Geoinformatica (2023) 27:737–758
https://doi.org/10.1007/s10707-022-00485-y

# Test-data generation and integration for long-distance e-vehicle routing

**Andrius Barauskas[1] · Agnė Brilingaitė[1] · Linas Bukauskas[1] · Vaida Čeikutė[1] · Alminas Čivilis[1] · Simonas Šaltenis[1]**

Received: 14 October 2021 / Revised: 17 November 2022 / Accepted: 20 December 2022 /
Published online: 26 January 2023


## Abstract

Advanced route planning algorithms are one of the key enabling technologies for emerging electric and autonomous mobility. Large realistic data sets are needed to test such algorithms under conditions that capture natural time-varying traffic patterns and corresponding travel-time and energy-use predictions. Further, the time-varying availability of charging infrastructure and vehicle-specific charging-power curves may be necessary to support advanced planning. While some data sets and synthetic data generators capture some of the aspects mentioned above, no integrated testbeds include all of them. We contribute with a modular testbed architecture. First, it includes a semi-synthetic data generator that uses a state-of-the-art traffic simulator, real traffic volume distribution patterns, EV-specific data, and elevation data. These elements support the generation of time-dependent travel-time and energy-use weights in a road-network graph. The generator ensures that the data satisfies the FIFO property, which is essential for time-dependent routing. Next, the testbed provides a thin layer of services that can serve as building blocks for future advanced routing algorithms. The experimental study demonstrates that the testbed can reproduce travel-time and energy-use patterns for long-distance trips similar to commercially available services.

✉ Simonas Šaltenis
simonas.saltenis@mif.vu.lt

Andrius Barauskas
andrius.barauskas@mif.vu.lt

Agnė Brilingait
agne.brilingaite@mif.vu.lt

Linas Bukauskas
linas.bukauskas@mif.vu.lt

Vaida Čeikutė
vaida.ceikute@mif.vu.lt

Alminas Čivilis
alminas.civilis@mif.vu.lt

[1] Institute of Computer Science, Vilnius University, Vilnius, Lithuania

**Keywords** Semi-synthetic data generation · Testbed · Electric vehicle · Long-distance EV routing · Time-dependent road network

## 1 Introduction

Like many areas of human activity, transportation is undergoing a profound transformation influenced by the continued digitalization of all aspects of the field. The change is driven by the emergence of new automotive technologies and business models, such as electric and autonomous vehicles, and ridesharing. For example, the efficiency of a fleet of autonomous electric vehicles will be highly dependent on effective routing and scheduling algorithms. Such algorithms will, in turn, depend on data-driven travel-time and energy-use predictions. Furthermore, as real-world routing problems are often formulated as multi-objective optimization involving multiple constraints, the optimal algorithms are intractable; thus, only heuristic algorithms are possible [1]. Only extensive experimental studies on large datasets and workloads can verify the efficiency and efficacy of such algorithms.

Real test datasets are readily available only at very few commercial service providers such as Google, TomTom, or HERE. Furthermore, making any such data sets public is often burdened by privacy concerns. This means that research on advanced routing algorithms in academia either has to resort to simplistic synthetic test data and workloads or, more often, is not tackled at all, looking instead for more accessible problems.

Research studies that do explore advanced routing problems expend much effort to prepare their experiments. For example, to implement Eur-PTV and Ger-PTV benchmarks [1], the following data from different sources were preprocessed and integrated: road network, elevation information, energy consumption, road traffic data, and locations of charging stations. In another study [2], to perform real-world experiments, the simulation framework [3] was extended, and traffic patterns were taken from LuST Scenario data [4].

This paper aims to do the tedious but necessary legwork for the road-network algorithms community. While there are a few traffic simulators and general-purpose spatial and graph data generators, we provide, to the best of our knowledge, the first comprehensive data generator and testbed for experimentation with advanced routing algorithms, particularly, algorithms for e-vehicles (EVs).

Consider a long-distance EV routing query to understand the richness and complexity of the data required by real-world routing algorithms. Such a query has to take into account the predicted traffic to estimate both the expected travel time and the expected energy consumption. Also, to plan charging stops, the algorithm has to use the latter information as well as information about the availability and power of chargers. Both the expected traffic and the expected availability of chargers are *time-dependent* (TD). Traffic volume and charger demand have peaks depending on the time of day, weekday, month, season, or other factors. Furthermore, we argue that any realistic long-distance routing system has to work with the inherent uncertainties of predictions. Thus, the travel time, the consumed energy, and the time waiting for charging are all modeled as *intervals* of expected values.

The contributions of this work are threefold. The paper contributes with a modular architecture and a data preparation workflow to generate realistic semi-synthetic EV-specific TD traffic data that captures uncertainty. Next, we provide a layer of services on top of the generated data to be used as building blocks of future advanced routing algorithms. Another contribution is identifying necessary data sources and tools, and the main lessons learned by integrating them. Finally, the experiments indicate that the proposed environment provides

data patterns similar to commercial ones. Hence, the environment enables testing of the TD routing of EVs.

This paper extends our preliminary work [5] by providing more comprehensive descriptions of all the stages of data generation, covering an important topic of ensuring that the so-called FIFO property holds for the generated data, and reporting on additional experiments.

The remainder of the paper is structured as follows. Section 2 gives a brief overview of related work. Section 3 introduces the theoretical background, testbed architecture, and the main steps of semi-synthetic data generation. Section 4 gives more specific implementation details. Section 5 presents the experimental evaluation of the testbed. Finally, Section 6 concludes the paper and discusses possible future work.

## 2 Related work

Developing a testbed for EV routing requires modeling traffic data and estimating travel costs (time and energy). Data for such estimation can be either retrieved from the existing commercial platforms, provided by other research projects [6], or generated using traffic simulators. We survey them in turn.

Several studies apply statistical and machine learning methods to forecast travel time and future congestion using Google Maps data. Traffic conditions on monitored locations are identified by capturing a traffic layer image and identifying *color* data on road segments: green, yellow, red, and dark red [7, 8]. Another approach, proposed by Zafar and Haq [9], is to use an Estimated Time of Arrival. They classified traffic states into five categories: smooth, slightly congested, congested, highly congested, and blockage. A substantial amount of data has to be collected to apply machine learning algorithms. Pramanik et al. [8] monitored traffic conditions for selected intersections and roads with 30-sec intervals for six months; Zafar and Haq [9] gathered data for three months.

Another possible approach to forecast traffic situations is to use historical GPS traces. Sapre et al. [10] employed raw GPS traces available in OpenStreetMap (OSM) to create a mobility model of the city. The model of traffic demand is the percentage of trips between two regions of the city to the total number of available trips. Traffic demand was scaled up to mimic the city vehicle population and was used to run a simulation in SUMO (Simulation of Urban MObility) [11]. Simulating such synthetic traffic data using the SUMO simulator and available GPS traces is a complex task. One has to preprocess GPS traces and OSM data before integrating them with SUMO. Raw GPS data preparation consists of several steps: identification of separate trips, map-matching GPS points to the nearest point on the road, and filtering of the trips. OSM data were adjusted by adding orientation and order to each segment. The limitation of this approach is GPS data availability, especially if traffic has to be generated for the whole country or continent, and data preprocessing.

This paper uses the Google Maps service for speed calibration purposes. The work focuses on open traffic data generation using a traffic simulator and congestion estimation based on typical traffic profiles.

Brinkhoff [12] pioneered a framework to generate moving objects on a road network. It allowed the user to define various properties of object classes, e.g., the number of moving objects and speed limits of road segments. Nevertheless, the framework did not consider microscopic traffic models, e.g., traffic lights, lane changing, and car following. Therefore, produced vehicle movements might be unrealistic. In contrast, the open-source microscopic traffic simulation tool SUMO [13] handles large-scale road networks by design. It

supports different types of moving objects, e.g., pedestrians, cars, and trains; it models lane-changing and car-following behavior. The tool is flexible and highly configurable, providing many features to generate desired and close-to-real traffic data. Another publicly available microscopic traffic simulator introduced by Yu et al. [14], GeoSparkSim, integrates the Spark-based spatial data management system GeoSpark [15]. Due to its scalability, data preparation and simulation time is superior to other existing solutions, including the SUMO simulator. SMARTS [16] traffic simulator allows data calibration by adjusting simulation parameters. It is recommended to run the simulation with different adjustments until the simulated data matches the actual data. AlDwyish et al. [17], when using the SMARTS simulator for navigation services, retrieve authentic traffic snapshots periodically and use them to validate and calibrate the simulation.

In this work, initial traffic data was generated using SUMO due to a wide range of available features, including the Vehicle Energy Model, needed to evaluate energy consumption along the given route.

Long-distance EV routing queries should consider battery recharging, and the data model should include charging stations. Baum et al. [18] augment the original road network graph with the charging-station sub-graph. Charging stations are represented as vertices, and each such vertice has a predefined state of charge range. This feature allows the implementation of charging restrictions caused by technical charging-station characteristics (e.g., regular or battery-swapping station) or user preferences. Edges in the charging-station sub-graph are energy-optimal paths between charging stations.

When an EV arrives at a charging station, it cannot be assumed that it will always find an available charger. Live and historical charger availability data will become an essential part of the data foundation for advanced route-planning algorithms. Thus, our Charging Stations component calculates charging and waiting time intervals at a particular station. Each charging station has a set of chargers with charging functions, a TD availability profile, and a set of coordinates.

## 3 Semi-synthetic data generation and testbed API

First, we present the theoretical background, architecture, and the main functionality of the proposed testbed. Then, Sections 3.3–3.6 describe the generation of semi-synthetic data. Finally, the FIFO property is discussed.

### 3.1 Background

A road network is based on a directed graph $G = (V, E)$, where $V$ and $E$ represent vertices and edges, respectively. Each edge $e \in E$ is a pair $(u, v)$ where $u$ and $v$ are the start and end nodes of the edge, $u, v \in V$. The edge represents a road segment $seg$, i.e., a polyline along the road with geographic properties. Notation $|seg|$ is used for a segment length. Path $P$ is a sequence of adjacent edges representing the trip from start $s$ to destination $d$. $P = \langle e_1, \ldots, e_N \rangle$, where $\forall i < N : e_i.v = e_{i+1}.u \wedge s = e_1.u \wedge d = e_N.v$.

Electric vehicle $EV$ has three essential features: $b$ is an $EV$ battery capacity, $f_c$ is a charging function, and $f_e$ is an energy consumption function. The energy consumption along the trip that started with the state of charge (SoC), $S_c$, is modeled as interval $\Delta c$ as it is an uncertain quantity.

In the time domain $T$, notations $\Delta t$ and $\bar{t} = (t^{\vdash}, t^{\dashv})$ represent duration interval and time interval of the day, respectively. For example, $\Delta t$ =(15 min, 17 min), and $\bar{t}$ =(13:00, 13:30).

The charging is available at charging stations. A set of charging stations is represented by $CH$. Waiting and charging times of the charging station $ch$ are represented by time intervals $\Delta t$, and they depend on the daytime intervals, $\bar{t}$.

The use of intervals to model travel time, energy use, and waiting time stem from a vision of a routing or trip-planning system that relies on data-driven prediction. In the prediction module of such a system, intervals would be computed as confidence intervals according to some confidence threshold. Routing and planning algorithms would then conveniently work with these predicted intervals.

### 3.2 Testbed architecture and functionality

As described in the introduction, generating and managing the test data calls for a multi-component architecture (see Fig. 1).

First, driving speed depends on the traffic at a particular time. Therefore, the TD Traffic Information component requires Traffic Simulation data and TD Traffic Statistics to define the parameters of road edges. Second, energy consumption depends on the physical road properties and the EV type. Thus, the Energy Consumption component includes elevation data and the consumption function that uses the EV properties as its parameters. Finally, long-distance EV routing requires charging stops along the road. Hence, the component of Charging Stations is supported by TD availability data of charging stations and charging function that uses the parameters of EV type.

While the main contribution and focus of this work is the generation of semi-synthetic data, we also propose a thin layer of services. Such services query and aggregate the data and can be used as the building elements of advanced routing algorithms. Figure 2 presents five API functions with optional parameters marked by $\star$. Function *findPath* uses a TD router to construct a path $P$ from the start $s$ to the destination $d$ and estimate the expected trip duration interval, $\Delta t$, and the expected energy consumption interval, $\Delta c$, when leaving sometime during the $\bar{t}$ time interval. The starting time is given as an interval, which is helpful if the function computes a leg of a longer route. For example, if a leg is a path between charging stops, then for the second leg and later legs, the departure time is usually uncertain, represented as an expected interval. If the initial SoC of the EV battery, $S_c$, is

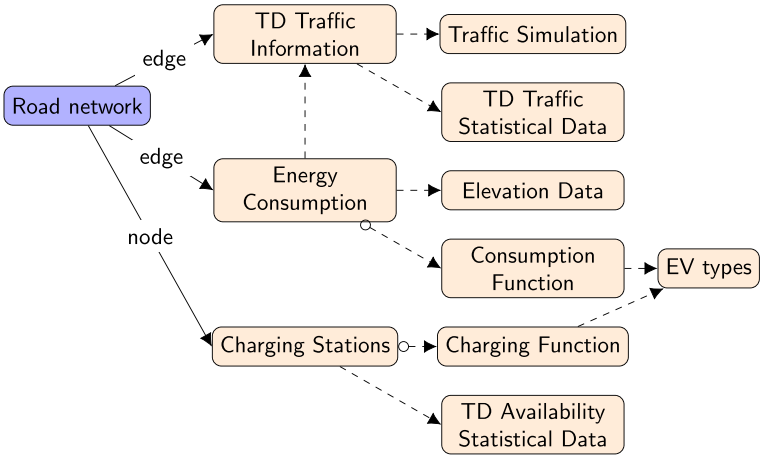



**Fig. 1** Components of the testbed architecture

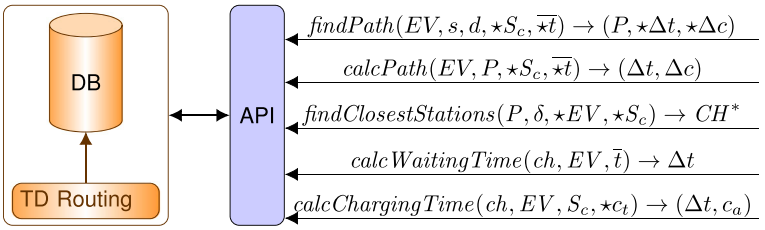



**Fig. 2** Testbed API

given, the returned $\Delta c$ is the final expected interval of the SoC of the battery rather than the consumed energy.

Function *calcPath* is calculates the same travel estimates on an already known path $P$. Function *findClosestStations* returns a set of charging stations $CH^*$ containing the stations within a Euclidean buffer $\delta$ around path $P$ and reachable by $EV$ when starting on the path with $S_c$. If parameters $EV$ and $S_c$ are not given, the function simply returns all stations within a buffer of the path. Finally, functions *calcWaitingTime* and *calcChargingTime* return the waiting-time and charging-time intervals $\Delta t$ at charging station $ch$ for $EV$. A waiting time interval depends on the daytime interval when the EV reaches the charging station. Also, a charging time interval depends on the SoC before starting the charging process. The target SoC, $c_t$, can be provided and the achieved SoC, $c_a$, is returned.

### 3.3 Map and traffic data

The traffic data preparation process is shown in Fig. 3, where the process steps are in the leftmost and rightmost columns of the figure, while the resulting data is modeled in the second and third columns of the diagram. During the road network (RN) preparation, first, map data is filtered, leaving only car roads. Next, the road network graph is made routable (a directed graph), and finally, routable network segments are augmented with length data and free-flow speed data (speed-limit data). This information is available at varying degrees in most existing map data sources and can be used to build a static road-network graph. Next, two primary sources are used to generate semi-synthetic TD weights of road-network segments (TD segment). TD traffic statistical data for a given region describes how traffic at large changes relative to the time of day. This property is essential to derive a traffic profile

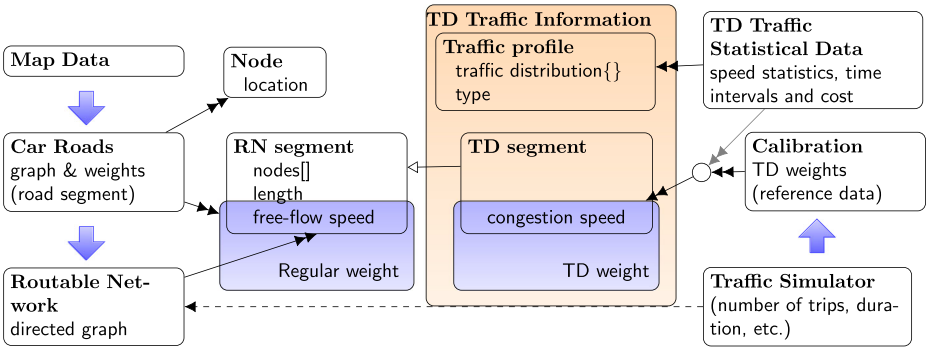



**Fig. 3** Traffic data preparation

(described in more detail later). Then, network segments are augmented with congestion speed data, i.e., either actual statistical data, if available, or synthesized data generated by the Traffic Simulator. Finally, the results of simulations are calibrated using commercial traffic data providers.

We use maps from the geographical information participatory project OpenStreetMap (OSM, [19]) for our testbed. The necessary region is directly downloaded from the repository [20]. The OSM map data includes geographical information, while traffic modeling and routing require just road-network data. An associated command-line tool `osmfilter` is used to manipulate and process large raw OSM data files and filter them for specific tags.

OSM network is not routable and has to be converted into a graph representation. The Java-based application `osm2po` [21] performs the conversion and can work on continent-wide networks. It outputs SQL INSERT scripts for a relational database management system. Executing the generated SQL scripts creates the edge and node tables containing the routable network data, such that each edge has links to source and target nodes in the node table.

### 3.4 Traffic data simulation and calibration

As shown in Fig. 3, semi-synthetic TD segment weights are composed of three types of information: edge-specific minimum traffic speed, edge-specific maximum traffic speed, and region-wide TD traffic volume distribution. Given the time of day, they are used to calculate edge-specific traffic speed as a weighted average of the minimum and the maximum traffic speeds of a segment. In our testbed, the maximum speed is the free-flow speed from OSM. The minimum speed is derived from congestion modeling using the open-source traffic simulator SUMO [11], and TomTom is the source for the TD traffic volume distribution. First, we discuss the main challenges of using SUMO and then cover TomTom traffic volume distribution.

SUMO takes a routable network as data input for traffic simulation and augments it with simulated traffic data. The testbed routable network is fed to SUMO using the `netconvert` tool. The simulation output is a travel time for each segment on the routable network during peak hours.

To perform a simulation using SUMO, the whole map is divided into regions. Each region is simulated separately as the population size and, consequently, the number of trips differ. The random traffic generation method [22] of the SUMO tool *randomTrips* is used. This method allows choosing different weights affecting the probability of selecting a segment for routing. Segment length is used as a weight; thus, dense regions like city centers get more traffic. Finally, the number of trips is calculated proportionally to the region's population size and distributed in an interval from 0 to 3600 seconds using the SUMO *randomTrips* tool.

Region-wide TD daily traffic volume distribution can be sourced from the traffic data providers such as TomTom. TomTom's Traffic Index data contains detailed historical data on traffic congestion levels worldwide. Congestion level shows travel time increase during a specific hour of the day compared to the free-flow situation. The year 2019 and a specific testing weekday, Tuesday, were chosen for our testbed. Any other weekday except Saturday and Sunday is appropriate for such a model. In accordance with transport-modeling good practice, Monday and Friday are also excluded due to different traveling patterns on these days. Figure 4 presents aggregated congestion cost coefficients for the whole of Germany, while separate daily traffic patterns are used to model traffic in major cities of Germany. In Fig. 4, value 1 on the y-axis represents maximal congestion (minimal traffic speed) and

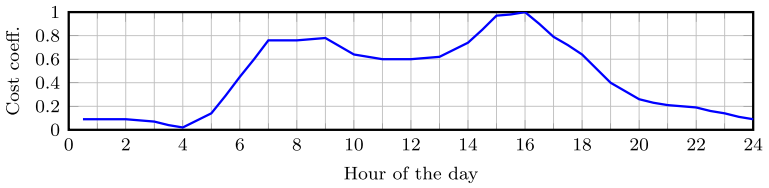



**Fig. 4** Relative Tuesday's traffic volume distribution in Germany

value 0 corresponds to the free-flow situation with no congestion or other adverse conditions (maximum traffic speed). Free-flow speed, congestion speed, and congestion coefficient at a given time of day, $t$, are the parameters to calculate the traffic speed at $t$.

**Definition 3.1** Each TD road segment is modeled as a 4-tuple $seg = (geo, v_f, v_c, tp) \in TS$ where $geo = \langle n_1, \ldots, n_m \rangle$ is a sequence of geographic coordinates, $n_i$, representing the road segment, $v_f$ is a free-flow speed, $v_c$ is a congestion speed, and $tp = \{(\bar{t}, cost) \mid \bar{t} = [t^{\vdash}, t^{\dashv}) \wedge cost \in (0; 1]\}$ represents the traffic-profile cost coefficients during time intervals of the day. The half-open intervals in $tp$ are disjoint and cover the whole day. Function $v : T \times TS \to \mathbb{R}$ returns the traffic speed at time $t \in T$ for a segment $seg \in TS$, where $TS$ represents a set of TD segments.

Travel speed along the segment $seg$ is a time function $v(t, seg)$ and varies between the two extremes. This is modeled via the $cost(t)$ function defined by the traffic profile (see Fig. 4):

$$\begin{aligned} v(seg, t) &= seg.v_f - (seg.v_f - seg.v_c) \cdot cost(t), \\ \text{where } cost(t) &= cost_1 \cdot \frac{t - t^{\vdash}}{t^{\dashv} - t^{\vdash}} + cost_2 \cdot \frac{t^{\dashv} - t}{t^{\dashv} - t^{\vdash}}, \\ \text{such that } (\bar{t}, cost_1) &\in seg.tp \wedge (\bar{\tau}, cost_2) \in seg.tp \wedge t \in \bar{t} \wedge t^{\dashv} = \tau^{\vdash}. \end{aligned} \tag{1}$$

Note that cost values are linearly interpolated, such that $cost(t)$ is a continuous function.

*Example 1* Let us assume an EV starts to travel at 6:20 along segment $seg$, where segment $seg = (geo, 13.89, 7.39, tp)$. Therefore, a free-flow speed is 13.89 m/s (50 km/h), and a congestion speed is 7.39 m/s (26.6 km/h). Let us assume the traffic profile is a set $tp = \{((00{:}00, 01{:}00), 0.09), \ldots, ((6{:}00, 7{:}00), 0.45), ((7{:}00, 8{:}00), 0.76), \ldots, ((23{:}00, 00{:}00), 0.14)\}$. Time profile divides a day into one-hour intervals. At 6:00 cost value is 0.45, at 7:00, $cost = 0.76$, and at 6:20, according to (1), $cost = 0.55$. Then traffic speed at 6:20 along $seg$ is equal to 10.29 m/s (37.1 km/h).

Assuring realistic generated data requires *calibration* of both the free-flow and congestion travel times. The calibration is implemented via two coefficients that multiply the congestion and free-flow speeds. The two coefficients are calculated by comparing simulated travel times with Google Maps travel times. First, two sets of routes were generated, inside cities and out of cities, for congestion and free-flow travel-time calibration, respectively. Second, travel times are computed at peak hours for the inside-cities set and off-peak hours for the out-of-cities set. Finally, the two generalized adjusting coefficients are obtained.

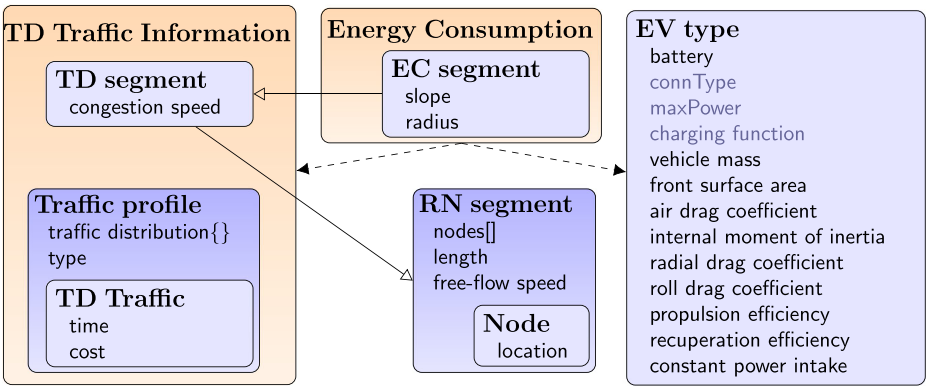



**Fig. 5** Domain model of traffic data and energy consumption

### 3.5 Data for energy consumption estimation

Energy consumption (EC) along a given route is estimated by adapting the Vehicle Energy Model (VEM) as introduced in the SUMO simulator [23]. In addition, the EC model considers traffic information discussed in Section 3.4 to estimate TD energy use along the route (see Fig. 5).

Energy consumption calculation uses two types of parameters, i.e., vehicle-specific and road-network dependent.

The following EV characteristics are employed (see Fig. 5): battery, vehicle mass, front surface area, air drag coefficient, an internal moment of inertia, radial drag coefficient, roll drag coefficient, propulsion efficiency, recuperation efficiency, and constant power intake. The constant power intake parameter could be extended and vary based on weather conditions for more precise modeling. The EV characteristics can be collected from various sources, including car manufacturers and EV enthusiasts that try to measure multiple parameters of their vehicles under specific conditions. The core road-dependent parameters, the slope and the radius, are precomputed for each EC segment and stored in the database. In addition, the segment inherits free-flow speed, length, and congestion speed and contains node coordinates (see RN, TD, and EC segments in Fig. 5).

CGIAR-CSI SRTM 90 m Digital Elevation Data [24] or other isosurface data sources can be used to calculate the slopes. Segment geometry is used to compute each segment length and radius. We deem the slope and the radius as the essential terrain approximation parameters. A single road segment might have several bends and/or hills. So to have a more precise energy model, each segment could be split into sub-segments based on these geometrical features.

### 3.6 Charging and waiting times at charging stations

The testbed component Charging Stations is responsible for storing charging-station data and calculating charging/waiting time at a particular station. Each charging station (see structure CHStation in Fig. 6) contains chargers and a TD availability profile (structure CHProfile in the Figure). The station also encompasses geographic coordinates (location), as it has to be mapped to the road network. There could be other features. For example, the parking can be free of charge, available 24/7, or located in a closed area; also, some

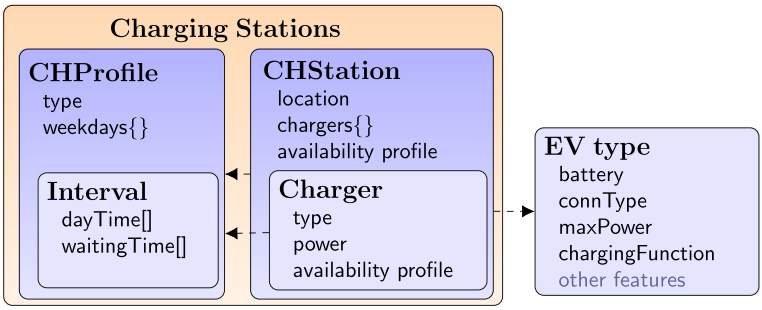


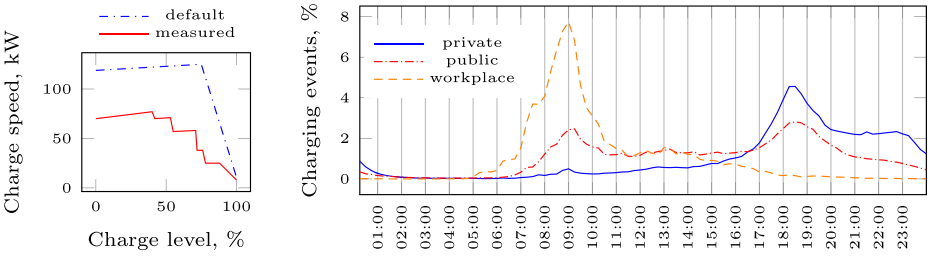

**Fig. 6** Domain model of charging stations

stations can have a connection fee. Similarly, the charging price can also be a parameter. These features can be used to set up user preferences for routing, but this paper considers only TD data, charger characteristics, and location.

The station can contain several chargers, and each charger is described by a connector type, power, and its availability profile. Charging station characteristics can be retrieved from charging service providers and other open data. The testbed integrates data from Open Charge Map [25].

Figure 7a presents two charging functions for 65 kWh battery—default and measured by observation. The charging function depends on EV features and charger properties (see EV type block on the right in Fig. 6). First, some EV types are limited by their own maximum charging power, maxPower, and predefined connection type, connType. Second, the charging process is slower when the battery's SoC is below 20% and above 80%, especially in the case of rapid charging. This protects the health of the battery. Charging functions are retrieved from open data available on the internet and integrated into the testbed. For example, for a number of EV types, the Open EV Data [26] project provides maximum power and power at different points of charging (piece-wise linear function) for different chargers.

The availability of a charging station or an individual charger can be represented by a piece-wise linear function of time. While the station can be available all day long, the particular charger availability might depend on its type. For example, in rural areas fast charging could be of no interest late in the evening, with a high probability that the station is available. Features like type, e.g., rural areas, and weekdays, e.g., Sunday and Saturday, define a particular availability profile. Figure 7b illustrates charging patterns in the Netherlands [27] for




(a) Charging function (b) Charging patterns (arrivals) in different areas [27]

**Fig. 7** Charging functions of two different 65 kWh batteries [26] and charging demands

private, public, and workplace charging points on workdays. The figure shows the percentage of charging cases throughout 24 hours. For example, at 9:00, the need for power is very high at workplaces. At night the number of charging times is low in all cases. Therefore, the availability profile can be constructed based on observation data with a high probability of a waiting time that depends on the charger power.

We propose calculating typical waiting time intervals for charging stations and applying usage profiles afterward. Algorithm 1 encapsulates the proposed heuristics to calculate a typical waiting time interval, $wt$, at a charging station, $ch$, supporting fast charging (power output $\geq$ 50 kW).

**IN:** $ch \in CH$
**OUT:** $wt = (wt_s, wt_e)$
1: Let $N$ be a number of chargers of the same type/power at $ch$
2: Let $K$ be a typical maximum charging time ▷ *E.*g. 15 min for rapid charging
3: Let $L$ and $U$ be the lower and upper bounds for a number of chargers
4: $wt = (wt_s, wt_e) \leftarrow (0, K)$
5: $\Delta \leftarrow wt_e/N$
6: $\nabla wt_e \leftarrow wt_e/2$
7: **if** $N \leq L$ **then**
8: $\quad wt \leftarrow (\nabla wt_e, wt_e + \Delta)$
9: **else if** $L < N \leq U$ **then**
10: $\quad wt \leftarrow (wt_s + \Delta, \nabla wt_e + \Delta)$
11: **else**
12: $\quad wt \leftarrow (wt_s, \Delta)$ ▷ $wt_s = 0$
13: **end if**
14: **if** *closeToHighway*$(ch)$ = **true then**
15: $\quad \delta = (wt_e - wt_s)/2$ ▷ Half of the interval length
16: $\quad wt \leftarrow (wt_s + \delta, wt_e + \delta)$ ▷ Shift right
17: **end if**
18: **return** $wt$

**Algorithm 1** Heuristic calculation of waiting time intervals.

The algorithm considers several chargers of the same type, $N$, at the charging station and a typical charging time, $K$ (from almost empty to a maximum allowed state of charge for this charging type). The latter value is chosen as a reference point, as the exact charging time depends on EV battery, e.g., rapid charging ($\geq$ 150 kW) takes 15 minutes, and fast 50 kW charging takes 40 minutes (40 kWh battery). The algorithm also uses two global constants, lower and upper bounds, $L$ and $U$, to distinguish charging stations based on the number of chargers of the same type. Bound $L$ represents a satisfactory number of chargers, e.g., 4, and bound $U$ represents a big number chargers, e.g., 10 in dedicated areas.

First, the interval $wt$ is initialized with values $(0, K)$, as $wt_s$ and $wt_e$ represent the minimum and maximum charging time, respectively. Then, two additional values are calculated. Value $\Delta$ represents the interval part divided by the number of chargers, and value $\nabla$ represents half of the interval. If the number of chargers is small (line 7), it might take time to wait, and the interval values are increased (the minimum waiting time is half of the initial interval length, and the maximum waiting time is shifted by $\Delta$, too). If the number of chargers is satisfactory (line 9), the interval length is smaller but shifted by $\Delta$. If the charging

stations contain a lot of chargers of a particular type, the waiting time interval is short, and the minimum waiting time is 0.

The last part of the algorithm considers the closeness of the station to the highway. The interval ends are shifted if the highway is close to the station as it might be in demand.

The availability profile of the charging station defines the demand during different times of the day. Therefore, waiting times should differ. We assume that the availability profile stores a coefficient $p$ for each discrete time value of the given granularity, e.g., each hour. For each timestamp, the coefficients of two adjacent time values are interpolated, similarly to the *cost* function (see (1)). Then, waiting time is calculated using (2).

$$wt = (wt_s + (wt_e - wt_s) \cdot p, wt_e + wt_e \cdot p) \tag{2}$$

If $p = 0$, waiting time is the default one, and if the charging demand for a particular time is decreased ($p < 0$), values of the interval are decreased, too. In the testbed, $p \in [-1; 0.5]$ (negative values are ceiled to 0 during calculations). The coefficient value sequence is created to reflect profiles of charging stations identified by ElaadNL [27].

Waiting times for slow charging can be calculated using other heuristics. As mentioned above, they have different profiles and demands. Slow charging can take several hours, and an occupied charger means a long waiting time. This property would significantly increase path costs in routing algorithms and is, thus, out of the scope of this work.

### 3.7 The FIFO property

To realistically model time-dependent travel times, the generated data must satisfy the FIFO property. The property, assumed by most work on routing in time-dependent road networks, is a natural requirement that, while vehicle $B$ leaving at a later time than vehicle $A$ can take less time to traverse a path, $B$ nevertheless can not arrive sooner than $A$ [28, 29]. If this property were not satisfied for a single segment of a road network, optimizing the earliest arrival time at the end of that segment may require simply stopping and waiting at the beginning of the segment so as to get short travel time for that segment and, in this way, overtake the cars that entered the segment earlier. This contradicts common sense and reduces pruning opportunities in routing algorithms.

In the following, we discuss when FIFO could be violated in synthetically generated TD weights of a road-network graph. We then show how we ensure that it is not violated both for the road-network TD weights and the waiting times of the charging stations.

There are several reasons why FIFO might be violated in traffic simulations. The segment-based simulation assumes that a vehicle drives at a constant speed on the entire simulated segment. The travel speed is determined by the speed function using a traffic profile (see Def. 3.1). Let us assume two e-vehicles $EV_1$ and $EV_2$ start a trip along segment *seg* at time $t_1^{\vdash}$ and $t_2^{\vdash}$, respectively, where $t_1^{\vdash} < t_2^{\vdash}$. E-vehicles $EV_1$ and $EV_2$ drive at speed $v_1$ and $v_2$, respectively. As both vehicles are simulated using constant speed starting from the segment start, the $i$-th vehicle will reach the segment end at time $t_i^{\dashv}$ (see (3)).

$$t_i^{\dashv} = t_i^{\vdash} + \frac{|seg|}{v_i},\ i \in \{1, 2\} \tag{3}$$

The FIFO property would be violated if $t_1^{\dashv} > t_2^{\dashv}$, i.e., vehicle $EV_2$ which started later were to reach the segment end sooner than $EV_1$.

This condition can be easily violated if speed was modeled as a step function. If that were the case, we could find two very close time points $t_1^{\vdash}$ and $t_2^{\vdash}$ ($\Delta t = t_2^{\vdash} - t_1^{\vdash}$) and while $\Delta t$

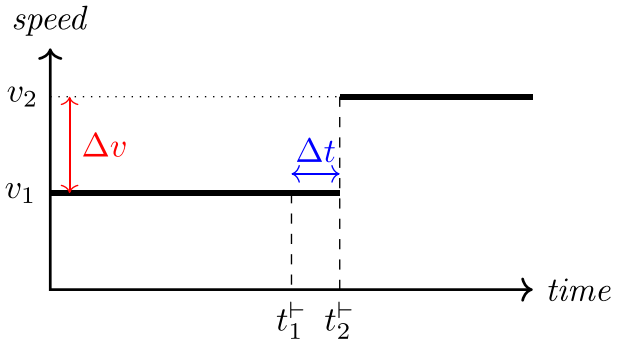



**Fig. 8** Example of the step function

could be made arbitrarily small, the speed difference remained constant $\Delta v = v_2 - v_1 = C$ (see Fig. 8.) If $\Delta t$ is almost zero, the FIFO violation condition can be simplified as follows:

$$\frac{|seg|}{v_1} > \frac{|seg|}{v_2} \Rightarrow \frac{1}{v_1} > \frac{1}{v_2} \tag{4}$$

This shows that FIFO will not be satisfied when speed values increase in the speed profile ($v_2 > v_1$). This problem can be solved by eliminating the step function using interpolation and replacing it with a piece-wise linear speed profile function. In our work, this is achieved via a piece-wise linear *cost* function (see (1)). Then, $\Delta t \to 0$ implies $\Delta v \to 0$.

Piece-wise linear speed profile function alone does not guarantee the FIFO property. Segment-based simulation assumes that vehicle speed is constant on the entire segment, i.e., equal to the speed at the time the EV enters the segment. Thus, $EV_2$ that enters a segment later than $EV_1$ but with a higher speed will always take over $EV_1$ in a sufficiently long segment. Violation of FIFO can be rewritten as presented in (5).

$$t_1^{\vdash} + \frac{|seg|}{v_1} > t_2^{\vdash} + \frac{|seg|}{v_2} \Rightarrow \frac{|seg|}{v_1} > \Delta t + \frac{|seg|}{v_2} \Rightarrow \Delta t < \frac{|seg|}{v_1} - \frac{|seg|}{v_2} \tag{5}$$

The time difference between two vehicles starting on the same segment is represented by $\Delta t$. Thus, one way to guarantee the FIFO property is to check segment lengths. Let $a = \Delta v / \Delta t$ be an "acceleration" value of the speed profile at a given time, then FIFO violation condition from (5) can be rewritten as a segment length estimation:

$$\Delta t < |seg|(\frac{1}{v_1} - \frac{1}{v_1 + \Delta v}) \Rightarrow |seg| > \frac{\Delta t}{\Delta v}({v_1}^2 + v_1 \Delta v) \Rightarrow |seg| > \frac{v_1^2}{a} + v_1 \Delta t \tag{6}$$

Equation (6) shows that the maximum FIFO-compliant segment length depends on the used speed and speed profile acceleration at a given time. To find the lower bound of the maximum FIFO-compliant length of a given segment ($lb(|seg|_{max})$), we assume $\Delta t$ to be close to zero and disregard the second term of the sum. Further, taking the possible range of speeds on the given segment—from $seg.v_c$ to $seg.v_f$—and combining it with the largest *cost* decrease per time unit using (1), we get the largest acceleration for this segment ($a_{max}(seg)$). Thus:

$$lb(|seg|_{max}) = \frac{seg.v_c^2}{a_{max}(seg)} \tag{7}$$

Such a bound can be easily computed for each segment. If there is a segment that is too long, it is broken in two or more pieces to guarantee the FIFO property.

*Example 2* Let us assume a free-flow speed, $v_f$, is 19.44 ms/s (70 km/h) and congestion speed, $v_c$, 1.39 m/s (5 km/h) along some segment. According to the speed values in the real data speed profile, the biggest speed increase (and accordingly the biggest profile acceleration value) is from 18:00 to 19:00 (see Fig. 4), where cost coefficient $cost(t)$ changes from 0.638 to 0.397, and simulated traffic speed changes from 7.9 m/s (28.5 km/h) to 12.28 m/s (44 km/h), respectively (see (1).) The profile's speed acceleration $a_{max}$ is 0.0012 m/s$^2$ in this period of time. Thus, according to (7), the segment can be up to $7.9^2/0.0012$ m, i.e., 52 km long without breaking FIFO.

It is easy to see that, if FIFO is not violated on every single segment of a path, then FIFO is not violated on the entire path. If two vehicles enter $seg_i$ with a positive time difference $\Delta t_i$, the time difference between the two vehicles reaching the end of the segment is positive as well when FIFO is satisfied. This time difference is then the time difference of entering the next segment on the path. In other words, $\Delta t_i > 0$ implies $\Delta t_{i+1} > 0$ and so on until the end of the path.

At the charging stations, waiting time should satisfy the FIFO rule. The EV that came first should be served first. Eqution (8) presents the FIFO condition for the waiting-time function.

$$\forall t_1 < t_2 : t_1 + \Delta wt_1 < t_2 + \Delta wt_2 \Rightarrow \Delta wt_1 - \Delta wt_2 < t_2 - t_1 \Rightarrow \frac{\Delta wt_1 - \Delta wt_2}{t_2 - t_1} < 1. \quad (8)$$

*Example 3* Let us assume $EV_2$ comes to the charging station 1 minute later than $EV_1$. But $EV_2$ waits for 5 minutes while $EV_1$ has to wait 15 minutes. Thus, $EV_2$ would leave the charging station 9 minutes earlier than $EV_1$, and would not satisfy the FIFO rule, $\frac{15-5}{1} = 10 > 1$.

As this paper uses the availability profile coefficient $p$ applied on a typical waiting time, to satisfy FIFO the profile coefficients should satisfy the condition in (9):

$$\frac{\Delta wt_1 - \Delta wt_2}{t_2 - t_1} < 1 \Rightarrow \frac{p_1 \cdot \Delta wt - p_2 \cdot \Delta wt}{t_2 - t_1} < 1 \Rightarrow (p_1 - p_2) < \frac{(t_2 - t_1)}{\Delta wt}. \quad (9)$$

Finally, in the testbed, waiting times are intervals. Therefore, (8) and (9) should be satisfied for both minimum and maximum values.

## 4 Testbed implementation

The testbed was implemented following the model and functionality described in Section 3. Figure 9 summarizes the process of semi-synthetic data preparation using open tools and data sources. PostgreSQL was chosen to support components of the testbed. Germany map was retrieved from OSM and filtered for vehicle roads using `osmfilter`. The `osm2po` tool generated a routable network as a sequence of SQL statements. PostGIS database extension was used to store and manage spatial road-network data in PostgreSQL. Finally, the

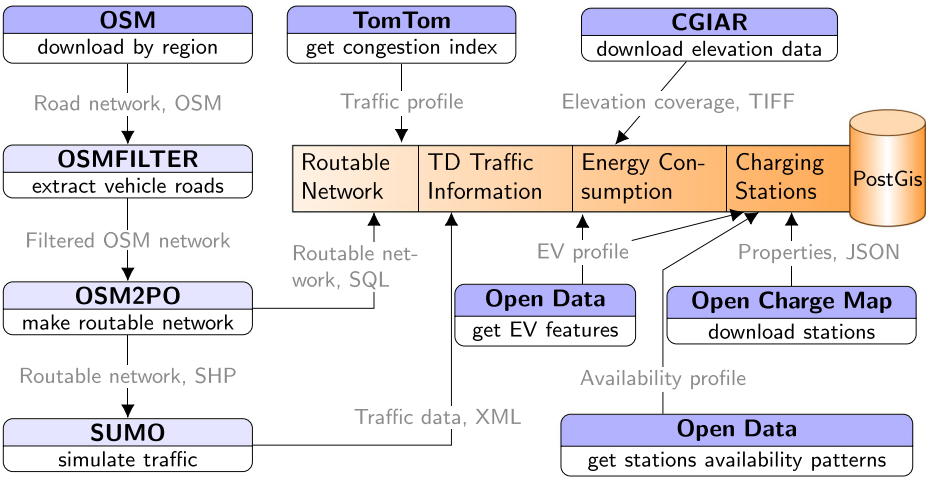



**Fig. 9** Data sources and processing

routable network was converted into a shapefile and fed to SUMO to create traffic data in XML.

The TomTom congestion index supplemented the traffic data. CGIAR-CSI SRTM Digital Elevation data in the TIFF format were downloaded and applied to compute elevation gain on each network-segment. Finally, EV features and charging stations with their availability patterns were set up using publicly available data.

While the detailed description of the implementation of the suggested testbed services (see Fig. 2) is out of the scope of this paper, in the following, we discuss the key aspects. The computation of the total energy and the total driving time on a route is at the core of *findPath* and *calcPath* functions. For each segment, the EC segment properties at a given time are used to calculate the travel time and energy required to traverse the segment. Then, the estimations for each route segment are added up to get the total energy and travel time.

The EC calculation evaluates each EC segment based on its properties. For some segments, it might result in a negative value. For example, going downhill, a car recuperates and charges its battery. Thus, if the functions are invoked with an initial SoC, the EC calculation considers that the battery cannot be charged more than the designed capacity, and the recuperated energy is lost.

The above calculations result in single-valued time and energy results for a given route. The testbed simulates the prediction uncertainty by assuming that the timing of peaks in the traffic profile might slightly shift from day to day. The testbed calculates two congestion cost values for each segment at a given time $t$: the minimal cost value and the maximal cost value. Cost values are calculated using the time window $[t-\epsilon, t+\epsilon]$, where $\epsilon$ is a testbed parameter controlling the uncertainty (see (10)).

$$[cost^{\vdash}(t), cost^{\dashv}(t)] = [\min cost(t), \max cost(t)] \quad \text{where} \quad t \in [t-\epsilon, t+\epsilon] \tag{10}$$

The default $\epsilon$ value is set to 30 minutes, but can be adjusted. Note that $t$ is a time when an EV reaches a given segment $seg_i$ along the route. Thus, it depends on the travel speed and departure time of the previous $i-1$ segments. Therefore, it is a sum of time-interval lengths required to pass all previous segments along the route added to the trip start time

$t_{start}$. Let us assume $\Delta t_i$ is the time required to pass segment $i$, then $seg_i$ entrance time $t_i$ is $t_i = t_{start} + \sum_{n=1}^{i-1} \Delta t_n$. For the whole route, the bounds of the estimated energy and time intervals are calculated in two iterations. The first iteration uses $cost^{\vdash}(t)$ as the cost function for the lower bound and the second iteration—$cost^{\dashv}(t)$ for the upper bound.

## 5 Experiments and results

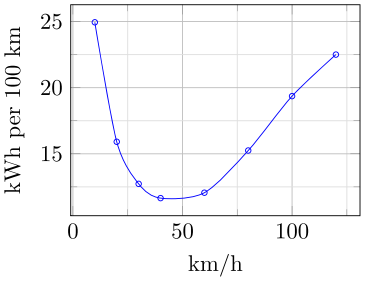
The testbed architecture requires a routing module that considers time-dependent weights. The KaTCH (Karlsruhe Time-Dependent Contraction Hierarchies) [30] is an algorithmic framework based on time-dependent contraction hierarchies. It enables queries that ask for a minimum travel time route for a start and a destination depending on a given departure time. The KaTCH [31] implementation was integrated into the testbed as a routing engine, and several tests were run to illustrate realistic results and appropriate scalability within the testbed. External forces like weather, traffic incidents, or road works are not included.

First, as a case study, sources and destinations were chosen for 8 representative trips in Germany. Then, the travel-time and energy-consumption intervals were calculated for all of them when traveling from a source to a destination and back—16 individual trips in total. Also, the departure times were set to 00:00 and 16:00 as non-congestion and congestion-time representatives. The tests were run in the testbed using the TomTom service as a reference.

To estimate energy consumption in both testing environments, the energy usage curve was constructed as a sequence of pairs (kWh per 100 km, km/h). The approximation of the energy consumption curve is presented in Fig. 10a. The modeled prototype vehicle had the set of parameters presented in Fig. 10b.

Figure 11 plots estimated travel time and energy consumption for different departure times on testbed (TB) with the results from TomTom (TT) shown for reference. For departure time 00:00, Figs. 11a and c present travel time and consumption energy, respectively.

The case study results show that the testbed is more conservative regarding travel time and energy consumption when leaving at the non-congestion time. In most cases, the difference is not significant, and the testbed-generated interval is very small. For a peak hour,




(a) Energy consumption curve

| | |
|---|---|
| mass | 1785 kg |
| battery | 62 kWh |
| constant power consumption | 1.5 kW |
| air drag coefficient | 0.28 |
| front surface area | 2.44 $m^2$ |
| propulsion efficiency | 0.8 |
| recuperation efficiency | 0.8 |

(b) EV features

**Fig. 10** Energy consumption for a EV with a defined feature set

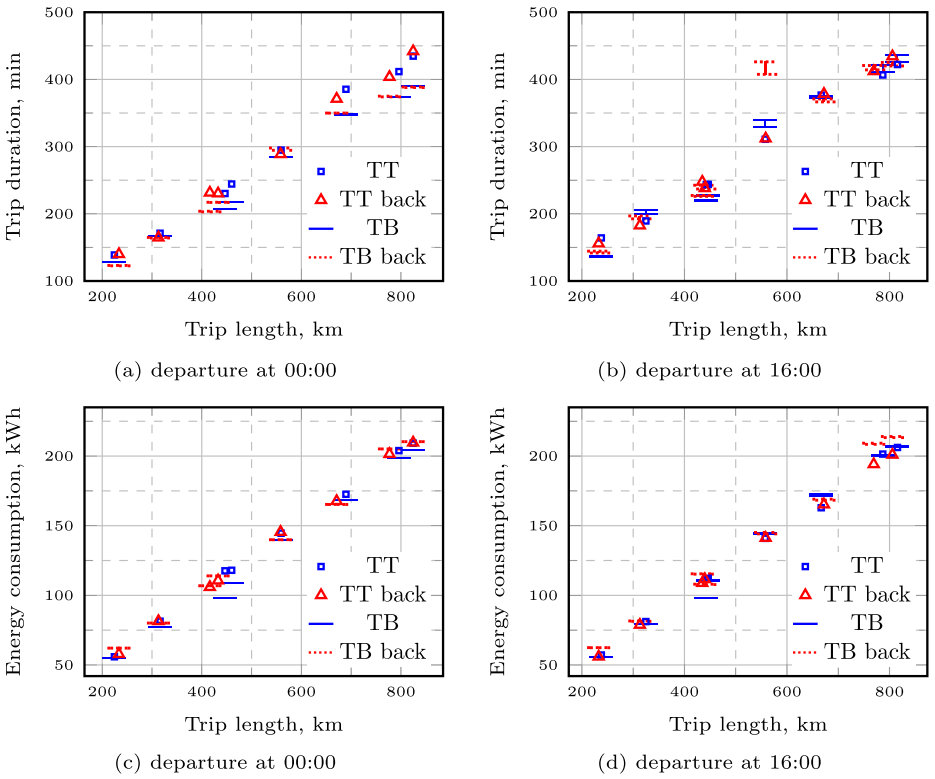



**Fig. 11** Travel time and energy consumption for different departure times

the testbed is more optimistic regarding travel time and energy consumption, and the generated uncertainty intervals are longer: for long trips interval length is approximately half an hour. The testbed has its own segment-level congestion data. Therefore, the different results make no surprise. The testbed provides different results for forward and backward trips, as the model considers elevation details and recuperation.

Additionally, tests were run for random trips of different lengths. For each length 125 pairs of start and destination were generated. Figure 12 shows aggregated statistical results with interquartile range, minimum and maximum values, and outliers.

During executions, charging time was subtracted. The results of the trip duration include only traveling time. These large-scale results show that TB underestimates trip duration for a non-peak hour. For a peak-hour, the difference is relatively small. TB energy consumption calculations are more conservative and predict larger consumption than TT.

Figure 13 shows scalability test results. The tests were run on a Linux workstation with Intel(R) 16 Core(TM), i9-9880H CPU @ 2.30GHz, 32GB RAM with an equivalent remote database server. For each trip length (air distance), 1000 source-destination pairs were generated, their routing was executed, and routes were saved in the database. The region was loaded into the main-memory KaTCH data structure, with approx. 21GB RAM used. The difference in the air distance and route length is app. 30%. The results show that the query cost without energy consumption calculation is almost constant, whereas energy computation grows linearly to the length of the path.

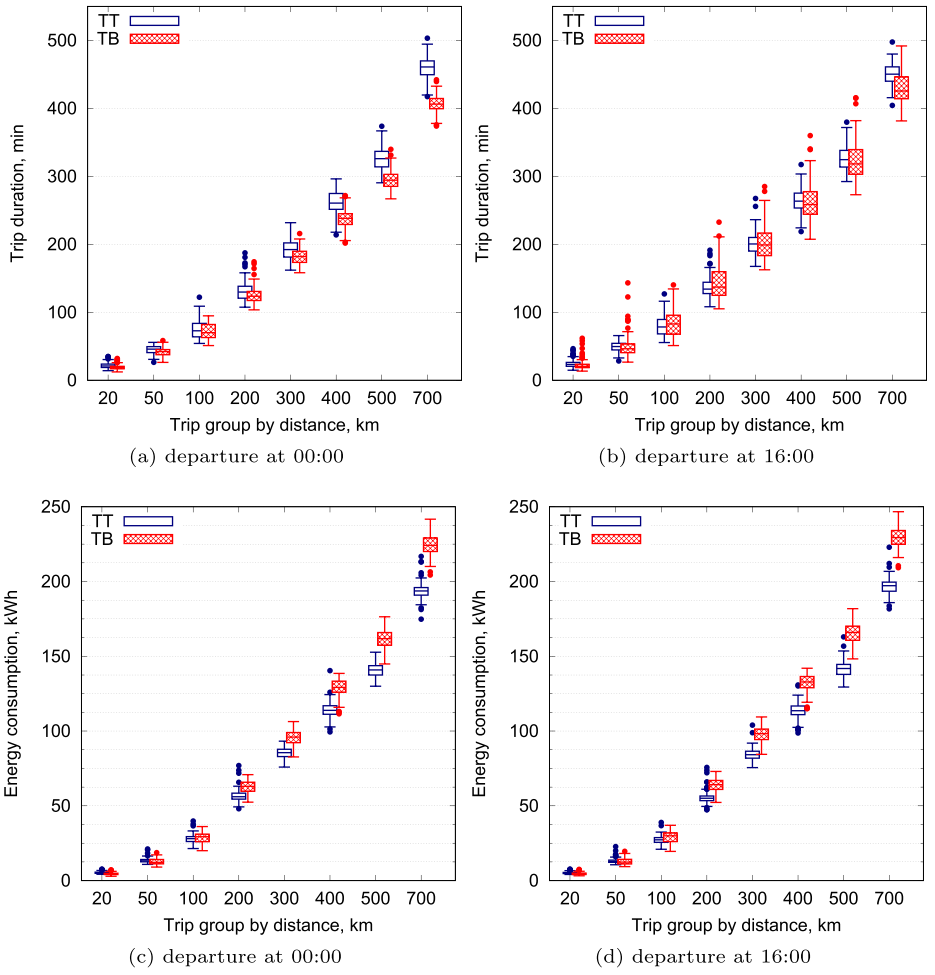



**Fig. 12** Travel time and energy consumption for different departure times

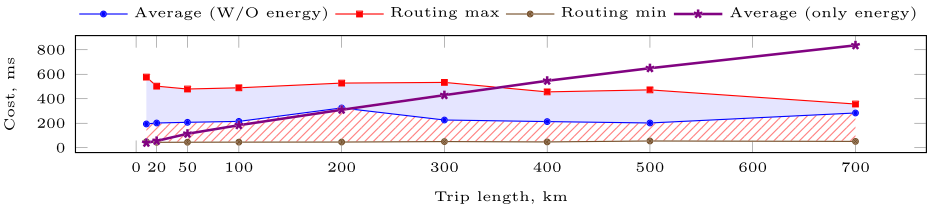



**Fig. 13** Scalability tests

## 6 Conclusions and future work

Motivated by the inherent complexity of testing advanced routing algorithms, the paper proposes a testbed that integrates state-of-the-art tools and provides a systematic approach to available open-source data. The proposed domain data models offer great opportunities for further research in the area of advanced EV routing. We believe the provided insights and the testbed itself will shorten the preparation phase of future experimental studies. The scalability and reference-based tests demonstrate the merits of the testbed.

The work can be extended in several directions. First, functions to switch among EV energy consumption and lifecycle profiles can be developed using real-world or semi-synthetic EV templates. Furthermore, the experimental environment could be enriched with a flexible setup for experiments with varying initial environmental or network conditions as parameters.

**Funding** This project has received funding from European Regional Development Fund (project No 01.2.2-LMT-K-718-02-0018) under a grant agreement with the Research Council of Lithuania (LMTLT).

**Availability of data and materials** The data and codes that support the findings of this study are available with the identifier at the National Open Access Research Data Archive (Midas) of Lithuania DOI: 10.18279/MIDAS.DALTRA.193594

### Declarations

**Disclosure statement** The authors declare that they have no known competing financial interests or personal relationships that could have influenced the work reported in this paper.



## References

1. Baum M, Dibbelt J, Wagner D, Zündorf T (2020) Modeling and engineering constrained shortest path algorithms for battery electric vehicles. Transp Sci 54(6):1571–1600. https://doi.org/10.1287/trsc.2020.0981
2. Åkerblom N, Chen Y, Chehreghani MH (2020) An online learning framework for energy-efficient navigation of electric vehicles. In: IJCAI, pp 2051–2057. https://doi.org/10.24963/ijcai.2020/284
3. Russo D, Roy BV, Kazerouni A, Osband I, Wen Z (2018) A tutorial on thompson sampling. Found Trends Mach Learn 11(1):1–96. https://doi.org/10.1561/2200000070
4. Codeca L, Frank R, Faye S, Engel T (2017) Luxembourg SUMO traffic (lust) scenario: traffic demand evaluation. IEEE Intell Transp Syst Mag 9(2):52–63. https://doi.org/10.1109/MITS.2017.2666585
5. Barauskas A, Brilingaite A, Bukauskas L, Ceikute V, Civilis A, Saltenis S (2021) Semi-synthetic data and testbed for long-distance e-vehicle routing. In: ADBIS, vol 1450, pp 61–71. https://doi.org/10.1007/978-3-030-85082-1_6
6. Tempelmeier N, Dietze S, Demidova E (2020) Crosstown traffic - supervised prediction of impact of planned special events on urban traffic. GeoInformatica 24(2):339–370. https://doi.org/10.1007/s10707-019-00366-x
7. Zhao X, Spall JC (2018) Modeling traffic networks using integrated route and link data. Preprint at arXiv:1811.01314

8. Pramanik A, Rahman M, Anam I, Ali AA, Amin A, Rahman M (2020) Modeling traffic congestion in developing countries using google maps data. Preprint at arXiv:2011.02359
9. Zafar N, Haq IU (2020) Traffic congestion prediction based on estimated time of arrival. PLoS One, vol 15(12). https://doi.org/10.1371/journal.pone.0238200
10. Sapre V, Kalambur S, Sitaram D, Bastian R (2018) Synthetic generation of traffic data for urban mobility. In: ICACCI, pp 2151–2157. https://doi.org/10.1109/ICACCI.2018.8554633
11. (2021). German aerospace center (DLR) and others: SUMO — simulation of Urban MObility. Accessed 28 Sept 2021. https://sumo.dlr.de/docs
12. Brinkhoff T (2002) A framework for generating network-based moving objects. GeoInformatica 6(2):153–180. https://doi.org/10.1023/A:1015231126594
13. López PÁ, Behrisch M, Bieker-Walz L, Erdmann J, Flötteröd Y, Hilbrich R, Lücken L, Rummel J, Wagner P, WieBner E (2018) Microscopic traffic simulation using SUMO. In: ITSC, pp 2575–2582. https://doi.org/10.1109/ITSC.2018.8569938
14. Yu J, Fu Z, Sarwat M (2020) Dissecting geosparksim: a scalable microscopic road network traffic simulator in apache spark. Distrib Parallel Databases 38(4):963–994. https://doi.org/10.1007/s10619-020-07306-x
15. Yu J, Wu J, Sarwat M (2015) Geospark: a cluster computing framework for processing large-scale spatial data. In: SIGSPATIAL, pp 70–1704. https://doi.org/10.1145/2820783.2820860
16. Ramamohanarao K, Xie H, Kulik L, Karunasekera S, Tanin E, Zhang R, Khunayn EB (2017) SMARTS: scalable microscopic adaptive road traffic simulator. ACM Trans Intell Syst Technol 8(2):26–12622. https://doi.org/10.1145/2898363
17. AlDwyish A, Xie H, Tanin E, Karunasekera S, Ramamohanarao K (2017) Using a traffic simulator for navigation service. In: SIGSPATIAL, pp 78–1784. https://doi.org/10.1145/3139958.3139998
18. Baum M, Dibbelt J, Pajor T, Sauer J, Wagner D, Zündorf T (2020) Energy-optimal routes for battery electric vehicles. Algorithmica 82(5):1490–1546. https://doi.org/10.1007/s00453-019-00655-9
19. (2021). OpenStreetMap foundation: openstreetmap. Accessed 28 Sept 2021. https://www.openstreetmap.org
20. (2021). Geofabrik GmbH: openstreetmap data extracts. Accessed 28 Sept 2021. http://download.geofabrik.de
21. Moeller C (2021) osm2po — openstreetmap converter and routing engine for java. Accessed 28 Sept 2021. https://osm2po.de
22. (2021). German aerospace center (DLR) and others: tools/trip. Accessed 28 Sept 2021. https://sumo.dlr.de/docs/Tools/Trip.html
23. Kurczveil T, López PÁ, Schnieder E (2013) Implementation of an energy model and a charging infrastructure in sumo. In: Simulation of urban mobility user conference. Springer, pp 33–43. https://doi.org/10.1007/978-3-662-45079-6_3
24. Jarvis A, Reuter HI, Nelson A, Guevara E (2021) Hole-filled seamless SRTM data V4. Accessed 28 Sept 2021. http://srtm.csi.cgiar.org
25. (2021). Open charge map: the open charge map API. Accessed 7 Mar 2021. https://openchargemap.org/site/develop/api
26. (2021). Chargeprice: open EV data. Accessed 7 Mar 2021. https://github.com/chargeprice/open-ev-data
27. (2021). ElaadNL: open data sets. Accessed 12 Mar 2021. https://platform.elaad.io
28. Sung K, Bell MG, Seong M, Park S (2000) Shortest paths in a network with time-dependent flow speeds. Eur J Oper Res 121(1):32–39
29. Kanoulas E, Du Y, Xia T, Zhang D (2006) Finding fastest paths on a road network with speed patterns. In: ICDE, p 10. https://doi.org/10.1109/ICDE.2006.71
30. Batz GV, Geisberger R, Sanders P, Vetter C (2013) Minimum time-dependent travel times with contraction hierarchies. ACM J Exp Algorithmics, vol 18. https://doi.org/10.1145/2444016.2444020
31. (2021). Institut fuer theroretische informatik, karlsruher institut fuer technology (KIT): KaTCH – Karlsruhe time-dependent contraction hierarchies. Accessed 16 Mar 2021. https://github.com/GVeitBatz/KaTCH

**Andrius Barauskas** holds a PhD in technological science from Vilnius Tech, Lithuania. He is a project researcher at Vilnius University in the Institute of Computer Science. His research interests focus on transport modeling, spatial planning.

**Agnė Brilingaitė** holds a PhD in computer science from Aalborg University, Denmark. She is an associate professor at Vilnius University in the Institute of Computer Science. Her research interests focus on spatial data modelling, location-based services, cybersecurity training, and education in computer science.

**Linas Bukauskas** holds a PhD in computer science from Aalborg University, Denmark. He is an associate professor in the Institute of Computer Science at Vilnius University. His research interests include Cybersecurity, Data processing, Database management systems, Data Mining, and Natural Language Processing.

**Vaida Čeikutė** holds a PhD in computer science from Aarhus University, Denmark. She is a project researcher at Vilnius University in the Institute of Computer Science. Her research interests include trajectory pattern mining, geo-context in location-based services, and intelligent transportation systems.

**Alminas Čivilis** holds a PhD in computer science from Vilnius University, Lithuania. He is an assistant professor and project researcher at Vilnius University in the Institute of Computer Science. His research interests focus on Intelligent Transportation Systems, Location-based Services.

**Simonas Šaltenis** holds a PhD in computer science from Aalborg University, Denmark. He is a research professor at the Institute of Computer Science, Vilnius University as well as an associate professor at the Department of Computer Science, Aalborg University. His research interests focus on spatial and spatio-temporal data management and intelligent transportation systems.