# Final Report - Simulation of Last Mile Logistics in Urban Areas

Richard Strunk, Yannis Matezki

January 2025

## Contents

## 1 Terminology

**API** Application Programming Interface. 10, 12

**BGP** Border Gateway Protocol. 3

**BPEX** Bundesverband Paket- und Expresslogistik. 6

**OSM** Open Street Map. 10, 12

**SUMO** Simulation of Urban MObility. 4, 5, 9, 10, 12

**TraCI** Sumo Traffic Control Interface. 5, 12

**TSP** Travelling Salesman Problem. 5

**VRP** Vehicle Routing Problem. 5, 12

**XML** Extensible Markup Language. 9

## 2 Background

### 2.1 Project Week

In mid-2024, the IT Department of the City of Munich asked their employees for challenging and socially-impactful ideas following the theme of ”digital participation” for students of the Technical University of Munich (TUM) to work on.
After choosing from these projects and forming groups in a first meeting, students from the course ”Application Project: Public Sector” were tasked to create a first pitch, and later participate in a 5-day project week, the 5th day being mostly dedicated to holding a final presentation on their work. Additionally this final report was required to fulfil the courses requirements.

### 2.2 Course Objectives

The core objective of the project week was to take the draft ideas given to actual implementation plans, considering scenario-specific factors like external stakeholders, practicality, sustainability, costs and inclusion. Notably, students did not need to follow the original challenge exactly and were allowed to deviate from it as long as the deviation was agreed upon with their city contact.
Students were also not required to finish their project, as long as their work was clearly documented.

## 3 Problem statement

The "Last-Mile Logistics in Urban Areas" problem was brought to us by Klaus Müller from the Open Source team of the cities IT-department.

### 3.1 Original

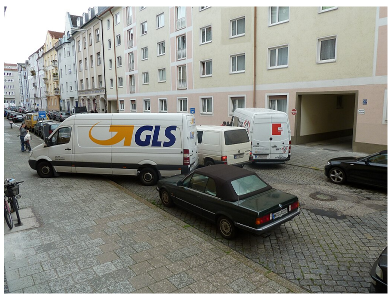


Figure 1: Two delivery vans blocking the street, picture by Klaus Müller

The problem statement presented to us by Müller was targetted at solving the issue of ineffective delivery of parcels from their last depot to the customers address, the so-called last-mile. Currently these deliveries are often done by large vans, many of which illegally stop or park on pedestrian or bike lanes, or simply block the road. Parcel carriers also usually do not cooperate with each other for their delivery services, leading to multiple carriers servicing the same area separate from each other; which in turn leads to a large number of trips being required to deliver parcels of all carriers. With good knowledge about traffic routing on the internet, the challenge essentially asked us to explore if backbone-concepts of the internet could be used to solve this issue by allowing parcel carriers to share resources and forward parcels between each other, mimicking the routing of the well-known Border Gateway Protocol (BGP) used to connect autonomous systems, like internet providers.

This also included drafting a decentralized protocol that could be used to communicate between stakeholders at time of delivery.

### 3.2 Modified

Since we were unsure if such a system could actually compete with existing models for delivering parcels on the last-mile, we decided on a different approach. Instead of formalizing an entirely new delivery model or even creating a technical protocol for it, we decided to build a vehicle simulation to analyze what changes in traffic, emissions, customer experience and cost we could expect when trying out different delivery policies. Once a simulation with acceptable accuracy exists, many different delivery models could be evaluated to objectively judge their performance on traffic impact, required work hours, emissions and more. This al-

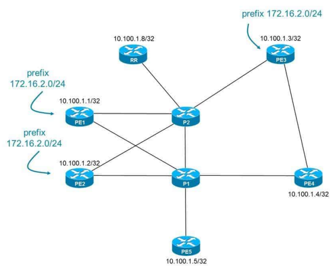


Figure 2: BGP outline, networkencyclopedia.com

lowed us to research the original problem, without
committing to a solution that was potentially not realizable. Müller agreed to our change in approach.

## 4 Methodology

### 4.1 Software

To simulate individual vehicles, we chose the open-source simulation software Eclipse Simulation of Urban MObility (SUMO), which is available under the Eclipse Public License v2.0 and GNU General Public License v2.0. The primary alternative to SUMO is PTV Vissim, effectively the industry standard. We chose not to use it, since the Student Version was strictly limited to an area of 1km x 1km and could only run simulations up to 600 seconds, which would not give us enough data to compare the different delivery models.
SUMO allows simulating thousands of individual vehicles and actors with many configurable parameters and actions like lane-changing strategies, distance between vehicles, impatience, or even automated driving with vehicle communication.

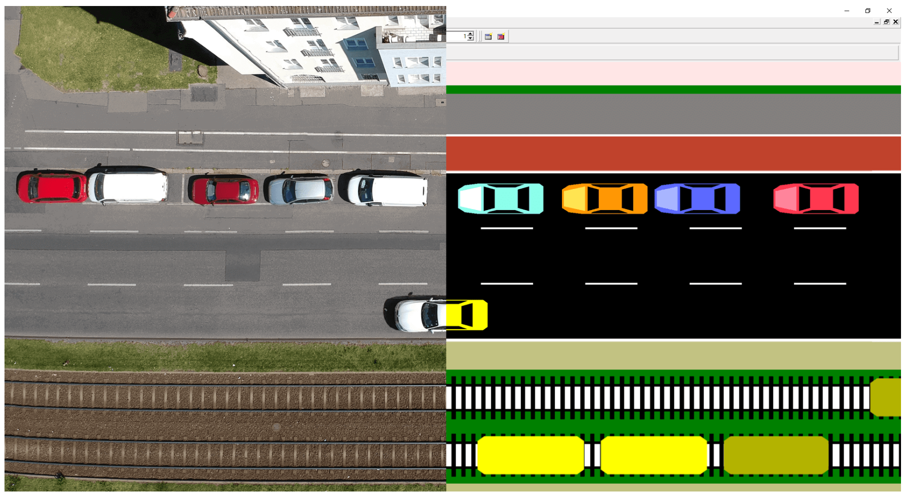


Figure 3: Using Sumo to model traffic

But working with our simulator SUMO alone was not sufficient, since we needed finer control about individual vehicles and routing decisions to correctly model our delivery vans and their stops. For that reason we introduced a custom simulation backend written in Java. This offers a good trade-off in terms of performance and complexity, and comes with a stack of high-quality libraries we can use. It was also the programming language that we were both most

familiar with out of the ones that provided bindings for SUMO.
To "connect" our Java backend to SUMO, we used the Sumo Traffic Control Interface (TraCI) Java bindings provided by the SUMO project, which were unfortunately difficult to navigate, understand and work with in general. The documentation for TraCI was sparse and a lot of our time was spent on finding the best way to interact with the vehicles in our simulation, as well as insert our custom logic into it. There was the option to avoid TraCI and run the simulation directly from out code, but that option was incompatible with the GUI delivered with SUMO, so there would have been no easy way of verifying our simulation visually.

In addition to SUMO we made use of the open-source Java library jsprit [1] which offers a toolkit to solve TSPs and VRPs configured in code. A Travelling Salesman Problem (TSP) describes the problem of visiting all points in a set under minimal cost, which is pragmatically very expensive to solve perfectly, so our solution is only an approximation, using the distance between the geographical points as our cost. A Vehicle Routing Problem (VRP) is a generalization of a TSP and describes the problem of finding an optimal set of routes to visit all points with a fleet of vehicles.
Jsprit is available under the Apache License v2.0 and allows us to solve these problems iteratively, using a defined amount of steps to constantly improve an initial solution. We utilized jsprit to find routes for our delivery vehicles in both last-mile models.

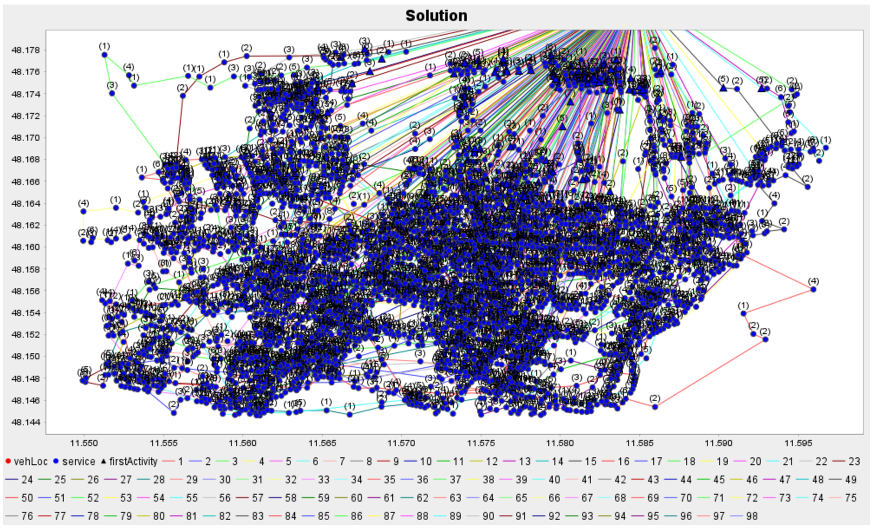


Figure 4: An iterative approximation to a VRP solution by jsprit

[1] https://github.com/graphhopper/jsprit

## 4.2 Model

### 4.2.1 Area

We choose a rectangular area of northern Munich for our simulation, encompassing Maxvorstadt and most of Schwabing, since we are familiar with the area and it seemed a good example of a densely populated urban area with a variety of traffic scenarios like larger and smaller streets, as well as areas like parks that are more accessible by bicycle. The rectangle stretched from the B 2R outer ring street in the north to Odeonsplatz in the south, and from the middle of Olympiapark in the west to include a section of Englischer Garten in the east, leading to a total area of roughly 13 square kilometers.

### 4.2.2 Parcel Demand

To estimate the parcel demand for our area we first estimated the number of residents using city statistics on population density, as well as overall population of the districts in our map area, arriving at an estimate of about 130000 residents.
Using the 2021 statistics by the Bundesverband Paket- und Expresslogistik (BPEX) on yearly customer parcel deliveries by city[2] we calculated that the average Munich resident receives about 37 deliveries per year. Considering that there is about 300 days in a year where parcel deliveries can take place due to public holidays or Sundays, we calculated that roughly 16000 parcels are delivered in our selected area per day.
To model the demand onto our map, we gathered a list of addressable locations within our map area and randomly distributed the parcels to these addresses. This could be done more accurately by taking business customers, as well as detailed population density into account. Alternatively, real delivery data could be used to have an accurate representation of demand.

### 4.2.3 Traffic model

In order to simulate what we called "background traffic", we both intuitively thought of a M-shaped curve, where congestion spikes early in the morning and later in the evening. In order to verify this data we looked to TomTom, a once popular navigation provider. On their website, they provided us with rudimentary traffic information for daily congestion in the Munich area that pretty much aligned with what we already expected[3].

[2]`https://bpex-ev.de/download.html?getfile=BIEK_Kompendium_2021_Regionale_Verteilung_Sendungsvolumen.pdf`
[3]`https://www.tomtom.com/traffic-index/`

In order to model the expected traffic at any time of the day, we choose the function

$$max(sin(b * sin(a * x + d) + c), 0) + e$$

with parameters $a = -5, b = -2.6, c = -0.1, d = 1.125, e = 0.01$ that were chosen by trial and error to match the M-shaped curve. We added parameter $e$ to have a small amount of background traffic during the night. We then multiply the result of this function by the number of vehicles we assume during rush hours, for cars 3000 and for bikes 1000. So whenever the number of driving actual cars or bikes in the simulation is less than our curve, more are to be added by our Java simulation backend. This concept is a big oversimplification, which fails to capture commuting or return trips. It assumes all drivers drive to a location uniformly distributed in our target area, which is hardly the case. Also, this simple model failed to take commuters from outside our map into account. To curb at least that problem somewhat, we choose 6 designated entry points and 4 designated exit points, that for cars have a 30% chance to be selected as the beginning or end of their trip. In practical observation this worked quite well, but of course added just more oversimplifications. While in theory we could have definitively added more realistic background traffic, we were quite limited on time and, since the overall result lined up well with real-life observations, didn't see it as a high priority. Our main objective still was comparing parcel services, so we assumed any bias in the background would affect all models roughly equally, which is debatable since our parcel models were not always using the same mode of transportation.

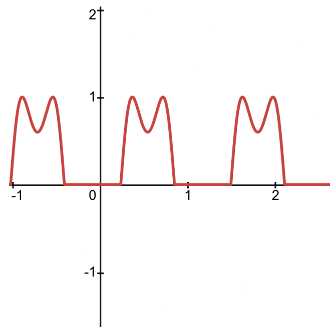


Figure 5: X from zero to one represents 0 to 24 hours, after which we go back to zero

### 4.2.4 Delivery durations

Additionally, particular care was taken when modelling delivery duration, which translated to the duration of a stop in our simulation. We tried to get a fairly realistic model which included factors like stairs the driver might have to climb, the number of packages, and the fact that it takes longer if a package is to be delivered to someone living higher up in the building. We assumed for simplicity that all packets were uniform in shape and weight and that no delivery drivers refused to carry a package to higher floors. We also estimated that drivers would wait an average of two minutes before aborting their delivery attempt. We did not include package weight or size in our estimation, since we did not have any good source of data and could therefore not make a reasonable assumption about their averages. A possible improvement could be including them into the process affecting all other random sub-processes. We made the assumption that 40% of deliveries would not involve climbing stairs, and the other 60% were uniformly distributed between one and six floors, quite common for Schwabing.

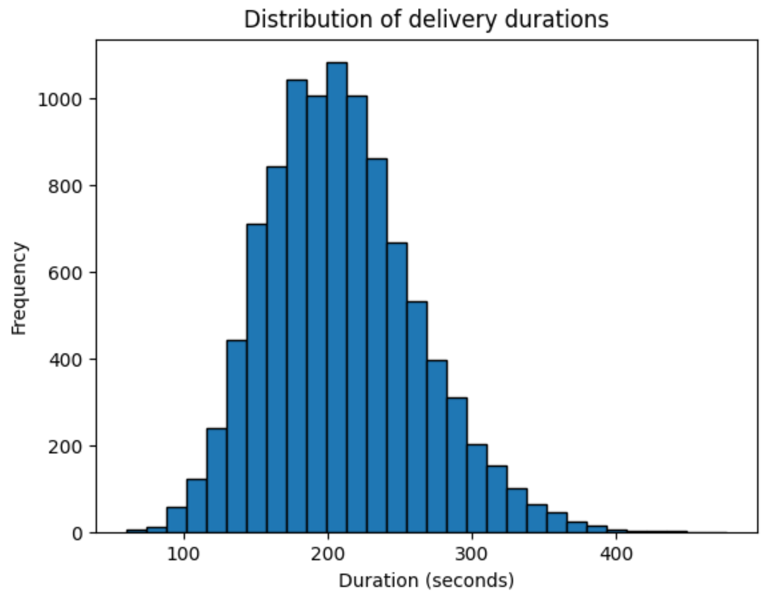
We assumed that it takes them about 15 seconds per floor on average, plus two and a half seconds per package on average. We also made the simplification that they take the same time getting down the stairs as getting up, including package duration penalties in the trip down to model potential exhaustion.

$$\begin{aligned}
N_{stairs} &\sim \begin{cases} round\_next(U(1,6)) & U(0,1) \geq 0.4 \\ 0 & else \end{cases} \\
T_{parking} &\sim Exp(12) \\
T_{approach} &\sim \mathcal{N}(30, 10) \\
T_{stair} &\sim \mathcal{N}(15 + 2.5 \times N_{packets}, 5) \\
T_{stairs} &\sim N_{stairs} \times T_{stair} \\
T_{wait} &\sim \begin{cases} \mathcal{N}(20, 5) & U(0,1) \leq 0.95 \\ \mathcal{N}(120, 5) & else \end{cases} \\
T_{loadunload} &\sim \mathcal{N}(60 + 20 \times N_{packets}, 20)
\end{aligned}$$

The total duration was then sampled as:

$$T_{total} \sim T_{parking} + 2 \times T_{approach} + 2 \times T_{stairs} + T_{wait} + T_{loadunload}$$

We then drew 10k delivery times from $T_{total}$ and plotted them to the corresponding histogram:



We compared the data to a news article from a delivery worker describing their work[4], and we think it roughly matches what we see - despite the article coming from an american worker. Take note that we nudged these values for different modes of transportation later on, for example less parking time for our bikes.

### 4.2.5 Regular Van Delivery Model

The next step was to replicate the most common delivery model used in nearly all cities today: large delivery vans transporting packages directly from a storage hub (in our case, located outside the city) to customers, following the shortest overall route. We simulated three different carriers, responsible for 50%, 25%, and 15% of total parcel deliveries, respectively. These proportions roughly align with the market share of Germany's three largest carriers - DHL, UPS, and DPD - while smaller competitors were excluded. Each delivery van was estimated to have a maximum capacity of 250 packages. Given the total number of parcels to be delivered daily, we calculated the required number of vans. However, we quickly found that operating with minimal excess capacity led to inefficiencies and suboptimal routing. To address this, we increased the number of delivery vehicles by 30% beyond the calculated minimum, assuming that delivery companies already maintain spare vehicles for operational flexibility.

We then took these randomly generated delivery locations and clustered them together when they were less than 20 meters apart. The location of the stops and the maximum number of vans with their respective capacity was used to configure jsprit to achieve a minimal total distance for all deliveries for each carrier. The routes generated by jsprit were then converted into an Extensible Markup Language (XML) file that could be loaded into SUMO to automatically dispatch the delivery vans at specific times within the simulation.

### 4.2.6 Micro-hub Bicycle Delivery Model

The micro-hub model utilizes small hubs around the district that are serviced in the morning. From there the parcels are distributed in a decentralized way with smaller vehicles holding much fewer parcels each. In our case, we opted for cargo bikes with a maximum capacity of 40 parcels. We allowed parcel carriers to cooperate by sharing hubs. However, given the short delivery trips each vehicle makes from the hubs, non-cooperation would have only a marginal impact on our model. This is because each vehicle must complete multiple trips, often delivering to stops that are in close proximity to previous ones..
To choose the locations of our micro-hubs we used the k-means clustering algorithm on our list of all addressable locations in the area to find 20 cluster centers that would service all locations belonging to that cluster. The number 20 was chosen, because that would mean that each micro-hub could be served

[4] `https://medium.com/the-post-grad-survival-guide/to-jeff-bezos-from-an-amazon-delivery-driver-5ccf39d5df7d`

by a single truck in the morning, based on the number of parcels assigned to it. We then applied the same procedure for modeling stops as in the regular delivery model, with two key differences: a smaller clustering radius of 5 meters and a shorter average stop time. The reduced stop time was based on two factors—cargo bikes do not require parking searches, and unloading is slightly faster since fewer parcels need to be sorted/searched-through to find the correct one at each stop.

### 4.3 Maps

The network for our traffic simulation (the map) is generated from Open Street Map (OSM) data, which can be downloaded through their public Application Programming Interface (API). The OSM map data is highly detailed, including streets with lane and turning restrictions, bike lanes, footpaths, points of interest, and basic geometry for rendering vegetation and buildings. It also contains much more information that was not utilized in our model. We chose to use OSM data because it is entirely free, open, and highly accurate. To convert the OSM file of our selected area into a SUMO network file, we used the included *netconvert* tool. We opted to discard railway and other public transportation data, as we assumed they had minimal impact on our model. Since pedestrians and public transport users contribute minimally to traffic congestion, excluding them allowed us to focus our limited time on more relevant factors.

## 5 Results

While we were only able to simulate two different last-mile policies during our limited time, we did manage to find some interesting differences in the data. To describe these differences, the Regular Van Delivery Model will from here out be referred to as Model A and the micro-hub bicycle delivery model will be referred to as Model B.
In Model A, the simulation generated a total of 81 trips to service our area, which were all done in parallel. Over these 81 trips, the vehicles drove 2183 km within 588 h, resulting in $CO_2$-Emissions of roughly 2000 kg, calculated with emission class LCV_diesel_N1-III_Euro-6d. Model B on the other hand required 381 smaller trips from the hubs and an additional 40 trips to service the hubs from outside our area. This resulted in a total route length of 981 km by bicycle and 123 km by truck, requiring 564 h of work and causing roughly 50kg of carbon dioxide to be emitted.
The much lower route length is achieved by the distribution of shared decentralized hubs, causing addresses to be served only once each day, instead of multiple times by each carrier.
We also estimated the parcel carrier cost in a very simplified way by calculating wages at 15€/h and fuel at 1.50€/l, ignoring all other costs. This lead us to a total cost of 9744.40€ for Model A and 8485.20€ for Model B. This reduction in cost is mostly caused by the smaller amount of work required to deliver all

Table 1: Results of a simulated day from 07:00 to 18:00

| | Model A | Model B (delivery + setup) |
|---|---|---|
| Trips | 81 | 381 + 40 |
| Trip duration ("Manhours") | 588h | 553h + 11h |
| Trip route length | 2183km | 981km + 123km |
| $CO_2$-Emissions | 2000kg | 0kg + 50kg |
| Average wait time for other traffic participants | 527s | 503s |

parcels in Model B. This is a very simplified calculation and an improvement in routing in either model could lead to major changes to the calculated cost. We therefore think more research needs to be done before changes can be implemented, even though our simulation shows that Model B is at least feasible. We also observed a reduction of average accumulated wait time for other traffic participants from 527 s in Model A to 503 s in Model B, due to reduced street congestion. It is also worth mentioning that vehicles by carriers with a higher market share in Model A are overall more effective, since they have smaller distances to drive between each delivery and therefore spend less of their time driving or in slowed traffic.
Our data also showed that for a few bicycle trips in Model B, the road time was much greater than the stop time due to large distances to the micro-hub. It is likely that these trips could be more effectively served by faster modes of transportation like vans. At last, we find it important to point out that we believe our findings not to be statistically significant, as we only ran the simulation once for each delivery policy. However, we still find the results interesting and think they warrant further research.

## 6 Impacts

Our project gives some insights into traffic management issues in the city, and could be used as a beginning to further investigate alternative solutions for last-mile logistics. It shows that the city should continue pushing pilot projects like the "Radlogistik-Hub" in Tumblingerstraße, due to the many advantages a co-operative micro-hub system could bring to the city.
In regards to digital participation, our project shows, that open data and open-source software is incredibly beneficial for a city and its population. Easy access to public data like accurate maps or traffic data opens up opportunities for citizens to work on research that can lead to the understanding and solving of problems the city is facing. The city should work on or incentivise the creation and upkeep of such public resources since it comes with higher government transparency, trust, self-empowerment for citizens and ultimately more participation in city issues and democratic decisions to solve them. The Open Data Hand-

book [5] by the Open Knowledge Foundation further dives into the advantages and disadvantages of open data and how it can be done in legally and socially safe manner.
Open-source software comes with very similar benefits and allows the city to use existing software to reduce development time while allowing for better collaboration between cities or even contributions from volunteers. Open-source projects encourage knowledge and skill sharing among citizens, creating a ecosystem that fosters innovation. Furthermore, the transparency of open-source code enhances trust in digital solutions, as anyone can inspect, modify, and verify the software's functionality.

## 7 Future Research

Further work is required in a few points to allow political decisions to be made based on a vehicle simulation of an urban environment. Most importantly the accuracy of the simulation could be improved greatly with more manual labor or open data. Correct traffic light programming, as well as real congestion data, for example crowd-sourced from citizens in the simulated area could lead to a much better depiction of real traffic conditions than our random simulated demand can.
Additionally, the delivery routes used by our simulated parcel carriers are likely not optimal, since only the air line distance between delivery points was taken into account when calculating delivery routes. This could be augmented with map data to use the actual distance to be driven by vehicles of different types to find a more optimized solution to the VRP. We tried this approach, but the computational cost was too large when using the routing implementation in our simulator.
We also believe, that simulation of other last-mile models, or a refined version of our model that better takes into account the time required to move parcels between different vehicles could lead to more insight, as well as running the simulation multiple times to gain further statistical significance.

## 8 Contributions

Before starting the work on the simulation the delivery models were chosen and formalized together, researching relevant data required to correctly model our traffic and delivery demand.
Matezki discovered SUMO and the possibility to import maps from OSM into it, Strunk got vehicle routing working properly via the TraCI Java API and added the sinus-sinus congestion curve to model rush-hour traffic.
Matezki added the vehicle routing library jsprit to solve the underlying VRP and imported the resulting vehicle data into the simulation, while Strunk worked on improving the efficiency of the process to allow for faster iteration times.

[5] `https://opendatahandbook.org/`

Strunk performed statistical analysis on the resulting trip data and created appealing visualizations for the final presentation, while Matezki implemented the micro-hub model with clustering and solving the resulting bicycle routes. All in all we consider the amount of work per contributor equally distributed, with all contributors working significantly more than the recommended on-site time. The project code and the git history are available at `https://github.com/yannismate/LMLSimulations`.