travel demand is expected to generate as the potential user group of autonomous vehicles is considerably larger than the one able to actively drive a conventional car. For this reason, vehicle cooperation and in particular platooning is needed to offset or reverse these negative effects on the capacity.

The capacity increase of vehicle platoons depends on many inter-related factors, such as

- the platoon length
- achievable headway between vehicles within one platoon
- the effective time gaps between platoons
- the platoon duration, or variation of its length
- the platoon formation

While a vehicle driving as part of a platoon, an on-board distances control system allows short headways to the car in front, based on the distance and speed of both vehicles. Both quantities can be transmitted vai V2V communications or directly measuremented via Radar, laser or optical devices. Headways within a platoon are typically sub-second and are considerably short compared to what human drives are able to achieve. The distance control system is designed to satisfy multiple requirements during all possible speed transitions [2] : safety, string stability and comfort. Concerning safety, there are different *vehicle spacing policies.* The relevant policies are the *constant time headway policy* and *constant safety policy.* The constant time headway policy offers a constant, speed independent carrying capacity [12]. However, collisions can theoretically occur at higher speeds. Despite this drawback, constant safety considerations have been the headway policy for the design of linear control system for platoon-join manœvers [4]. Fewer literature can be found on feedback controllers that follow the constant safety separation policy. A constant safety controller needs by definition a nonlinear, quadratic feedback and does therefore not fit into concepts of standard linear control theory. A common approach to overcome this difficulty is the separation into a "safe trajectory generator", driving a linear feedback controller [4, 13]. An alternative, with a non-linear control-feedback has also been proposed [7]. Concerning string stability, it is required that the speed changes of the platoon leader are not amplified by the following vehicles. In case of linear controller, string stability can be proven easily [12]. The design of a non-linear feedback controller providing string stability is more complex, but also possible, for example by means of the Popov criteria [?]. A critical issue for the string stability is the time delay of the communication or measurement devices: a delay between an the occurance of speed-change of the lead vehicle and the moment when the follower vehicle recognizes it, is crucial to string stability. A large delay is typically jeopardizing string stability. V2V communication for position- and speed communications is typically faster than measuring via radar. If the leader communicated its speed and position to all followers simultaneously, the string stability would also be guaranteed. The issue regarding the inter-platoon space depends on many factors: a basic requirement for lasting platoons is obviously that all vehicles in a platoon have a common route, at least in part – the shorter this common route the the lower is the time vehicles stay together is a platoon. The ability for CAVs to join a platoon depends also on the provided infrastructor. For example extra lanes or ramps to accelerate the vehicle before joining the platoon. Concerning comfort criteria, it is necessary that the control system of CAVs incolved in a platoon guarantees also speed, acceleration and jerk limits.

Concerning the capacity increase, one needs to distinguish between ideal environments and real environments (urban environments in the present article). Shladover, (2012) [11] who has micro-simulated CAVs on a one-lane, intersection-free highway at steady-state traffic flows, has shown an 80% increase in capacity, assuming all vehicles are CAVs. However, micro-simulating CAVs in an urban environment with random trips results in much lower capacity gains of approximately 16%, due to the network-level effect [5]. Clearly, the dynamics in intersections and the durations of platoons appear to have a significant effect on the capacity [3].

Apart from the performance of the CAVs themselves, there is the question on the impact on other road users. One particular issue is the interaction with pedestrians on mixed access roads or at pedestrian crossings, where the average travel speed may reduce for both, pedestrians and C, dependent on the vehicle flows and pedestrian flows. Changes in travel time will in turn change demand and consequently flows of vehicles and pedestrians. See [15] for gap acceptance of pedestrians crossing a road with platooned CAVs.

These examples suggest that, in general, small, microscopic and large-scale macroscopic models cannot be simulated separately, which means only a large-scale microscopic model will ensure that microscopic dynamics will correctly determine the traffic flows and vice versa, thus global, network-level effects needs to be taken into account.

For these reasons, the present work evaluates the performance of platooned CAVs with a large-scale micro-simulation scenario with a realistic demand, including all relevant modes of transport. The next section describes some details about the methods and used parameters, while section 3 presents and discusses the results. Section 4 draws the main conclusion and points to limitation and potential future research.

2 Methods

This section is separated in two subsections, where the transport scenario and the CAV and platooning related aspects are explained.

2.1 The microsimulation scenario

The used validated microsimulation model for the medium size city of Bologna, Italy, with approximately 500,000 inhabitants is explained in detail in [9]. The entire demand has been created through the tool SUMOPy [8] while the simulation itself is performed by the SUMO simulator [1]. in brief, the road network of Bologna city has been converted from OSM to a SUMO XML format by SUMO's "netconvert" program and edited manually with SUMOs "netedit" . In addition, connectivity problems have been identified by matching GPS traces to the network. Traffic lights are an OSM node attribute, but the signals have been generated by heuristics. Large traffic light systems in and around the center have been edited manually based on traffic light plans provided by the city of Bologna. The road-network of the city of Bologna with surrounding towns is the core simulation area, covering approximately 50 km^2. The core area has a detailed street network, including bikeways and footpath. The entire metropolitan area of Bologna covers a wider area of 3.703 km^2. Figure 1 shows the traffic assignment zones (TAZs) of the core area from the 2001 national population census. As there is a substantial traffic between the core simulation area and the extra-urban TAZs, the citys road network has been manually expanded by a simplified road network – using again SUMO's netedit and satellite imaginary.

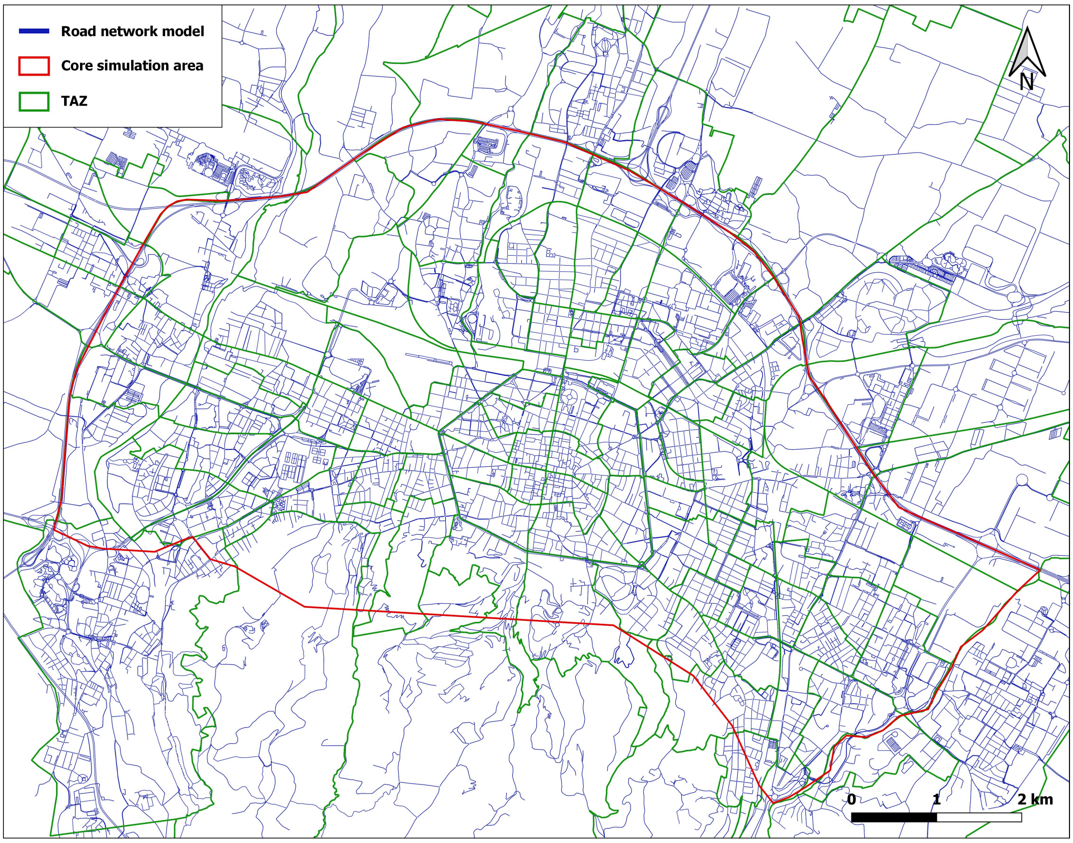


Figure 1: Bologna core network with TAZ.

The total number of road links is 32,409 with a total length of 3,316.20km. The share of major road (with priority level greater 7) is 20.11% of the total length or 667.05km. Moreover, there are 59,218 link connections and 14,724 intersections, 530 of which are regulated by a traffic light. The geometric shapes, heights and type of 58,421 buildings in the core simulation area have also been imported from OSM. Buildings are associated with activity locations of persons in the synthetic population model. In addition, on-street parking lots have been created with some heuristics along suitable roads.

The entire public transport service provided by the local operator (Tper) has been realistically modelled within the core simulation area by generating bus lines based on data from GTFS (General Transit Feed Specification). More specifically, 234 bus lines have been imported from GTFS for a workday in May 2018 during the time from 6:00 to 9:00 a.m. The service frequency has been averaged for this time interval for modeling purposes.

Demand has been generated mainly by disaggregating origin-to-destination matrices (ODMs) valid for a work-day from 7:00 to 8:00. In particular, a virtual population has been created by disaggregating OD-flows to single buildings, while taking their volume into account [10]. ODMs of the modes car, scooter, bus and walking has been used to generate the population, but only for demand flows between TAZs inside the core simulation area. Bicycle demand has been estimated from GPS traces recorded by citizens on a volunteer bases using Smartphone. Each GPS trace describes the movements of each participating cyclist through a sequence of time-stamped and geo-referenced Lat/Lon locations. For the used scenario, GPS traces recorded during the European Cycling Challenge campaign in Bologna in May 2016 have been used. The GPS traces have been filtered and mapmatched to create bicycle routes as a sequence of network edges. Only traces during rush hours have been relevant, more precisely between 8:30 and 10:30 a.m. The number of GPS trips need to be scaled to the effective number of trips, using a scale factor from a previous publication [59]. Also each GPS trace has been associated with a building, by analyzing the terminal GPS points.

At this point the assumed home and work location is defined for each person, and also a "preferred" mode of transport has been associated with each person, depending on the type of ODM or bike in case of GPS-generated persons. Next, plans have been created for each person, connecting the building of home with work activity and by involving the preferred mode. Afterwards some relaxation and calibration has been applied in order to achieve a higher consistency between mode choice and building location.

An external car demand (cars with origin or destination outside the core area) has been created by disaggregating ODMs on edges, considering all OD flows between external zones and zones inside the core area as well as external to external zones that are crossing the core area. With this method, external ODMs have been created for the modes car, scooters. External demand for bicycles have been created from again from GPS traces, following the same selection strategy as used for the OD flows.

Finally the dynamic user equilibrium has been determined for the scenario, optimizing the plans of the virtual population the trips of the external demand components.

2.2 Implementation of CAVs

The demand for CAVs has been generated by simply substituting all car trips with vehicles controlled by the algorithms of the SIMPLA [14] module through TRACI. SIMPLA is deciding over platoon join maneuvers, but it is not controlling the distances between vehicles. Instead, the distances are controlled by the car follower algorithm that can be specifies as a vtype in SUMO. The follower algorithms of can be dynamically changed at any time via Traci. This is necessary, for example if the vehicle changes from follower to leader or vice versa. Parameters of follower algorithms in normal/leader and follower/catchup mode are summarized in Table 1. The speed factor for catchup vehicles is set to two, which means CAVs could go as much as twice the speed to catch up with the leader. This high number is needed to allow a reasonable platoon formation within short distances. The default vehicle model is the standard Krauss model while during platoon operations the Shladover CACC model is used. Platoon update time has been set to 2s, platoon gap equals 15m, platoon split-up time is 3s, and catchup distance equals 100m. As there were frequent congestions due to deadlocks at roundabouts and successive intersections, two SIMPLA underwent two modifications: A vehicle would not change from normal to follower mode unless

Table 1: Vehicle type attributes of leader/normal and follower/catchup.

Attribute	value as leader/normal	value as follower
Length [m]	4.3	
Width [m]	1.8	
Height [m]	1.5	
Passengers	1	
Capacity	4	
Max. speed [m/s]	50	
Speed factor	1.0	2
Speed dev.	0.1	
Speed mode	7	1
Max. accel. [m/s^2]	2.9	5.0
Max. decel. [m/s^2]	3.0	
Emergrncy decel. [m/s^2]	8.0	
Reaction [s]	0.8	0.2
Driver	0.5	0
Min. gap [m]	1.0	0.3
boarding time [s]	4	
loading time [s]	4	
vClass	passenger	
Emission type	average passenger car (all fue types)	
Impatience	1.0	

- the common route between the vehicle and the potential leader have at least a minimum distance (default is 500m)

- the common route between the vehicle and the potential leader have at least a certain number of edges in common (default is 3)

With these modifications deadlocks on networks with a high node density have been entirely avoided. However, it has been observed that SIMPLA works only with the standard lane change model, not with the sublane model.

3 Results

Essentially the above described scenario has been simulated with normal cars and with CAVs and the two results have been compared. The results are shown in Tab. 2, distinguishing vehicles of the virtual population (these are vehicles circulating in the core area) and the vehicles of external demand (these are vehicles circulating also in the periphery).

Table 2: Performance results for virtual population (core area) and external demand (periphery).

Quantity	Normal cars	CAVs	Difference in %
Completed trips, external demand	44,941	47,590	5,89
Completed trips, virtual population	12,061	12,288	1,88
Av. Trip duration, external demand [s]	969.56	903.87	-6.77
Av. Trip duration, virtual population [s]	646.27	607.26	-6.04
Av. Waits, duration external demand [s]	72.16	65.89	-8,69
Av. Waits, duration virtual population [s]	135.30	130.76	-3.36
Av. speed, external demand [km/h]	45.00	54.25	20.56
Av. speed, virtual population [km/h]	26.21	28.19	7,55

The following observation can be made: for the scenario with CAVs, the completed trips (within 1 hour simulated time) is higher, the average trip duration is shorter, the wait times are shorter and the average speeds are higher with respect to the scenario without CAVs. There is a remarkable difference between the improvements made in the core simulation area (with a high node density) and the periphery (with a lower node density): the external car trips in the periphery do clearly profit more of CACs, while the advantages in the dense core area is less pronounced. The increase in completed trips indicates less congestion due to a higher road capacity. The increase in average speed (and consequently a decrease in average travel time) is mainly due to the higher speeds of the catchup followers, but also due to less congestion.

Average velocity and waiting times of bikes and pedestrians did not change significantly (improvements below 1% difference between normal car and CAVs), which is reasonable because both modes have their own network and do not interfere much with the road traffic, except at intersections. The waiting time of pedestrians (this is the time during pedestrians stand still) increases, but not significantly in the absolute sense (increase from 19.59s with normal cars to 20.30 with CAVs). This short average waiting time of pedestrians seems surprising and needs to be further investigated. Another reason for this small influence of CAVs on pedestrians could be because in the core area, where pedestrians are walking, the formation of platoons is less pronounced, as mentioned above. Also public transport average speed and travel time is almost unchanged by the presence of CAVs.

Instead, scooter seem to profit of the vehicle platooning with an increase of 5.28% in velocity and a decrease of 3.84% in trip duration. This means scooters can increase speed by following catchup vehicles, which do overspeed.

Concerning fuel consumption, the scenario with CAVs consumes 4.73% more fuel with respect to the ordinary car scenario, assuming that both scenarios use the same motor technology. Emissions are up 4.72% for CO_2 and 7.16% for all Particle Matter PM_x. Even though CAVs are expected to have cleaner motors or even full electric, it remains the fact that there seems to be an increase in energy consumption of CAVs with respect to normal cars, which is in line with the increase in average speed.

4 Conclusions

A realistic large-scale micro-simulation scenario during one rush hour in Bologna has been simulated – with and without CAVs. In the CAV scenario all private car trips of the normal scenario were substituted by CAVs. The simulated CAVS have been controlled by SIMPLA, with some modifications that avoid deadlock behavior. One of the most significant parameter of the platooning algorithm is the catchup speed which can be as much as twice the ordinary speed. A main limitation of SIMPLA is that it does not run well with the sublane model and it does not explicitly model communication links –these need to be modelled by parametrizing the car following model parameters of the used vehicles. SIMPLA has apparently no issues with string stability, most likely due to the use of simplified car follower models and the absents of communication delays between the vehicles.

The results indicate that the CAV scenario increases average speed by approx 6% in the periphery and 2% in the core area, made of a dense street network. Simulations showed that the presents of CAVs have a very limited effect on the speed and travel time of bikes, pedestrians and public transport. On the other hand, scooters can profit of the higher speed of catchup vehicles, which must overspeed in order to reach their lead vehicle.

The higher average speeds of CAVs is also responsible for a higher fuel consumption and emissions. Therefore, the role of the catchup speed is important as low catchup speeds will not lead to platoon creation over short distances and too high catchup speeds will lead to an increase of fuel/energy consumption, and will potentially compromise safety at certain locations. In summary, the speed gain of a few percent points can only be achieved by allowing extreme overspeed, otherwise the advantage of platooning would hardly be measurable.

The current study has some important limitations: 1) it considers only one rush hour and 2) no V2I communication is implemented; it can be assumed that if vehicles could communicate with traffic lights then this would further boost lane capacity at critical junctions of the network. For this reason it is planed to include the GLOSA module to adapt traffic light cycles to the arrival of vehicle platoons. A further project would be to introduce vehicle sharing, with the consequence that additional empty vehicles would circulate.

5 Acknowledgments

The authors would like to thank Roberto Todisco for fixing the network and for performing the numerous simulation during his master thesis.

References

[1] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz. Sumo - simulation of urban mobility. In *SIMUL 2011, The Third International Conference on Advances in System Simulation.*

[2] R. Horowitz and P. Varaiya. Control design of automated highway system. *IEEE Proc.*, 88(7), July 2000.

[3] Seungjae Lee, Benjamin Heydecker, Jooyoung Kim, and Sangung Park. Stability analysis on a dynamical model of route choice in a connected vehicle environment. *Transportation Research Procedia*, 23:720–737, 12 2017.

[4] P. Li, L. Alvarez, and R. Horowitz. AHS safe control laws for platoon leaders. *IEEE Trans. Contr. Syst. Technol.*, 5(6):615–628, November 1997.

[5] Qiong Lu, Tamas Tettamanti, Dniel Hrcher, and Istvn Varga. The impact of autonomous vehicles on urban traffic network capacity: an experimental analysis by microscopic traffic simulation. *Transportation Letters The International Journal of Transportation Research*, 12, 09 2019.

[6] Margarita Martnez-Daz and Francesc Soriguera. Autonomous vehicles: theoretical and practical challenges. *Transportation Research Procedia*, 33:275–282, 2018. XIII Conference on Transport Engineering, CIT2018.

[7] J. Schweizer. Non-linear feedback control for short time headways based on constant-safety vehicle-spacing. In *IEEE Intelligent Vehicles Symposium*, pages 167 – 172, S.N. – ITA, Giugno 2004. s.n.

[8] J. Schweizer. Sumopy wiki, 2016. http://www.sumo.dlr.de/wiki/Contributed/SUMOPy.

[9] Joerg Schweizer, Cristian Poliziani, Federico Rupi, Davide Morgano, and Mattia Magi. Building a large-scale micro-simulation transport scenario using big data. *ISPRS International Journal of Geo-Information*, 10:165, 03 2021.

[10] Joerg Schweizer, Federico Rupi, Francesco Filippi, and Cristian Poliziani. Generating activity based, multi-modal travel demand for sumo. In Evamarie Wiessner, Leonhard Lücken, Robert Hilbrich, Yun-Pang Flötteröd, Jakob Erdmann, Laura Bieker-Walz, and Michael Behrisch, editors, *SUMO 2018- Simulating Autonomous and Intermodal Transport Systems*, volume 2 of *EPiC Series in Engineering*, pages 118–133. EasyChair, 2018.

[11] Steven Shladover, Dongyan Su, and Xiao-Yun Lu. Impacts of cooperative adaptive cruise control on freeway traffic flow impacts of cooperative adaptive cruise control on freeway traffic flow. volume 2324, 01 2012.

[12] J.T. Spooner and K.M. Passino. Fault tolerant control for automated highway systems. *IEEE Trans. Veh. Technol.*, 46(3):771–785, August 1997.

[13] M. Szillat. PRT system simulation. In *7th Int. Conf. on Automated People Movers'99*, pages 31–33, IDA House, Kalvabod Brygge 31-33,DK-1780 Copenhagen V, May 1999. IDA.

[14] Sumo user. Sumo wiki, 2016. lased visited 28.02.2022.

[15] Roger Woodman, Ke Lu, Matthew Higgins, Simon Brewerton, Paul Jennings, and Stewart Birrell. Gap acceptance study of pedestrians crossing between platooning autonomous vehicles in a virtual environment. *Transportation Research Part F: Traffic Psychology and Behaviour*, 67:1–14, 11 2019.

SUMO User Conference 2022
https://doi.org/10.52825/scp.v3i.111

Published: 29 Sept. 2022

Topology-Preserving Simplification of OpenStreetMap Network Data for Large-scale Simulation in SUMO

Zhuoxiao Meng[1,2*], Xiaorui Du[1,2*], Paolo Sottovia[1], Daniele Foroni[1], Cristian Axenie[1], Alexander Wieder[1], David Eckhoff[2,3], Stefano Bortoli[1], Alois Knoll[2], and Christoph Sommer[4]

[1] Intelligent Cloud Technologies Laboratory, Huawei Munich Research Center, Munich, Germany
[2] Department of Informatics, Technical University of Munich, Munich, Germany
[3] TUMCREATE, Singapore, Singapore
[4] TU Dresden, Faculty of Computer Science, Dresden, Germany

✉Zhuoxiao Meng zhuoxiao.meng@huawei.com
✉Xiaorui Du xiaorui.du@huawei.com
✉Christoph Sommer https://www.cms-labs.org/people/sommer/

Abstract

Converting OpenStreetMap (OSM) data to a road network suitable for microscopic traffic simula-tion keeps being a challenging task: both missing information and excessive details, as well as wrong typologies present in the dataset frequently confuses automatic converters. In this paper, we present a method along with a reference implementation, Traffic Simulation Map Maker (TSMM), which aims at substantially increasing the automation level of road network prototyping by simplifying the OSM data while preserving important topology information. The main objective of this work is to enable the study of traffic simulation dynamics at scale using real-world road networks, while minimizing the need for solving the long tail of problems related to the road network generation. Our proposed approach yields what we believe is a good trade-off between precision and automation, making bold yet acceptable decisions that solve most of the errors at the source, i.e., the map. While there is definitely a loss in fidelity with respect to the real world, many properties of the road network are preserved. We argue that TSMM greatly improves the availability of arbitrarily large and usable road networks on top of available OSM maps by reducing the complexity for conversion tools and traffic simulation researchers alike. A proof-of-concept study using OSM data from Binjiang, China, demonstrates that TSMM is able to generate a road network with well-preserved topological information which avoids the many errors and deadlocks that occur when building the network using the original input sources.

[*] Zhuoxiao Meng [https://orcid.org/0000-0001-7224-2461] and Xiaorui Du contributed equally to this work

1 Introduction and Motivation

Microscopic traffic simulation is a mature domain which has been studied for decades [1]–[3]. A well-established traffic scenario is often the basis for a meaningful simulation study. However, building such a scenario is, in most of the cases, a challenging task, especially when it comes to generating the road network. Considerable effort is usually required to gather, process, and transform the collected data from different government agencies and map providers. Achieving a highly automated efficient road network production is thus becoming more and more pressing in recent years both for academics and traffic engineering practitioners.

At present, OpenStreetMap (OSM) [4] data is a common data source for the road network generation to support researchers and practitioners in generating proof of concept scenarios. As a publicly available platform, OSM offers geographic data for most of the areas of the world in a standardized XML format. Traffic simulators such as SUMO [5] are able to convert it into simulator-native road network formats. However, the road networks obtained from a direct conversion of OSM data often contain crucial errors and inconsistencies due to ambiguous and often erroneous long tail of details and corner cases. Hence, cumbersome and time-consuming manual efforts for post-processing are thus still necessary, outright precluding the generation of large scale scenarios.

There are three reasons why converting OSM data directly to a road network for microscopic traffic simulation is problematic: (i) Intersections are not modeled with an explicit data structure in OSM, requiring the conversion process to guess relevant information like intersection geometry, waiting lines, and lane-to-lane connections[1]. (ii) Considering that most of the OSM data is produced by volunteers, human errors are very frequent. Incorrect connections, gaps, misclassifications, and broken `ways` are common issues found in the road network for a given region [6], which could result in disconnections and inconsistencies in the traffic simulation. (iii) Variability in how real world geometry is represented in OSM data is another reason. Although there are conventions, they are not always followed due to the sheer number of editors and covered areas. For example, some bi-directional roads are represented by a single bi-directional `ways`, while others by two separate uni-directional `ways`. Turn lanes are sometimes classified separately and sometimes they share the same road type as the joining streets. Such variability leads to a substantial increase in map data complexity and further aggravates the challenges of automatically converting OSM data to a road network suitable for microscopic traffic simulation.

In this paper, we present the Traffic Simulation Map Maker (TSMM)[2] toolchain that follows a novel approach to convert OSM data to SUMO networks. So far, researchers attempting to generate a road network for traffic simulations based on OSM data must first simplify and fill in the data manually. TSMM substantially increases the automation level of this process. By extracting data from OSM, TSMM can generate a lane-level SUMO network for any regions. In a step-by-step fashion, the complexity of the OSM data is gradually reduced by eliminating auxiliary elements. At the same time, most important data and information is kept and processed to ensure the consistency. TSMM aims to support researchers in overcoming the challenges caused by incorrect and incomplete data in original OSM map data. The road network generation process proposed in this work allows to produce a fully connected and positionally accurate road network through a one-click automated process.

The goal of TSMM is to create error-free road networks at scale. This translates into a higher priority for correctness compared to realism. The generated road network retains the original road layout, road classification, and intersection locations, while complex side roads, lane connections within intersections, ramps, etc., are simplified and represented in configurable templates. Typical use cases of such simplified networks include, but are not limited to: (i) Generating a road network which features the typical mix of low and high traffic density roads and can therefore be used in simulation studies of

[1] https://wiki.openstreetmap.org/wiki/Junctions

[2] TSMM source code and data repository URL is hosted on Zenodo: https://doi.org/10.5281/zenodo.6482355

Inter-Vehicle Communication [7] applications. (ii) Providing large-scale road networks for researchers who are simulating traffic with High Performance Computing frameworks to evaluate the reliability and performance of relevant designs and algorithms. (iii) Offering general-purpose road networks which can be used to evaluate traffic policies that are not city-specific and rather target a certain type and scale, e.g., a middle size European city or a densely populated Asian metropolitan area.

For usage scenarios which require higher details, on the other hand, we hope that TSMM can accelerate the process of road network prototyping.

2 Data Representation

OSM is a crowdsourced publicly accessible platform which provides rich Volunteered Geographic Information (VGI) that is free to use and edit [8]. Since the focus of this paper is to generate road networks for microscopic traffic simulation, only the following elements regarding the road network in OSM are relevant: `nodes`, `ways`, and `relations`. We briefly describe these elements in the following.

Nodes: A `node` is the basic element in the OSM data model. Each `node` represents a single point and is defined by a coordinate. Usually each `node` should have a unique `node` id.

Ways: A `way` is the representation of (part of) a street. A `way` consists of several `nodes`, which define the shape of the street. Each `way` is characterized by a set of tags, defining the typology and specifications of the street. These tags show, among the others, the road level (e.g., `motorway`, `primary`, `trunk`) and the number of lanes. Another important tag is `oneway`. When a `way` is tagged as `oneway = yes`, the `way` is then uni-directional and only allows vehicles to travel from its first `node` to the last one, conversely is then prohibited. Besides real one-way roads in small districts, dual carriageways are also often marked with this tag and then represented by two approximate parallel uni-directional `ways`, each in an opposite direction. However, this will cause problems and confuses the conversion tools when generating a road network for traffic simulation, as we will discuss later.

Relations: In the OSM data model, `ways` are connected by default when they share the same `nodes`. That is, vehicles are allowed to travel from one `way` to another via their common `nodes` unless other rules apply. Access restrictions for connected `ways` can be defined using `relations`, for instance, to prohibit U-turns or right turns (with `no_u_turn` or `no_right_turn`, respectively).

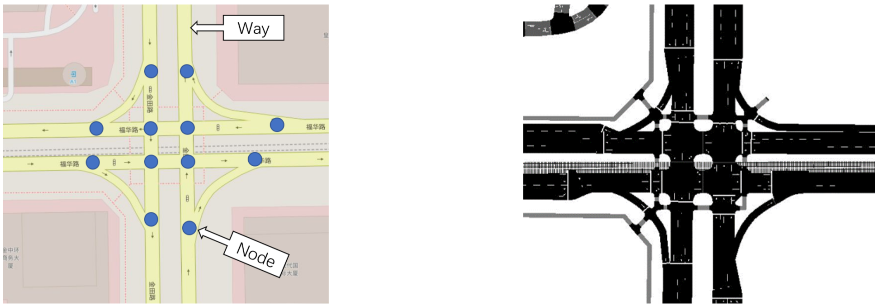
An example illustrating a junction represented by OSM data is depicted in figure 1a.

Similar to the OSM data model, a SUMO network also consists of nodes and edges, the latter describing, e.g., streets, bike lanes, or walkways [3]. SUMO provides several tools for generating SUMO road networks from OSM data: NETCONVERT and OSMWebWizard are both able to convert OSM data to SUMO networks, while NETEDIT is a graphical network editor that can be used to manually fill in missing information and correct errors in the converted network files.

A major problem of converting OSM data to a SUMO network is to identify the association of nodes to intersections. As shown in figure 1a, one intersection could consist of multiple `nodes` and `ways`, particularly when the `ways` are modeled by two parallel uni-directional roads with opposing directions. While this approach for modeling intersections is well suited for navigation or routing services, and allows for appealing visual map rendering, it poses challenges for the conversion to SUMO road networks.

Automatic conversion using, for instance, NETCONVERT, requires the correct identification of `nodes` and `ways` that represent regular streets or are part of an intersection. Incorrect identification will lead to

[3] Map data and OSM tiles © OpenStreetMap contributors; terms: www.openstreetmap.org/copyright



(a) original OSM data[3]

(b) SUMO map after conversion; multiple intersections are generated instead a complete one

Figure 1: Representation of a intersection in OSM and the result of converting it to SUMO without simplification.

inconsistencies, such as the one shown in figure 1b, where a single intersection is interpreted wrongly as multiple sub-intersections. This can cause vehicles to misbehave, triggering deadlocks, as we will show later, e.g., in figure 12a. Such problems are discussed at length in literature on medium-scale scenarios such as the work by Codecá et al. [9] and Rapelli et al. [10] regarding the generation of the SUMO Luxembourg and Turin scenarios. A commonly employed solution is therefore manual editing of large parts of the network.

3 Related Work

The challenges of road network generation from OSM data has been addressed in prior work. Some studies [11], [12] suggest that nodes which represent the same intersection can be integrated into one relying on reasoning over geometric connection relations, such as the spacing among the nodes, and whether they can form a circuit or not. However, this approach can typically handle intersections consisting of only a small number of nodes, and it struggles over complex intersections with extra turn lanes. Other researchers presented methods to use the semantic information in the OSM data to help locate intersections [7]. However, such methods highly depend on the homogeneous and consistent quality of the OSM data available.

Other studies [13], [14] aim to capture parallel dual carriageways in OSM data through machine learning. However, the focus of these studies is to identify multi-lane roads for road hierarchy analysis rather than traffic simulation. The integration of multi-lane roads, the processing of single-lane roads, as well as the identification and representation of intersections for traffic simulation are not in the scope of these studies.

In this paper, we present a novel approach that is applicable to a wide variety of cases to simplify the OSM data with a strong emphasis on generating an error-free and consistent road network. This is done by firstly merging multiple OSM `ways` (in particular those trying to approximate a median strip separating opposing lanes of the same road) into single `ways` using a buffer method to represent the streets, and then using the crossing points among all the simplified `ways` to represent the intersections. In the end, the resulting simplified OSM network includes only streets and intersections which can be straightforwardly converted to a SUMO network by Netconvert, as discussed before.

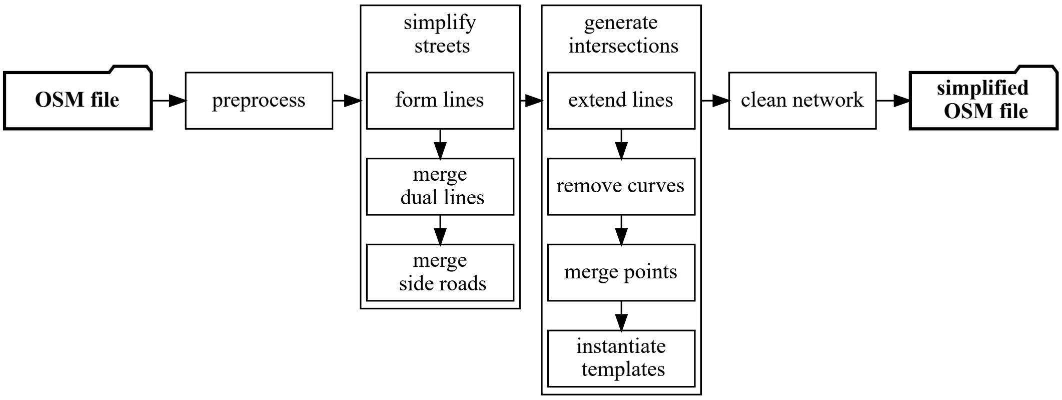


Figure 2: Steps performed by TSMM.

4 Methodology

TSMM takes OSM data as input and produces simplified OSM data as output, which is suited for conversion with NETCONVERT. The simplification of the OSM data is performed in a sequence of steps building upon each other, as illustrated in figure 2. We detail these individual processing phases in the following.

4.1 Preprocessing

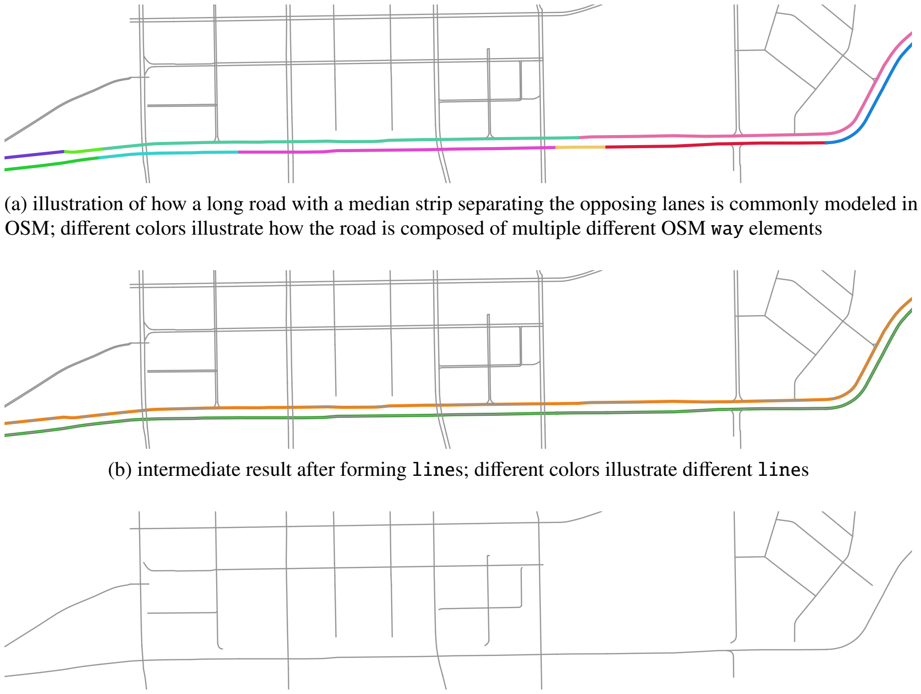
At present, the emphasis of TSMM is solely on generating road networks for motor vehicles. Roads used primarily by other traffic participants, such as bicycle lanes, crosswalks, and bus lanes, are beyond the scope of this work. Therefore, the raw OSM data is first pre-processed by filtering out irrelevant elements. In particular, only roads of type `motorway`, `trunk`, `primary`, `secondary`, `tertiary`, and `residential` are extracted from the raw OSM data; other types are discarded.

4.2 Simplifying Streets

The first step of simplification is to form `lines`. Note that a `line` is not the name of a data structure that is available in OSM nor in SUMO. Instead, in this paper, we use the concept of a `line` to refer to an ordered set of `ways`; these `ways` have the same road type and are connected to adjacent `ways` within the group trough their first or last `node`.

Forming `lines` is straightforward, starting from an arbitrary `way`, searching forward and backward for its adjacent `ways` and, if the angle between them is less than a certain value, adding them to the `line`. One notable situation is that it is possible for a `way` to have multiple other `ways` connected at its end. In this case, that `way` among all connected `ways` which has the smallest angle with the `line` will be selected to continue the `line`. Exemplary input and output data of this step is shown in figures 3a and 3b, respectively.

Two approximately parallel `lines` (such as those highlighted in figure 3b) often represent a dual carriageway in the real world. Thus, by identifying and merging these into one single `line`, the raw data can be substantially simplified. Typically, the two `lines` in the dual `lines` are not too far apart since they

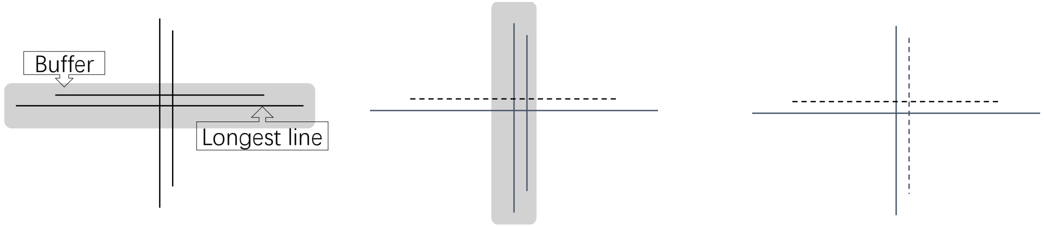


(a) illustration of how a long road with a median strip separating the opposing lanes is commonly modeled in OSM; different colors illustrate how the road is composed of multiple different OSM `way` elements

(b) intermediate result after forming `lines`; different colors illustrate different `lines`

(c) result after merging dual `lines`

Figure 3: Simplification of `ways` in OSM data.



(a) after creating a buffer based on the longest `line`

(b) after deleting `lines` within the buffer and creating the next buffer; removed `ways` illustrated as dashed lines

(c) after merging all dual `lines`; removed `ways` illustrated as dashed lines

Figure 4: Buffer method for merging dual `lines`.

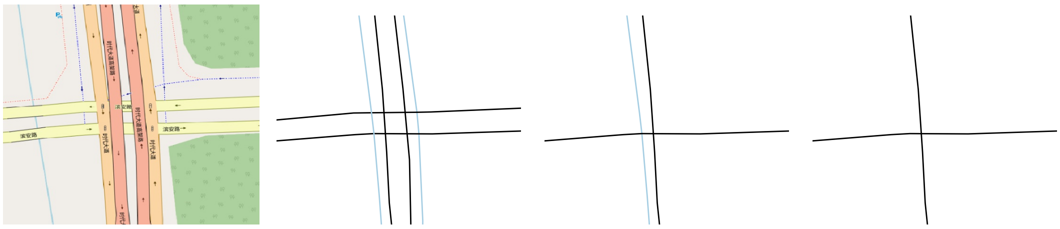
are representing the same street. For the same reason, their shape and alignment should also be relatively similar.

Therefore, we propose a *buffer* approach to merge the dual `lines`, as shown in figure 4. We use the term *buffer* to denote a polygon with a fixed width, which is generated around a `line`. First, such a buffer is generated based on the longest `line` l in the current map data. All other `lines` with same road type as l and are covered by the buffer over a certain percentage with respect to the length will be then deleted in the map data. Consequently, `line` l will be then tagged as a bi-directional street. The amount of its

Algorithm 1 Buffer method for merging dual lines Algorithm

```
Input: set of all lines, $L = \{l_0, l_1, ...\}$
Output: set of remaining lines, $R = \{\}$
  while $L$ is not empty do
    Pick longest line $l$ from $L$;
    Add $l$ to $R$;
    Generate buffer polygon based on $l$;
    Remove any line from $L$ that is of the same type and of which at least $p\%$ length is inside the buffer;
  end while

  return $R$
```



(a) rendering of side roads in OSM map[3]

(b) representation of side roads (blue) and main roads (black) with `lines`

(c) after merging dual `lines`

(d) after merging side roads

Figure 5: Process of merging side roads into a main road.

lanes is thus doubled to ensure a consistent road capacity as before. Next, the buffer is generated starting from the longest of the remaining `lines`, and the process is repeated until all `lines` are processed. The pseudo-code for the buffer algorithm can be found in algorithm 1.

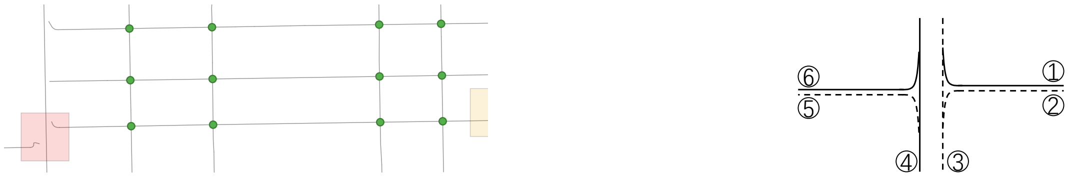
As shown in figure 3c, all the dual `lines` in the original OSM data are represented by a single `line` each now. As can be seen, the resulting network has a similar road layout compared to the original map while its complexity is substantially reduced.

One remaining problem are side roads of a different road type than the main road (figures 5a and 5b). As figure 5c shows, side road and main road are still represented by two approximate parallel `lines` after the buffer approach if only roads of the same type are considered. To solve this, the buffer method is thus applied again – the only slight difference being that, this time, the road merge process also considers any road which has a "lower" type (e.g., `secondary` vs. `primary`) than a given road as being eligible for merging into the present road. In order to ensure the consistency of the data, the number of lanes of the main roads is increased with the corresponding lanes amount of side roads, same as the approach taken by Wang et al. [11].

At this point, the simplification for streets is completed. Next, we focus on the corresponding operations for generating the intersections.

4.3 Generating intersections

As shown in figure 6a, after the last step, streets are now all represented by only one `line`. In this case, the crossing points among the remaining `lines` (illustrated as dots) are naturally the location of the



(a) generation of intersections after simplifying streets; t
hand rectangle marks an area where a disconnection occu
ure 6b); the right-hand rectangle marks an area where the m
process created a sharp right-turn curve (figure 6c).

(c) disconnection and sharp turn curve caused by simplifying streets

Figure 6: Illustration of issues for generating intersections.

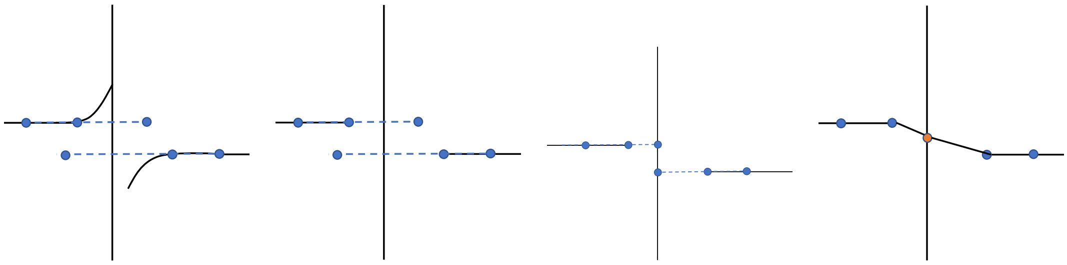
intersections in real world. Thus, the next step is to calculate the coordinates of these crossing points and insert them as `nodes` into the corresponding `ways`. While this approach works in most cases, additional processing is required for certain issues, as explained below.

Several streets are disconnected after the previous simplification, as is highlighted with the bottom left rectangle in figure 6a. One case of such disconnections is shown in figure 6b. In this case, the `way` (dashed line) which connects the other two `ways` was filtered out during the data pre-processing because it is not tagged with an appropriate road type (e.g., `unclassified`) or tagged as auxiliary elements (e.g., `link`). This caused the break of those two originally connected `ways`. Such an issue occurs more frequently when data is less complete and accurate. Another case of disconnection happens after the dual `lines` merging process, as is illustrated in figure 6c. Originally, `line` 1 is connected with `line` 3. However, `line` 3 is merged into `line` 4 as both `lines` jointly represent the dual street. This leads to a disconnection between `line` 1 and the dual street in the simplified road network.

Apart from the problem of disconnections, unreasonable intersection shapes (bottom right rectangle in figure 6a) caused by the simplification process also need to be considered. The reason for this issue is shown in figure 6c as well. After `line` 5 was merged, `line` 6 became the complete dual street. Unlike `line` 1, the connection between `line` 6 and the dual street represented by `line` 4 and `line` 3 is still ensured. However, in the original data, `line` 6 has a left-turn curve. If the turn curve is directly converted to a bi-directional edge in the SUMO network, an intersection with an unreasonable shape will be generated: Right-turning vehicles driving from `line` 6 to `line` 4 are confronted with a sharp turn angle. This can lead to severe congestion in the simulation that is purely an artifact of the unreasonable intersection shape.

Therefore, there is an automatic pipeline in TSMM, illustrated in figure 7, which aims to solve the issues mentioned above: First, all `lines` are extended by a certain length at the head and tail ends. If a curve is detected there, the extension will be based on the overall angle of `line` and the curve will be subsequently removed. By doing this, the `lines` remaining in the map will reconnect to the others and crossing points can be formed to represent the intersections. Next, among the crossing points formed by the extended `lines`, very closely spaced points can be further merged together and excess trimmed. In this manner, intersections that were not able to be captured previously can be created, and sharp turn curves can be straightened out to form normal streets. We found that the length the `lines` need to be extended and the proximity parameter for merging intersections play an important role. TSMM suggests different values based on the road level, and users can tune them according to the roads characteristics (e.g., road density) of their target city.

Regarding the representation of intersections, TSMM provides three different types of templates which it instantiates depending on the level of roads that connect to the intersection.



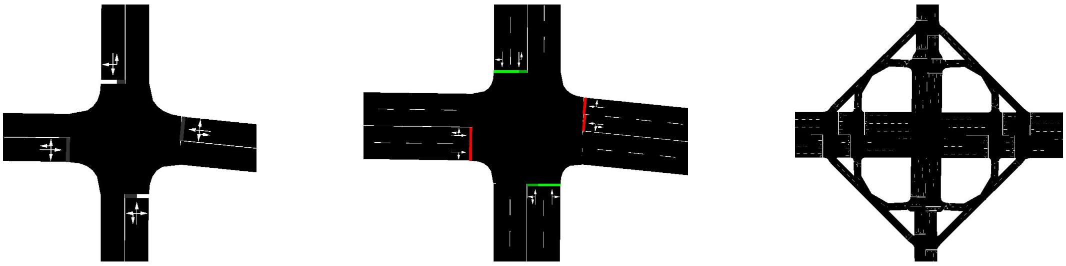
(a) extending roads along main direction

(b) removing curves near the intersection

(c) creating nodes at the intersection

(d) merging nearby nodes

Figure 7: Steps in the `line` extension process.



(a) uncontrolled intersection

(b) traffic light controlled intersection

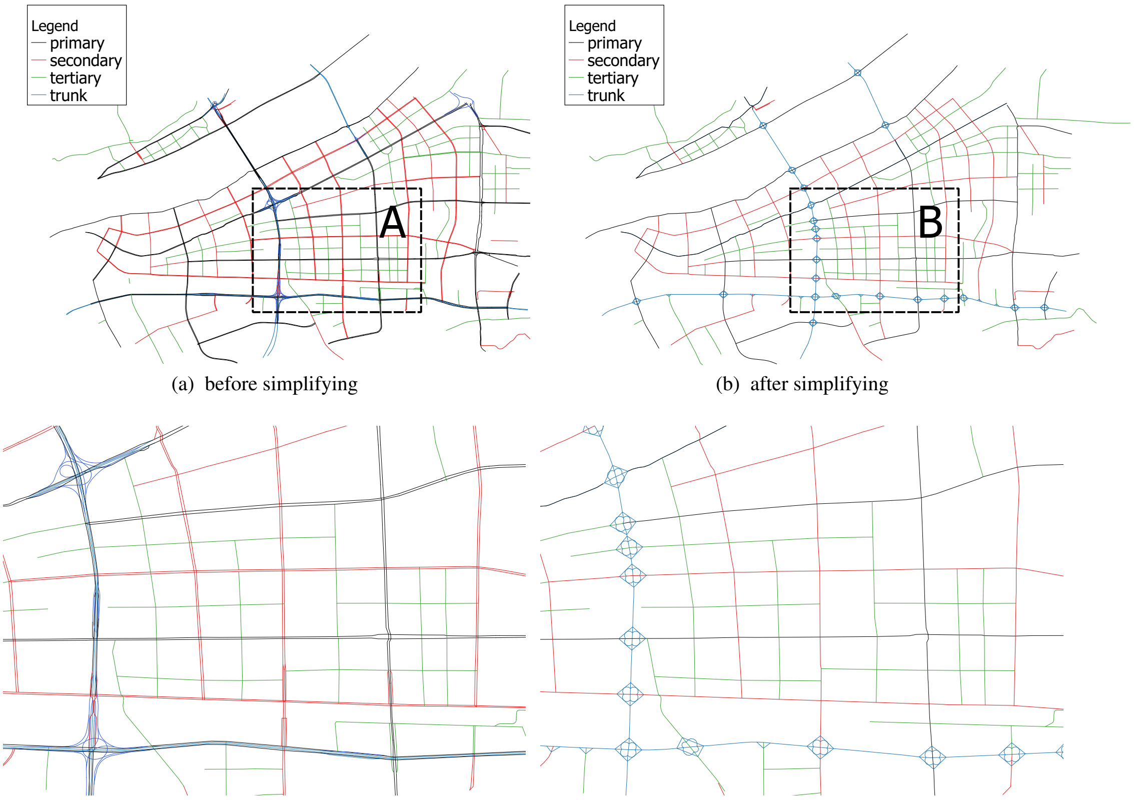
(c) grade separated intersection

Figure 8: Classification of intersections.

Uncontrolled intersection (see figure 8a): In the real world, these intersections are regulated only by right-of-way rules. They are found mostly among low-level streets, like residential `ways`. For this type of intersections, no additional operations have to be applied in TSMM. When a `node` is shared with more than two `ways`, by default this is converted to an uncontrolled intersection with NETCONVERT.

Traffic light controlled intersection (see figure 8b): Some intersections require traffic lights. The standard identification of traffic light controlled intersections in NETCONVERT is through the information provided by the tags of the `nodes`. For instance, if a `node` is tagged with `traffic_signals`, the intersection formed by this `node` will be converted to a traffic light controlled intersection. Standard traffic light programs will also be provided by NETCONVERT if there is no additional data filled in by the users. In addition, with TSMM, users can define which level of intersections should be controlled by traffic lights. TSMM is then able to identify and tag these additional `nodes` to ensure a proper conversion later.

Grade separated intersection (interchange) (see figure 8c): For streets like `motorway` and `highway`, it is necessary to separate the traffic in the vertical grade in order to maintain high driving speeds of vehicles and maximize traffic throughput. Users of TSMM can therefore define which type of street should be served with interchange. So far, TSMM always uses clover leaf interchanges to represent all grade separated intersections.



(a) before simplifying

(b) after simplifying

(c) zoomed view of figure 9a

(d) zoomed view of figure 9b

Figure 9: Comparison before and after simplifying.

4.4 Network cleaning

Full connectivity of the road network plays an important role in many SUMO simulations, yet mapping errors in OSM data can sometimes leave spurious roads in the network. After simplifying the road network from the previous steps, TSMM therefore uses the Breadth-first search (BFS) algorithm to group all intersections into different clusters according to their connectivity to each other. In the end, the cluster with the highest number of intersections will be defined as the true road network – and intersections and streets which are not in it will be removed from the road map. In general, the removed parts are supposed to occupy only a small proportion, otherwise, the user needs to reconsider the parameters being used.

5 Evaluation

The purpose of this experimental case study is to evaluate the performance of our tool using real OSM map data. The road network in Binjiang, a central district of the Chinese city Hangzhou, which covers 72.2 km^2 in total, was tested. The road network contains 161 km roads and 128 intersections, among which 28 are grade separated intersections.

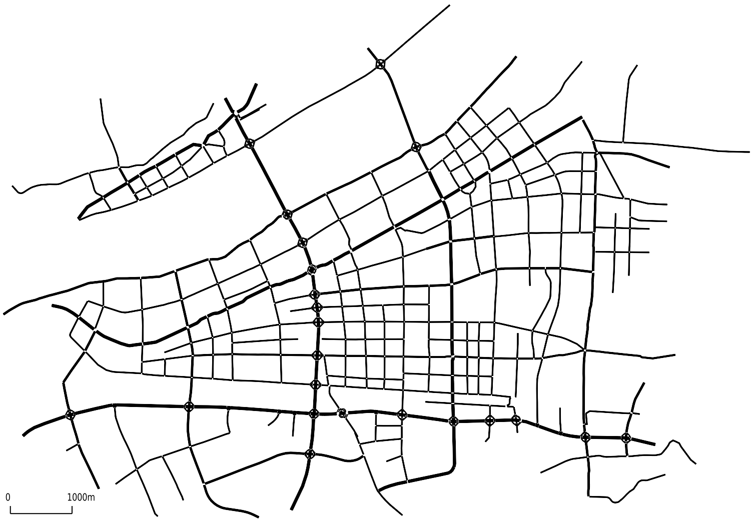


Figure 10: SUMO network of Binjiang.

As shown in figure 9a and figure 9c, the original OSM network mainly consists of dual uni-directional roads. As can further be seen, `primary` (in black) roads appear in several places as the side roads of `trunk` roads (in blue). This means that the vehicles driving on the `trunk`s should not have conflicts with traffic from other roads; the connection among them should be achieved through the `primary` side roads. However, the design of these side roads is often so complicated and non-standardized that converting them into a SUMO network directly through Netconvert rarely succeeds. Such poor identification often even leads to broken roads and deadlocks.

Therefore, as described in section 4, TSMM merges side roads into main roads and inserted clover leaf interchanges for all the intersections on the merged roads during the simplification process. The simplification was executed on a mid-range Desktop machine operating at 3.90 GHz taking 21 seconds with non-optimized code.

Based on the proposed methodology, TSMM is able to simplify the original road network while retaining the topological information with respect to location of intersections, overall road layout, and road classifications. In this manner, although the layout visually differs from reality, it still ensures to a large degree the topology of the road network regarding traffic characteristics (figure 9b and figure 9d).

5.1 Preparation

After the simplification, the OSM road network is fed into Netconvert to be converted to a SUMO readable network file (figure 10).

For comparison, the non-simplified OSM road network is converted as well. Netconvert itself contains many parameters and switches; for a comprehensive comparison, we converted the non-simplified road network with two different versions. The first version, same as converting the simplified OSM network, is converted with default options, as follows:

```
netconvert --osm-files <osm-file> -o <sumo-network-file>
```

For the second version, it is converted with additional options which are frequently recommended[4]

[4] https://sumo.dlr.de/docs/Networks/Import/OpenStreetMap.html

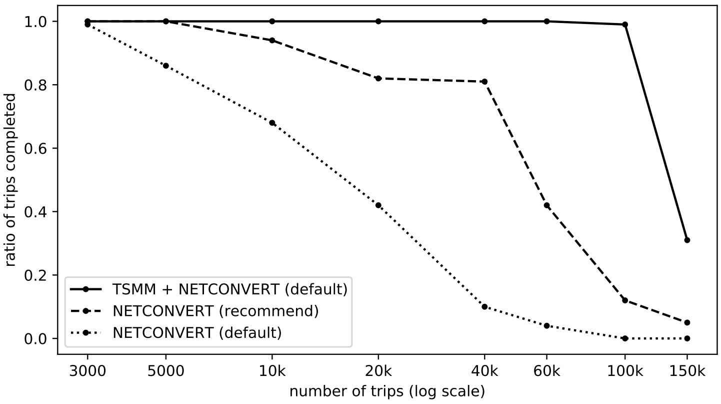


Figure 11: Rate of completed trips vs. increasing travel demand. Different road networks can sustain differently high demand.

for NETCONVERT. Conversion with these additional parameters is able to simplify the raw OSM data automatically, joining the nodes heuristically, guessing location of ramps, and inserting traffic lights for uncontrolled intersections. The parameters are:

```
netconvert --osm-files <osm-file> -o <sumo-network-file> --geometry.remove
    --ramps.guess --junctions.join --tls.guess-signals --tls.discard-simple
    --tls.join --tls.default-type actuated
```

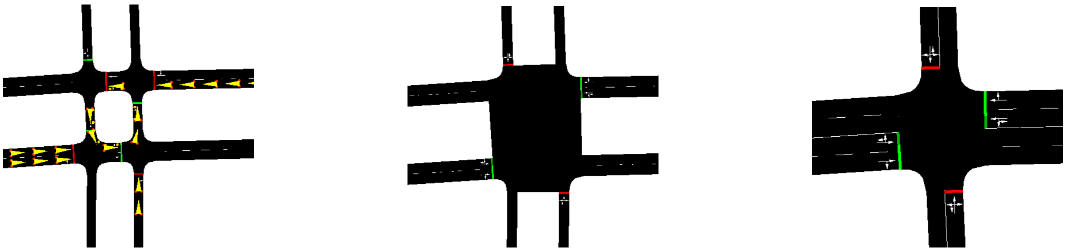
We also prepare traffic demand for the experiments as follows. To ensure the consistency across all scenarios, the trips files for each network are taken from the same dataset. In the dataset, a certain amount of trip data is stored, where each trip consists of a coordinate pair within the study area, and a randomly assigned departure time within the time interval 0–180 min. In the next step, the trip data is converted to SUMO readable trips files for each network. The departure time is kept identical to the one in the dataset, while the origin and destination places are set to the closest edges to the coordinates.

The simulation time is set to 6 hours, which is sufficient for all the vehicles to finish their trip if there is no congestion. The teleporting of the vehicles (the fail-safe procedure employed by SUMO when unexpected events such as deadlocks happen) is disabled unless a collision has occurred or if no valid route has been found for a vehicle. At the end of each simulation, by analyzing the car counts statistics and visually inspecting the network to identify bottlenecks, the performance of each road network can be assessed.

5.2 Assessment

As illustrated in figure 11, continuously increasing the initial amount of traffic reveals that the tested road networks start to show uncompleted trips when the traffic volume reaches 5000, 10 000, and 100 000 trips, respectively. Much more so than the quantitative performance of the networks, however, we are interested in why bottlenecks materialize.

At the end of the simulation we therefore inspect visualizations of jammed intersections. This analysis of the causes of bottlenecks allows to determine whether the congestion is caused by excessive traffic volume or by errors in the road network itself.

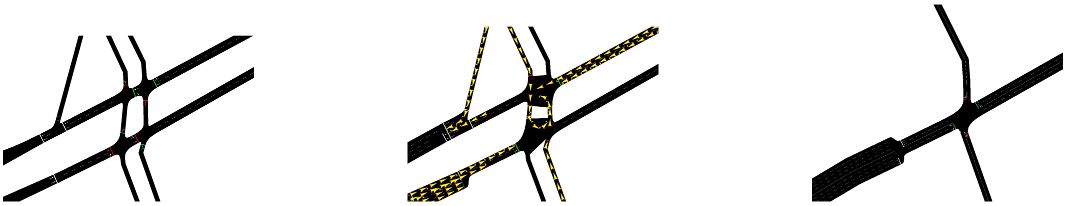


(a) a representation of congested intersections at the end of simulation; converted via default settings

(b) same intersection as figure 12a converted via recommended settings

(c) same intersection as figure 12a converted via TSMM

Figure 12: Analysis of uncompleted trips in the road network converted directly via default settings with 5000 initial trips.



(a) same intersection as figure 13b converted via default settings (shown without traffic)

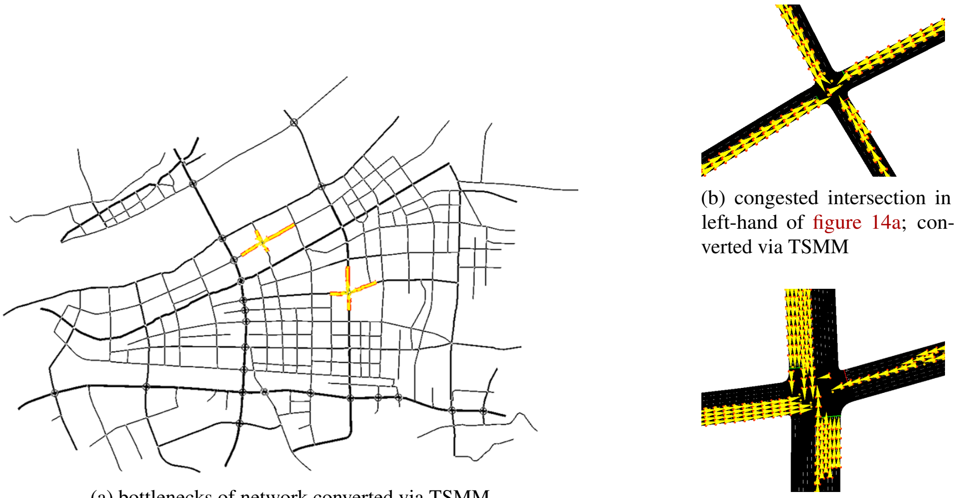
(b) a representation of congested intersections at the end of simulation; converted via recommended settings

(c) same intersection as figure 12b converted via TSMM

Figure 13: Analysis of uncompleted trips in the road network converted directly via recommended settings with 10 000 initial trips.

The results after the simulation reveal that the non-simplified road network generated using the default settings has the lowest traffic capacity compared to others. When 5000 cars were imported into the network as input demand, nearly 400 cars were already unable to complete their trip. Congestion occurred at many intersections at the end of the simulation. Through observing one of their representatives (figure 12a), it is revealed that the congestion was not caused by excessive traffic but rather by errors in the road itself. This is because, by default, Netconvert treats all nodes crossing different ways in OSM data as intersections; four intersections, each with an independent traffic light controller, are hence created here. Consequently, when too many vehicles enter this complex multi-intersection area, the vehicles block each other due to traffic lights and narrow spaces. A deadlock was thus created and henceforth obstructed all vehicles that were supposed to pass by. On the contrary, the same intersection generated via recommended parameters (figure 12b) and via TSMM (figure 12c) are free of such deadlocks.

The road network, converted with the recommended parameters via Netconvert shows a remarkable improvement in the deadlock discussed above. However, the algorithm it uses can only cope with simple cases when multiple nodes form one intersection in OSM data. It still lacks the ability to address more complex situations. Thus, at the end of the simulation, unresolved traffic congestion at many intersections still appeared across several intersections. One of these intersections can be seen in figure 13b. Part of the



(a) bottlenecks of network converted via TSMM

(b) congested intersection in left-hand of figure 14a; converted via TSMM

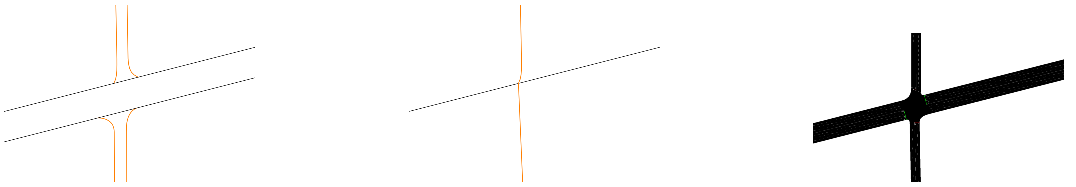
(c) congested intersection in right-hand of figure 14a; converted via TSMM

Figure 14: Analysis of uncompleted trips in the road network converted via TSMM with 100 000 initial trips.

nodes in this widely spaced intersection are clustered together and two independent traffic light controlled intersections are generated in the end. Although better than the result converted via default settings (figure 13a, shown without traffic), where four intersections are created, a deadlock is still unavoidable when a large amount of traffic is flowing into this area. On the contrary, TSMM is able to identify all the nodes associated with this intersection properly and generate a deadlock-free intersection (figure 13c).

Still, even the simplified network generated by TSMM is not free of congestions. As shown in figure 11, however, the simplified road network generated via TSMM can accommodate nearly 100 000 vehicles to finish their trips. This is a multi-fold increase in capacity compared to the other two networks and might hint at less deadlocks. Figure 14a depicts the location where congested vehicles gathered after the simulation. By zooming in on these two intersections, as shown in figure 14b and figure 14c, respectively, it can be confirmed that the congestion is solely caused by excessive traffic volumes, as undesired network errors are not revealed in either area.

Summing up, in our experiments, errors were found in both networks generated without TSMM, most owing to the data quality of OSM data. These errors caused deadlocks and further limited the capacity that the networks should have reached. As discussed, the errors are mainly caused by the failure of identifying the correct OSM nodes to form the intersections, especially in cases involving dual carriageways and side roads. This matches exactly with the goals of TSMM. Our proposed approach proved its ability to simplify OSM data while preserving the topology of the network (at the cost of a less detailed road network representation). After the simplification with TSMM, the generated road network not only reaches the largest capacity, but also avoids any deadlocks caused by road network errors.



(a) original OSM data: Straight ahead is not allowed in highlighted streets.

(b) simplified OSM data: No turn restriction is applied in highlighted (yellow) streets.

(c) converted SUMO network: straight ahead is allowed in each entry.

Figure 15: At the moment, TSMM is incapable of identifying turn restrictions.

6 Limitations

While already a worthwhile tool for the simplification of OSM data, follow-up research will be carried out to address some limitations still present in TSMM.

One limitation concerns turn restrictions. Currently, TSMM is discarding turn restrictions modeled in OSM – both explicitly and implicitly. The simplified road network is always fully connected at every intersection, i.e., straight ahead, right turn, left turn, and U-turn are all allowed at every intersection. This is also due to the simplification process. For instance, figure 15a shows an intersection as modeled in the original OSM data, where traffic on the highlighted road is restricted to right and left turns only; driving straight through the intersection from north to south or vice versa is not possible. In the simplified road network generated via TSMM, however, this complex intersection is replaced with one crossing point (figure 15b). Consequently, the implicit turn restrictions are lost in the final generated SUMO network, as shown in figure 15c.

Another limitation concerns roundabouts. The method proposed in this paper is not applicable to roundabouts. In OSM data, a roundabout is always a closed circular loop consisting of one or several ways. Thus, when forming `ways` into `lines`, the algorithm of TSMM would be trapped in an infinite loop. The current interim approach to avoid an infinite loop is to identify and filter out any roundabout during the pre-processing phase, and filling it back in after the step of simplifying dual streets. However, the roundabouts are often cut by extended lines when generating the intersection points. In the future, a specific algorithm for the roundabouts therefore needs to be proposed.

Yet another limitation concerns extensibility. TSMM is hoped not be limited only to OSM data. To meet the different needs of map detail, more Application Programming Interfaces (APIs) must be developed to be able to consume other data sources. Speed limit, lane count, layout of intersections, and traffic lights can all be configured in an incremental way within the pipeline only if additional data sources and corresponding APIs for processing them are available.

Lastly, by leveraging validated scenarios such as SUMO Luxembourg [9], quantitative evaluations of similarity of the generated road network must be performed. The comparison of metrics such as travel-times and travel-distances will reveal how close the one-click automatically generated road network via TSMM is to a real one prepared over several months, indicating if TSMM might not just work for large-scale simulations interested in more macroscopic effects, but also for high-detail simulation.

7 Conclusion

In this paper, we presented a method along with a reference implementation, Traffic Simulation Map Maker (TSMM), to substantially increase the automation level of road network prototyping by simplifying the OpenStreetMap (OSM) data while preserving important topology information. The main objective was to enable the study of traffic simulation dynamics at scale using real-world road networks, while minimizing the need for solving the long tail of problems related to the road network generation.

Primarily this is achieved by merging dual carriageways and side roads into single `lines`, then instantiating intersection templates. We believe this is a good trade-off between precision and automation that solves many conversion problems at the source, i.e., the map, most of the errors. While there is definitely a loss in fidelity with respect to the real world, many properties of the road network are preserved. We thus argue that TSMM greatly improves the availability of arbitrarily large and usable road networks on top of available OSM maps by reducing the complexity for conversion tools and traffic simulation researchers alike.

A proof-of-concept study using OSM data from Binjiang, China demonstrated that TSMM is able to generate a road network with well-preserved topological information within seconds. Compared to the non-simplified OSM data, the SUMO network generated from the simplified version effectively avoided all artificial deadlock situations.

The reference implementation of TSMM is publicly available to interested researchers.[2] In future work, it will be continually upgraded to address more sophisticated situations like roundabouts and to enhance the realism of the resulting network by incorporating more details.

References

[1] M. Fellendorf and P. Vortisch, "Microscopic traffic flow simulator VISSIM," in *Fundamentals of traffic simulation*. Springer New York, 2010, pp. 63–93. DOI: 10.1007/978-1-4419-6142-6_2.

[2] D. Zehe, S. Nair, A. Knoll, and D. Eckhoff, "Towards CityMoS: A Coupled City-Scale Mobility Simulation Framework," in *5th GI/ITG KuVS Fachgespräch Inter-Vehicle Communication (FG-IVC 2017)*, Erlangen, Germany: FAU Erlangen-Nuremberg, 2017.

[3] P. A. Lopez et al., "Microscopic Traffic Simulation using SUMO," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, Nov. 2018. DOI: 10.1109/ITSC.2018.8569938.

[4] J. Bennett, *OpenStreetMap*. Packt Publishing Ltd, 2010.

[5] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, "SUMO (Simulation of Urban MObility)-an open-source traffic simulation," in *Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002)*, 2002, pp. 183–187.

[6] F. Tabet et al., "OSMRunner: A System for Exploring and Fixing OSM Connectivity," in *22nd IEEE International Conference on Mobile Data Management (MDM 2021)*, IEEE, 2021, pp. 193–200. DOI: 10.1109/MDM52706.2021.00039.

[7] C. Bewermeyer, R. Berndt, S. Schellenberg, R. German, and D. Eckhoff, "Poster: cOSMetic – towards reliable OSM to sumo network conversion," in *2015 IEEE Vehicular Networking Conference (VNC)*, IEEE, 2015, pp. 151–152. DOI: 10.1109/VNC.2015.7385562.

[8] M. Haklay and P. Weber, "OpenStreetMap: User-Generated Street Maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, Oct. 2008. DOI: 10.1109/MPRV.2008.80.

[9] L. Codecá, R. Frank, S. Faye, and T. Engel, "Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 2, pp. 52–63, 2017. DOI: 10.1109/MITS.2017.2666585.

[10] M. Rapelli, C. Casetti, and G. 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)*, IEEE, Oct. 2019. DOI: 10.1109/DS-RT47707.2019.8958652.

[11] H. Wang, J. Wang, P. Yu, X. Chen, and Z. Wang, "Extraction and construction algorithm of traffic road network model based on OSM file," in *2021 4th International Conference on Advanced Algorithms and Control Engineering (ICAACE 2021)*, vol. 1848, Sanya, China: IOP, Jan. 2021, p. 012 084. DOI: 10.1088/1742-6596/1848/1/012084.

[12] T. Ziemke and S. Braun, "Automated generation of traffic signals and lanes for MATSim based on OpenStreetMap," *Elsevier Procedia Computer Science*, vol. 184, pp. 745–752, 2021. DOI: 10.1016/j.procs.2021.03.093.

[13] Y. Xu, Z. Xie, L. Wu, and Z. Chen, "Multilane roads extracted from the OpenStreetMap urban road network using random forests," *Wiley Transactions in GIS*, vol. 23, no. 2, pp. 224–240, Dec. 2019. DOI: 10.1111/tgis.12514.

[14] Q. Li, H. Fan, X. Luan, B. Yang, and L. Liu, "Polygon-based approach for extracting multilane roads from OpenStreetMap urban road networks," *Taylor & Francis International Journal of Geographical Information Science*, vol. 28, no. 11, pp. 2200–2219, May 2014. DOI: 10.1080/13658816.2014.915401.

SUMO User Conference 2022
https://doi.org/10.52825/scp.v3i.123

Published: 29 Sept. 2022

Implementation of a Perception Module for Smart Mobility Applications in Eclipse MOSAIC

Robert Protzmann[1], Karl Schrab[2], Moritz Schweppenhäuser[1], and Ilja Radusch[2]

[1]Fraunhofer Institute FOKUS, Berlin, Germany
{robert.protzmann, moritz.schweppenhaeuser}@fokus.fraunhofer.de
[2] Daimler Center for Automotive IT Innovations (DCAITI),
Technical University of Berlin, Germany
{karl.schrab, ilja.radusch}@dcaiti.com

Abstract

Nowadays, smart mobility applications could benefit from environment perception, en-abled by evolving sensor technology and processing capabilities available for traffic entities. On the application level, in many cases, information about detected objects is required in-stead of the raw sensor data. Developing and evaluating the impacts of such applications can be done in co-simulation frameworks, which combine the modeling of different domains such as application, communication, and traffic. Eclipse MOSAIC is a suitable solution for this task, combining the traffic simulation of Eclipse SUMO with other simulators, such as the integrated Application Simulator, or OMNeT++ and ns-3 for modeling commu-nication. However, a model for perceiving surrounding traffic entities, such as vehicles, traffic signals, and traffic signs, is only available to a limited extent. In this paper, we introduce an object-level perception module to the MOSAIC Application simulator. It takes advantage of state-of-the-art spatial indexing methods to get rapid access to traffic objects, especially moving objects, within a defined field of view. We furthermore evaluate the computational performance of the indexing techniques as well as the integration with the traffic simulator SUMO using *TraCI* and *Libsumo*. With the aid of this model, novel connected applications that analyze or share surrounding objects, e.g. for an improved traffic state estimation, can now be evaluated with Eclipse MOSAIC.

1 Introduction

Safety, efficiency, sustainability - connected vehicles and associated applications are promised to improve future mobility for more than a decade now. In order to evaluate the impact of these applications by simulation, we developed the framework Eclipse MOSAIC (formerly VSimRTI), which combines simulators from the different domains of applications, traffic, communication, and others, needed for holistic modeling of Intelligent Transport Systems (ITS) [4, 11].

From the view of the distributed applications, especially the MOSAIC Application simulator plays an essential role as it covers the most important participants in a traffic scenario. For this purpose, it models several entities, which are connected, and which are equipped with an application logic. First of all, *Vehicles* are considered as moving entities, whereas their movements and states are simulated in a vehicle simulator like PHABMACS [10] or Carla [2], or a traffic simulator like Eclipse SUMO [9] and synchronized to the Application Simulator. Due to the bidirectional coupling, applications in the Application Simulator could control probable state changes, which then influence the behavior in the vehicle or traffic simulator. *Pedestrians* are another kind of moving entity. As stationary entities, *Traffic Signals* are usually also directly related to the vehicle and traffic simulators, due to their close and also pre-defined interaction with the moving participants such as vehicles. Additional stationary entities are *Charging Stations*, *Road Side Units (RSUs)* and (novel) *Traffic Signs*. These entities might be part of physical infrastructure on or beside streets, but are not necessarily related to the vehicle and traffic simulator since their interaction might be part of the conducted research and require a dedicated implementation. *Cloud/Edge Servers* and *Traffic Management Centers* (which are specialized Servers) are also stationary, but usually not part of road infrastructure. They realize interaction with other entities by communication.

With evolving sensor technology and processing capabilities, more and more data are handled by the traffic entities, enabling novel mobility applications. Nowadays, radar, cameras, and even LiDAR have found a permanent place in vehicles for the detection of traffic objects in the surrounding area of the vehicle. The perceived information supports safety applications, but also efficiency could benefit. One example is estimating the traffic state on the road by considering surrounding vehicles, e.g., estimating the density on roads. Furthermore, parking space solutions could be imagined with direct detection by vehicles. Not only vehicles could be equipped with the perception technology, but also stationary entities like traffic signals or RSUs at certain intersections, or gantries on highways. All in all, there are many constellations of how traffic entities could perceive each other. For the simulation system, perception implies new interaction possibilities between entities.

On the one hand, common research in the field of simulated perception sensors has mainly been motivated by the implementation and validation of Advanced Driver Assistance Systems (ADAS) [6, 8, 12]. This results in detailed models trying to closely mimic the behavior of the physical sensors. Hence a vehicle simulator might be the best choice for realization. Yet, generating such sensor data with a vehicle simulator and handling them in application models is computationally expensive and might be only applied, when needed. On the other hand, there are many applications, which rely on pre-processed data like *Object Information* instead of *Raw Data* like sensor images. The object-related data may contain the positions, dimensions, and directions of other traffic entities. In turn, research questions, about how such applications improve traffic efficiency, might be investigated in large-scale scenarios. To enable this is the main goal of this paper.

Accordingly, this paper presents a Perception Model on Object-Level in the MOSAIC Application Simulator to be used in combination with large-scale traffic simulations. The implementation aims for a generic solution with a consistent interface for applications supporting the seamless exchange of different simulators such as Carla, PHABMACS, and SUMO, which are all coupled to MOSAIC. Yet, it uses most of the features of the traffic and vehicle simulators themselves. Specifically for this paper, the implementation details are presented on the basis of SUMO and in turn on SUMOs Traffic Control Interface (*TraCI*) and the newer and faster solution *Libsumo*, as well as the available Context Subscriptions to retrieve relevant object information. These objects will then be maintained by the Application Simulator in a spatial index for fast search of objects in sight of view. Currently, the solution supports the most relevant case that vehicles perceive vehicles, meaning moving participants perceive moving participants, which implies the highest requirements for the spatial search. We choose a simplified perception model for the start. Error models or occlusion of traffic objects are not in the scope of this paper but will be part of further work.

The paper is structured as follows. At first, Section 2 discusses data structures and methods for fast handling (update and search) of spatial objects. Section 3 presents how the new perception module is integrated into Eclipse MOSAIC. Specifically, it covers the implementation in the Application Simulator, but also the interfaces for coupling SUMO. In Section 4, the implementation is evaluated regarding performance. In fact, we implemented several solutions, which are compared in different scenarios. Finally, Section 5 concludes the paper and gives an outlook for ideas of further integration and also of using the perception module for investigations.

2 Spatial Search

In the context of mobility simulation, all objects have a specific location on a 2-dimensional plane (ignoring the altitude). Those objects are static or can move on the plane. Perceiving other objects (e.g. from the viewpoint of a vehicle) requires an algorithm to find all objects within a given area. On a 2-dimensional plane, the problem of finding localized objects within a range can be solved using spatial search algorithms. There mainly exist two different types of spatial search:

- Finding the k-nearest neighbors given an arbitrary point in the search space.
- Finding a set of objects or points within a given range, which could be a rectangle, a circle, or any other geometrical shape.

In order to model a perception module supporting moving objects, it requires us to find all vehicles (or any other object) within the sight of any other vehicle. Therefore, an algorithm for a spatial range search needs to be considered.

In a very naïve approach, this could easily be solved by checking all simulated objects in a loop. This approach, however, would become increasingly slow with large object quantities, as for every vehicle all other existing vehicles (and objects) need to be considered. This always results in a complexity of $O(n^2)$ for n objects. Much faster approaches employ some form of spatial index to speed-up queries. In general, spatial indexes work by grouping spatially close objects in some form. A common approach is to use spatial trees, such as K-D-trees, R-trees[5], or Quad-trees. These data structures allow a fast search of localized objects, resulting in a complexity of $O(\log(n))$. A major problem with these kinds of data structures is, however, that they are fast at finding points, but not at updating the structures when objects are moving. Furthermore, the issue of moving objects can be solved by using a grid-based data structure, which stores the objects in cells spanned across the map.

The choice of the best suited spatial index highly depends on the type of data, the amount of data, the dimensionality of the data, and the dynamics of the use case. Therefore, a lot of research has been conducted to compare different spatial indexes on different datasets. For example, Kothuri et al.[7] conducted an in-depth comparison of Quad-trees and R-trees on spatial data and concluded that index building and update operations are faster in Quad-trees, while query operations perform faster on R-trees on average. In search of a well-suited spatial index for our purposes, we will further describe and evaluate those approaches.

2.1 Range Search in Spatial Trees

Trees are a simple yet effective structure to find specific data points in a given set. For example, binary search trees are used to find one-dimensional data, such as objects in a hash table. For spatial data, e.g. localized objects within a 2-dimensional plane, a special kind of tree is required to gain similar results. All existing tree-like structures used for spatial search follow the *branch-and-bound* principle, which arranges data points within certain bounds on different levels. These bounds can be used during a search in the tree to discard branches that lay outside the range to search in.

An **R-tree** groups nearby points and creates bounding rectangles around those. This procedure is repeated within each bounding rectangle, representing a second level of the search tree. Depending on the number of points existing in the search space, this is done multiple times resulting in m levels. During search for a range (or single point), only those bounding rectangles of the first level which contain the search range are considered for deeper search.

A **K-D-tree** shows a very similar structure. Instead of bounding rectangles, a median line on one axis through the search space is calculated, dividing the set of points into two halves. This is again done for each of the two halves but using the other axis. The whole process is executed several times depending on the number of points, resulting in different levels and branches to be used during search. The main disadvantage compared to the R-tree is, that no items can be added or removed without re-building the entire tree.

In a **Quad-tree**, the search space is divided into four quadrants. This is again done recursively for each quadrant, depending on the number of points inside. This results in a tree with nodes having zero or four children each. Similar to the R-tree, insert and remove operations can be implemented efficiently. However, if the resulting tree is quite unbalanced (i.e. the points in the search space are unevenly distributed), the complexity of all operations search, insert, remove, can increase up to $O(n)$. Therefore, the Quad-tree requires a suitable configuration of the size of the quadrants (tile size), which is determined from the initial bounds the tree is covering.

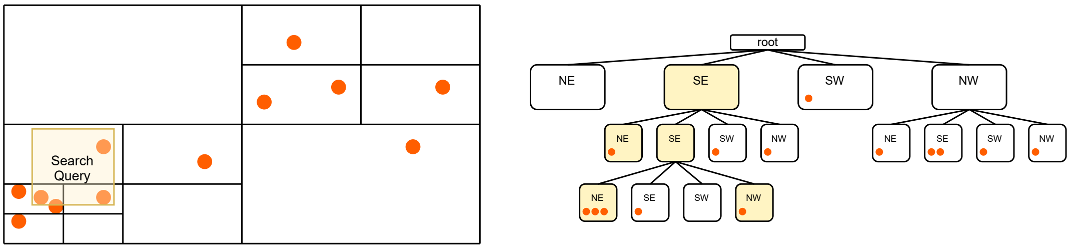


Figure 1: Example of a Quad-tree with a split size of 4. Only those quad-tiles which intersect with the search query are considered.

All tree variants described above are optimized on search, but lack on updating the objects in the search space. The K-D-tree requires a complete re-build when adding or removing objects, and the R-tree requires the bounding rectangles to be continuously updated when objects are moving. The Quad-tree, however, uses fixed tile sizes and could be a good candidate as it gives room for improvements that are suitable for our use cases. Therefore, we propose to use a Quad-tree like structure, with the additional properties (see Fig. 1):

- Each node of the Quad-tree stores multiple objects. Only if a maximum number of objects per node (*split size*) is exceeded, the node is split and all objects are distributed into four child quad tiles. Also, if all child quadrants of a node together contain less than a minimum number of objects per node (*join size*), all child nodes are joint. The actual values for *split size* and *join size* might depend on the expected total number of objects present in the search space.

- The previous improvement also supports moving objects. Therefore, we check if the moved object is still within the bounds of its tile. Only if it is outside, the object is completely removed from the tree and added again.

2.2 Range Search Using Spatial Grid

A different approach to index spatial data is using a grid-like structure instead of t rees. In this case, the search domain is split by a grid into equally sized cells. To address cells by given coordinates, the x and y-coordinates are converted accordingly. The formula for this conversion depends on the size of the grid and the area bounding the spatial data to search for. Adding or moving data points can be done in $O(1)$ time, as the cell address can be derived from its coordinates. To find data, all cells intersecting with the search range need to be considered, and all data points within those cells need to be checked (see Fig. 2). Therefore, the performance of this index depends on the grid size and the number of data points in each cell. One disadvantage of this data structure is memory consumption, as cells are always allocated even if they do not contain any data points. This could be problematic, especially in large-scale scenarios. When using range queries, the size of the search area (e.g. a rectangle or circle) affects the number of cells to be considered. It is advised to use range queries not larger than one cell, limiting queries to four grid cells at maximum.

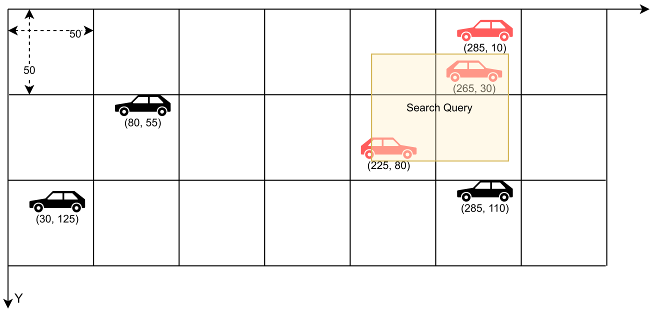


Figure 2: Searching spatial items using a grid index with a cell size of 50. Objects in cells are selected according to the search query.

2.3 Finding Traffic Objects in Sight of View

To model a perception facility, we define a field of view for every vehicle. All other traffic objects within this field of view should be detected by this model with the aid of the previously described methods for spatial search. Any occlusion or error models are neglected here but could be added as an additional filter. The field of view is defined by a sight distance h and a view angle γ, resulting in a circle around the center of the vehicle bound by two vectors (see Fig. 3). The perception is done in two steps:

1) An axis-aligned minimum bounding rectangle (MBR) around the field of view is calculated. Using the sight distance h as the radius of a circle around the center position of the vehicle, vectors b and c are calculated using the opening angle γ. Furthermore, any of the four additional vectors $\vec{a_0}, \vec{a_1}, \vec{a_2}$, and $\vec{a_3}$ with a length of h in both directions of both axes are considered, if they lie within the opening angle γ (see Fig. 3). Based on these vector points, the MBR can be easily calculated and is then used to query a range search in the given spatial index.

2) For all items returned by the spatial range search we check if they are inside the field of view of the vehicle. For each traffic object m at relative location $\vec{m}$, we use the dot product with vectors $\vec{b}$ and $\vec{c}$ to determine if object m lies inside or outside the bounds (Eq. 1 and 2). This approach has the advantage, that it does not rely on trigonometric functions during requests and can therefore be executed faster. However, this method only works for viewing angles $< 180°$. An additional check by comparing the magnitude of $\vec{m}$ with the sight distance h reveals if the object m is visible or not (Eq. 3).

$$\vec{c} \cdot \vec{m} \geq 0 \tag{1}$$

$$\vec{m} \cdot \vec{b} \geq 0 \tag{2}$$

$$|\vec{m}| \leq h \tag{3}$$

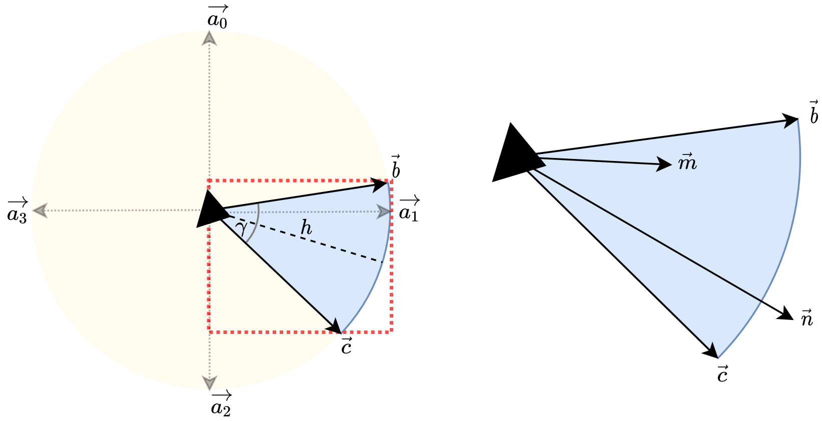


Figure 3: Field of view as a circle bound by two vectors $\vec{b}$ and $\vec{c}$ derived from sight distance h and a view angle γ. Left: Determining an axis-aligned minimum bounding rectangle (MBR, red) to use as search query. Right: Using vectors $\vec{b}$ and $\vec{c}$, and the sight distance h from the center of the vehicle, it can easily be checked if traffic objects at $\vec{m}$ or $\vec{n}$ are visible.

3 Implementation in Eclipse MOSAIC

The implementation of a perception module in the simulation framework Eclipse MOSAIC affects various components. To comprehend our design decisions, we give a brief overview of the concepts used in the MOSAIC framework. MOSAIC includes a middleware written in Java, which couples various simulators with each other. This middleware, the runtime infrastructure (RTI), uses *interactions* to exchange data among all coupled facilities. The coupling of an individual simulator is following the *federate* and *ambassador* principle, in which the actual simulator is wrapped by a federate interface that exchanges information with the ambassador. The ambassador is therefore coupled directly with the RTI by implementing and employing provided interfaces. A strict separation of the federate and ambassador is not mandatory, as a simulator that is already written in Java can be directly integrated into an ambassador implementation.

Modeling of smart mobility applications for different entities is done in the Application Simulator bundled with Eclipse MOSAIC. The movements of individual vehicles are usually modeled in a vehicle or traffic simulation, e. g. with PHABMACS or Eclipse SUMO. In order to provide a perception module for individual applications, the Application Simulator requires positions of all perceivable objects, especially vehicles in the traffic simulation. The following subsections briefly describe both the application simulator and the SUMO coupling, before continuing with the actual implementation of the perception module.

3.1 The MOSAIC Application Simulator

Eclipse MOSAIC supports applications for several entities such as vehicles, pedestrians, traffic signals, RSUs, servers, traffic management centers, and charging stations. The MOSAIC Application Simulator, therefore, provides an API to easily integrate custom application models. It allows, for one thing, using various facilities available on the respective unit (see Fig. 4). For example, applications have access to a communication module for building and sending V2X messages. Additionally, vehicles are provided with a navigation module to calculate routes on the road network. Furthermore, application code can read vehicle data, such as position, speed, heading, or sensor data, such as the distance to its leader. This information is usually derived from the traffic or vehicle simulator coupled with MOSAIC, e.g. SUMO. Additionally, many vehicle-actions can be triggered, e.g. stopping or re-routing the vehicle, or changing speed and lanes.

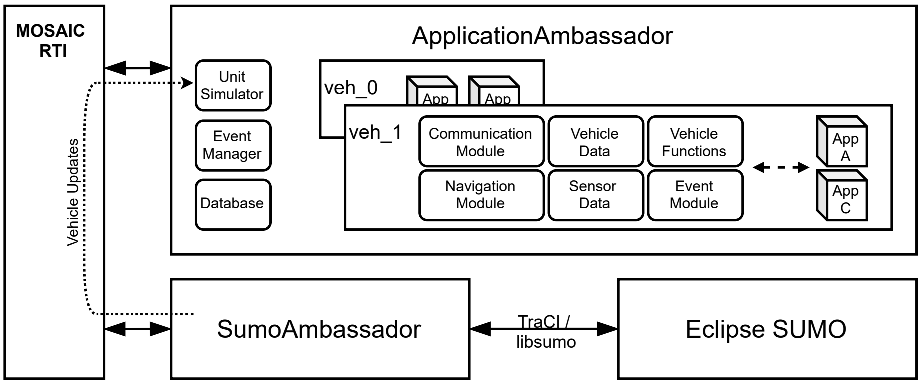


Figure 4: Overview of the Application Simulator bundled with Eclipse MOSAIC.

3.2 Coupling Traffic with Eclipse SUMO

In the context of MOSAIC, the traffic simulation of SUMO is used to handle the movements of individual vehicles. A coupling interface is used to realize the integration since MOSAIC is written in Java and SUMO in C++. In the current version, the *Traffic Control Interface* (TraCI) is used for this purpose. This socket-based protocol allows exchanging information between SUMO and any other application. The protocol is well-documented and could be easily adapted by the contributors of MOSAIC and is used for many years. However, the main issue of this coupling method is its performance, as it hardly depends on the I/O capabilities of the machine SUMO is running on. To reduce I/O overhead, SUMO uses the concept of *subscriptions*, which allows the client to define in advance which data from which vehicles should be sent via *TraCI* after each simulation step. This indeed reduces much of the protocol overhead, but the requested data still must be sent via the socket.

The *SumoAmbassador*, which implements the *TraCI* client on the MOSAIC side, could already be configured in a way, that only vehicles equipped with MOSAIC applications are subscribed. This is helpful in many use cases, especially in those with low penetration rates. However, when thinking of perception, the position of all entities (i.e. vehicles) is again required to insert them into the search space, which would require basic position data for all vehicles to be subscribed.

As an alternative to *TraCI*, recent developments introduced *Libsumo* as a second method to integrate SUMO into Java- or Python-based programs. This library makes use of the SWIG framework to create bindings for different programming languages, such as Java, allowing to integrate SUMO as a native library using the Java Native Interface (JNI). With this method being used, data is not sent via a socket anymore but is read directly from the memory. This integration method has been adopted by the MOSAIC coupling implementation of SUMO, enabling faster simulations, especially when subscribing to many vehicles.

The requirements to use the perception module by MOSAIC applications include, that the positions of all present vehicles must be known at any time. This implies that the coupling implementation of SUMO with MOSAIC has to request basic data for all vehicles, such as position, heading, and speed. To achieve this, basic subscriptions for all vehicles are used in the *TraCI* implementation or read directly from SUMO using the *Libsumo* implementation. In that context, SUMO already comes with a feature that could solve the problem of object perception already without the need for dedicated spatial indexing and search in the MOSAIC Application Simulator. With *context subscriptions*, any client can subscribe for data of all surrounding vehicles for a given ego-vehicle. Additionally, SUMO can already filter surrounding vehicles by defining a field-of-vision, which is expected to give equal results as our own perception model. This indeed would reduce the amount of data to read, as only (single and context) subscriptions for equipped vehicles would be required. However, the main goal is to provide a unified API in the application simulator regardless of the traffic or vehicle simulator used. Therefore, we decided to implement two solutions: 1) makes use of context subscriptions and the field-of-vision filter in SUMO, and 2) implements our own perception module which is fed with basic position data of all vehicles in the simulation. In Section 4.2 the solution of using context subscription is directly compared with our perception module, which is described in the following.

3.3 Perception Module

The perception module is integrated into the MOSAIC Application Simulator in a way, that each application has access to it. As a result, every entity can request surrounding objects in its field of view. Next to vehicles that perceive other traffic participants, this allows modeling road side units or traffic signals having access to a sensor detecting other traffic objects. The API usage from inside an application can be read in Listing 1.

The spatial index itself, be it Grid or Quad-tree, is only instantiated once and can be accessed globally by all active perception modules. To avoid unnecessary update calls, e.g. if the perception is called rarely or not used at all, the index is only updated on request (lazy loading pattern). To achieve this, we provide a parameter `spatialIndexUpdateInterval` to define how long (in simulation time) the index is used for requests without being updated. Each spatial search request comes with a `PerceptionRange` parameter, which provides a bounding rectangle as the search range query. The result of the index is then filtered by the perception module according to its field of view (see Section 2.3).

The actual implementation of the spatial index variants Grid and Quad-tree can be found in the Eclipse MOSAIC repository [1]. The implementations are interchangeable (see Fig. 5) as the best-performing index might be chosen dependent on scenario size or view range. A hash table is used for both implementations to fast update the actual traffic objects in the index. The index itself is updated once the location of all traffic objects has been changed. If the grid cell index of the moved object has not changed, or it is still within the bounds of its current quad-tile respectively, then no update operation for that particular object is performed.

Listing 1: Activating and using the perception module in a MOSAIC application.

```java
public class PerceptionApp extends AbstractApplication {

  @Override
  public void onStartup() {
    getOs().getPerceptionModule().enable(
        new SimplePerceptionConfiguration(60.0, 200.0)
    );
  }

  @Override
  public void onVehicleUpdated(VehicleData previous, VehicleData updated) {
    List<VehicleObject> vehicles =
        getOs().getPerceptionModule().getPerceivedVehicles();
    // do something with the perceived vehicles
  }

}
```

To optimally perform in different scenarios the Quad-tree can be parametrized. A quad-tile is not split until more objects than `splitSize` are stored in that particular tile. If `splitSize` is exceeded, four more quad-tile are created and all objects are spread over these according to their positions. To avoid unnecessary split and join operations, the parameter `joinSize` defines when four quad-tiles are joined together. This parameter must be lower than `splitSize` to function. Furthermore, the `maxDepth` parameter is used to determine how many levels of split quad-tiles can exist at maximum. If the maximum depth is reached, a quad-tile can hold more than `splitSize` objects.

Parameters for the Grid index only adjust the width and height of each cell. Therefore, the actual size of the grid (number of cells) varies with the scenario boundary. The four corners of the bounding rectangle of the search query are used to find the minimum and maximum row and column indexes. All objects of the cells within these bounds are collected and filtered to fit into the search range query.

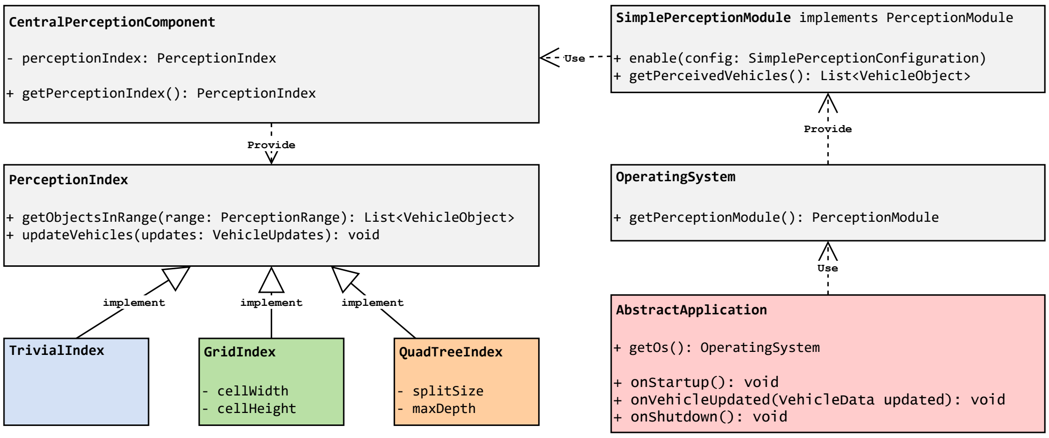

CentralPerceptionComponent
- perceptionIndex: PerceptionIndex
+ getPerceptionIndex(): PerceptionIndex
Use
SimplePerceptionModule implements PerceptionModule
+ enable(config: SimplePerceptionConfiguration)
+ getPerceivedVehicles(): List<VehicleObject>
Provide
Provide
PerceptionIndex
+ getObjectsInRange(range: PerceptionRange): List<VehicleObject>
+ updateVehicles(updates: VehicleUpdates): void
OperatingSystem
+ getPerceptionModule(): PerceptionModule
Use
implement
implement
implement
TrivialIndex
GridIndex
- cellWidth
- cellHeight
QuadTreeIndex
- splitSize
- maxDepth
AbstractApplication
+ getOs(): OperatingSystem
+ onStartup(): void
+ onVehicleUpdated(VehicleData updated): void
+ onShutdown(): void

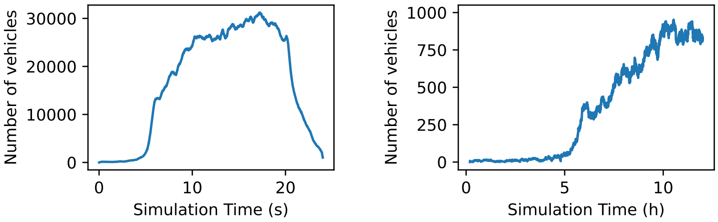
Figure 5: Class structure of the perception integration in the MOSAIC Application simulator.

4 Evaluation

In this section, we compare the performance of the spatial indexes implemented in MOSAIC. In addition, we also compare our solution with built-in features in SUMO to find out which implementation is more suitable for specific use cases. For our experiments, we employ various simulations with MOSAIC coupling the traffic simulation of SUMO. For the actual simulation of the traffic, we defined two simulation scenarios of different sizes. Firstly, we use a full-day city scenario with several million vehicles in total, in order to measure the performance on a large scale. Secondly, we use an inner-city scenario with several hundreds of vehicles with varying traffic demand, which is a more common size for a mobility simulation.

The **Berlin** scenario contains traffic demand with around 1,8 million trips during 24 hours within the whole city of Berlin, Germany. The traffic demand was generated by extracting 80 % of vehicle trips from the MATSim Open Berlin [13]. Routes for these trips have been iteratively calibrated using the `duaIterate.py` script[1]. This scenario is furthermore planned to be published under an open-source license on GitHub [3].

The **Charlottenburg** scenario covers a smaller area of Berlin. It was created by cutting out traffic from the Berlin scenario using the `cutRoutes.py` script[1]. The simulation duration was reduced to 12 hours including the morning peak only. In total, this scenario contains 67 000 vehicle trips, with 900 vehicles being present simultaneously at maximum (see Fig. 6).



(a) Berlin

(b) Charlottenburg

Figure 6: Traffic volumes modelled for each simulation scenarios.

Each simulation scenario was configured to run with MOSAIC in order to examine the different spatial index implementations. The perception module in MOSAIC can only be used by applications mapped to individual vehicles. We therefore configured a mapping to deploy a simple application that solely requests the surrounding vehicles within its field of vision in every simulation step (= every second) (see Listing 2). The field of vision was configured with a 200 m range and an opening angle of 60°.

Listing 2: Mapping configuration to deploy a perception module on 10 % of the vehicles.

```
{
  "prototypes": [ {
      "name": "DefaultVehicle",
      "applications":[
        "org.eclipse.mosaic.app.tutorial.vehicle.PerceptionApp"
      ],
      "weight": 0.10
  } ]
}
```

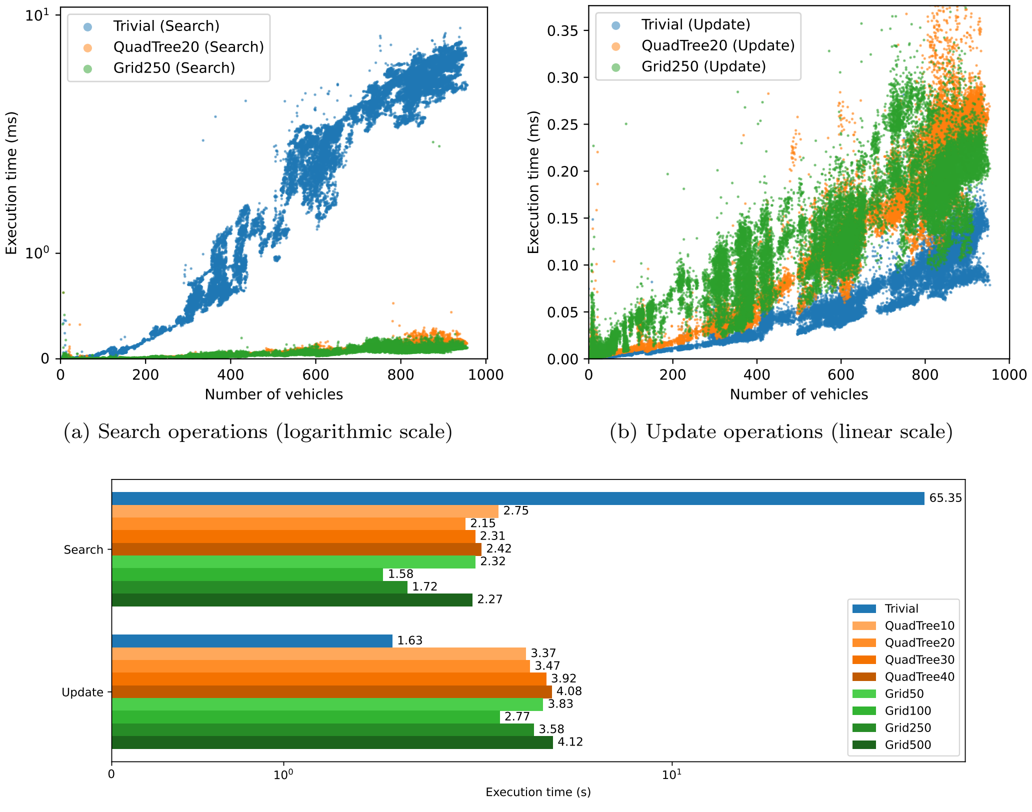
[1] This script is delivered with the SUMO installation [9].

4.1 Performance of Spatial Indexes

For the performance evaluation of the different index implementations, we defined the following configurations to compare against each other:

- **Trivial** - No spatial index. A view check is done for all existing vehicles in a for-loop.
- **QuadTreeXX** - Quad-tree implementation with a configuration of a maximum depth of 12, and a maximum number of vehicles per leaf/tile of [10, 20, 30, or 40].
- **GridYYY** - Grid-based index implementation having different cell sizes. The cell size is either [50, 100, 250, or 500] m in length and height.

For these experiments, 10% of the vehicles are equipped with the perception module. Each equipped vehicle requests the vehicles in its field of vision in every simulation step. During simulation, we measure the duration of every call of the atomic operations *update*, *search*, and *remove*. We furthermore store the current number of vehicles present in the simulation for each measurement, in order to find a relation between duration and index size.



(a) Search operations (logarithmic scale)

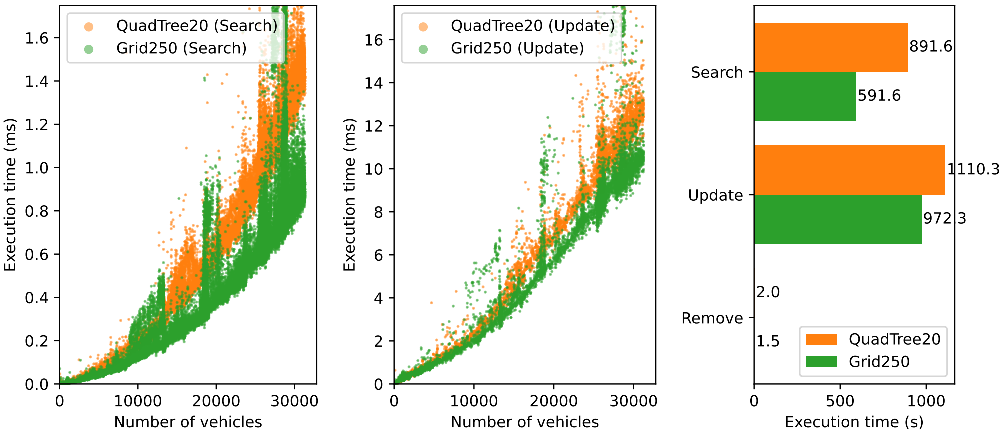
(b) Update operations (linear scale)

(c) Accumulated durations (logarithmic scale)

Figure 7: Individual and accumulated durations of search or update operations on each spatial index implementation in the Charlottenburg scenario.

For a first comparison, we run the Charlottenburg scenario with the nine different index configurations. Results can be found in Figure 7. Not surprisingly, the search queries require a lot of time in the trivial approach compared to Grid or Quad-tree. Even in situations with a low number of vehicles (e.g. less than 200), a spatial index is already much faster. In general, Grid and Quad-tree show very similar performance results. Search and update performance of the Grid is slightly better than the Quad-tree. Having a look at the individual configurations show, that both, lower tile capacity in the Quad-tree and very small cell sizes in the Grid, have disadvantageous effects on the performance. Best results can be achieved by using a Quad-tree with a maximum capacity of 20 or 30 vehicles per tile, or a Grid with a medium-large cell size of 100 or 250 m. Choosing a cell size also most likely depends on the viewing range, as it affects the number of cells in the Grid to be initially chosen for the range check. However, we did not investigate the effects of the viewing range on the performance of the Grid. In comparison to the total simulation time of 240 seconds, the overhead of 4–7 seconds produced by the spatial index is minimal and does not require further examination.

In a second series of experiments, we configured the Berlin scenario with the Quad-tree and Grid implementation, and again an equipment rate of 10%. Since this massive scenario already requires several hours to complete, we skipped the Trivial approach. Figure 8 shows that the Grid can outperform the Quad-tree in large-scale scenarios. Search operations with the Grid are 30% faster, update operations show 10% better performance. The total overhead of the index for search and update operations compared to a total simulation time of roughly nine hours is found to be at 5% for the Grid, and 6% for the Quad-tree.



(a) Search operations

(b) Update operations

(c) Accumulated durations

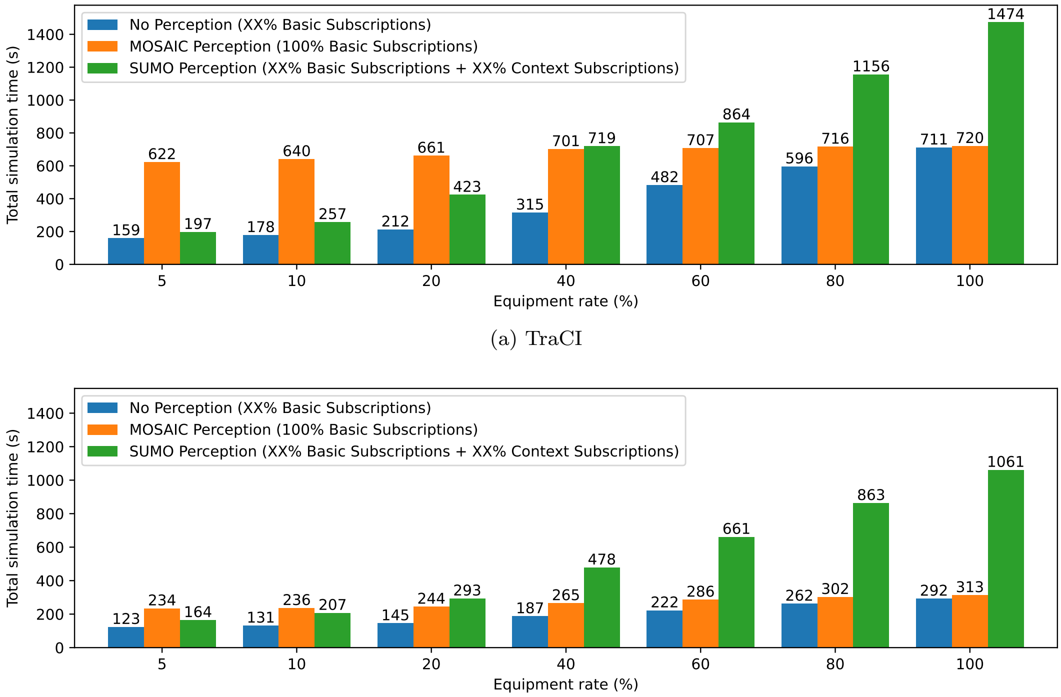
Figure 8: Durations of update and search operations in the Berlin scenario.

4.2 Performance of Coupling with SUMO

The following experiments help to understand the overhead that comes with the coupling of the traffic simulation of SUMO. This comparison includes two aspects. Firstly, the two different coupling interfaces using *TraCI* or *Libsumo* are compared with each other. Secondly, our implementation of the perception module is compared with the built-in features provided by the interfaces of SUMO (i.e. context subscriptions), as explained in Section 3.2. We, therefore, define the following simulation runs, which will furthermore be executed using the two different coupling methods:

- **No Perception -** As a baseline, basic vehicle data is subscribed for [5, 10, 20, 40, 60, 80, or 100]% of the vehicles, which are equipped with any application. No perception index is used at all. This allows measuring the total overhead each of the subsequent solutions requires.
- **MOSAIC Perception -** Basic vehicle data is subscribed for *all* vehicles in the simulation. [5, 10, 20, 40, 60, 80, or 100]% of the vehicles are equipped with a perception module using a QuadTree20 as a reference for a spatial index implementation.
- **SUMO Context Subscriptions -** Basic vehicle data is subscribed *only* for equipped vehicles. Additionally, context subscriptions are used for equipped vehicles to retrieve vehicles within the field of vision. The perception module in MOSAIC uses the results of these context subscriptions rather than building its own spatial index. The equipment rate is either [5, 10, 20, 40, 60, 80, or 100]%.

The following experiments include simulations of the Charlottenburg scenario. The equipment rate is varied from 5, 10, 20, 40, 60, 80, to 100%. As a coupling method, *TraCI* is compared to *Libsumo* to find the performance improvement this new coupling method comes



(a) TraCI

(b) Libsumo

Figure 9: Total simulation duration using *TraCI* and *Libsumo* coupling under different equipment rates in the Charlottenburg scenario.

with. Figure 9 shows, that with an increasing equipment rate, the simulation time of all variants also increases. The solution which uses the spatial data structures in MOSAIC (i.e. MOSAIC Perception) exhibits a certain time overhead at all equipment rates. The main reason here is the required subscriptions of all vehicles in the simulation, which need to be transferred to MOSAIC, in case (a) *TraCI* via TCP. Using SUMOs context subscriptions to obtain surrounding vehicles (i.e. SUMO Perception) shows a large benefit at lower equipment rates, but inferior scalability compared to MOSAIC Perception at higher equipment rates. Eventually, in the 100% case the SUMO Perception variant requires twice the time as running SUMO without any context subscriptions, and almost ten times as long as the 5% case.

Using the new *Libsumo* coupling interface on the other hand can improve simulation time significantly (see case (b) in Fig. 9). The simulation time of the MOSAIC Perception case can be reduced almost by a factor of three. Again, the difference between a low and high equipment rate is minimal in the MOSAIC Perception case and only depicts the overhead produced due to application handling and utilizing the spatial index. Comparing with the SUMO Perception variant, it is already enough to deploy 20% of the vehicles to find better performance with the MOSAIC Perception. Still, the SUMO Perception case increases simulation time with the equipment rates.

4.3 Evaluation Summary

In various conducted experiments, we investigated two aspects of the perception module implementation in MOSAIC. For one thing, a detailed examination of the index implementations Grid and Quad-tree and their configuration options showed, that both variants work well without producing much computational overhead. The Grid performs slightly better, especially in large scenarios with several thousands of vehicles being simulated in parallel. Here we found, that middle-sized cells of 100 to 250 m work best. The ideal cell size, however, is expected to depend on the view range and angle as this affects the number of cells being selected during search. Nevertheless, the Quad-tree implementation still shows good performance even with a large amount of moving objects in the index, for both update and search operations.

Another aspect is the interaction with SUMO. The amount of data captured from SUMO during simulation has an immense influence on the overall performance of the simulation, especially since the perception module implementation in MOSAIC requires the positions of all existing vehicles. Here we found, that using *TraCI* results in large overhead, which can mostly be eliminated by using the new *Libsumo* interface.

However, we also found that the built-in feature of SUMO, which is able to obtain surrounding vehicles as well, shows decreased performance on higher equipment rates. A deeper look into the actual MOSAIC simulation using a profiler revealed, that most of the additional time comes not only with the additional results from context subscriptions to read. The simulation call itself, using `Simulation.simulateUntil()` requires more than twice as much time as soon as context subscriptions are added for 40% of the vehicles. Additional time is then spent read-

Experiment	**Total**	**Simulate Step**	**Read single subscriptions**	**Read context subscriptions**	**MOSAIC overhead**
40% Single subscriptions	187 s	107 s	58 s	0 s	22 s
40% Single subscriptions + 40% Context Subscriptions	478 s (+ 156%)	259 s (+ 142%)	60 s (+ 3%)	133 s (+∞%)	26 s (+ 18%)

Table 1: Increase of simulation duration due to use of context subscriptions in *Libsumo*.

ing the context subscription results using `Vehicle.getAllContextSubscriptionResults()`. Detailed results of this brief analysis can be found in Table 1.

5 Conclusion and Outlook

Extracting object information from raw sensor data (e.g. Camera) and feeding them into smart mobility applications can help to improve safety and traffic efficiency. Co-simulation frameworks such as Eclipse MOSAIC may be used to develop and test such applications. In order to enable object detection on an application level, the integrated models need to provide information about surrounding objects. Such objects can be stationary (e.g. traffic signs, traffic signals), or moving (e.g. vehicles, pedestrians). Obtaining moving objects within a field of view of a stationary or other moving traffic participant requires a fast method to prevent low-performing simulations.

In this paper, we solved this problem by introducing a perception module to the Application Simulator of MOSAIC. For finding moving objects within a given range (e.g. inside the field of view of another vehicle), we implemented and evaluated different spatial index implementations based on a grid or Quad-tree. We evaluated the implementation with large-scale simulation scenarios in combination with the traffic simulation of Eclipse SUMO, with a view to the computational performance. The perception method using a grid to find surrounding objects worked best, with varying performance depending on the size of the included cells. Using a Quad-tree instead resulted in slightly but not significantly slower simulations. The modeling of perception in MOSAIC required reading basic data (e.g. position) for all moving objects (here vehicles) from SUMO during the simulation. This led to a bottleneck when the socket interface *TraCI* was still used to exchange data. In the course of this paper, we improved the integration of SUMO by implementing the *Libsumo* interface. Evaluations showed, that with *Libsumo* the simulation time could be improved by a factor of three in the presented scenarios. Only with the use of *Libsumo* feasible simulation times are possible in combination with our perception module implementation.

In following work, we want to use object perception for improving traffic efficiency. Here we believe that object detection through cameras in vehicles could be used to pre-process and pre-estimate traffic states on road segments. Such pre-estimated data could be shared and used to improve overall traffic state estimation, which is currently only achieved by collecting floating-car-data from a large fleet of vehicles.

In the current implementation of the perception module only moving objects have been considered. This work will be published with the next release of Eclipse MOSAIC and is already integrated into the source code repository in [1]. Future work will also allow the perception of stationary objects, such as traffic signs and signals. Furthermore, additional models which filter the set of surrounding objects due to occlusion are subject to ongoing work. The traffic scenario used in the evaluation section of this paper is planned to be published under an open-source license on GitHub soon [3].

Acknowledgment

This work is part of the “eBaseCamp” project (funding number 01ME19002C), which is funded by the Federal Ministry for Economic Affairs and Climate Action (BMWK), Germany. We furthermore want to thank Fabian M. Thiele and Trung Han Nguyen for their implementation work on the spatial index implementations.

References

[1] MOSAIC contributors. Eclipse MOSAIC Source Code. Eclipse Foundation on Github, [online], 2020-2022. https://github.com/eclipse/mosaic.

[2] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[3] Eclipse MOSAIC Core Team. Berlin SUMO Traffic (BeST) Scenario. https://github.com/mosaic-addons/best-scenario, 2022.

[4] Eclipse MOSAIC Core Team. Eclipse MOSAIC: A Multi-Domain and Multi-Scale Simulation Framework for Connected and Automated Mobility. https://eclipse.org/mosaic, 2022.

[5] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57, 1984.

[6] Maike Hartstern, Viktor Rack, Mohsen Kaboli, and Wilhelm Stork. Simulation-based evaluation of automotive sensor setups for environmental perception in early development stages. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 858–864. IEEE, 2020.

[7] Ravi Kanth V Kothuri, Siva Ravada, and Daniel Abugov. Quadtree and r-tree indexes in oracle spatial: a comparison using gis data. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 546–557, 2002.

[8] Clemens Linnhoff, Philipp Rosenberger, Simon Schmidt, Lukas Elster, Rainer Stark, and Hermann Winner. Towards serious perception sensor simulation for safety validation of automated driving-a collaborative method to specify sensor models. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 2688–2695. IEEE, 2021.

[9] 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 *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[10] Kay Massow, Fabian Maximilian Thiele, K Schrab, BS Bunk, I Tschinibaew, and Ilja Radusch. Scenario definition for prototyping cooperative advanced driver assistance systems. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE, 2020.

[11] Robert Protzmann, Björn Schünemann, and Ilja Radusch. Simulation of convergent networks for intelligent transport systems with vsimrti. *Networking Simulation for Intelligent Transportation Systems: High Mobile Wireless Nodes*, pages 1–28, 2017.

[12] Francisca Rosique, Pedro J Navarro, Carlos Fernández, and Antonio Padilla. A systematic review of perception system and simulators for autonomous vehicles research. *Sensors*, 19(3):648, 2019.

[13] Dominik Ziemke, Ihab Kaddoura, and Kai Nagel. The matsim open berlin scenario: A multimodal agent-based transport simulation scenario based on synthetic demand modeling and open data. *Procedia Computer Science*, 151:870–877, 2019. The 10th International Conference on Ambient Systems, Networks and Technologies (ANT 2019) / The 2nd International Conference on Emerging Data and Industry 4.0 (EDI40 2019) / Affiliated Workshops.

SUMO User Conference 2022
https://doi.org/10.52825/scp.v3i.109

Published: 29 Sept. 2022

Building a real-world traffic micro-simulation scenario from scratch with SUMO

Maria Laura Clemente[1][https://orcid.org/0000-0002-5952-2810]
[1] CRS4, Italy
clem@crs4.it

Abstract

Simulation of Urban Mobility (SUMO) is a powerful traffic simulation program which can work at different scales, from sub-microscopic to macroscopic. Depending on the available input dataset, it is possible to build lots of different configurations, changing routing and car-follow algorithms, and many parameters. Building a basic SUMO scenario is a multi-step activity involving the followings: preparing the transportation network, traffic definition, setting a routing algorithm, and running the simulation. The aim of the present work is to show a detailed real case study explaining how to build a complete scenario and run simulations, starting from the preparation of the network from Open Street Map. The last part of the present paper is about how to use the SUMO output files with MongoDB in order to keep track of significant information resulting from each simulation.

1 Introduction

Transport planning is based on three pillars: demand analysis (the number of movements performed by people, either walking or using a vehicle), offer analysis (the transportation network and public transport services), and the traffic assignment with the interaction models to get a configuration of equilibrium (or a sequence of equilibrium, in the dynamic models) between the demand and the offer. In this big picture, it's understandable that a reliable demand computation is the basis for a good simulation; for this purpose, two approaches have commonly been used: the more traditional one is the trip-based, also called *four-step model*, which puts the trips of each vehicle at the core; the second one, more recent, is called *activity-based* because it starts from the analysis of activities performed by the users to understand how they will move during the day (McNally, 2000; McNally et al. 2007). Hybrid approaches are possible as well.

Many platforms for traffic simulation are available and they can be grouped by main features: commercial versus free and open-source, scale of simulation (if performing macro meso or micro simulation), routing algorithms (if trip based or activity based), if altimetry is supported, possibility of online interaction, documentation availability, etc.

An updated and comprehensive comparison of all the software available for traffic simulation is provided in (Ullah et al., 2021). The most popular commercial tools for macro-simulation are Cube Voyager, Visum and OmniTRANS, while VISSIM is very common for the micro-simulations. Talking about free and open-source solutions for micro-simulation, Matsim (https://www.matsim.org/) and SUMO (https://sumo.dlr.de) are the most widely used.

The present activity was carried out with SUMO (Alvarez Lopez et al., 2018) for many reasons. First, it is free and open source (under the Eclipse Public License EPL v.2), is highly portable, and is continuously improved by the German Aerospace Center and a huge community of users. Another important feature is that it allows to manage a big area micro-simulation, in fact it has been used for whole cities (Maiorov et al., 2019; Bachechi et al., 2019): Bologna, Brunswick, Dublin, Ingolstadt, Luxembourg, Monaco, Stuttgart, Turin, Cologne (https://sumo.dlr.de/docs/Data/Scenarios.html).

A simulation built with SUMO is continuous in space (vehicles can be in any position on the street) and discrete over time (the mobility model uses uniform time steps), supports multi-modality, is highly configurable, and extendible (it's also possible to add new algorithms). Depending on the traffic dataset, SUMO allows to build different types of micro-simulations, from the classic Origin Destination Matrices to data from sensors (Po et al. 2019).

SUMO provides command line tools specific to each of the steps required to build a whole simulation. A first confusion can arise from the name, because sumo is the name of the whole platform but also the name of the last step, which is the one to run the simulation; there is also the command sumo-gui which differs in that it also shows the graphical visualization of the running simulation. The sumo platform has a solid and efficient base of algorithms written in C++, but it includes also a great number of useful Python scripts to facilitate many operations, working as a sort of wrapper over the C++ core.

All the files required in input to a simulation are in xml format. So, all the procedures related to the input file preparation have the target of translating the available information into a set of input xml files suitable to be understood by the SUMO world.

The present paper describes one of the many possible ways to build a sumo micro-simulation, referring to a real scenario, the corridor SS195 to Cagliari (Italy), for which some Origin Destination Matrices were available; the available input dataset drove the choice toward a classical 4-step model. So, this paper shall not be considered as a complete tutorial of this powerful program, for which the official website is the best reference (http://sumo.dlr.de/userdoc/), but it shows a detailed roadmap of the steps required to get a real-world simulation up and running, enriched with practical tips to overcome tricky obstacles along the way, learned by experience.

Having said that, the following set of files makes up what is usually called a 'scenario':

- Information about the transportation network (in short, net),
- Information about the Traffic Assignment Zones (TAZ),
- Information about the Origin and Destination Matrices (ODM).

These compound the smallest set of input to run a real-world simulation, which could be extended with many further information about: buildings, types of vehicles, traffic lights, Public Transport Services and bus stops, and so on.

A SUMO simulation can have many different configurations, based on a great number of parameters, some of which are particularly important, such as:

- *route-choice-method (*input of the dynamic User Assignment for route building, default is Gawron*),* further information at https://sumo.dlr.de/docs/Demand/Dynamic_User_Assignment.html
- *routing-algorithm*, the famous Dijkstra is set by default, but other possibilities are alpha star (astar), Contraction Hierarchies (CH), and CHWrapper, as explained at https://sumo.dlr.de/docs/Simulation/Routing.html

- car-following-models (input of sumo/sumo-gui), explained in detail at https://sumo.dlr.de/docs/Car-Following-Models.html

Providing details about the choice of these and the many other parameters is out of the scope of the present paper; it's important to know that they are the core of a traffic micro-simulation and each of them involves further parameters that must be tuned during a careful process of model calibration (Krajzewicz et al., 2002; Aminia et al., 2019), when a sort of ground truth dataset is available. The official SUMO documentation gives many explanations about them. Being an open source, an interesting and evolving research activity consists in further analysis of custom-developed algorithms. In this presentation we limit to provide an example of sequence of command lines applied to the real-world case study. To help beginners, SUMO provides default values for all these parameters.

This paper doesn't cover the interactive "Traffic Control Interface" (TraCI), provided by SUMO software to allow accessing to a running traffic simulation in order to retrieve values of simulated objects and change their behavior "on-line".

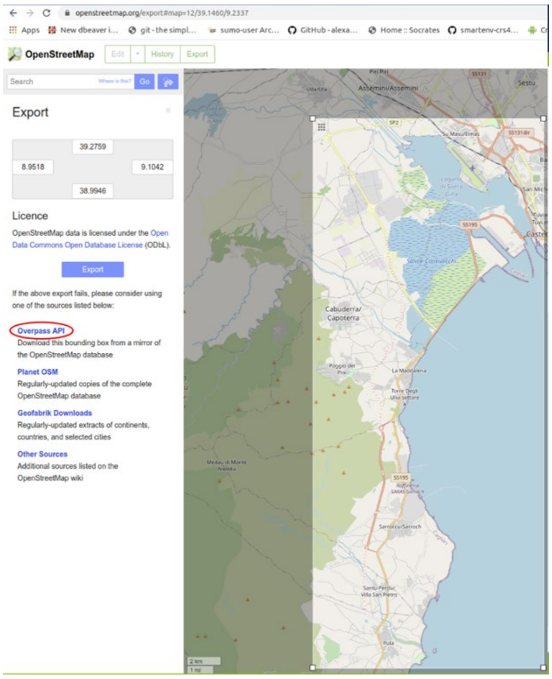
The last part of the present paper proposes a way to model the output dataset of a simulation using the Mongo Atlas cloud database in order to allow a quick and effective comparison of the results coming from all the simulations related to the same scenario.

2 Network preparation

A traffic micro-simulation starts with the choice of the boundaries of the study area. Working on a road, like the SS195, which collects traffic flows from a big area and brings it to Cagliari, as the most attractive point of that area, cannot be limited only to the road, but needs a careful analysis of the whole area involved; in other words, here it is particularly important to analyze a reasonable extension of the 'catchment area'. On the other hand, it is recommended to avoid exaggerating in this operation because the bigger is the area, the more time consuming will result, as side effect, working on it.

2.1 Downloading the map from Open Street Map

Once the proper area has been carefully chosen, the next step consists in downloading it from Open Street Map (openstreetmap.org), in short OSM.



Export

39.2759
8.9518
9.1042
38.9946

Licence

OpenStreetMap data is licensed under the Open Data Commons Open Database License (ODbL).

Export

If the above export fails, please consider using one of the sources listed below:

Overpass API
Download this bounding box from a mirror of the OpenStreetMap database

Planet OSM
Regularly-updated copies of the complete OpenStreetMap database

Geofabrik Downloads
Regularly-updated extracts of continents, countries, and selected cities

Other Sources
Additional sources listed on the OpenStreetMap wiki

Figure 1: Downloading the map from OSM

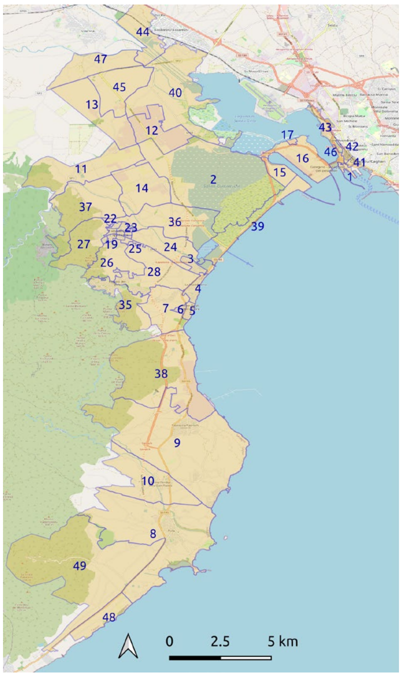


Figure 2: The Traffic Assignment Zones

Figure 1 shows the area required for the SS195 road. But when the area is as big as this one, OpenStreetMap API answers to the request with the following error message*: "You requested too many nodes (limit is 50000). Either request a smaller area or use planet.osm"*. A work around to this problem is selecting the link 'overpass API' (shown with a red contour in Figure 1) which allows to download the selected bounding box from a mirror of the OpenStreetMap database and in fact the downloading will immediately start.

3 Adding Altimetry to the OSM map with osmosis

Unfortunately, OSM does not carry elevation data. There are some ways to add them (https://sumo.dlr.de/docs/Networks/Elevation.html); for the case study, altitudes have been downloaded from Shuttle Radar Topography Mission (SRTM); in particular, for Cagliari the altimetry data are in the file N39E009.hgt.zip which includes points from N39 to N40 and from E009 to E010.

Once this file is available, *osmosis* must be installed (*sudo apt-get install osmosis*). First of all, it's better to check the validation of the OSM map file in *osmosis*, by reading it (the flag *–write-null* avoids any output building):

```
osmosis --read-xml SS195_map.osm.xml --write-null
```

Then if everything goes well, it's possible to run *osmosis* in order to have the altitudes in the osm map file, giving the local directory of the SRTM file, and specifying to set the tag element *ele* for the altitude, which is the trick to have the file imported in SUMO (through the command *netconvert*, as explained later):

```
./osmosis --read-xml SS195_map.osm.xml --write-srtm locDir=~/SRTM/ tagName=ele --write-xml
SS195_map_ele.osm.xml
```

With this command, the output file *SS195_map.osm.xml* will have each node element enriched with the extra tag about elevation, as shown in Figure 3.

```
<node id="31483414" version="1" timestamp="2019-07-12T09:51:28Z" changeset="1"     lat="39.2842393"
lon="9.0286643">
     <tag k="ele" v="6.808519999996889"/>
</node>
```

Figure 3: The detail of a node in the OSM file with the elevation extra tag.

4 SUMO

Once the OSM map is ready, it’s time to start using SUMO and its command lines (here after, cml) and Python scripts. In fact, as already explained, SUMO is made of tools designed to be used independently from each other. SUMO installation is explained from the official site web https://sumo.dlr.de/docs/Installing/index.html.

The presented simulation has been carried out with sumo installed from source code (https://github.com/eclipse/sumo) on ubuntu version 20.04, in order to be able to keep it frequently updated.

To have SUMO command lines available, it’s useful to set up the environment variable SUMO_HOME in the .bash file:

```
export SUMO_HOME=/mypathtosumo/Sources/sumo_sourcecode/
```

Each command could also take information (input files and parameters) from a xml configuration file, which maybe would reduce error-proneness; on the other hand, for the case study, all the values for each parameter were defined inside the same shell script and all the command lines put in sequence inside nested loops in order to run all the possible combinations of parameters. This structure allows to run many configurations one after the other and to manage all the parameters involved in all the different configurations from just one file. Each command line used will be described in the following paragraphs.

4.1 Netconvert: converting the OSM map into the xml format input for SUMO

The OSM map file, an xml format, needs to be translated into the SUMO xml format (they are compliant to two different xml schemas). To do this, the SUMO command line required is *netconvert*.

An important flag to add when performing this transformation is *–no-turnarounds*. Without this flag, the vehicles moving in the net, arriving at the end of a road, at the boundary of the map, will make a U conversion, going back along the lane in the opposite direction. The *netconvert* command will be:

```
netconvert -v –osm.elevation true --osm-files SS195_map_ele.osm --no-turnarounds -o
SS195_osm_ele.net.xml
```

The network file obtained as output running this command is called *SS195_osm_ele.net.xml* and can be used as network for the rest of operations with SUMO. Optionally, in order to get an improved model for the visibility of the roundabouts, two further steps can be done. The first one splits the parts of the network into different files:

```
netconvert -s SS195_osm_ele.net.xml –plain-output-prefix SS195
```

This command produces in output: *SS195.edg.xml* (file of only edges), *SS195.nod.xml* (files of only nodes), *SS195.con.xml* (file of only connections), *SS195.tll.xml* (file of the traffic lights), *SS195.typ.xml* (file of the types). Now, calling again the *netconvert* command, giving as input all these files, produces a better network file in output (as anticipated):

```
netconvert -e SS195.edg.xml -n SS195.nod.xml -x SS195.con.xml -i SS195.tll.xml -t SS195.typ.xml -o
SS195.net.xml
```

The output *SS195.net.xml* is the definitive network file which will be used from now on.

In order to add some environment elements to the bare map, there is another command line provided by SUMO, which is *polyconvert*. It requires in input three files: the OSM map file *SS195_map.osm* (as downloaded from OSM), the output file of *netconvert*, which is *SS195_osm.net.xml*, and also a file that is available in the SUMO code, *osmPolyconvert.typ.xml*, that is necessary to define the types of environments. The complete command line will be:

```
polyconvert --net-file SS195.net.xml --osm-files SS195_map.osm --type-file
$SUMO_HOMEdata/typemap/osmPolyconvert.typ.xml -o SS195.poly.xml
```

It must be said that there is also a quicker way to perform a starting scenario, using the SUMO tool *osmWebWizard.py* which allows to select the study area. In this case *sumo-gui* will import the net from OSM to run a simulation with randomly generated traffic; this approach can be very useful to quickly build all the required set of files, that can be corrected and integrated afterward.

In any case, the map generated must be carefully checked and corrected through another important tool of SUMO, called *netedit*. Typical mistakes are extra lanes/crossings, missing accelerating/decelerating lanes, etc.; *netedit* allows to correct all of them, by hand and one by one.

Another useful tool to make things easier is *sumopy*, a free open-source python library (https://github.com/schwoz/sumopy) gathering all the steps required to build a sumo simulation in a very comprehensive Graphic User Interface (Schweizer, 2013). *Sumopy* was used to build the real-world scenario of the city of Bologna (Schweizer et al., 2021).

4.2 Polyconvert: building the Traffic Assignment Zones (TAZs)

When input traffic data about the demand is available, they define an Origin/Destination matrix, in terms of number of vehicles moving from each TAZ to the others; to add this information to the scenario, a shapefile of the zones must be defined through a GIS editor, like QGIS (https://www.qgis.org). A good starting point is the official shapefile of the census regions, from where it is possible to group regions into sensible TAZs. Doing this operation, it’s possible that a geographical conversion is required to have all the map layers properly matching in QGIS. Typical systems are *EPSG 3003, 4326,* and *32632*. When a shapefile doesn’t show this information, the web service *epsg.io* comes in help, revealing the current coordinates system. Knowing the exact coordinates system allows to make the proper transformations with QGIS, ending with all the layers matching.

The 49 Traffic Assignment Zones of the real case study are shown in Figure 2. The shapefile of the TAZs built with QGIS must be translated into the format required by SUMO, using also the network file *SS195_osm.net.xml* as input. The conversion of this file to the proper format requires to call again the *polyconvert* cml, this time using other arguments, and specifying the name used for the TAZ column:

```
polyconvert -v --shapefile-prefixes TAZs --shapefile.guess-projection true --shapefile.id-column TAZ -n SS195.net.xml -o TAZs.taz.xml
```

It's important to note, that in order to work properly, each TAZ zone in the XML file, in addition to the boundary definition, must also have at least one edge of the network declared as TAZ source, and another as TAZ sink. As suggested by the name, the TAZ sources are all the edges from where the trip can start, while the TAZ sinks are all the edges in the Zone where the trips can end. This information should be added carefully by hand, directly in the SUMO xml file, which is the output of the *polyconvert* command line. In fact, although there is the python script *edgesInDistricts.py* (one of the available SUMO tools) performing this operation automatically, it adds all the edges in the TAZ as sources and as origins, with no distinctions. For the case study, a textual file with a list of the tazSources and tazSinks for each TAZ were prepared to be added running a custom Python script. An example of the structure of a TAZ xml file is shown in Figure 4.

```
<tazs>
   <taz id = “48” shape=”2995.4312,1572.2758, …” >
     …
     <tazSource id=”2940” weight=”0.07”/>
     <tazSource id=”2946” weight=”0.07”/>
      …
     <tazSink id=”2940” weight=”0.07”/>
     <tazSink id=”2946” weight=”0.07”/>
      …
   </taz>
   …
</tazs>
```

Figure 4: The TAZ file XML format

4.3 Od2trips: importing the traffic demand from OD matrices into individual trips

Figure 5 shows a very basic vehicle definition used for the case study, considering only cars (in SUMO called '*passenger'*) and *trucks* (generic name for heavy vehicles). Lots of more details could be added to specify a long list of vehicles and related features, even pedestrians.

```
<additional>
   <vType id=“passenger” vClass=“passenger” length=”4.60” color=”1,1,1”
deepartSpeed=”max” />
   <vType id=“truck” vClass=“truck” length=”10.22” color=”0,1,1” deepartSpeed=”max” />
</additional>
```

Figure 5: A basic XML file for vehicle definition

The Sumo *od2trips* cml imports the traffic demand, as a list of origin, destination, and number of vehicles, into an xml output file made of trips; each trip is defined by an *id* with starting and ending time (included inside the given time-lapse); this command must be run once for each transport mode and for each time-lapse and produces as output an xml file for the input transport mode and time-lapse. The TAZs file is also required as input for this operation. The following is the command line used in the case study to obtain the xml file of the trips of cars inside the 7-8 a.m. time-lapse:

```
od2trips -v --taz-files TAZs.taz.xml --vtype passenger --prefix car --od-matrix-files SS195_ODM_7_8_cars.od
-o $OUTPUT_DIR/SS195_ODM_7_8_cars.odtrips.xml
```

An OD input file in the *O-format*, is an example of suitable input for the *od2trip* command line:

- any line starting with an asterisk is a comment which is ignored from *od2trip*,
- the first line is an identification code of the format of the file ($OR;D2),
- the second significant line defines the time-lapse,
- the third line is a scaling factor for the number of vehicles,
- from the fourth line to the end of the file, the format is:
  - id of the Origin TAZ,
  - id of the Destination TAZ,
  - number of vehicles moving during the specific time-lapse from the Origin to the Destination TAZ.

A sample of the output of the *od2trip* cml is shown in Figure 6.

```
<routes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://sumo.dlr.de/xsd/routes_file.xsd">
        <trip id="61" depart="28800.79" from="0" to="118" fromTaz="1" toTaz="44" departLane="free"
departSpeed="max"/>
        <trip id="7048" depart="28801.27" from="4489" to="118" fromTaz="7" toTaz="44" departLane="free"
departSpeed="max"/>
        <trip id="2917" depart="28801.88" from="56" to="19" fromTaz="40" toTaz="13" departLane="free"
departSpeed="max"/>
        <trip id="268" depart="28802.83" from="3017" to="2879" fromTaz="10" toTaz="8" departLane="free"
departSpeed="max"/>
```

Figure 6: A sample of the Origin-Destination trips file

4.4 Duarouter: from trips to routes

The files output of *od2trips*, one for each transport mode, are the input of the SUMO cml *duarouter*, which for each vehicle builds the complete route, in terms of sequence of edges along with the starting time; this file is the input of the simulation with the *sumo* command (or *sumo-gui*).
Calling *duarouter* an important input parameter is the *route-choice-method*, which can be *Gawron* (DUA, used by default), *logit*, or *lhose*.
An example of the duarouter command line used for the case study is the following:

```
duarouter -v -n SS195.net.xml --route-files cars.odtrips.xml, SS195_trucks.odtrips.xml
--additional-files vtypes.xml --xml-validation never --no-step-log true -o SS195_duarouter.odtrips.rou.xml
```

A sample of the output of *duarouter* is shown in Figure 7. As explained in 4.6, the duarouter cml can be used as first step of the iterative algorithm to approximate the Dynamic User Equilibrium (DUE).

```
<routes xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:noNamespaceSchemaLocation=http://sumo.dlr.de/xsd/routes_file.xsd>
         <vType id=”passenger” length=4.60” minGap=”2.00” maxSpeed=”30.00” speedFactor=”normc(1.00,0.00)”
vClass=”passenger” color=”white” carFollowModel=”IDM” accel=”1.0” decel=”1” tau=”1.4” />
         <vehicle id=”car8370” type=”passenger” depart=”25200.42” departLane=”free” departSpeed=”max”
fromTaz=”47” toTaz=”38”>
```

```
            <route edges=”47874#0 198047874#1 198047874#2 198047906#0 198047906#1 -373339455“ />
        </vehicle>
        <vehicle id=”car61” type=”passenger” depart=”25200.79” departLane=”free” departSpeed=”max”
fromTaz=”1” toTaz=”43”>
            <route edges=”27752726#0 27752726#1 116050065 11605029 27752728 666975680 11361107” />
        </vehicle>
        <vehicle id=”car7069” type=”passenger” depart=”25201.27” departLane=”free” departSpeed=”max”
fromTaz=”46” toTaz=”41”>
            <route edges=”138413006 26617614#0 26617614#1 26617616 26617617 113611070#0” />
        </vehicle>
```

Figure 7: A sample of the *duarouter* output

4.5 Sumo cml: running the micro-simulation

The command line *sumo* runs the simulation without the graphical user interface. To better understand this core part, it’s useful to know that *sumo* performs a time-discrete simulation, where the default step (parameter *step-length*) is 1 second but can be reduced up to 1µs; the simulation model is space-continuous and the position of each vehicle at each simulation step is defined by the *id* of the lane and the distance from the beginning of that lane. By default, the speed of each vehicle is computed using an extension of the stochastic car-following model developed by Stefan Krauss (Krauß, 1998), which is faster and simpler than others. SUMO provides also several other algorithms for this task: Wiedemann 74 and 99 (the ones used by VISSIM), IDM (very popular), but particularly important is the *KraussPS* version because it uses road slopes in the computation, information obtained from the elevation value of the nodes in the network file.

One of the parameters for *sumo* is the *routing-algorithm*, which can be Dijkstra by default (Dijkstra, 1959), alpha star (in short, astar), CH, CHWrapper.

When a vehicle is stopped in the traffic jam for too much time (which is another configurable parameter), it is automatically moved from its position to another one. It’s possible to avoid this, by setting the *time-to-teleport* parameter to -1. The simulation allows to set a particular action to be taken in case of collisions, for instance, send a warning in standard output. The list of all additional files, such as information about the buildings, can be passed to the simulation through the *additional-files* flag.

Another important input is related to the time-lapse, for instance 7-8 A.M., because it must be converted into seconds of simulation:

- *start time 7 A.M. = 3600 s * 7 = 25200 s*
- *end time 8 A.M. = 25200 s + 3600 s = 28800 s*

To highlight the relevance of this aspect, it must be said that if start and end time of the simulation are not set accordingly to the ones specified in the Origin/Destination file, the simulation will not have any vehicle running in the network.

An example of the sumo command line is the following:

```
    sumo -v -n SS195.net.xml --route-files SS195.odtrips.rou.xml --duration-log.statistics true -b 25200 -e 28800
--time-to-teleport -1 --collision.action warn  --step-length 0.5 --routing-algorithm dijkstra --time-to-impatience -1
--additional-files SS195.poly.xml
```

4.6 dualterate.py: the iterative algorithm to approximate the Dynamic User Equilibrium (DUE)

After running *duarouter* the first time, its output (SS195_duarouter.odtrips.rou.xml, related to all the transportation modes, which in this example are cars and trucks) can be given as input to a python script, *dualterate.py*, which performs Gawron's algorithm (Gawron, 1998) by calling both commands, *duarouter* and *sumo* (already explained in previous paragraphs), iteratively in order to approximate the Dynamic User Equilibrium (DUE), where the number of iterations depends on the scenario. A scheme of this process is shown in Figure 8.

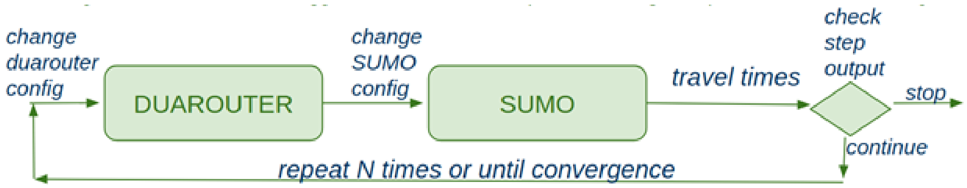


Figure 8: A scheme of how dualterate.py works.

An important thing to know is that when *dualterate.py* is run, the input parameters for the *sumo* command are distinguished from the ones for *duarouter* by the prefix *sumo–*.

A key parameter for *dualterate.py* is the *max-convergence-deviation*: at each iteration the standard deviation related to the average travel time is computed allowing to use convergence as threshold to stop the iterations.

An example of this command for the case study is reported here after:

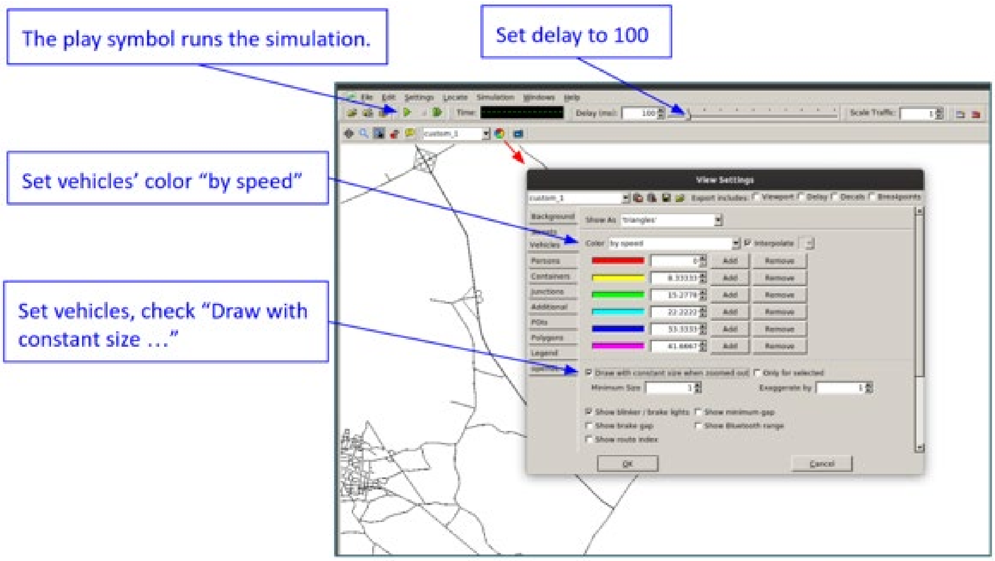
```
$SUMO_HOME/tools/assign/dualterate.py --router-verbose -n SS195.net.xml -D TAZs.taz.xml -r SS195.odtrips.rou.xml -l 50 -b 25200 -e 28800 --max-convergence-deviation 0.01 sumo--step-length 0.5 sumo--routing-algorithm dijkstra sumo--vehroute-output vehroute.xml
```

4.7 Running a simulation with sumo-gui

The command sumo-gui, as already mentioned, provides a graphical user interface (GUI) which allows to watch the vehicles moving in the network during a simulation. In the GUI it's possible to set the colours of vehicles and lanes according to various criteria. Effective choices are:

- to be able to watch the simulation, set the delay time to 100, otherwise it runs too fast
- to have an idea of the jammed edges, the colour of vehicles can be set according to speed (red when stopped, etc.)
- to set a constant size for the vehicles, which otherwise are too small.

Figure 9 shows how to achieve this configuration in the sumo GUI.



The play symbol runs the simulation.

Set delay to 100

Set vehicles' color "by speed"

Set vehicles, check "Draw with constant size …"

View Settings

Delay (ms): 100

Show As 'triangles'

Color by speed ☑ Interpolate

0
8.33333
15.2778
22.2222
33.3333
41.6667

☑ Draw with constant size when zoomed out □ Only for selected

Minimum Size 1 Exaggerate by 1

☑ Show blinker / brake lights □ Show minimum gap

□ Show brake gap □ Show Bluetooth range

□ Show route index

Figure 9: How to get a more readable simulation

During the simulation, while the vehicles are moving, many useful operations are possible. One of these consists in closing an edge, or only a lane, in order to analyse the effects on the traffic flow; another useful feature is following a vehicle in its route (for instance, to follow an emergency vehicle).
Figure 10 shows a particular of the running simulation.

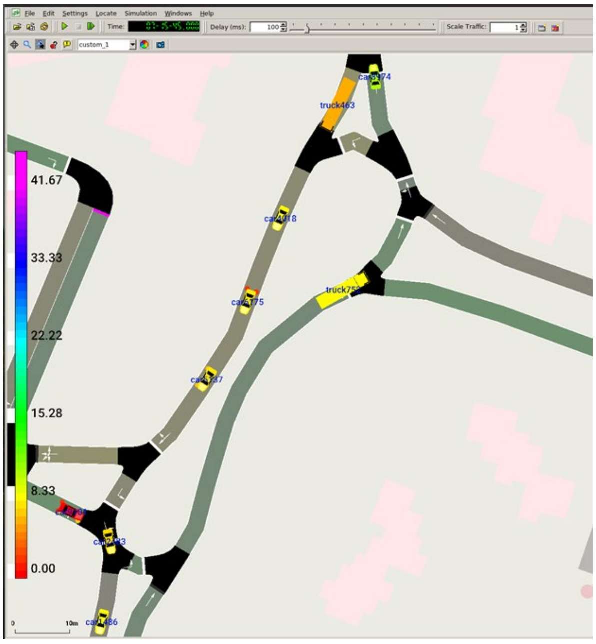


Figure 10: An image of the simulation with sumo-gui

4.8 The xml files output of a simulation

The user can choose how much information get as output of a simulation, which will be written after the simulation ends, by specifying some parameters in the input to the cml:

- *tripinfo-output*, to have detailed information about the trips, such as: departure speed, arrival speed, duration, waiting time, time-loss, etc.
- *vehroute-output*, to have detailed information for each vehicle: vehicle id, departure, route (a list of all the edges), arrival, fromTaz, toTaz, etc.
- *fcd-output*, to have the floating car data, i.e., the coordinates of each vehicle at each instant of the simulation (a very big data file)
- *summary*, to have some general information for each step of the simulation: number of vehicles running, mean speed, mean travel time, etc.

To sum up, the previous parameters can be given as input to run the simulation with the graphical user interface, through the following cml:

```
sumo-gui -v -n SS195.net.xml --route-files SS195.odtrips.rou.xml --duration-log.statistics true -b 25200 -e 28800 --time-to-teleport -1 --collision.action warn  --step-length 0.5 --routing-algorithm dijkstra --time-to-impatience -1  --additional-files SS195.poly.xml  --tripinfo-output SS195.tripinfo.out.xml  --vehroute-output SS195.vehrou.out.xml --summary SS195_summary.out.xml
```

SUMO provides many Python scripts to read these output files (in the directory sumo/tools/output/), which can also be used as a starting point to customize a solution accordingly to specific requirements. Particularly useful in this task is the Python library *sumolib*, included in the tools of SUMO.

Figure 11 shows a scheme which summarises the main 5 steps required to build a traffic micro-simulation with SUMO.

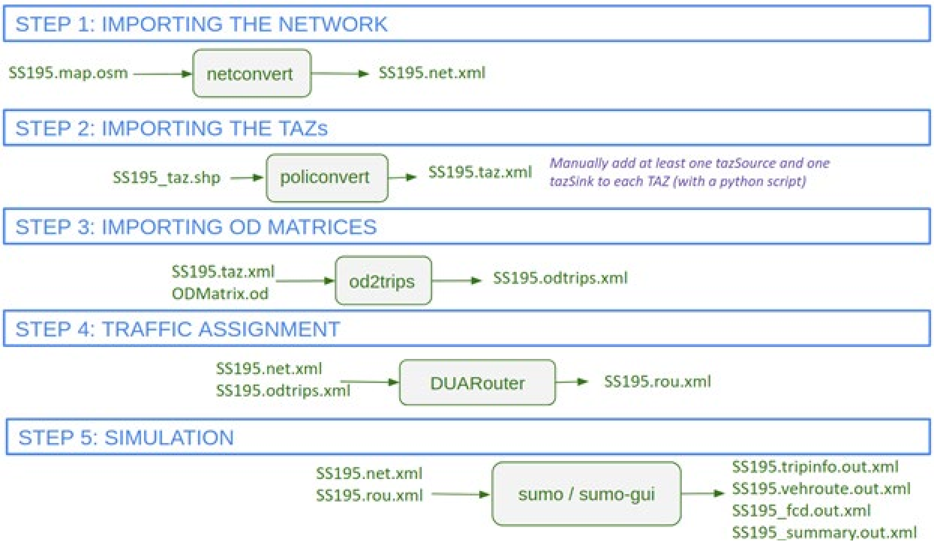


Figure 11: A scheme of the steps to build a SUMO micro-simulation.

5 A way to store information about simulations into MongoDB

A research activity of calibration, changing parameters many times can become difficult to be managed unless a way to record the important information about each run is put in place.

In fact, elaborating statistics and Key Performance Indices (travel time, number of vehicles, average speed, time-loss, etc.) is easier with the help of a database. Figure 12 shows a scheme of input files of a simulation, output files and its parsing through a custom python code to save the related information in a database.

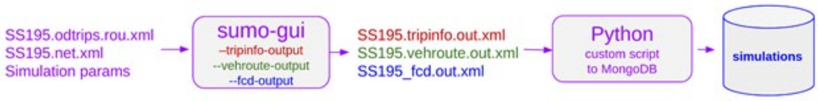
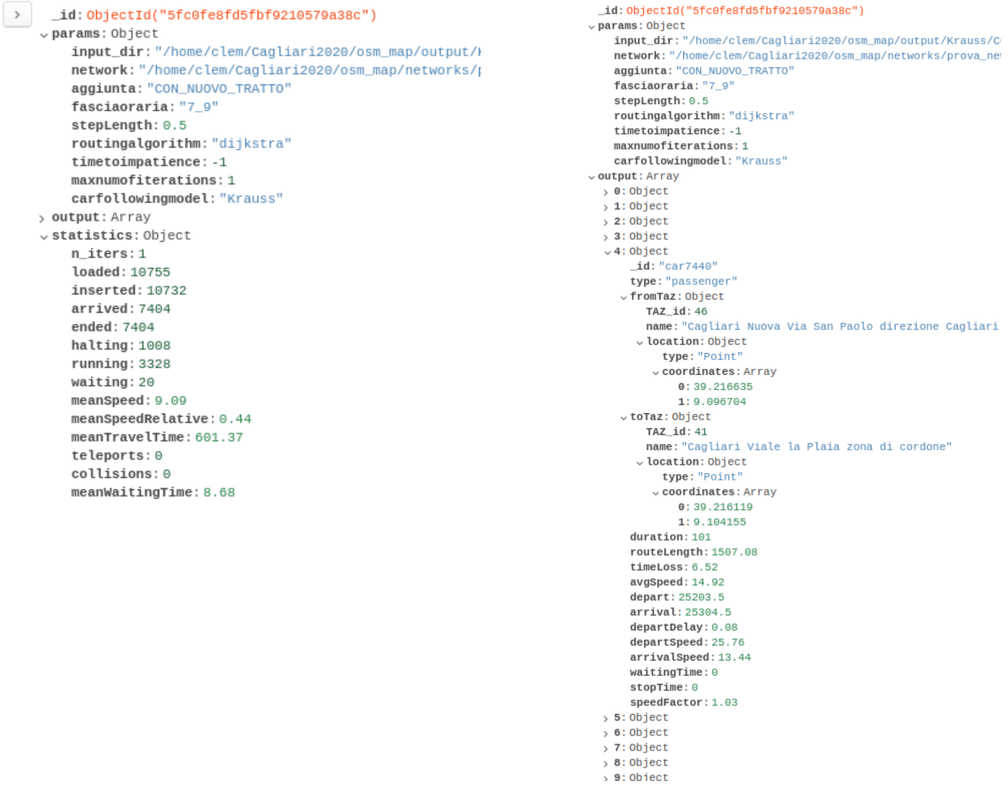


Figure 12: input and output files of a simulation

For the case study MongoDB was used, after a data modeling analysis tailored to the specific application case of simulation recording. A simple combination of the Python libraries *pymongo* and the sumo library *sumolib* allow to save locally or to Atlas (in cloud) a complete set of information about each simulation (as shown in Figure 13).



```
 _id: ObjectId("5fc0fe8fd5fbf9210579a38c")
⌄params: Object
   input_dir: "/home/clem/Cagliari2020/osm_map/output/
   network: "/home/clem/Cagliari2020/osm_map/networks/
   aggiunta: "CON_NUOVO_TRATTO"
   fasciaoraria: "7_9"
   stepLength: 0.5
   routingalgorithm: "dijkstra"
   timetoimpatience: -1
   maxnumofiterations: 1
   carfollowingmodel: "Krauss"
›output: Array
⌄statistics: Object
   n_iters: 1
   loaded: 10755
   inserted: 10732
   arrived: 7404
   ended: 7404
   halting: 1008
   running: 3328
   waiting: 20
   meanSpeed: 9.09
   meanSpeedRelative: 0.44
   meanTravelTime: 601.37
   teleports: 0
   collisions: 0
   meanWaitingTime: 8.68
```

```
 _id: ObjectId("5fc0fe8fd5fbf9210579a38c")
⌄params: Object
   input_dir: "/home/clem/Cagliari2020/osm_map/output/Krauss/C
   network: "/home/clem/Cagliari2020/osm_map/networks/prova_ne
   aggiunta: "CON_NUOVO_TRATTO"
   fasciaoraria: "7_9"
   stepLength: 0.5
   routingalgorithm: "dijkstra"
   timetoimpatience: -1
   maxnumofiterations: 1
   carfollowingmodel: "Krauss"
⌄output: Array
  › 0: Object
  › 1: Object
  › 2: Object
  › 3: Object
  ⌄ 4: Object
      _id: "car7440"
      type: "passenger"
     ⌄fromTaz: Object
        TAZ_id: 46
        name: "Cagliari Nuova Via San Paolo direzione Cagliari
       ⌄location: Object
          type: "Point"
         ⌄coordinates: Array
            0: 39.216635
            1: 9.096704
     ⌄toTaz: Object
        TAZ_id: 41
        name: "Cagliari Viale la Plaia zona di cordone"
       ⌄location: Object
          type: "Point"
         ⌄coordinates: Array
            0: 39.216119
            1: 9.104155
      duration: 101
      routeLength: 1507.08
      timeLoss: 6.52
      avgSpeed: 14.92
      depart: 25203.5
      arrival: 25304.5
      departDelay: 0.08
      departSpeed: 25.76
      arrivalSpeed: 13.44
      waitingTime: 0
      stopTime: 0
      speedFactor: 1.03
  › 5: Object
  › 6: Object
  › 7: Object
  › 8: Object
  › 9: Object
```

Figure 13: An example of simulation data in Atlas

The python code steps to be implemented are summarized here after:

- step 1: build a simulation dictionary, called *sim*, with all the input parameters;
- step 2: build a TAZ dictionary (reads TAZ names from a csv/txt file);
- step 3: build a dictionary with the statistics from *SS195.summary.out.xml* file and get average simulation performance indicators from last line in *summary.out.xml* file;
- step 4: build a list of information about each vehicle, called *all_vehicles_list*, getting information from output files *SS195_vehroute.out.xml* and *SS195_tripinfo.out.xml*, adding the names of the TAZs from the dictionary built in step 2;
- step 5: the previous steps collect all the data, while this last step stores all the data in the MongoDB Atlas cloud database (as shown in the python method in Figure 14).

```
# step 5: store all simulation info in a new collection of db
def store_in_mongodb(SUMO_DB_NAME, sim, all_vehicles_list, statistics):
    myclient =      pymongo.MongoClient(“mongodb+srv://root:PSSWD@cluster0.plnvt.mongodb.net/” +
SUMO_DB_NAME + “?retryWrites=true&w=majority”)
   dblist=myclient[SUMO_DB_NAME]
   mycollection=mydb[“simulations”]
   simulation = {‘params’:sim, ‘output’:all_vehicles_list, ‘statistics’:statistics}
   sim_id = mycollection.insert_one(simulation)
   print(‘sim_id = %s’ % sim_id.inserted_id)
   return sim_id.inserted_id
```

Figure 14: A python script to store the information about a simulation in the MongoDB Atlas cloud DB.

6 Conclusions and Future work

A real case study was presented as an example of how a micro-simulation scenario can be built with a map from Open Street Map and the traffic simulation platform SUMO. A detailed description was provided, starting from the network preparation, the definition of the Traffic Assignment Zones, and the trips obtained from Origin-Destination Matrices. A micro-simulation was run and some suggestions about how to use the output files and store the important information in a Mongo database.

This activity could be further developed in many directions: extending the study area, collecting further ground truth demand data to work on model calibration, considering public transport services, etc.

The SUMO platform is plenty of tools properly developed to import data collected by sensors or coming from video cameras, even from drones. These data would also allow a more detailed definition of the vehicles that could be used not only to better model the traffic, but also to compute the emission of pollutants. All these developments would provide a useful support to decision makers to prevent events of traffic congestions and environmental risks.

7 Acknowledgements

This work was partially supported by Italian Ministry of University and Research (MIUR), within the Smart City framework (Project Cagliari 2020, ID: PON04A2_00381).

A special thanks to MLabs srl (www.mlabls.eu) for providing the traffic assignment zones and related OD Matrices.

References

Alvarez Lopez, P., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and Wießner, E. (2018). Microscopic Traffic Simulation using SUMO. *IEEE Intelligent Transportation Systems Conference (ITSC)*.

Aminia, S., Tilga, G., Buscha, F. (2019). Calibration of mesoscopic simulation models for urban corridors based on the macroscopic fundamental diagram. *Proceedings of hEART 2019*. 8th Symposium of the European Association for Research in Transportation.

Bachechi, C., Po, L. (2019). Traffic Analysis in a Smart City. *WI'19 Companion*, 2019, Thessaloniki, Greece, ACM.

Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1), pp. 269–271.

Gawron, C. (1998). *An Iterative Algorithm to Determine the Dynamic User Equilibrium in a Traffic Simulation Model*. International Journal of Modern Physics C Vol. 9 (3), pp. 393-407

Krajzewicz, D., Hertkorn, G., Rössel, C., Wagner, P. (2002). An Example of Microscopic Car Models Validation using the open source Traffic Simulation SUMO. *Proceedings of Simulation in Industry, 14th European Simulation Symposium*. SCS European Publishing House (pp. 318-322). Dresden.

Krauß, S. (1998). *Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics*. Hauptabteilung Mobilität und Systemtechnik des DLR Köln, ISSN 1434-8454

Maiorov, E.R., Ludan, I.R., Motta, J.D. (2019). *Developing a microscopic city model in SUMO simulation system*. Journal of Physics Conference Series 1368 0420812019.

McNally, M.G. (2000). The activity-based approach. *UCI-ITS-AS-WP-00-4*, Center for Activity Systems Analysis, University of California, Irvine, CA. https://escholarship.org/uc/item/5sv5v9qt

McNally, M. G., Rindt C. (2007). The Activity Based Approach. *Handbook of Transport Modelling*. Emerald Group Publishing Limited. (pp. 55-73). ISBN: 978-0-08-045376-7.

Po, L., Rollo, F., Bachechi, C., Corni, A. (2019). From Sensors Data to Urban Traffic Flow Analysis. *IEEE International Smart Cities Conference ISC2 2019*. Casablanca.

Schweizer, J. (2013). Sumopy: an advanced simulation suite for sumo. *Simulation of Urban MObility User Conference*. Springer, Berlin, Heidelberg.

Schweizer, J., Poliziani, C., Rupi, F., Morgano, D., Magi, M. (2021). Building a Large-Scale Micro-Simulation Transport Scenario Using Big Data. *ISPRS International Journal of Geo-Information*. 10, 165.

Ullah, M.R., Khattak, K.S., Khan, Z.H., Khan, M.A., Minallah, N., Khan, A.N. (2021). Vehicular Traffic Simulation Software: A Systematic Comparative Analysis. *Pakistan Journal of Engineering and Technology*, PakJET. Volume: 04, Number: 01, (pp. 66-78).

Web references

SUMO, Simulation of Urban MObility, http://sumo.dlr.de/userdoc/, accessed on 9 February 2022

Sumo source code https://github.com/eclipse/sumo

SUMO whole cities scenarios https://sumo.dlr.de/docs/Data/Scenarios.html

Sumo docs: Including elevation data in a network
https://sumo.dlr.de/docs/Networks/Elevation.html

sumo car-following-models
https://sumo.dlr.de/docs/Definition_of_Vehicles%2C_Vehicle_Types%2C_and_Routes.html#car-following_models

sumopy documentation
https://sumo.dlr.de/docs/Contributed/SUMOPy.html

sumopy code on github https://github.com/schwoz/sumopy

mongo db
https://mongodb.com