Exploring accelerated evolutionary parameter search for iterative large-scale transport simulations in a new calibration testbed

Sebastian Hörl*[1]

[1]PhD, Institut de Recherche Technologique SystemX, France

Paper presented at hEART 2022, 1-3 June 2022, Leuven, Belgium

SHORT SUMMARY

Large-scale agent-based transport models of whole territories have become an important tool in research and planning of new services and policies. Yet, studies based on those tools are rarely reproducible due to the complexity of data sources and modeling processes. One important element towards fully replicable simulations is automatic calibration of behavioral and infrastructural model parameters. The present paper contributes to standardizing the calibration process by describing a consistent framework for benchmarking calibration objectives and optimization algorithms. Furthermore, the paper advances the current state of the art by exploring the integration of a search acceleration method for iterative simulators (*opdyts*) with sample-based evolutionary search algorithms. In a use case for Paris and the MATSim simulator, we demonstrate the applicability of the framework. We show that *opdyts* accelerates the parameter search process, although its comparative runtime benefits decrease with higher availability of computational resources.

Keywords: transport simulation, calibration, parameter search, acceleration, mode share, MATSim

1. INTRODUCTION

Agent-based transport simulations have gained in interest over the past years as they allow studying in detail the highly dynamic interactions between customers, mobility providers and system control measures in increasingly digitalized transport systems. By adding behavioral components to simulate the decision-making of people, agent-based simulations have been used to construct scenarios for future transport technologies, mobility services, and policies.

The present paper is embedded in a larger effort of making such large-scale agent-based transport simulations reproducible. (Hörl & Balac, 2021b) propose a method to create synthetic travel demand data sets consisting of synthetic households, persons and their daily activity patterns from open and publicly available data with a reference use case for Paris and Île-de-France. By adding network data from OpenStreetMap and regional public transport schedules, the pipeline allows producing open and runnable simulations for the agent-based transport simulation framework MATSim (Horni, Nagel, & Axhausen, 2016). While the demand data can serve as input to any agent-based transport simulator, it is currently optimized to be used in MATSim's *eqasim* extension (Hörl & Balac, 2021a), which focuses on mode decisions which are governed by discrete choice models. In the full model set up, agents interact in a traffic simulation, perceive traffic

conditions and perform mode decisions for the trips in their daily schedules after a one-day simulation. After, their updated daily schedules are simulated again, until the decision of the agents and the traffic conditions reach a stochastic equilibrium state. The process is iterative and dependent on input parameters that value travel time, monetary costs or line switches when comparing different mode alternatives.

While the choice model parameters can be estimated from (proprietary) survey data, key metrics such as mode shares or network flows will still show differences when compared with reference data sets. Hence, both demand-side (behavioral) and supply-side (network) parameters need to be adjusted to achieve a high correspondence with reference data.

So far, the calibration process has been performed manually with a researcher proposing a set of parameters, running the simulation and comparing the resulting metrics with reference data. However, within the scope of reproducibility, we are interested in establishing an automated pipeline from raw data sets to the final calibrated simulation. In this context, the process of model parameter calibration needs to be automated. Having in mind that large-scale agent-based transport simulations have high demand in computation time, using efficient and adapted algorithms is a key requirement.

2. BACKGROUND

In the context of MATSim, some efforts of automatic calibration have been presented. (He et al., 2021) regard MATSim as a black box with input parameters and an objective value as output to calibrate network flows using the SPSA approach (Spall, 1998). (Agarwal, Flötteröd, & Nagel, 2017) describe a proof-of-concept for calibrating mode choice parameters using a search acceleration strategy proposed by (Flötteröd, 2017). Apart from those efforts, no readily useable calibration framework and benchmark for MATSim or similar simulators exist, which is an endeavor that will be backed by the present research.

A common approach to perform parameter optimization for stochastic functions without calculable derivatives is black-box optimization. Various known optimization approaches such as SPSA. A common class of approaches are covered by Evolutionary Search (ES) strategies. Those algorithms are usually population-based in that in every iteration, a set of potential candidates is generated. Usually, these candidates are generated by mutating or recombining well-performing parameter choices from previous generations. A frequently used approach is Covariance Matrix Adaptation ES (Hansen, 2006) with CMA-(λ,1)-ES being one specific version. Briefly summarized, the algorithm is based on a normal distribution with a mean and covariance matrix that is adapted, based on the performance of new proposed parameters. Specifically, the simple (λ,1) variant prescribes to sample λ new candidates from the given distribution in each iteration and replaces the current mean with the most successful candidate if it is better than the current best. An implementation of CMA-(λ,1)-ES in pseudocode can be found in (Igel, Suttorp, & Hansen, 2006).

Any black box algorithm can be used to calibrate model parameters, as long as the specific model can be packaged as a function with input parameters and a resulting output value. It is, hence, possible to calibrate a MATSim simulation using CMA-(λ,1)-ES. However, speeding up the calibration process means exploiting specificities of the simulator. One of the major features of MATSim is that it progresses iteratively, and that the calculations in one iteration are only dependent on the previous state. The latter property especially means that the simulation state can be saved and restarted later on. Based on these properties, (Flötteröd, 2017) proposes the search acceleration method *opdyts* that not only operates on full simulation runs, but takes into account transitional states. In brief, the algorithm requires the user to propose a set of N parameter candi-

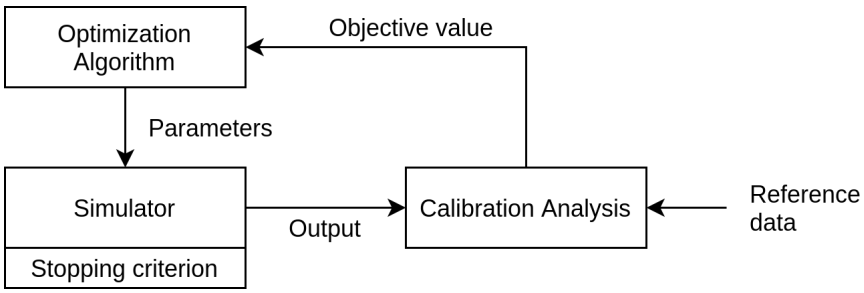

Figure 1: Structure of the calibration loop

dates, which are then implemented in respective simulation runs indexed by $k \in \{1,...,N\}$. Each run is then transitioned for T steps, which are represented, for instance, by the iterations of a MATSim simulation. After, sequentially, one run k is selected based on the transitional performance of all existing runs and transitioned for another T steps. This process is repeated and aims at pushing forward runs with promising transitional states and ignoring others early on in the process. Once a full run completes, it is registered as the new best candidate. Subsequent *opdyts* iterations are started from the last state of the best run. (Flötteröd, 2017) leaves mostly open how to generate parameter candidates initially or before an *opdyts* iteration. In the paper, an example of a city tax simulation is provided in which hourly fees are varied in discrete quantities of ± 0.25. Hence, a full enumeration of changing the fee levels is proposed.

In more general cases, parameters may be continuous, a scenario for which no advice is given by (Flötteröd, 2017). Structurally, a mecahnism is required that (1) allows a user to sample N different parameter vectors from a proposal distribution, and (2) allows letting the user feed back exactly one winning parameter set and its objective value. These properties are fulfilled by many sampling-based black-box optimization algorithms, notably elitist ES such as CMA-(λ,1)-ES.

3. METHODOLOGY

The larger aim of the research presented here is to systematically benchmark different calibration objectives, optimization algorithms and their hyperparameters to arrive at automatically calibrated versions of standard simulations within the MATSim framework, such as the *eqasim* use case for Île-de-France.

For that purpose, a calibration/optimization framework has been developed that provides implementations for a large range of existing black box algorithms and standardized components for MATSim that allow to extract information to calculate various calibration objectives.

Optimization loop

We propose a classic optimization loop in which an algorithm proposes one or multiple parameter vectors that are executed in a simulation environment. Once the simulation runs have finished, an objective value is calculated for each run which quantifies the mismatch between simulation and reference data. The objective values are fed back to the optimization algorithm, which enters into the next iteration. The process is visualized in Figure 1.

The process is implemented in a modular Python framework, which is available online[1] and which can be easily applied to any (transport) simulator by exposing a number of actions to the framework:

[1] https://github.com/sebhoerl/boptx

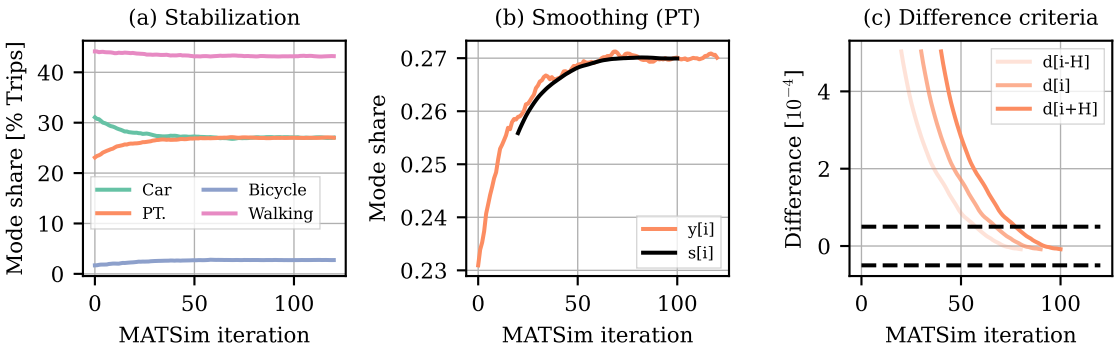

Figure 2: Mode shares in MATSim and stopping criterion

- `submit(id, parameters, information)`: A unique identifier (`id`) generated by the framework is passed to the simulation environment and asks it to start a new simulation with a vector of `parameters` chosen by the selected optimization algorithm. Furthermore, the algorithm may pass additional `information`. This field is especially useful for advanced use cases such as *opdyts* which uses the interface to start individual T-step transitions by passing the unique identifier of the previous (prematurely aborted) simulation to advance it further.
- `get(id)`: The framework requests the simulation environment to return information on the simulation run. This call should block the system until a run has finished and return the objective value.
- `clean(id)`: The framework requests the simulation environment to clean up any results of a previous simulation run because it is not needed anymore.

Currently, the framework provides a range of commonly used algorithms such as various random sampling approaches, elitist and non-elitist CMA-ES and NES, or SPSA. Especially, sampling-based methods are defined through dedicated interfaces which can then be used directly in *opdyts*.

Stopping criterion

A crucial component that is implicitly part of the calibration loop is the stopping criterion that is used to decide when a simulation has finished. Today, MATSim simulations are usually run for an a priori fixed number of iterations. Often, researchers would simulate an initial baseline case, then wait until key metrics such as the overall mode share stabilize (as shown in Figure 2a) and note down the appropriate number of iterations. However, in an automated calibration loop, simulations need to be stopped such that simulations runs with different parameters (1) are as long as necessary to be comparable and not still in a transitional state when they are stopped, and (2) are as short as possible to save computational time. (Hörl, Becker, & Axhausen, 2021) describe a computational stopping criterion that is adapted here. We propose to examine the overall mode share as the major output of an *eqasim* simulation using the following equations. We denote the share of any mode in iteration $i \in \mathbb{N}$ as $y_i \in \mathbb{R}$ and first smooth the value over a horizon $S \in \mathbb{N}$ both left and right of i:

$$s_i = \frac{1}{2S} \sum_{j=i-S}^{i+S} y_j \tag{1}$$

We then approximate the derivative of the mode share signal with a two-sided difference equation

with horizon $H \in \mathbb{N}$:

$$d_i = \frac{s_{i+H} - s_{i-H}}{2H} \tag{2}$$

We stop the simulation as soon as for any d_i three criteria hold. The horizon H is reused to check that the value d_i at the center, as well as d_{i-H} and d_{i+H} must not exceed a predefined threshold $T \in \mathbb{R}^+$:

$$\mid d_{i+u} \mid \leq T \quad \forall\, u \in \{-H, 0, H\} \tag{3}$$

Hence, if a simulation has already advanced until iteration $I \in \mathbb{N}$, valid values for d_{i+H} are only available for $i \leq I - 2H - S$, which also means that at least $2H + S$ iterations of a simulation run need to be performed. In our experiments, we require that the condition be met for all transport modes. The process is visualized for one mode in Figure 2 with $S = 20$, $H = 10$, and a threshold of $T = 0.5 \times 10^{-2} \times 10^{-2}$ requiring changes of less than 0.5% over a period of 100 iterations.

Calibration objectives

Three major objectives have been implemented in the MATSim-specific part of the framework that allow to compare the simulation with open and publicly available reference data for French use cases in terms of (1) mode shares, (2) network flows, (3) and travel times.

Information on network flows comes from the open data portal of the city of Paris, where flow information from all loop detectors are published on a daily basis. The reference network can be map-matched to the one used in the MATSim simulation, and the difference Δ_j between reference and simulated counts can be calculated for all matchable road links j.

For travel time comparison, data from Uber Movements can be used, which gives hourly travel time information between fine-grained zones. The data can be compared by aggregating car flows in the simulation on the same origin-destination (OD) zones to calculate the offsets in travel times as Δ_j for each OD pair j.

Finally, modal shares in the simulation can be compared with reference shares either from the National Household Travel survey (which is available as open data) or from the regional survey, which is available upon request. To avoid overfitting of the Multinomial Logit model that is used in our *eqasim* implementation with mode-specific constants, we are interested in shares by Euclidean distances. Figure 3 visualizes the reference data, which shows that there is naturally a strong dependency of mode use on distance, which we aim to recover in our simulations. To define the mismatch, we divide all trips in increasingly large distance bins and note down the mode shares in each distance bin. The difference between simulation and reference is denoted as Δ_j for each mode-distance bin j.

Finally, in all three cases, a single objective value $J(\Delta)$ can be defined by using a norm of the mismatch vector, e.g.

$$J(\Delta) = \sum_j |\Delta_j| \quad \text{or} \quad \sqrt{\sum_j \Delta_j^2} \quad \text{or} \quad \max_j \left\{\Delta_j\right\} \tag{4}$$

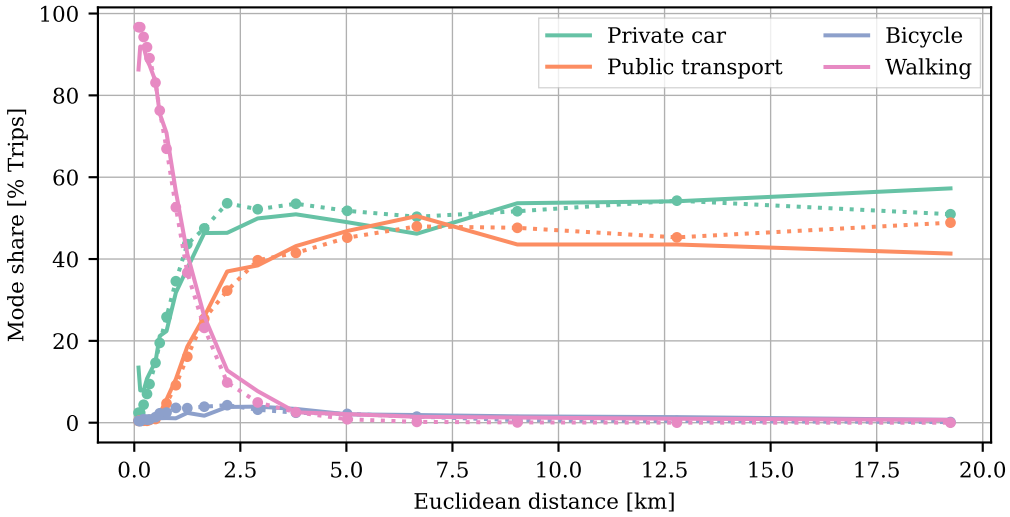

Figure 3: Mode shares by Euclidean distance between simulation output (solid) and reference data (dotted)

Parameters

To guide the simulation, we currently expose the following parameters in our *eqasim* simulations:

- **Mode choice parameters** describe the parameters of the integrated Multinomial Logit model. The model structure and (manually) calibrated parameter values have been documented in (Hörl, Balac, & Axhausen, 2019) for a case study of automated taxis in Paris.
- **Link capacities** are initially defined based on information from the underlying OpenStreetMap (OSM) network and the *pt2matsim*[2] converter. The calibration parameters are given as scaling factors that are provided per OSM road category.
- **Link freespeeds** are also initially based on OpenStreetMap and can be controlled through calibration factors per OSM road category.

While other parameters can be added, the ones listed above can easily be defined using the current state of the Python framework that is wrapped around MATSim.

4. USE CASE

The present use case has the aim to demonstrate a relevant application of the proposed framework and to perform a benchmark between the bare CMA-(λ,1)-ES algorithm and its use within *opdyts*. In both cases, the ES algorithm is configured with the same hyperparameters and such that it proposes $\lambda = 8$ samples. *Opdyts* is configured to perform transitions of $T = 20$ iterations in MATSim.

As the calibration objective, we choose mode share by Euclidean distance with L_1 norm, and we choose to calibrate the alternative-specific constants $\beta_{ASC,car}$, $\beta_{ASC,bicycle}$, and $\beta_{ASC,walk}$, the marginal utility of travel time by *car* ($\beta_{tt,car}$), and a uniform capacity scaling factor s for all road

[2] https://github.com/matsim-org/pt2matsim

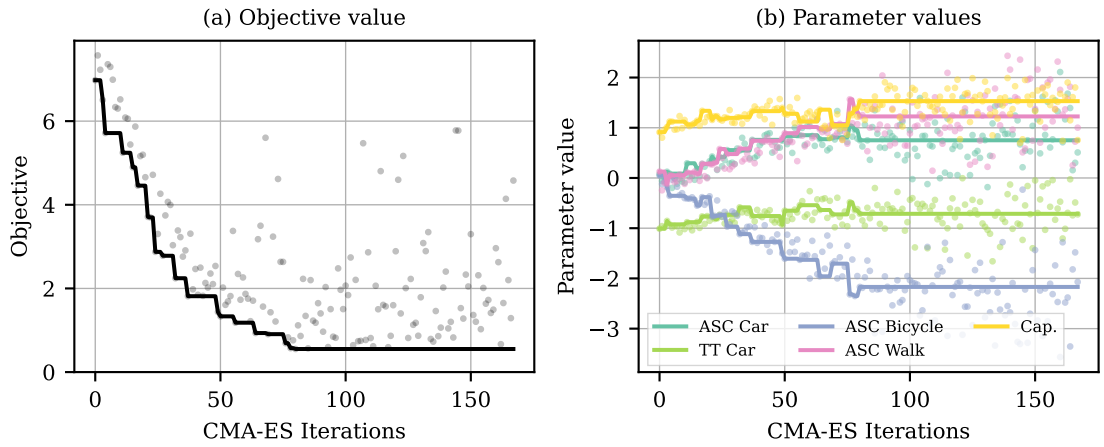

Figure 4: Objective and parameter values for elitist CMA-ES

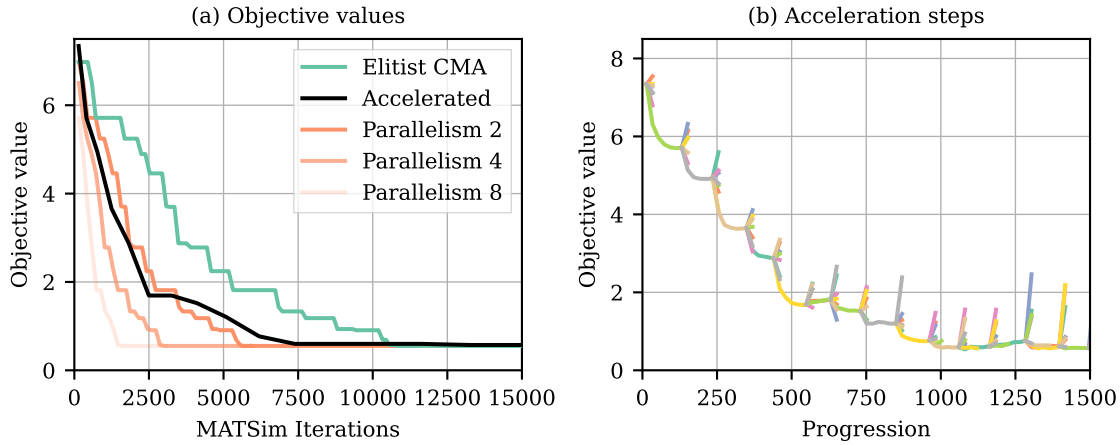

Figure 5: Comparison with accelerated calibration

links. For testing, we use a 1% sample of our synthetic population for Île-de-France and reset the initial parameter values to -0.1 For better tractability, $\beta_{tt,car}$ is rescaled by a factor of 10 and the capacity factor is initialized to 1.0.

Figure 4 shows the calibration run for the elitist CMA-ES approach. On the left, the progression of the objective value is shown, while on the right, the evolution of the current best parameter combination is indicated. We can see how the algorithm efficiently performs a search for the best parameter combination. After about 75 iterations, the objective can not be improved further. Figure 3 shows the final obtained mode shares in comparison to the reference values. Note that every iteration in Figure 4 represents eight (λ) times about 120 to 160 MATSim iterations, depending on the individual choice by the stopping criterion.

Since *opdyts* operates on partial simulations, only a comparison in terms of *MATSim* iterations makes sense. Figure 5 shows the direct comparison between the standard and accelerated case on the left. Clearly, *opdyts* is able to reduce the number of MATSim iterations, and, hence, the computational effort for the calibration by early dropping unpromising simulation runs. This behavior can be seen on the right in Figure 5 where each color represents one parameter candidate. The runs are plotted such that, from a common starting point after each *opdyts* iteration, the evaluated runs are displayed in parallel. One can see that, initially, always a clear candidate can be identified early on, while at low objective values often multiple runs are transitioned for a while before a clear winner is identified.

Finally, it should be mentioned that the results shown in Figure 5 are valid if we think of a sequential execution of both algorithms, i.e. all simulations are run after another, even when they are submitted in batch. However, the framework is able to perform MATSim runs in parallel if memory and computational power are sufficient. In fact, the experiments were performed on a machine that allows four MATSim runs in parallel. This means that the CMA-(λ,1)-ES algorithm effectively executes the first four and the second four runs of one iteration in batch. In contrary, in *opdyts*, only the first T MATSim iterations of each run are performed in parallel (as we need to know their initial performance), while the remaining transitions are necessarily performed sequentially. While we do not provide a direct comparison in runtime, as the calibrations have been performed on different machines, Figure 5 compares the process when accounting for iterations that can be performed in parallel when either two, four, or eight runs fit into memory.

5. DISCUSSION

While the general applicability of the framework has been shown, particular aspects of the present approach need further discussion and research.

We have shown that the combination of *opdyts* and a sampling-based black-box optimization algorithm such as elitist CMA-ES is a viable way to speed up the calibration process. However, the sequential nature of *opdyts* can limit performance when computational resources are abundant. It would be interesting to reflect upon ways to further increase parallelism in *opdyts*, e.g. by running multiple, potentially communicating, chains of the algorithm in parallel.

Regarding the stopping criterion, the results in Figure 5 indicate that further work is needed. While simulations should be stopped when their objective value has stabilized, some runs still exhibit slight transitional dynamics. This behavioral results from the fact that the stopping criterion in MATSim (overall mode shares) is not congruent with the objective calculated in the optimization loop (mode shares by distance). The stopping criterion in MATSim should, hence, be adapted to represent the objectives calculated in the surrounding Python wrapper.

For future work, a systematic benchmark of calibration objectives and algorithms is planned to give structured advice to researchers on how to configure the framework for specific use cases. These cases should also include applications to other simulators, such as SUMO.

Finally, additional calibration techniques can be added. Promising pathways can be found in Bayesian Optimization and surrogate modeling. Furthermore, multi-fidelity approaches, which can adaptively request information either for full-scale or computationally efficient down-sampled simulations, may speed up calibration processes.

6. CONCLUSION

In conclusion, we present a functional and ready-to-use framework for automatically calibrating well-defined large-scale agent-based transport simulations in France based on the *eqasim* framework. While the system already provides value for these specific cases, the framework can easily be extended to other simulators. In the future, it can be used to benchmark different combinations of calibration tasks and algorithms.

As a second insight, we propose a consistent way of using the *opdyts* method developed by (Flötteröd, 2017) in the context of continuous parameter calibration, by connecting it with common black-box optimization algorithms from the facility of Evolutionary Search approaches. Our initial experiments show that *opdyts* is improving the computational efficiency, but that its benefits

decrease if computational resources are sufficient to perform large parallel batches of simulations.

ACKNOWLEDGMENT

This research work has been carried out in the framework of IRT SystemX, Paris-Saclay, France, and therefore granted with public funds within the scope of the French Program "Investissements d'Avenir".

REFERENCES

Agarwal, A., Flötteröd, G., & Nagel, K. (2017). Calibration of behavioral parameters in an agent-based transport simulation.

Flötteröd, G. (2017). A search acceleration method for optimization problems with transport simulation constraints. *Transportation Research Part B: Methodological*, *98*, 239–260.

Hansen, N. (2006). The CMA Evolution Strategy: A Comparing Review. In J. A. Lozano, P. Larrañaga, I. Inza, & E. Bengoetxea (Eds.), *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms* (pp. 75–102). Berlin, Heidelberg.

He, B. Y., Zhou, J., Ma, Z., Wang, D., Sha, D., Lee, M., … Ozbay, K. (2021). A validated multi-agent simulation test bed to evaluate congestion pricing policies on population segments by time-of-day in New York City. *Transport Policy*, *101*, 145–161.

Horni, A., Nagel, K., & Axhausen, K. W. (Eds.). (2016). *The Multi-Agent Transport Simulation MATSim*. Ubiquity Press.

Hörl, S., & Balac, M. (2021a). Introducing the eqasim pipeline: From raw data to agent-based transport simulation. *Procedia Computer Science*, *184*, 712–719.

Hörl, S., & Balac, M. (2021b). Synthetic population and travel demand for Paris and Île-de-France based on open and publicly available data. *Transportation Research Part C: Emerging Technologies*, *130*, 103291.

Hörl, S., Balac, M., & Axhausen, K. W. (2019). Dynamic demand estimation for an AMoD system in Paris. In *2019 IEEE Intelligent Vehicles Symposium (IV)* (pp. 260–266). Paris, France.

Hörl, S., Becker, F., & Axhausen, K. W. (2021). Simulation of price, customer behaviour and system impact for a cost-covering automated taxi system in Zurich. *Transportation Research Part C: Emerging Technologies*, *123*, 102974.

Igel, C., Suttorp, T., & Hansen, N. (2006). A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation - GECCO '06* (p. 453). Seattle, Washington, USA.

Spall, J. C. (1998). Implementation of the simultaneous perturbation algorithm for stochastic optimization. *IEEE Transactions on Aerospace and Electronic Systems*, *34*(3), 817–823.