

RidePy: A fast and modular framework for simulating ridepooling systems

Felix Jung^{1,*} and Debsankha Manik¹

¹*Chair of Network Dynamics, Institute of Theoretical Physics and Center for Advancing Electronics Dresden (cfaed),
TUD Dresden University of Technology, 01062 Dresden, Germany*

(Dated: December 4, 2023)

SUMMARY

RidePy enables fast computer simulations of on-demand mobility modes such as ridehailing or ridepooling. It strongly focuses on modeling the mobility service itself, rather than its customers or the environment. Through a combination of Python [1], Cython [2] and C++ [3], it offers ease of use at high performance. Its modular design makes customization easy, while the included modules allow for a quick start.

STATEMENT OF NEED

An accelerating climate change and congested cities both call for an urgent change in the way we move [4]. To reduce carbon dioxide emissions as well as the number of vehicles on the road, digitally managed on-demand mobility services such as ridehailing and ridepooling are explored in research [5, 6] and on the road. Unfortunately, physically experimenting with such services for research purposes is extremely cost- and time-intensive. However, the operational properties of such systems are largely predefined in terms of the scheduling backend that manages them. This makes it possible to replace physical experiments with computer simulations, substituting virtual vehicles for actual ones and modeling the incoming mobility demand by sampling either historic requests or synthetic distributions. Another advantage of simulations is that the degree to which they represent reality may be freely adjusted. This makes it possible to both answer concrete operational questions [7–13] and investigate idealized system behavior, gaining deeper insights into the general properties of on-demand mobility systems [14–19].

In this context, a simulation framework should appropriately allow for vastly different system sizes and degrees of realism. The system size incorporates the number of simulated vehicles as well as the extent of the space they operate on: A small system may consist of a single vehicle serving a network of just two nodes, while an example of a large system could be a fleet of several thousand vehicles operating on the street network of a large city. The degree of realism may be varied, for example, by sampling requests from either a uniform distribution or recorded mobility demand, or by operating on a continuous Euclidean plane versus a realistic city street network. Another option is to adjust the constraints imposed, such as the time windows assigned to stops or the vehicles’ seat capacities.

Finally, an on-demand mobility simulation framework should be fast, easy to use and adaptable to various applications.

A number of open-source simulation software projects are already being used to investigate on-demand mobility services. Some of them focus on microscopic modeling in realistic settings, through which concrete predictions for service operation are enabled, guiding urban planning. Prominent examples are MATSim [20], which performs agent-based simulations of individual inhabitants, and Eclipse SUMO [21], a microscopic traffic simulator. Both rely on additional packages to model on-demand mobility, such as AMODEUS [22] for MATSim and Jade [23] for SUMO.

FleetPy [24], a recently released on-demand mobility simulation, is primarily aimed at realistic modeling of the interactions between operators and users, specifically incorporating multiple operators. While its technical approach is similar to ours, integrating Python with fast Cython and C++ extensions, the project is predominantly focused on applied simulations, although its framework architecture promises to allow for adjustment of the model detail level.

Perhaps the most idealized approach is taken by the Julia [25] package `RidePooling.jl` [26] which was developed in support of a recent scientific contribution [17].

A very different yet interesting route is taken by MaaSsim [27], which models on-demand mobility in the realm of two-sided mobility platforms such as Uber [28] and Lyft [29].

RidePy extends this landscape by providing a universal and fast ridepooling simulation framework that is highly customizable while still being easy to use. It is focused on modeling the behavior of a vehicle fleet while covering a broad scope in terms of system size and degree of realism.

PHILOSOPHY AND USAGE

RidePy simulates flexible mobility services based on *requests*, *dispatchers* and *vehicles*. The vehicles continuously move along routes defined by scheduled *stops*. At each stop, passengers are picked up or dropped off, leading to a change in seat occupancy aboard the vehicle. A `RequestGenerator` supplies requests for mobility that are submitted to the simulated service, consisting of origin and destination locations and optional constraints. A `dispatcher` processes these incoming requests. If a request cannot be fulfilled given the constraints (e.g., time windows, seat capacity), it is rejected upon submission. Otherwise, pick-up and drop-off stops are scheduled with a vehicle, respectively.

All individual components of the simulation framework may be customized or replaced. This includes `RequestGenerators`, `dispatchers`, and the `TransportSpace` which the system operates on. Examples for `TransportSpaces` include the continuous Euclidean plane and arbitrary weighted graphs (e.g., street networks). Several components of RidePy are implemented in both pure Python and Cython/C++. While their pure Python versions are easier to understand, debug and modify, the Cython/C++ versions make large-scale simulations tractable.

Running a RidePy simulation yields a sequence of `Events`. The included analytics code consumes these events and returns two extensive Pandas [30] `DataFrames`: `stops` and `requests`. `stops` contains all stops that have been visited by each vehicle, along with additional information such as the vehicles' passenger occupancy. `requests` similarly contains all requests that have entered the system, enriched with secondary information such as the time riders spent on the vehicle.

Additional included tooling allows for the setup, parallel execution, and analysis of simulations at different parameters (parameter scans). This includes the serialization of all simulation data in JSON format [31].

To ensure valid behavior, RidePy incorporates an extensive automated test suite [32].

AVAILABILITY

RidePy is available from PyPI [33]. The source code is hosted on GitHub [34]. Extensive documentation can be found on the project's webpage [35].

ACKNOWLEDGEMENTS

We kindly thank Philip Marszal, Matthias Dahlmanns, Knut Heidemann, Malte Schröder, and Marc Timme for their input and advice.

This project was partially supported by the Bundesministerium für Bildung und Forschung (BMBF, German Federal Ministry of Education and Research) under grant No. 16ICR01 and by the Bundesministerium für Digitales und Verkehr (BMDV, German Federal Ministry for Digital and Transport) as part of the innovation initiative mFund under grant No. 19F1155A.

COMPETING INTERESTS

Debsankha Manik was employed at MOIA GmbH when this research was conducted. MOIA GmbH neither sponsored nor endorses his research.

* felix.jung@tu-dresden.de

- [1] G. Van Rossum and F. L. Drake Jr, *Python Reference Manual* (Centrum voor Wiskunde en Informatica Amsterdam, 1995).
- [2] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith, *Comput. Sci. Eng.* **13**, 31 (2011), ISSN 1521-9615.
- [3] B. Stroustrup, *The C++ Programming Language* (Pearson Education India, 2000).
- [4] L. Winkler, D. Pearce, J. Nelson, and O. Babacan, *Nat Commun* **14**, 2357 (2023), ISSN 2041-1723.
- [5] R. Engelhardt, F. Dandl, A. Bilali, and K. Bogenberger, in *2019 IEEE Intell. Transp. Syst. Conf. ITSC* (2019), pp. 2992–2997.
- [6] P. Santi, G. Resta, M. Szell, S. Sobolevsky, S. H. Strogatz, and C. Ratti, *PNAS* **111**, 13290 (2014), ISSN 0027-8424, 1091-6490.

- [7] A. de Ruijter, O. Cats, J. Alonso-Mora, and S. Hoogendoorn, *Transp. Plan. Technol.* **46**, 407 (2023), ISSN 0308-1060.
- [8] A. Henao and W. E. Marshall, *Transportation* **46**, 2173 (2019), ISSN 1572-9435.
- [9] C. Lotze, P. Marszal, M. Schröder, and M. Timme, *New J. Phys.* **24**, 023034 (2022), ISSN 1367-2630.
- [10] C. Ruch, C. Lu, L. Sieber, and E. Frazzoli, *IEEE Trans. Intell. Transp. Syst.* pp. 1–6 (2020), ISSN 1558-0016.
- [11] G. Wilkes, R. Engelhardt, L. Briem, F. Dandl, P. Vortisch, K. Bogenberger, and M. Kagerbauer, *Transp. Res. Rec.* **2675**, 226 (2021), ISSN 0361-1981.
- [12] F. Zwick, N. Kuehnel, R. Moeckel, and K. W. Axhausen, *Procedia Computer Science* **184**, 662 (2021), ISSN 1877-0509.
- [13] F. Zwick, N. Kuehnel, and S. Hörl, *Transportation Research Part A: Policy and Practice* **165**, 300 (2022), ISSN 0965-8564.
- [14] S. Herminghaus, *Transp Res Policy Pr.* **119**, 15 (2019).
- [15] D. Manik and N. Molkenhain, *Appl Netw Sci* **5**, 49 (2020), ISSN 2364-8228.
- [16] N. Molkenhain, M. Schröder, and M. Timme, *Phys. Rev. Lett.* **125**, 248302 (2020).
- [17] S. Mühle, *Multimodal Transportation* **2**, 100080 (2023), ISSN 2772-5863.
- [18] R. Tachet, O. Sagarra, P. Santi, G. Resta, M. Szell, S. H. Strogatz, and C. Ratti, *Sci Rep* **7**, 42868 (2017), ISSN 2045-2322.
- [19] R. M. Zech, N. Molkenhain, M. Timme, and M. Schröder, *Sci Rep* **12**, 10880 (2022), ISSN 2045-2322.
- [20] ETH Zürich, A. Horni, K. Nagel, TU Berlin, and K. W. Axhausen, eds., *The Multi-Agent Transport Simulation MATSim* (Ubiquity Press, 2016), ISBN 978-1-909188-75-4, URL <http://www.ubiquitypress.com/site/books/10.5334/baw/>.
- [21] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner, in *2018 21st Int. Conf. Intell. Transp. Syst. ITSC* (2018), pp. 2575–2582, ISSN 2153-0017.
- [22] C. Ruch, S. Horl, and E. Frazzoli, in *2018 21st Int. Conf. Intell. Transp. Syst. ITSC* (IEEE, Maui, HI, 2018), pp. 3639–3644, ISBN 978-1-72810-321-1 978-1-72810-323-5.
- [23] M. Behrisch, D. Krajzewicz, and M. Weber, eds., *Simulation of Urban Mobility: First International Conference, SUMO 2013, Berlin, Germany, May 15-17, 2013. Revised Selected Papers*, no. 8594 in Information Systems and Applications, Incl. Internet/Web, and HCI (Springer Berlin Heidelberg : Imprint: Springer, Berlin, Heidelberg, 2014), 1st ed., ISBN 978-3-662-45079-6.
- [24] R. Engelhardt, F. Dandl, A.-A. Syed, Y. Zhang, F. Fehn, F. Wolf, and K. Bogenberger, *FleetPy: A Modular Open-Source Simulation Tool for Mobility On-Demand Services* (2022), 2207.14246.
- [25] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, *SIAM Rev.* **59**, 65 (2017), ISSN 0036-1445, 1095-7200.
- [26] S. Mühle, *RidePoolingSimulations* (2022), URL <https://github.com/SteffenMuehle/RidePoolingSimulations>.
- [27] R. Kucharski and O. Cats, *PLOS ONE* **17**, e0269682 (2022), ISSN 1932-6203.
- [28] Inc. Uber Technologies, *Uber* (2023), URL <https://www.uber.com/>.
- [29] Inc. Lyft, *Lyft* (2023), URL <https://www.lyft.com/>.
- [30] Wes McKinney, in *Proc. 9th Python Sci. Conf.*, edited by S. van der Walt and Jarrod Millman (2010), pp. 56–61.
- [31] T. Bray, *The JavaScript Object Notation (JSON) Data Interchange Format* (2017), URL <https://www.rfc-editor.org/rfc/rfc4180.txt>.
- [32] H. Krekel, B. Oliveira, R. Pfannschmidt, F. Bruynooghe, B. Laughner, and F. Bruhin, *Pytest* (2004), URL <https://github.com/pytest-dev/pytest>.
- [33] Felix Jung and Debsankha Manik, *Ridepy - PyPI* (2023), URL <https://pypi.org/project/ridepy/>.
- [34] Felix Jung and Debsankha Manik, *PhysicsOfMobility/ridepy - github* (2020), URL <https://github.com/PhysicsOfMobility/ridepy>.
- [35] Felix Jung and Debsankha Manik, *RidePy documentation* (2023), URL <https://ridepy.org/>.