SoftwareX 34 (2026) 102745

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

TUD-SUMO: A research-oriented SUMO wrapper for traffic simulation in python

C. Evans [a,*], M. Rinaldi [a], H. Taale [b], S.P. Hoogendoorn [a]

[a] Delft University of Technology, 2628 CD Delft, the Netherlands
[b] Rijkswaterstaat, Lange Kleiweg 34, 2288 GK, Rijswijk, the Netherlands

ARTICLE INFO

Keywords:
Traffic simulation
Adaptive traffic control
SUMO
Python

ABSTRACT

TUD-SUMO is a Python wrapper for SUMO, a traffic simulation software, designed to support the development of traffic control systems, particularly adaptive systems where data is frequently transferred between a controller and the traffic environment. It provides automated data collection and a set of modular, extensible tools allowing for a wide range of scenarios and control strategies to be simulated and compared. These capabilities are accessed through a simplified interface that enables rapid prototyping of control strategies with complex interactions using minimal code, promoting ease of use and portability. TUD-SUMO has already been employed in multiple projects at Delft University of Technology, including two Horizon Europe projects and 2 transportation engineering courses.

Metadata

Code metadata.

Nr.	Code metadata description	Metadata
C1	Current code version	v3.3.2
C2	Permanent link to code/repository used for this code version	https://github.com/DAIMoNDLab/tud-sumo/tree/v3.3.2
C3	Permanent link to Reproducible Capsule	https://github.com/DAIMoNDLab/tud-sumo-examples/tree/v1.0.0
C4	Legal Code License	Apache-2.0 License
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	Python
C7	Compilation requirements, operating environments & dependencies	Python ≥ 3.10 and packages; tqdm, matplotlib, mpl-tools, shapely, requests, moviepy, sumolib & traci.
C8	If available Link to developer documentation/manual	https://tud-sumo.github.io/docs/
C9	Support email for questions	C.Evans@tudelft.nl

1. Motivation and significance

Traffic simulation is a fundamental tool used by researchers, engineers and transport planners to predict the impact of changes to a road network prior to conducting field trials. For instance, planners can test the effectiveness of implementing a control system in a simulated environment before much more costly and potentially safety-critical real-world evaluations. Traffic simulation can also be used to analyse planning scenarios, including those involving changes to infrastructure or land use that may significantly affect the likelihood of congestion forming. In addition, high-precision data and visualisation tools enable more detailed analysis of results and provide a greater understanding of the underlying dynamics within the road network. Such tools further play a fundamental role in machine learning-based traffic management

* Corresponding author.
Email addresses: C.Evans@tudelft.nl (C. Evans), M.Rinaldi@tudelft.nl (M. Rinaldi), Henk.Taale@rws.nl (H. Taale), S.P.Hoogendoorn@tudelft.nl (S.P. Hoogendoorn).

https://doi.org/10.1016/j.softx.2026.102745
Received 1 April 2026; Received in revised form 10 May 2026; Accepted 21 May 2026
Available online 26 May 2026

research, where algorithms often require thousands of simulated hours of training in order to develop an optimal control policy.

Generally, traffic models can be categorised as either macroscopic, mesoscopic or microscopic [1]. Macroscopic models represent traffic as a continuum flow, similar to fluid modelling, such as the Lighthill-Whitham-Richards (LWR) model. On the other hand, microscopic models simulate the lateral and longitudinal positions of individual vehicles according to car-following and lane-changing models. Lastly, mesoscopic models aim to strike a balance between the two. One such example of microscopic traffic simulation software is "Simulation of Urban MObility" (SUMO) [2], a project developed by employees of the Institute of Transportation Systems at the German Aerospace Centre in Berlin. SUMO has become very commonly used within the transportation research field due to its accessibility and wide-ranging functionalities. Specifically, SUMO is designed to simulate multimodal traffic across large transportation networks and is open-source, unlike many other similar software, such as AIMSUN or VISSIM. Its highly granular level of control also allows for a wide variety of traffic control techniques to be developed and evaluated, such as adaptive traffic signal control [3], variable speed limits [4] and ramp metering [5]. SUMO operates as a standalone application, including its own GUI; however, the simulation can be directly interacted with using the provided Traffic Control Interface (TraCI) API. TraCI allows users to retrieve information from an active simulation and dynamically modify the behaviours and characteristics of objects within. However, SUMO, and TraCI by extension, exhibit a steeper learning curve than comparable traffic simulation software [6,7]. This can be particularly true for students and interdisciplinary researchers with less experience in programming or traffic simulation, as it requires extensive programming, setup and data management to fully exploit its capabilities.

We therefore introduce TUD-SUMO, a research-oriented SUMO wrapper developed as part of the DAIMoNDLab research group at Delft University of Technology (TU Delft) in the Netherlands. Its primary goal is to act as a research acceleration tool, enabling rapid prototyping of control strategies and reducing the learning curve for researchers wanting to use traffic simulation as a result. This is achieved through a simplified yet powerful interface with several automation and quality-of-life features emphasising ease-of-use and flexibility. Importantly, as a wrapper, TUD-SUMO leverages TraCI for interactions with the simulation, and so it is not able to perform granular actions that are not implemented in TraCI itself. However, by acting on a higher level without limiting TraCI's capabilities, TUD-SUMO delivers complex engineering-oriented functionality for advanced scenarios and control methods to be easily simulated with minimal code, including aspects such as weather, incidents or metered junctions, as demonstrated in Section 3. Ultimately, these functionalities allow users to focus on traffic control system or policy design over simulation and data management. TUD-SUMO has been in continual development since April 2023, with its first public release on PyPI in July 2024 at version 3.0.0. It has so far shown steady growth in its use and has been used as part of two Horizon Europe projects and multiple transportation engineering modules at TU Delft, as outlined in Section 4.

2. Software description

2.1. Software architecture

Fig. 1 shows how TUD-SUMO operates in a typical adaptive traffic controller use case, where a controller adapts its actions based on the current conditions in the simulated road network.

TUD-SUMO requires the standard SUMO input files, which define the road network, vehicle demand and placement of vehicle detectors. A separate, optional file containing the setup parameters for some TUD-SUMO objects can be included, increasing portability and reducing the need to hardcode these objects on a case-by-case basis. The user then interacts with SUMO entirely through TUD-SUMO and can fetch aggregated data from the simulation in order to generate control actions, such as new phase timings for an intersection traffic signal controller. TUD-SUMO subsequently actuates the control actions using TraCI and automatically collects traffic data relevant to the controller. At the end of the simulation, the user can save all data in a compact standardised data format, either as a single JSON or binary file. Users can generate figures from these final outputs, or during a simulation, visualising basic network statistics or more complex object-specific information.

2.2. Software functionalities

The two main contributions of TUD-SUMO are the simplified simulation interface and automatic data collection. The interface is designed to allow for a wide range of complex interactions, primarily those useful for the development and testing of dynamic traffic controllers, without requiring significant amounts of code and simulation management. Therefore, the main TUD-SUMO component handles the base operation of any controllers, such as the tracking and updating of traffic signal states or redirection of vehicles under a dynamic route guidance system. Importantly, TUD-SUMO does not include any control policies, but instead provides a variety of endpoints for fetching relevant data. This may include, for example, the average vehicle density over a set of detectors in the past five minutes. This is done alongside the automatic data collection, which aims to reduce required simulation management by automatically storing simulation statistics at each time step. In a basic simulation, this includes data such as the number of vehicles in the network or trip information. However, additional TUD-SUMO objects, such as traffic controllers or tracked edges, lanes or junctions, can be created to collect important use-case specific data. For example, if a junction is

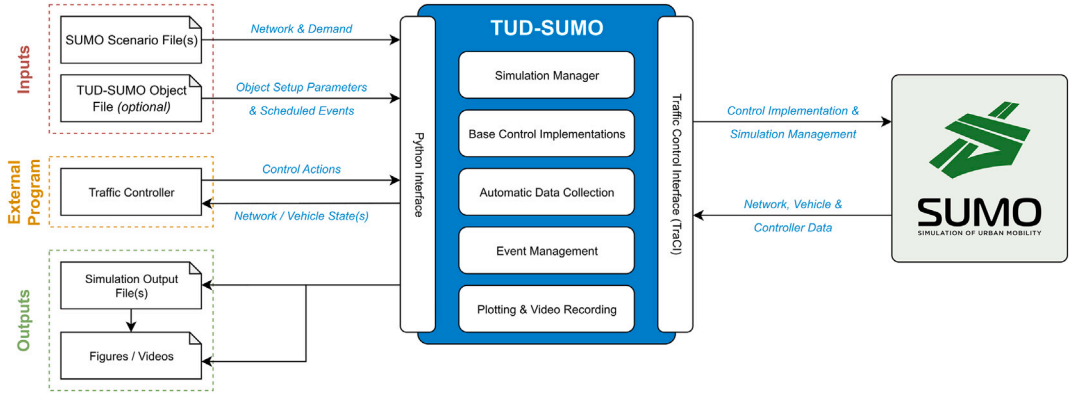

Fig. 1. TUD-SUMO software architecture diagram for a typical traffic controller example.

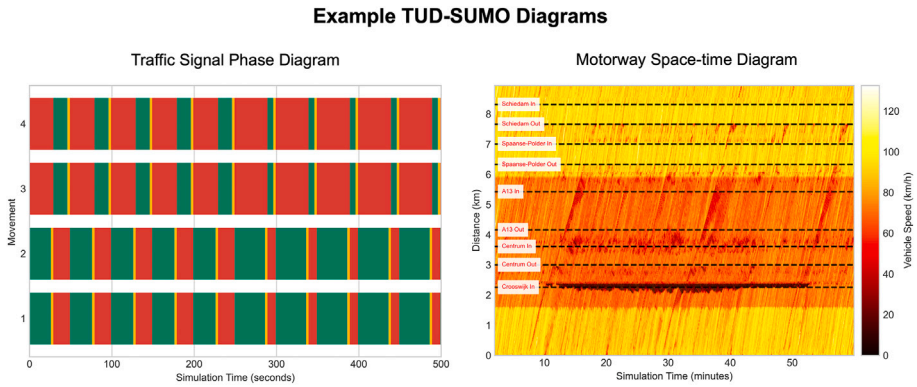

Fig. 2. Examples of diagrams generated by TUD-SUMO.

tracked and certain vehicle detectors are specified, inflow and outflow are automatically calculated and stored. In contrast, when using solely TraCI, all of this data would need to be manually fetched and processed by the user, inevitably increasing implementation time. In particular, this could become an even larger issue when managing data from multiple controllers and detectors, especially as these processes are often reimplemented for each new project.

TUD-SUMO provides data visualisation functions that can be used to quickly generate traffic engineering-focused figures, either from active or completed simulations. Any of the basic information automatically collected from the simulation can be easily plotted, such as the average speeds measured by detectors or network-wide vehicle delay. Further, context-dependent visualisation capabilities include, for instance, the plotting of phase timings under an adaptive traffic signal controller or the metering rate during the operation of a ramp metering system. Similarly, tracking several consecutive edges on a motorway allows for the creation of a space-time diagram, showing vehicle speeds throughout the simulation. Examples of these are shown in Fig. 2. If the user runs a simulation with the GUI active, TUD-SUMO is also able to create videos following specific vehicles or centred on parts of a network.

TUD-SUMO further adds an events layer to SUMO simulations, which allows for events to be dynamically started during a simulation or scheduled at initialisation. This aims to expand the types of scenarios that can be evaluated, including incidents and road closures. However, most network or vehicle parameters can be modified during a custom event, enabling a large range of situations to be simulated in an easily replicable way. For instance, this includes weather-related impacts on vehicle speeds and accelerations, either network-wide or locally. As a result, controlled experiments in specific scenarios can be made significantly more reproducible, particularly by saving event settings to a separate file for shared use.

3. Illustrative examples

To evaluate TUD-SUMO and demonstrate its functionalities, we adopt 5 benchmark tests that aim to cover many of the most common uses of SUMO when applying it to the development and evaluation of adaptive traffic control systems. All tests create a simulation statistics file and use the motorway on-ramp scenario network shown in Fig. 3 with a constant on-ramp demand of 500 vehicles/hour and a mainline demand of 1000 vehicles/hour. The scenario lasts 2000s, with mainline demand increasing to 1500 vehicles/hour between 400-1600s. The test code is shown in Listings 1–5 and uses the functions described in Table 1. Each test aims to do as follows:

1. Simple execution of the simulation without any interaction.
2. Generation of floating-car data from a simple execution of the simulation without any interaction.
3. Generation of a space-time diagram from the minimum required amount of simulation data.
4. A ramp metering scenario with ALINEA applied to the on-ramp, demonstrating traffic signal management and interaction with the simulation.
5. A scenario with an incident active on the road segment immediately downstream of the on-ramp.

4. Impact

To quantitatively demonstrate the advantages of TUD-SUMO, we evaluate how it performs on the tests outlined in Section 3, considering the following Key Performance Indicators: elapsed computational time (s), average CPU utilisation (%), maximum memory usage during execution (MB), code cyclomatic complexity (the number of if-then-else paths and loops during execution), total required lines of source code and total number of function calls to TraCI and simulation/plotter objects. Whilst SUMO lists several tools and extensions in its documentation [9], including some packages with features akin to TUD-SUMO, we limit our comparison to standard TraCI. This is primarily due to the fact that many listed extensions are no longer maintained, or are designed and optimised for very specific tasks. For example, SUMOPy [10], a very comparable wrapper tool, has not been updated past Python 2.7, which reached its end-of-life in January 2020. Meanwhile, SUMO-RL

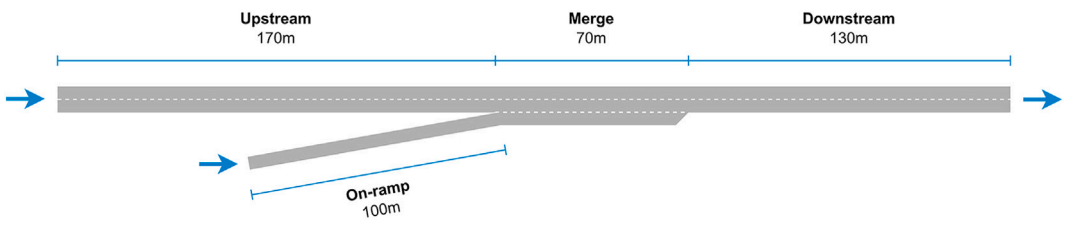

Fig. 3. On-ramp scenario network.

```python
sim = Simulation("TUD-SUMO test 1", verbose=False)
sim.start(SCENARIO_CONFIG, get_fc_data=False, seed=RANDOM_SEED)

sim.step_through(n_steps=MAX_SIM_DURATION)
sim.end()

stats_file = f"{outputs_loc}test_1/stats.txt"
sim.print_summary(stats_file)
```

Listing 1: Test A, showing a simple run of a simulation without any interactions.

```python
sim = Simulation("TUD-SUMO test 2", verbose=False)
sim.start(SCENARIO_CONFIG, get_fc_data=True, seed=RANDOM_SEED)

sim.step_through(n_steps=MAX_SIM_DURATION)
sim.end()

t_outputs_loc = f"{outputs_loc}test_2/"
stats_file = f"{t_outputs_loc}stats.txt"
sim.print_summary(stats_file)

data_file = f"{t_outputs_loc}fc_data.pkl"
sim.save_fc_data(data_file)
```

Listing 2: Test B, showing the generation of floating-car data, containing the position and speed of every vehicle during each time step.

```python
sim = Simulation("TUD-SUMO test 3", verbose=False)
sim.start(SCENARIO_CONFIG, get_fc_data=False, seed=RANDOM_SEED)

edges = ["e_upstream", "e_weave", "e_downstream"]
sim.add_tracked_edges(edges)

sim.step_through(n_steps=MAX_SIM_DURATION)
sim.end()

t_outputs_loc = f"{outputs_loc}test_3/"
stats_file = f"{t_outputs_loc}stats.txt"
sim.print_summary(stats_file)

data_file = f"{t_outputs_loc}all_data.pkl"
sim.save_data(data_file)

plt = Plotter(data_file)

plt.plot_space_time_diagram(
    edges,
    fig_title="TUD-SUMO Implementation",
    save_fig=f"{t_outputs_loc}st_diagram.png",
)
```

Listing 3: Test C, showing the generation of a space-time diagram from the minimum required amount of simulation data.

```python
sim = Simulation("TUD-SUMO test 4", verbose=False)
sim.start(SCENARIO_CONFIG, get_fc_data=False, seed=RANDOM_SEED)

sim.add_tracked_junctions(
    {"J2": {"meter_params": {"min_rate": MIN_RATE, "max_rate": MAX_RATE}}}
)

prev_rate = MAX_RATE
while sim.is_running() and sim.curr_step + CONTROL_INTERVAL <= MAX_SIM_DURATION:

    sim.step_through(n_steps=CONTROL_INTERVAL)

    curr_occupancy = (
        sim.get_interval_detector_data(["dn_1", "dn_2"], "occupancies", CONTROL_INTERVAL)
        * 100
    )
    rate = get_metering_rate(prev_rate, curr_occupancy, 0_CR, K_R)
    sim.set_tl_metering_rate("J2", rate)
    prev_rate = rate

sim.end()

t_outputs_loc = f"{outputs_loc}test_4/"
stats_file = f"{t_outputs_loc}stats.txt"
sim.print_summary(stats_file)
```

Listing 4: Test D, showing a run of a motorway on-ramp scenario with ramp metering applied (ALINEA [8]), demonstrating traffic signal management and interaction with the simulation.

[11] and FLOW [12] are specifically designed to generate environments for machine learning-based traffic control systems.

Therefore, all of the benchmark tests in Section 3 were identically implemented using only TraCI for the traffic simulation, as well as Matplotlib [13] for figures in Test C. Results are shown in Fig. 4, and summarised for each test in Fig. 5. All tests used the same scenario as in Fig. 3 with an identical SUMO seed, in order to avoid any variation in the performance of SUMO itself, and were run on the same Apple M1 MacBook Pro with 16GB RAM. All test code for TUD-SUMO and TraCI is

```python
sim = Simulation("TUD-SUMO test 5", verbose=False)
sim.start(SCENARIO_CONFIG, get_fc_data=False, seed=RANDOM_SEED)

sim.step_through(n_steps=INCIDENT_START)

veh_id = random.choice(sim.get_geometry_vals(VEHICLE_LOCATION, "vehicle_ids"))

sim.cause_incident(
    INCIDENT_DURATION, vehicle_ids=veh_id, edge_speed=15, position=VEHICLE_POSITION
)

sim.step_through(n_steps=MAX_SIM_DURATION - INCIDENT_START)

sim.end()

stats_file = f"{outputs_loc}test_5/stats.txt"
sim.print_summary(stats_file)
```

Listing 5: Test E, showing a run of a simulation with a dynamically created incident.

available in the TUD-SUMO examples GitHub repository [14]. Repeated runs of each test on a comparable system indicated high levels of performance stability, with a coefficient of variation (CV) across 10 runs below 3% for all metrics, with most below 1.5%. Peak memory utilisation showed a brief warm-up phase before stabilising with a CV below 4%.

The tradeoff between computational load and user-friendliness is very evident, where TUD-SUMO performs better in terms of usability metrics and memory utilisation, yet worse in runtime and CPU utilisation. For example, in tests C-E, the required lines of code were 50.9-52.9% lower and the cyclomatic complexity was 72.7-87.5% lower, as shown in Fig. 4(a)-b. Tests C and D are of note as they represent 2 important use cases for motorway traffic control, which are the generation of a space-time diagram and the application of an adaptive ramp metering system, here using the ALINEA algorithm. In the first case, to produce the same result in the TraCI example, floating car data has to be manually processed, and many more calls to a separate plotting package, Matplotlib, are also required. Both result in a cyclomatic code complexity 8 times higher. Similarly, in test D, the ramp meter state has to be manually tracked and its input collected and aggregated over the control interval. Cyclomatic complexity is, therefore, 3.7 times higher using standard TraCI and the implementation required slightly over double the lines of code than with TUD-SUMO. This demonstrates how TUD-SUMO minimises code without limiting capability, here replacing several TraCI calls and lines of simulation management with individual powerful functions listed in Table 1.

Then, whilst peak memory utilisation with TUD-SUMO is 4% and 80% higher in tests A and B, respectively, it is around 25% lower for tests C-E. From this, we can see in simple use cases that TUD-SUMO is less memory efficient, particularly in test B, where floating car data needs to be specifically processed on request from tracked edge data. Yet, in more complex scenarios, TUD-SUMO has a lower peak memory utilisation, through its efficient data storage and aggregation when performing more elaborate tasks. As TUD-SUMO only adds functionality on top of TraCI, it exhibits comparatively longer total runtimes and significantly higher CPU utilisation. Specifically, as shown in Fig. 4(e)-f, the runtime for each test was between 1.7 and 3.6 times longer using TUD-SUMO, and CPU utilisation was mostly between 3.3 and 8.5 times higher. TraCI is, therefore, more CPU efficient as utilisation only approaches that of TUD-SUMO in test B, where Matplotlib has to be used to generate a space-time diagram, as is done in TUD-SUMO. The difference in computational load is also especially noticeable in test B, where the runtime increased from 2.2 to 7.9 s due to the aforementioned processing of floating car data. However, this complexity and computational tradeoff is still within reasonable bounds and is often acceptable in many research and teaching situations where runtime is not a direct priority. This is also particularly true as the complexity of the simulated scenario or control system increases. The tools and endpoints provided by TUD-SUMO are intended to be significantly flexible and capable of being combined for use in very niche or technical research problems, meaning that the corresponding increase in code complexity should also still be much more manageable than without TUD-SUMO.

Table 1
Descriptions of all functions used within the benchmark tests shown in Listings 1–5.

Test	Function	Description
A-E	`start()`	Initialises a SUMO simulation and necessary TUD-SUMO data structures.
A-E	`step_through()`	Advances the simulation by any number of steps.
A-E	`end()`	Ends the SUMO simulation.
A-E	`print_summary()`	Prints a summary of a simulation, including vehicle statistics and outline of TUD-SUMO objects.
B	`save_fc_data()`	Saves all floating car data from the simulation.
C	`add_tracked_edges()`	Initialises tracking of more detailed edge-specific data.
C	`save_data()`	Saves all collected data, including step vehicle, detector and trip data.
C	`plot_space_time_diagram()`	Plots a space-time diagram using floating-car data from tracked edges.
D	`add_tracked_junctions()`	Initialises tracking of more detailed junction-specific data.
D	`is_running()`	Returns whether the simulation is active, with still vehicles in the network or waiting to be inserted.
D	`get_interval_detector_data()`	Returns aggregated detector data, averaged over a given time interval and/or multiple detectors.
D	`set_tl_metering_rate()`	Sets the metering rate for a ramp meter after it is set as a tracked junction.
E	`cause_incident()`	Simulates an incident by temporarily halting a vehicle on its subsequent edge.

In general, TUD-SUMO can require a much higher level of computational resources to perform the same task, yet this is somewhat inevitable as a wrapper that expands upon the functionalities of TraCI. Primarily, despite the simulation outcome being identical, the design principles behind TUD-SUMO mean that the package defaults to a larger coverage of high-quality data instead of offloading any tedious data management to the user. As a result, more computational resources are required in order to produce a richer and more useful output without harming usability, particularly as TraCI is still used by TUD-SUMO as the primary method of interacting with SUMO. This tradeoff is demonstrated in Fig. 5, which shows how both packages perform in each test across all metrics tested. The values are normalised using total sum scaling, with lower values and a smaller overall profile being more favourable. Here, it can be clearly seen how TUD-SUMO improves upon usability at the cost of performance for the same task. Importantly, several optimisations are included to reduce this performance gap, such as toggleable data collection and automatically managing subscriptions, an optional feature of TraCI that reduces message overhead produced from fetching simulation data. Including modular objects also means that users can choose to collect useful, yet more case-specific, data only when necessary, such as tracking vehicle speeds along an edge only when it is relevant to their work.

TUD-SUMO has been made openly available through GitHub and PyPI distributions, with its use at TU Delft, at its Transport & Planning department and DAIMoNDLab research group, steadily growing. For instance, according to PyPI download statistics queried using 'pypinfo,' TUD-SUMO was downloaded 778 times in 2024 and 2025, which does include automated installations. It has also been used in 2 Horizon Europe projects, DIT4TraM [15,16] and ACUMEN [17]. The initial development of TUD-SUMO in April 2023 was motivated by a DIT4TraM pilot where the algorithm for an adaptive traffic signal controller had already been largely implemented, yet required integration with a simulation environment. The algorithm used an auction-based mechanism for handling cyclist priority in signalised intersections, where their positions were communicated to a dedicated app and server. As a result of the system complexity, avoiding dependencies and changes to the existing codebase was considered important, and TUD-SUMO was a significant aid in producing results. Development continued after the pilot project was completed, leading to the public release of TUD-SUMO at v3.0.0 in July 2024.

Its successful use in teaching also demonstrates the robustness and accessibility of the package, as TUD-SUMO is now being used in 2 courses at TU Delft. The courses, with codes CT3505-24 and CIEM6110, focus on traffic flow theory and traffic management and require traffic simulation work in assignments or student workshops. Specifically, in CT3505-24, TUD-SUMO was integrated into a web interface to familiarise students with car-following models and ramp metering systems. The module is offered to multidisciplinary students with varying levels of prior knowledge; in this context, TUD-SUMO allowed pupils to focus on the intended learning outcomes, with minimal time wasted on coding requirements. In CIEM6110, which involved evaluating the impact of automated vehicle penetration rates, TUD-SUMO was used to instantiate vehicle classes and easily run and compare multiple simulations with increasing rates through its interface and visualisation tools. In both cases, the same features designed to support research are also very advantageous for teaching, where the focus is on understanding and using traffic control theory rather than managing a simulation. TUD-SUMO, overall, has been applied in research since the beginning of its development [18] and is planned for continued use in future projects.

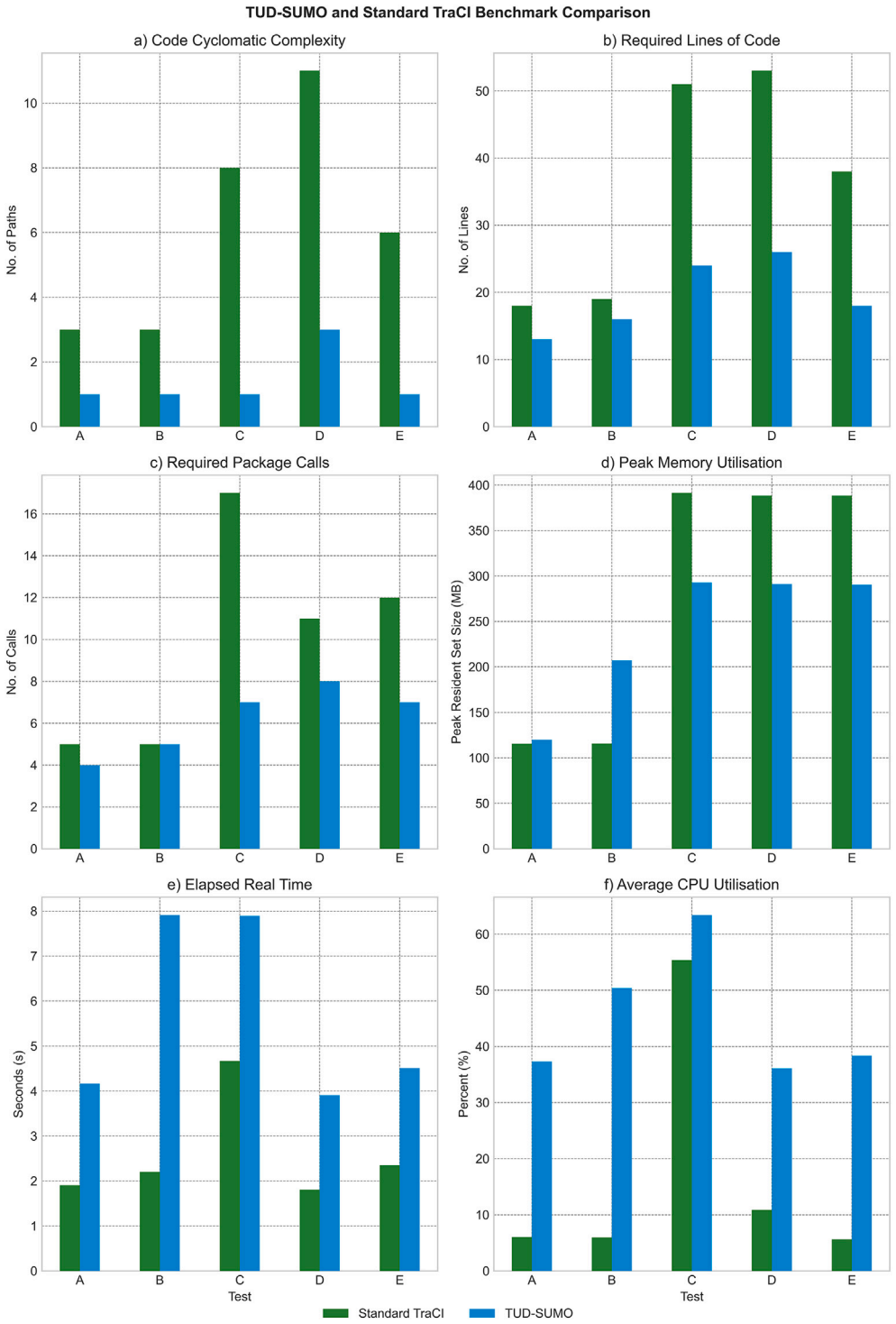

Fig. 4. Benchmark comparison of TUD-SUMO against standard TraCI. a) Code cyclomatic complexity, representing the number of paths through the test code. b) Required lines of code. c) Total number of function calls to TraCI or Simulation/Plotter objects. d) Maximum recorded memory usage during execution. e) Elapsed duration of the test. f) Average CPU utilisation.

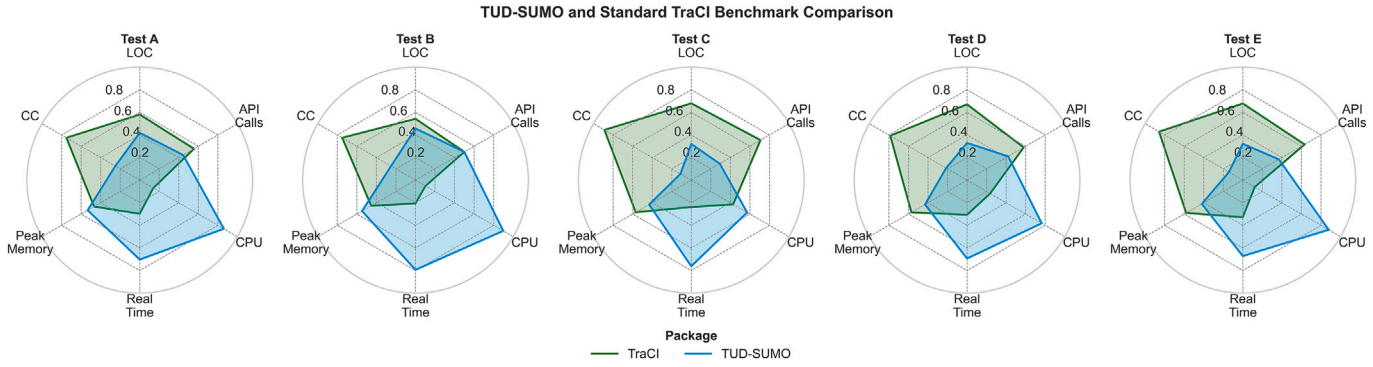

Fig. 5. Benchmark test results, normalised using total sum scaling for easy comparison. a) code cyclomatic complexity (CC). b) required lines of code (LOC). c) total number of function calls to TraCI or Simulation/Plotter objects. d) maximum recorded memory usage during execution. e) elapsed duration of the test. f) average CPU utilisation.

5. Conclusions

We presented TUD-SUMO, a wrapper for the traffic simulation software SUMO, that is specifically designed to accelerate the use of microscopic traffic simulation in both research and teaching applications. SUMO is a powerful tool that is very commonly used in the transportation research field, yet it has a steep learning curve, and its use can often require significant implementation time and programming. TUD-SUMO instead aims to streamline this process. This is achieved through a simplified interface for interacting with SUMO using its API, TraCI, adding several automation and quality-of-life features, such as automatic data collection, events and visualisation tools. As demonstrated, whilst TUD-SUMO introduces additional computational load, it is proven effective in reducing user complexity for a given modelling and simulation task. For example, in the more complex benchmark tests, the required lines of code were approximately 50% lower when using TUD-SUMO in comparison to standard TraCI, whilst runtime increased by a factor of 1.7-3.6. This tradeoff is inevitable; however, its success in research and teaching projects demonstrates that TUD-SUMO is a very effective tool for facilitating the use of microscopic traffic simulation in cases where simulation management is not a priority.

The design of TUD-SUMO is intended to be modular, so it can be easily maintained and extended. This could include features such as an interface for greater control of public transport or improved scenario creation tools with the ability to procedurally generate scenarios based on real-world data.

CRediT authorship contribution statement

C. Evans: Writing – review & editing, Writing – original draft, Visualisation, Software, Methodology, Formal analysis, Conceptualization. **M. Rinaldi:** Writing – review & editing, Supervision, Software, Funding acquisition, Formal analysis, Conceptualization. **H. Taale:** Writing – review & editing, Supervision, Funding acquisition, Formal analysis, Conceptualization. **S.P. Hoogendoorn:** Writing – review & editing, Supervision, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships that may be considered potential competing interests:

Callum Evans reports that financial support was provided by Rijkswaterstaat Water Traffic and Living Environment. Henk Taale reports a relationship with Rijkswaterstaat Water Traffic and Living Environment that includes employment. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The author thanks Dr. Alex Roocroft for detailed and insightful feedback which helped improve the framing of this work.

This work is part of the project “AI in Network Management,” funded by Rijkswaterstaat, grant agreement no. 31179439, under the label of ITS Edulab. One co-author is employed by the Rijkswaterstaat, however, authors report that this affiliation has not influenced the research. There are no other competing interests to declare.

References

[1] van Wageningen-Kessels F, van Lint H, Vuik K, Hoogendoorn S. Genealogy of traffic flow models. EURO J Transp Logist Dec 2015;4:445–73.
[2] Lopez PA, Behrisch M, Bieker-Walz L, Erdmann J, Flötteröd Y-P, Hilbrich R, Lücken L, Rummel J, Wagner P, Wießner E. Microscopic traffic simulation using SUMO. In: The 21st IEEE international conference on intelligent transportation systems. Maui, HI, USA): IEEE; 2018.
[3] McKenney D, White T. Distributed and adaptive traffic signal control within a realistic traffic simulation. Eng Appl Artif Intell Jan 2013;26:574–83.
[4] Guo Y, Xu H, Zhang Y, Yao D. Integrated variable speed limits and lane-changing control for freeway lane-drop bottlenecks. IEEE Access 2020;8:54710–21.
[5] Gu C, Zhou T, Wu C. Deep Koopman traffic modeling for freeway ramp metering. IEEE Trans Intell Transp Syst Jun 2023;24:6001–13.
[6] Ullah MR, Khattak KS, Khan ZH, Khan MA, Minallah N, Khan AN. Vehicular traffic simulation software: a systematic comparative analysis. Pak J Eng Technol Mar 2021;4:66–78.
[7] Saidallah M, Fergougui AE, Elalaoui AE. A comparative study of urban road traffic simulators. In: MATEC web of conferences, vol. 81; 2016, p. 05002.
[8] Papageorgiou M, Hadj-Salem H, Blosseville J-M. ALINEA: a local feedback control law for on-ramp metering. Transp Res Rec 1991;(1320):58–64. ISBN 9780309051538..
[9] Simulation of Urban MObility (SUMO) Development Team. Contributed. Feb 2026. https://sumo.dlr.de/docs/Contributed/index.html. (Accessed: 24/ 02/ 2026).
[10] Schweizer J. SUMOPy, GitHub. Aug 2024. https://github.com/schwoz/sumopy.
[11] Alegre LN. SUMO-RL, GitHub. May 2024. v1.4.5. https://github.com/LucasAlegre/sumo-rl.
[12] Wu C, Kreidieh A, Parvate K, Vinitsky E, Bayen AM. Flow: a modular learning framework for mixed autonomy traffic. IEEE Trans Robot Apr 2022;38, arXiv:1710.05465 [cs], pp. 1270–86.
[13] Hunter JD. Matplotlib: a 2D graphics environment. Comput Sci Eng 2007;9(3):90–5.
[14] DAIMoNDLab. tud-sumo-examples, GitHub. Feb 2026. https://github.com/DAIMoNDLab/tud-sumo-examples/tree/v1.0.0.
[15] European Commission. Distributed intelligence and technology for traffic and mobility management. Tech. Rep. Grant agreement No. 953783 European Commission; Aug 2024. https://doi.org/10.3030/953783
[16] Rinaldi M, Hegyi A, Taale H, Tavassoli K. Bordeaux pilot preparations: theory, algorithm and simulation. Tech. rep. Delft University of Technology; Jan 2023.
[17] European Commission. AI-aided deCision tool for seamless mUltiModal nEtwork and traffic managemeNt. Tech. Rep. Grant agreement No. 101103808 European Commission; May 2026. https://doi.org/10.3030/101103808
[18] Evans C, Rinaldi M, Taale H, Hoogendoorn SP. Impact of pre-training on deep reinforcement learning ramp metering systems. In: Proceedings of the national academy of science’s transportation research board 104th annual meeting. Washington DC, USA; Jan 2025. TRBAM-25-01884.