Data Exchange Format and Software Interfaces

Grant Agreement N°: 875022

Project Acronym: **SORTEDMOBILITY**

Project Title: Self-Organized Rail Traffic for the Evolution of Decentralized MOBILITY

Funding scheme: Horizon 2020 ERA-NET Cofund

Project start: 1 June 2021

Project duration: 3 Years

Work package no.: 4

Deliverable no.: 1

Status/date of document:

Due date of document:

Lead partner for this document: DTU

Project website: www.sortedmobility.eu

Dissemination Level		
PU	Public	
RE	Restricted to a group specified by the consortium and funding agencies	
CO	Confidential, only for members of the consortium and funding agencies	

Revision control / involved partners

Following table gives an overview on elaboration and processed changes of the document:

Revision	Date	Name / Company short name	Changes
1	16/09/2022	DTU	First draft
2	04/10/2022	CNR and univEiffel	Sec. 2, 3, 4.1, 5.3 and 5.4
3	27/10/2022	TUDelft	Sec. 4.2.1 and 5.2
4	07/11/2022	All	All Figures
5	25/11/2022	DTU	Overall Revision

Following project partners have been involved in the elaboration of this document:

Partner No.	Company short name	Involved experts
1	univEiffel	Paola Pellegrini, Federico Naldini, Bianca Pascariu
2	SNCF	
3	DTU	Carlos Lima Azevedo, Felipe Rodrigues, Georges Sfeir, Johan Victor Flensburg, Mayara Monteiro
4	BDK	
5	ISTC-CNR	Vito Trianni
6	RFI	
7	TUDelft	Egidio Quaglietta, Konstantinos Rigos

Executive Summary

The objective of D4.1 is to describe the data exchange format and software interfaces of SORTEDMOBILITY.

After an introduction in Chapter 1, Chapter 2 presents the modelling framework of the decentralized traffic management system of SORTEDMOBILITY, Chapter 3 provides an overview of the software framework, Chapter 4 describes the different component of the frameworks presented in the previous chapters, and finally Chapter 5 describes the data content and format of the needed software interfaces.

Table of contents

Table of figures

Tables

Table of abbreviations

TSP	Traffic State Prediction
RTTP	Real Time Traffic Plan
rtRTMP	Real-time Railway Traffic Management Problem
MILP	Mixed-Integer Linear Programming
OD	Origin-Destination
EGTrain	Environment for the desiGn and simulaTion of RAIlway Networks
GUI	Graphical User Interface
PT	Public Transit
LT	Long-Term
MT	Mid-Term
ST	Short-Term
TAZ	Traffic Analysis Zone
DAS	Day Activity Schedule
GTFS	General Transit Feed Specification
PAP	Passenger Assignment Plan

1 INTRODUCTION

SORTEDMOBILITY (Self-Organized Rail Traffic for the Evolution of Decentralized MOBILITY) aims at developing concepts, models and algorithms for self-organizing management of public transport operations in urban and interurban areas, specifically focusing on rail transport as a mobility backbone. In addition, a detailed simulation assessment platform will be developed to assess the proposed self-organization approach against a centralized one.

This deliverable reports on the different data exchange formats and software interfaces between SORTEDMOBILITY's different modules. The report starts by presenting the modelling frameworks of the centralized and decentralized traffic management systems of SORTEDMOBILITY. Next, it provides an overview of the software architecture and describes its different components. Finally, it describes the data content and format of the needed software interfaces.

2 MODELLING FRAMEWORK

Conceptually, the modelling framework identifies the different actors and processes that are involved in the train traffic management. We have developed a framework for the decentralised solution depicted in **Error! Reference source not found.**. The framework determines the way in which the control architecture interacts with the transport (simulation) system, identifying relevant flows of information to be included in a Traffic State and Demand Prediction, to determine and implement the Real Time Traffic Plan (RTTP), which is the description of the train routes and schedules that is used within the project.

The main difference between centralised and decentralised model resides in the fact that the former implements a global conflict detection and resolution to produce a new RTTP, while the latter has autonomous train agents that interact to determine the best RTTP to be implemented from the individual train point of view (here referred to as neighbourhood RTTP as it is related to a train and its immediate neighbourhood, see also Section 4.1.2). The decision made by the trains is fully decentralised. However, a centralised merge of the individual decisions is needed to provide the transport system with a unique global RTTP. This step is however much simplified when consensus is reached among trains and is based only on simple heuristics to aggregate the information provided by multiple trains.

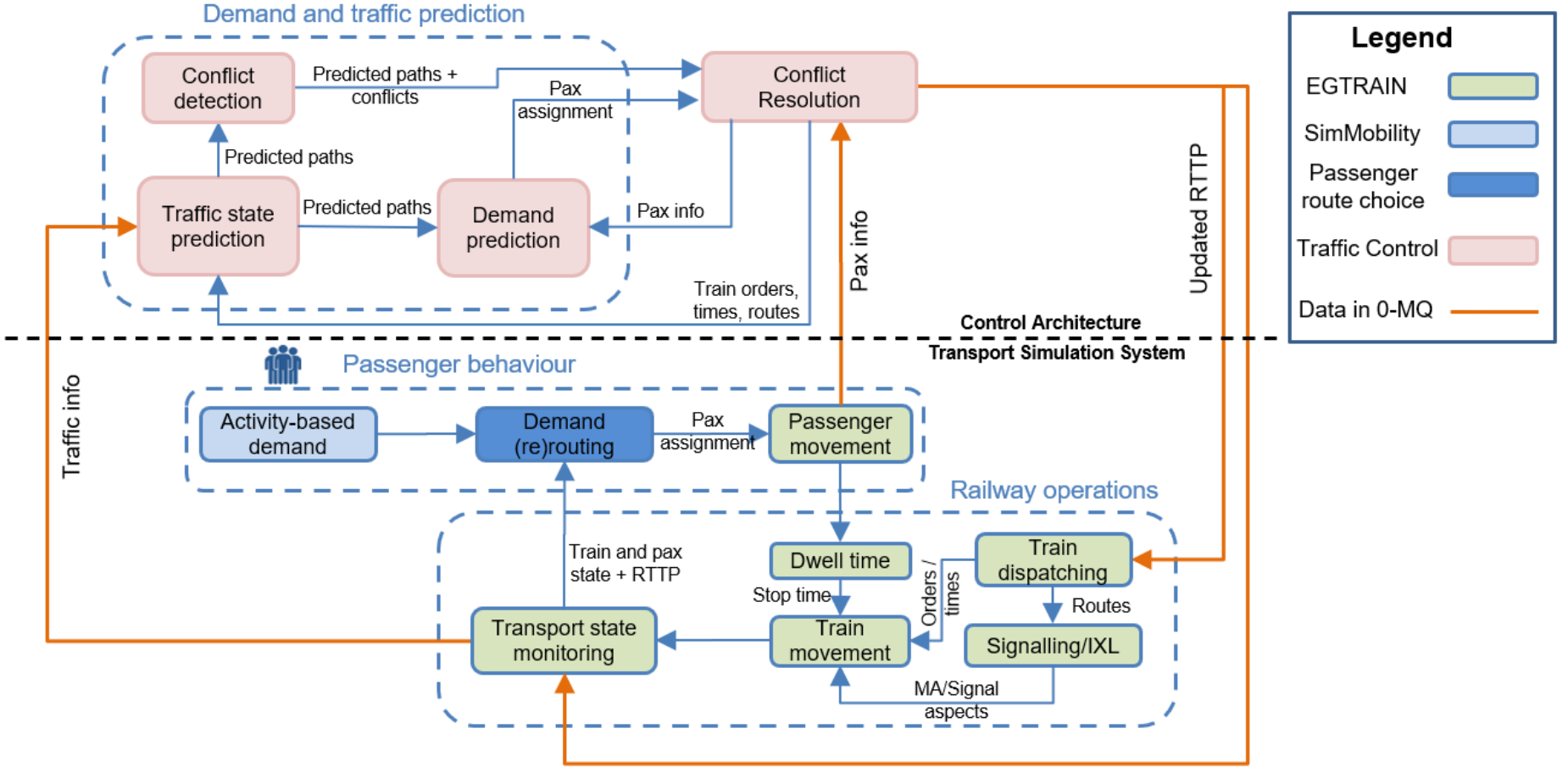

Figure 1: Modelling framework of the centralized traffic management

3 SOFTWARE ARCHITECTURE

The software architecture of SORTEDMOBILITY contains two major components, the transport simulation system architecture (that simulates a realistic transport system and its performance) and the control architecture. As the experimental activities of the project are not deployed in a real application, the **transport simulation system** architecture must encompass a series of modules necessary to simulate the real world in terms of network infrastructure, rolling stock and passengers. The transport simulation system is based mainly on two software suites:

- A microscopic simulation of train traffic based on the EGTRAIN simulator (Quaglietta, 2011)
- A microscopic, event-based simulation of travel demand, based on the SimMobility Mid-Term Pre-day simulator (Adnan et al., 2012)

On the other hand, the **control architecture** contains two components necessary for traffic management:

- A conflict detection and resolution system (either centralised or decentralised), to determine the best possible train route and schedule, which relies on traffic states predictions (future train positions);
- A demand prediction module that forecasts expected demand in the short-run given the current state of the system, the known historical demand patterns and a given train route and schedule plan.

Altogether, these software modules have been conceived independently for fast prototyping and testing, but they interact with each other providing the relevant information needed to simulate all the components required for the system optimisation.

4 COMPONENTS

This chapter presents the components of the two main modules, control architecture and transport simulation system.

4.1 Control Architecture

4.1.1 RECIFE-MILP

RECIFE-MILP is a mixed-integer linear programming (MILP) based algorithm for real-time railway traffic management. It makes centralized decisions on the best schedule and routes to be taken by a set of trains traversing a given infrastructure in a time horizon.

It is a two-step algorithm. In the first step, a pure scheduling problem is solved, in which trains use a previously defined route and precedencies on common tracks must be decided. In the second step, a routing and scheduling problem is solved, in which the choice of one out of a set of alternative routes must be performed for each train. The solution of the first step is used to initialize the search in the second.

The basic functioning of RECIFE-MILP consists in solving the MILP formulation detailed in Pellegrini et al. (2015) with a commercial solver. This is done in both steps.

The objective function typically optimized is the minimization of total delay, but various others have been tested.

RECIFE-MILP has been successfully used for optimizing traffic management in various French and European infrastructures (Pellegrini et al., 2015; Quaglietta et al., 2016).

4.1.2 Self-organizing decentralized system

The decentralised approach to traffic management is realised by means of a software pipeline that emulates how multiple autonomous intelligent trains make a shared decision about the schedule and routes that each train should follow. The pipeline is composed by several modules, which are displayed in **Error! Reference source not found.** and described below:

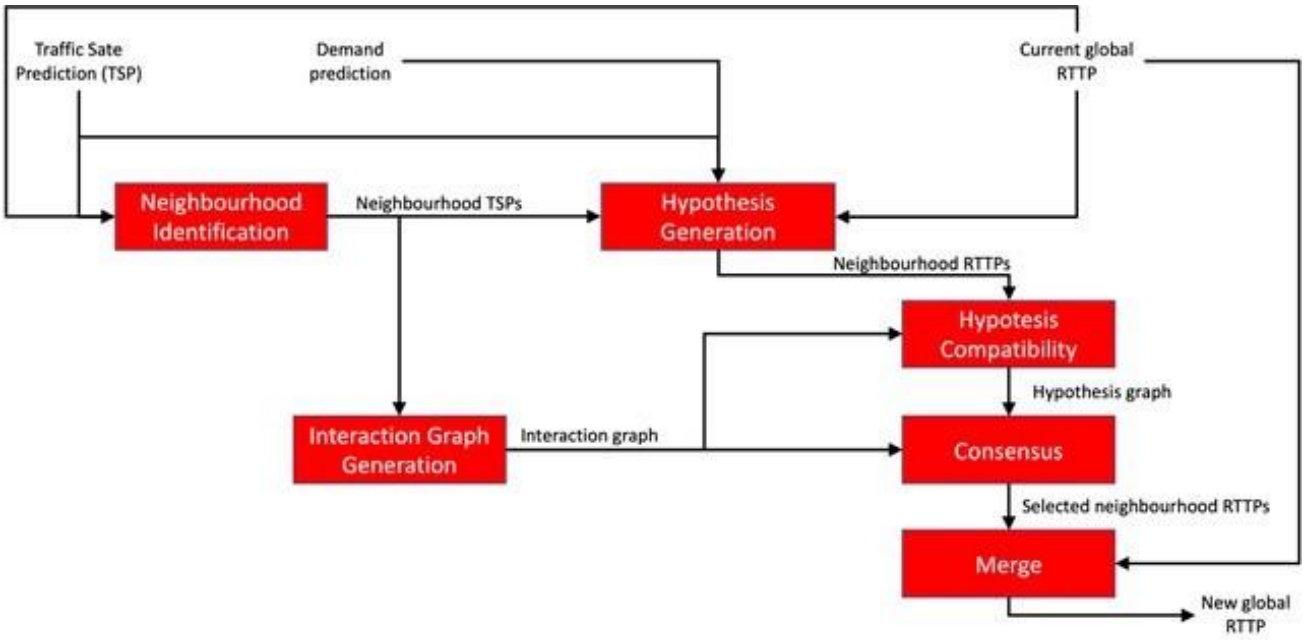

Figure 2: Schematics of the different modules of the pipeline for self-organised decision making on train traffic management.

- **Neighbourhood identification**: This module defines, for each train, which other trains should be considered for the local definition of schedules and routes. It takes as input the actual traffic state prediction and the previous RTTP, on which is based the selection of relevant trains (i.e., the neighbourhood of the focal train). The output is a modified Traffic State Prediction (TSP) where the neighbouring trains are identified, so that they can be involved in the consensus process.
- **Interaction graph generation**: This module generates a graph in which nodes are trains and edges exist when two trains are in the same neighbourhood. This is a convenient representation to support the consensus module and enables to mimic communication among autonomous trains.
- **Hypothesis generation**: This module is responsible for generating new scheduling and routing hypotheses for each train, taking into account the train neighbourhood and a train-specific objective function. The generation of hypotheses is implemented as a modified version of RECIFE-MILP, and uses the information about the schedule of trains outside the neighbourhood as a constraint, generating a number of hypotheses that will be negotiated during the consensus process. Each hypothesis is a

"neighbourhood RTTP", that is, a RTTP in which re-scheduling and re-routing is limited to the given neighbourhood.

- **Hypothesis compatibility**: This module determines whether the hypotheses from two trains are "compatible", that is, they provide solutions that can be merged without any issue. A hypothesis graph is a graph in which nodes represent hypotheses and edges connect hypotheses that are compatible and originate from trains belonging to the same neighbourhood.
- **Consensus**: This module implements the self-organised consensus strategy, whereby trains interact on a given network (the interaction graph) and hypotheses are confronted according to compatibility (the hypothesis graph). As a result of the consensus process, each train selects a single hypothesis to be deployed.
- **Merge**: This module takes as input the neighbourhood RTTPs generated by each train and selected by the consensus process, and produces a new global RTTP that will be then implemented by traffic control centre.

As mentioned above, the modules of this software pipeline actually emulate the processes that would be deployed in an actual decentralised system. Specifically, neighbourhood identification, hypothesis generation, hypothesis compatibility and consensus are processes that are executed in parallel by each train. Instead, the interaction graph generation is replaced by train-to-train communications limited to trains in the same neighbourhood for exchanging hypotheses.

4.1.3 Demand prediction

The demand prediction module consists of two parts, a prediction model for the origin-to-destination (OD) passenger demand and a model for assigning predicted demand to trains, thereby producing a prediction of the number of passengers in each train, their destinations and their route choice.

At the beginning of a short time interval e.g., of 5, 10 or 20 minutes, the prediction model forecasts OD passenger demand within this interval and a certain small number of intervals into the immediate future. Since the traffic management and control requires information on demand in terms of passengers in each train and the passengers' destinations, a passenger assignment model transforms the OD

prediction into a passenger assignment to individual trains in accordance with the real-time traffic plan (RTTP).

The input to the demand prediction model comprises:

- At the start of operations, historical records of observed demand for all OD pairs
- At the start of operations, historical records of observed traffic in network including realised timings at stations
- During operations, records of observed demand in terms of tap-in and tap-out locations and times of passengers in real-time
- During operations, records of observed and predicted traffic plans including realised and predicted timings at stations

The output from the prediction model is the estimated OD demand of travellers starting their journey in the current (and possibly near-future) time interval(s).

The prediction model is a deep learning model consisting of multilayer artificial neural networks and exists on two operational levels. The first level is the training model which essentially calibrates the parameters within the model with the goal of achieving the best generalized performance at deployment. The training is carried out using a large training data set and a smaller validation data set, each consisting of features related to historical demand observations as well as observations on reliability of supply such as frequency of service, train delays, and cancellations. The structure of the data associates several features with each time interval for which a demand prediction is needed. These features include observed demand from time intervals immediately preceding the one to be predicted as well as older observations providing context about the historical demand patterns.

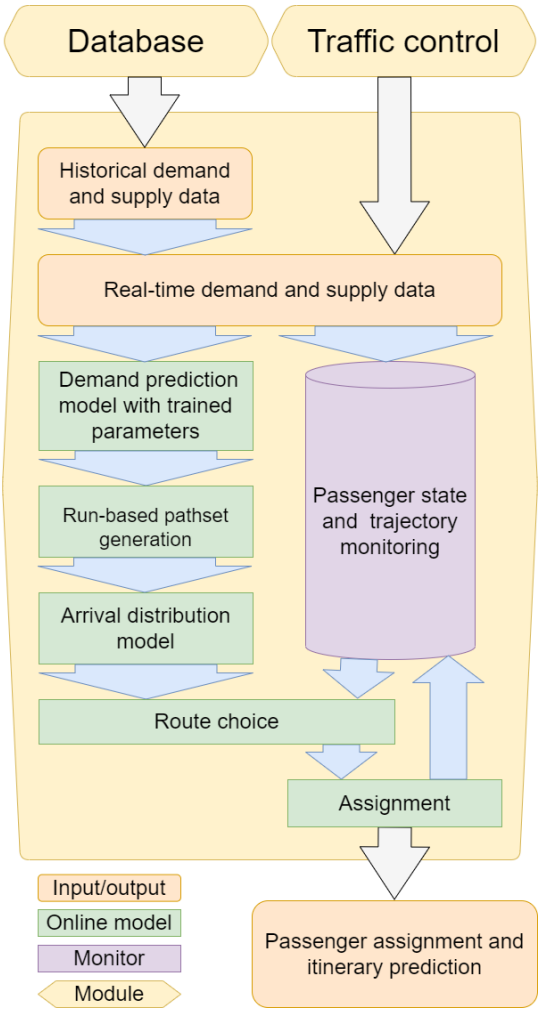

Figure 3: Demand prediction and assignment module

The features are extracted from the data such that no time interval features contain information that would only be available after the start of that interval, thereby simulating real-time information at each time interval. During training, the model is fed features for each time interval, the output of the model is compared to the target observed demand for that time interval, and an optimization algorithm uses the deviation from target to update the parameters of the model in an iterative manner. The generalized performance of the model is estimated by running the model on the validation data set which is not used for parameter optimization. The training process is time consuming but is only required to be re-run occasionally offline to update parameters with larger chunks of new data. The second level is the deployment model, which is used in real-time with the same feature types to predict the demand for the current time interval. The

deployment process is fast and is integrated with the traffic control algorithms in an online fashion.

As the control algorithms do not accept the OD demand directly as input and instead need information on the distribution of this demand over the trains in the system, an assignment model is built on top of the OD demand prediction model. The goal with the assignment model is to assign the OD demand for a time interval to specific trains in order to maintain information for each train of its passengers, their origins, destinations and their personalized journey plan, including start and end times of journey and transfers.

The model has several stages, see Figure 3**Error! Reference source not found.**. First, using knowledge on the static choice set of lines to take for a given OD pair, the model constructs a choice set of specific trains to take for each OD pair relevant to the passengers in the current time interval. Then, based on the train departures at the origin station, the distribution of arrivals of the passengers at the station for the OD demand is estimated. This will associate a subset of paths available to each group of passengers with the same OD and similar arrival time. Based on the real-time traffic plan and the arrival times, the discrete choice attributes of each path and group is computed. Finally, the OD demand estimated by the demand prediction model is distributed on these paths according to a discrete choice model. The resulting passenger distribution is used as input to the control algorithms along with the information about those passengers' itineraries.

4.2 Transport Simulation System

The Transport Simulation System is composed by three key simulation modules: (1) EGtrain, (2) Demand (re)routing and (3) SimMobility MT Pre-day, represented by the green, dark blue and light blue boxes in Figure 1, respectively. We note that all three modules, its variables used and associated interfaces, may not be used in every case-study of the SORTEDMOBILITY.

4.2.1 EGTrain

Environment for the desiGn and simulaTion of RAIlway Networks (EGTRAIN) is an object-oriented model developed in C++, which represents microscopic information about the infrastructure namely gradients, radii, speed limits, switches and stations, the vehicles (available tractive effort, deceleration rate, number of wagons, masses of the coaches), the signalling and the ATP systems (position of signals, aspects, ATP speed codes, braking behaviour), and the timetable (arrival/departure times at/from stations) (Quaglietta, 2011).

The high-level design of EGTRAIN incorporates four primary modules (Figure 4). These are:

1. **Infrastructure module**. Models the rail network as a link-oriented graph at the microscopic level. Tracks as the edges of the tracks contain the necessary attributes such as speed limits, gradients, curvature radii, as well as the coordinates of infrastructure elements (e.g., positions of signals and stations).

2. **Rolling stock module**. Rail vehicles are represented in a detailed object-oriented model considering both their physical and mechanical characteristics such as the "tractive effort-speed" curve of the traction unit, the maximum deceleration rate, the jerk value, as well as the train composition (number of wagons, masses of the coaches and the traction unit, etc.)

3. **Signaling system module**. In this module, the interactions between vehicles and signaling equipments are modeled, for different types of signaling systems: the Italian BACC system, ETCS level 1, and ETCS level 2. The signaling system assures the respect of safety requirements by train operations, regulating the movement of trains on the track.

4. **Timetable module**. Incorporates the Departure/Arrival times and minimum dwell times at stations, described in the Timetable. Moreover, it is possible to introduce disturbances to ordinary train operations, by imposing delays to a

specific train at a certain station. Furthermore, train dwell times at stations can be also modeled or train reorderings at stations.

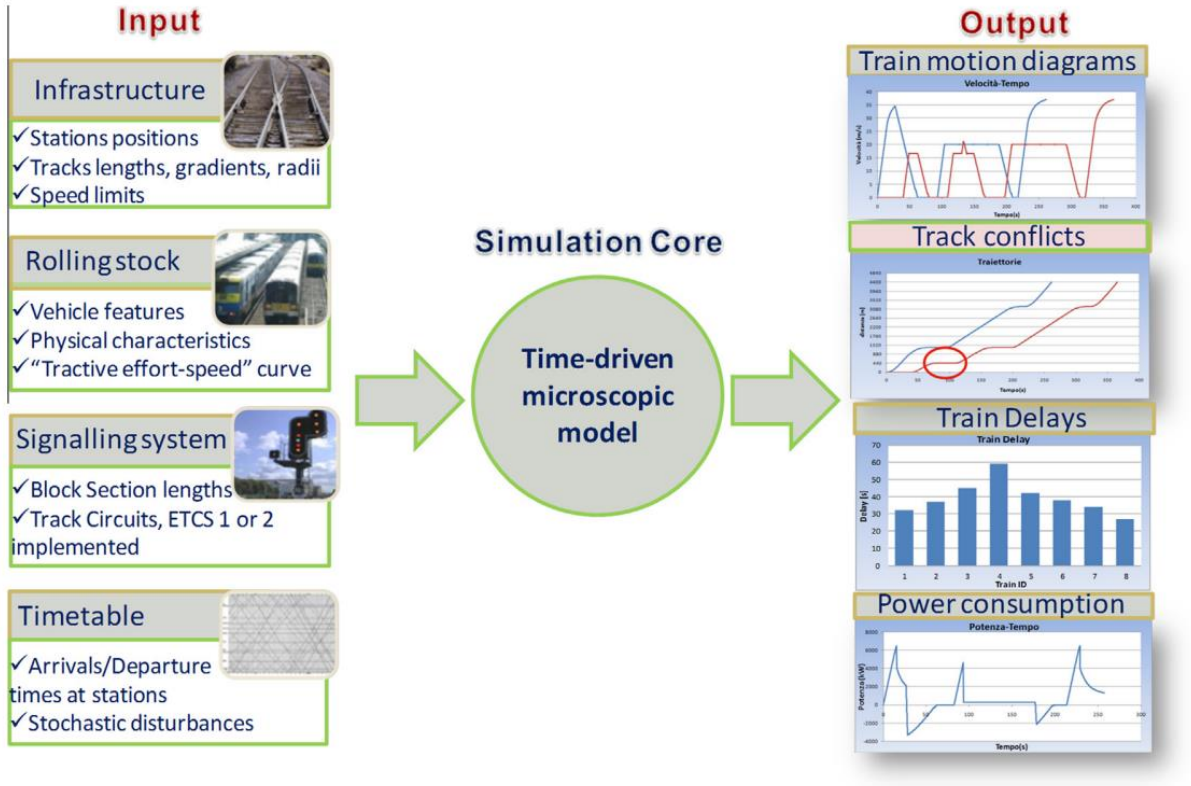

Figure 4: High-level design of EGTrain

EGTRAIN also provides a Graphical User Interface (GUI) (Figure 5). Using the GUI, it is possible to visualize the simulation, and explore all the input data. The GUI is built on top of EGTRAIN software and does not interfere with its process. The GUI only reads its data and controls its flow.

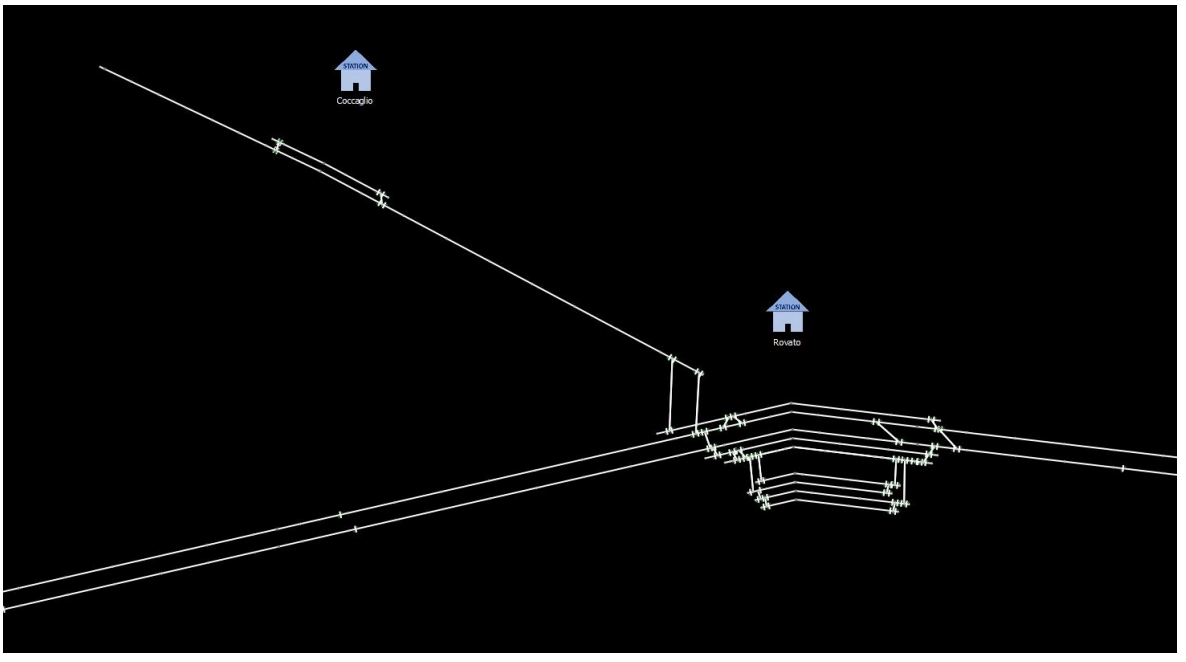

Figure 5: GUI of EGTrain

All simulation modules of EGTRAIN have been implemented in C++ using an object-oriented programming technique. The GUI is written in C++ and built using the Qt cross-platform development framework.

The input files are grouped in 4 categories:

1. Infrastructure files: they describe microscopically the topology and the attributes of the tracks (length, curvature, speed limits, station points, line switches)
2. Rolling stock files: the mechanical characteristics of the trains (tractive force, braking force, weight, energy consumption, passenger capacity)
3. Signaling systems files: system used (ETCS L1, L2, CBTC) topology of the signals
4. Timetable files: the arrival and departure times, dwell times at stations, frequency of the services.

4.2.2 Demand (re)routing

4.2.2.1 Background

Route choice models are needed to analyze, understand, and predict travelers' behavior under current and hypothetical conditions on transport networks. The developed route choice model will be integrated with EGTrain.

4.2.2.2 High-level structure

The route choice model is developed as a discrete choice model, a random utility model framework (McFadden, 1974). Such model assumes that each traveler associates a utility to each path alternative between an origin and destination and then selects, out of a set of discrete path alternatives, the one that maximizes his/her utility.

A route choice model has two main components: 1) pathset generation and 2) a route choice model estimation.

The first component consists of generating a set of possible route alternatives between each origin and destination while the second component calculates the probability of a given route being selected from the pathset.

Pathset generation

In the real world, when a traveler wants to commute by public transit (PT) to his/her trip destination, he/she does not plan the route on the road network. A traveler plans in-terms of which bus-line or train-line he/she needs to take, which stop or station he/she must board and alight and where he/she may need to transfer. In other words, a PT path is not the sequence of roads that starts at an origin location and leads to the destination. It is the sequence of access trip, public transit trip, transfers, and egress trip. Therefore, a PT graph, that is different from a road network graph, is needed to generate a PT pathset. A graph contains a set of vertices and a set of edges connecting pairs of those vertices.

The vertex set consists of:

- nodes from the road network
- bus stops in the network
- train and metro stations in the network.

The edge set which connects pairs of these vertices can be categorized into 3 edge types:

- bus edge: If a bus-line serves bus-stop and then eventually bus-stop b, there is an edge (a, b) in the edge set representing this bus-line service
- rail edge: If a train-line serves station c and then eventually station d, there is an edge (c, d) in the edge set representing this train-line
- walk edge: There is a walk edge connecting each node vertex to each bus-stop or train-stop vertex (and vice-versa) if the distance between the node and the PT stop is "walkable". These walk edges would represent the access and egress trips of PT paths. Similarly, there is a walk edge between every pair of PT (bus, train, or metro) stops that are at a "walkable" distance from one-another. These walk edges would represent the walk-transfers between those stops.

The following conceptual steps are accomplished to produce the pathset:

- The flows of travellers moving from one node to another in the network, based on the activities they have to accomplish, are computed. More precisely, an Origin-Destination matrix based on the Day Activity Schedule is generated. Note that the sequence of activities that a person decides to accomplish in the day translates into a sequence of trips from an origin node to a destination node. Each trip generates a flow between these nodes, that is added to the Origin-Destination matrix.
- Each of the trips mentioned above will be realized on a path, i.e., a sequence of links that starts at the origin of the trip and terminates at the destination. Therefore, for each trip, a pathset is generated (via specific algorithms such as shortest path, link elimination, and random perturbation), which is a set of paths connecting the origin to the destination of the trip. When a traveler will actually perform the trip, he/she will choose one of the paths in the pathset.

The pathset is generated offline, i.e. before the actual simulation of EGTrain and is case-specific. A graph needs to be generated a-priori using existing data (e.g.: GTFS) and SortedMobility follows the graph and pathset generation method presented in: https://github.com/smart-fm/simmobility-prod/wiki/Implementation---PT-Demand/.

Route choice model

The utility of a path alternative is specified as a linear-in-the-parameters function of, but not limited to, the alternative (path) attributes (e.g., in-vehicle travel time, walking time, waiting time, number of transfers etc.), in addition to a random term that represents the effect of unobserved variables.

The utility of a traveler n choosing a path j during time period t is expressed as:

$$U_{njt} = \beta_X X_{njt} + \varepsilon_{njt}$$

Where X_{njt} is a vector of path attributes related to alternative j at time t, β_X the corresponding vector of unknown parameters that need to be estimated statistically, and ε_{njt} a random disturbance term that is assumed to follow an independently and identically distributed (iid) Extreme Value Type I distribution over alternatives and travelers.

The key path attributes to be considered are, but not limited to, in-vehicle travel time, walking time during transfers, waiting time at each boarding stop/station, number of transfers along the path, and pathsize (the degree of overlap for each path within its path set). Note that access and egress walk times are not included as only stop-to-stop path selection is considered in the route choice model.

The probability of traveler n choosing path j during time period t can be then written as follows:

$$P_{njt} = \frac{e^{\beta_X X_{njt}}}{\sum_{j'=1}^{J_{nt}} e^{\beta_X X_{nj't}}}$$

Where J_{nt} is the set of available alternatives for traveler n at time t.

4.2.2.3 Platform

The mid-term module of SimMobility is used to generate the pathset while pandasbiogeme (Bierlaire, 2020) is used to estimate the route choice model.

4.2.2.4 Input

The inputs required for the pathset generation are: 1) a list of edges that make up the public transport graph (Table 1), 2) a list of vertices that make up the public transport graph (Table 2), and 3) an OD matrix (Table 3) that contains a

list of origin and destination node ids for which public transit paths are to be generated during pathset generation.

Table 1: PT edge

Column name	Description
start_stop	Stop id for the starting vertex.
end_stop	Stop id for the ending vertex.
r_type	Service Line type, can be "BUS", "TRAIN" or "WALK".
service_line	If the edge is a route segment, it will have the primary bus service line in that route segment. If it is a walking leg, it will have string "Walk".
road_index	Index for road type 0 for BUS, 1 for TRAIN/METRO, 2 for Walk.
stop_link_sequence	The sequence number of the stop at this edge along the service_line.
road_edge_id	Strings of passing road segments by current edge as specified in edge_id. Example:"4/15/35/43", each number corresponds to a road segment between two adjacent stops along service lines. Road_edge_id containing single "edge_id" are referring to the road segment between two adjacent stops along service lines as specified in R_service_lines. Road_edge_id containing more than 1 "edge_id" are virtual edges representing traversing the edges sequentially without a transfer in between.
r_service_lines	If the edge is a route segment, it will have bus service lines in that route segment. If it is a walking leg, it will have string "Walk".

Column name	Description
link_travel_time	Estimated travel time (in seconds) on current edge. If it is a transit leg, it is the estimated in-vehicle travel time on current route segment. If it is a walking leg, it is the estimated walking time on current walking leg.
edge_id	Id for current edge.
wait_time	Estimated waiting time to embark on current edge. Data obtained from scheduled service line information.
walk_time	Estimated walk time in the current edge, estimated based on direct distance and 4km/h walking speed in seconds.
transit_time	Estimated travel time on transit legs for bus and train/metro in seconds.
transfer_penalty	Transfer penalty used to impose penalty on transfers for shortest path searching which assumes path cost is the addition of edge attributes in seconds.
day_transit_time	Estimated transit time for buses and trains/metros in day time in seconds.
dist	Estimated distance from travel time table in kilometers.

Table 2: PT vertex

Column name	Description
stop_id	Stop id for stops as specified by public transport body. It can be bus stops, train/metro stations or SimMobility nodes (Starts with N_).
stop_code	Stop codes for stops specified by public transport body.
stop_name	Name of stop as per public transport body.
stop_lat	Latitude of the stop.
stop_lon	Longitude of the stop.
ezlink_name	EZlinkName of the stop, usually same as stop code.
stop_type	Type of stop, can be '0' for SimMobility nodes, '1' for Bus stops and '2' for train/metro Stations.
stop_desc	Description of stops. Usually, street where the stop is located.

Table 3: PT OD

Column name	Description
origin	Path origin node id.
destination	Path destination node id.

As for the route choice model, it requires two main inputs: a pathset (Table 4) which is an output of the pathset generation and an OD matrix that contains a list of origin and destination node ids (Table 3).

Table 4: PT Pathset

Column name	Description
pathset_origin_node	Node id of the path's origin node.
pathset_dest_node	Node id of the path's destination node.
scenario	Contains the algorithm used to generate the path and its iteration number.
path	Comma separated list of pt_edge ids that make up the path.
path_travel_time_secs	Estimated travel time to traverse the path in seconds.
total_distance_kms	Estimated distance traversed over path in kilometers.
path_size	The degree of overlap for each path within its path set.
total_cost	The computed total cost for traversing this path.
total_in_vehicle_travel_time_secs	Estimated travel time spent in a vehicle in seconds.
total_waiting_time	Estimated travel time spent waiting in seconds.
total_walking_time	Estimated travel time spent walking in seconds.
total_number_of_transfers	Total number of public transport transfers required while traversing this path.
is_min_distance	Indicates that this path is the minimum distance between the two nodes.
is_valid_path	Indicates that this path is valid.
is_shortest_path	Indicates that this path is the shortest between the two nodes.

Column name	Description
is_min_in_vehicle_traveltime	Indicates that this path is the minimum in vehicle travel time between the two nodes.
is_min_number_of_transfers	Indicates that this path has the minimum number of public transport transfers between the two nodes.
is_min_walking_distance	Indicates that this path has the minimum walking distance between the two nodes.
is_min_travel_on_mrt	Indicates that this path has the minimum travel on MRT between the two nodes.
is_min_travel_on_bus	Indicates that it the minimum travel on Bus between the 2 nodes.

4.2.2.5 Output

The output of the pathset generation is the pathset table presented in the previous section (Table 4) which is later used as an input for the route choice model.

The output of the route choice model is the vector of parameters β_X that can be used to estimate the probabilities of choosing each available alternative between each OD pair in the network.

4.2.3 SimMobility

4.2.3.1 Background

SimMobility is an open-source integrated agent- and activity-based simulation platform that has been developed by the Massachusetts Institute of Technology (MIT) and the Singapore-MIT Alliance for Research and Technology (SMART) since 2012 (Adnan et al., 2012). It supports the activity-based modeling paradigm and allows for individual behavior simulation in different time (from seconds to years) and spatial scales, simultaneously simulating demand and supply.

4.2.3.2 High-level structure (SimMobility)

The high-level design of SimMobility incorporates three primary modules (Figure 6), which consider different timeframes: long-term, mid-term and short-term.

The Long-term (LT) module simulates year-to-year dynamics, capturing long-term land use and economic changes. The Mid-term (MT) module simulates day-to-day dynamics of travel demand, i.e., agents' daily travel and activity patterns. The Short-term (ST) module simulates what happens within a day, i.e., the movement of agents at a microscopic granularity.

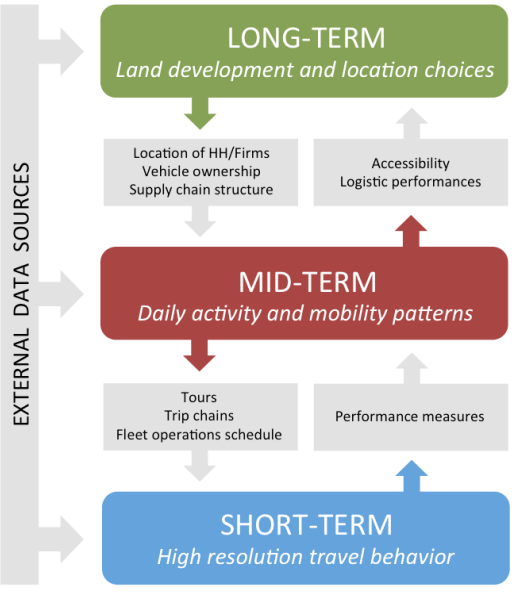

Figure 6: SimMobility Modelling Framework

4.2.3.3 High-level structure (SimMobility Mid-Term)

The Mid-term (day-to-day) simulator handles transportation demand for passengers and goods. It is a mesoscopic simulator that simulates agents' behavior which includes their activity and travel patterns using activity-based models, with explicit pre-day and within-day behavior including re-routing and re-scheduling, and multiple transport modes. The Mid-term module of SimMobility consists of three interacting simulators (Figure 7):

- Pre-day (which is used by SORTEDMOBILITY): simulates individuals' daily activity and travel patterns according to the activity-based models estimated. More details can be found below;
- Within-day: simulates both departure times and route choice behavior, allowing for re-scheduling depending on real-time network conditions (provided by the supply simulator);

- Supply: simulates the transport network and its attributes, including both public and private transport.

The Mid-Term simulator takes input in the form of synthetic population that contains detailed characteristics of each agent in the simulation region, and processes the day activity schedule of each agent. The pre-day models follow an enhanced version of econometric Day Activity Schedule approach to decide an initial overall daily activity schedule of the agent, particularly its activity sequence (including tours and sub-tours), with preferred modes, departure times by half-hour slots, and destinations. This is based on sequential application of hierarchical discrete choice models using a Monte-Carlo simulation approach. As the day unfolds, the agents apply the within-day models to find the routes for their trips and transform the activity schedule into effective decisions and execution plans. Agents may get involved in a multitude of decisions, not constrained to the traditional set of destination, mode, path and departure time depending upon their state in the event simulation cycle, such as re-routing in the middle of a trip (including alighting a bus to change route).

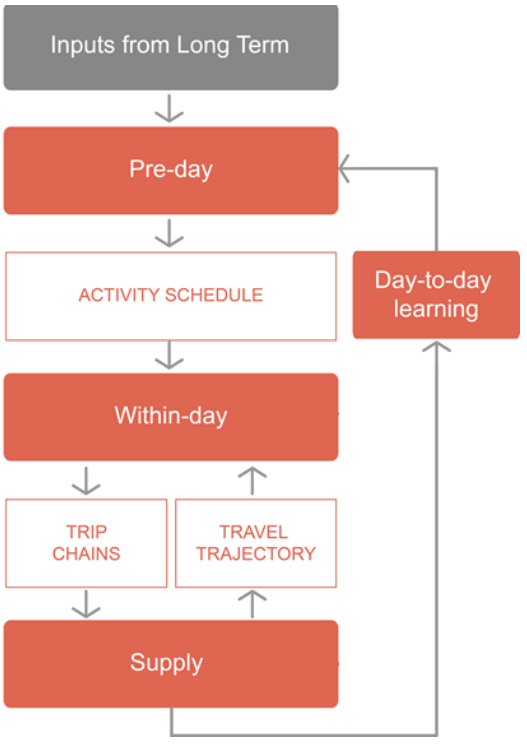

Figure 7: SimMobility Mid-Term Model

4.2.3.4 Platform

SimMobility is entirely developed in C++, using boost threads library, for parallelization. It is able to do runtime load balancing by taking advantage of individual agents' context (e.g., neighbor agents can be grouped together; agents of similar type can be grouped together). SimMobility takes advantage of state-of-the-art computational efficiency tools to increase scalability. The embeddable and light-weight scripting language LUA is used for implementing the models' specifications (Viegas de Lima et al., 2018) (Lu et al., 2015), due to its relatively simpler and more intuitive language syntax. For more information on installing and running simmobility, please refer to the following links:

- Running Mid Term · smart-fm/simmobility Wiki (github.com)
- Simmobility Installation · smart-fm/simmobility Wiki (github.com)

The next sub-sections will present the standard version of the Pre-day models, which are being slightly adapted for the case of Copenhagen.

4.2.3.5 Input

Mid-term pre-day has two modes of execution - logsum and simulation. Logsums are used as accessibility measures from lower- to higher-level choice models (Adnan et al., 2016). All necessary logsums from pre-day choice models are pre-computed in the logsum mode. The simulation mode uses these pre-computed logsums as it iterates through the system of choice models to build an activity schedule for each individual in the population.

The inputs required for the logsum mode are population and zone level inputs.

Population

The population data from SimMobility LT is an input for MT pre-day models. Every individual in the population is a potential source of travel demand. Various characteristics of individuals are used by the pre-day models to determine their activity schedule. Below (Table 5) is a list of attributes of each person that MT pre-day requires to be able to construct an activity schedule.

Table 5: Population

column	LT-table	description
id	individual	individual (person) id
employment_status_id	individual	person type id
gender_id	individual	gender of person
education_id	individual	student type category id (refer education table in LT database descriptions)
vehicle_category_id	individual	vehicle ownership type (refer vehicle_category table in LT database for descriptions)
age_category_id	individual	age category id (refer age_category table in LT database for descriptions)
income	individual	income of person
work_at_home	individual	whether the person works from home
car_license	individual	whether the person has licence to drive car
motor_license	individual	whether the person has licence to drive motor-cycle
vanbus_license	individual	whether the person has licence to drive heavy vehicles
time_restriction	job	whether the person has strict work hours
fixed_workplace	job	whether the person has a fixed work location
is_student	job	is the person a student
sla_address_id	establish-ment	address of primary activity location (school location if student, work location if employed)
id	household	id of household to which the person belongs
sla_address_id	fm_unit_res	person's home address id
size	household	size of person's household
child_under4	household	number of household members under 4 years of age
child_under15	household	number of household members under 15 years of age
adult	household	number of adults in household
workers	household	number of earning individuals in the household

Zone level

Pre-day model uses the following socio-demographic data for generating activity schedules:

- Traffic Analysis Zones (TAZ) – Table 6
- Skim matrices (Level of service information for 10 time intervals) – Table 7
- Departure and arrival time dependent TAZ-TAZ travel times for public transit and private vehicles interpreted from the skim matrices. There are two tables in the Simmobility database in the demand schema holding TAZ-TAZ departure and arrival-based travel times for every TAZ pair for each of the 48 intervals which consists in dividing the day into 30-minute intervals. The table tcost_car holds private vehicle travel times and the table tcost_bus holds public transit travel times.

Table 6: Traffic Analysis Zones (TAZ)

column	description
zone_id	TAZ id
zone_code	zone code
area	area of the TAZ
population	population in the TAZ
shop	number of shops within the TAZ
central	whether this TAZ in a central zone
parking_rate	parking rate in the TAZ
resident_workers	number of working individuals residing in the TAZ
employment	number of employed people in the TAZ
total_enrollment	number of students enrolled in schools within the TAZ
resident_students	number of students residing in the TAZ
cbd	whether this TAZ is in the central business district

Table 7: Skim matrices

column	description
origin_zone	origin TAZ code
destination_zone	destination TAZ code
distance	distance between the zones in km
car_cost_erp	Electronic Road Pricing cost between the zones
car_ivt	in vehicle car travel time in hours between the zones
pub_ivt	in vehicle public transit travel time in hours between the zones
pub_walkt	Public Transit (PT) access/egress walk time in hours between the zones
pub_wtt	public transit waiting time in hour between the zone
pub_cost	public transit cost between the zones
avg_transfer	average number of PT transfers between the zones
pub_out	public transit out of vehicle travel time in hours

The inputs required for the simulation mode are the abovementioned inputs population and zone level in addition to logsums.

Logsums

The logsum of a choice model is given by the equation below:

$$logsum_i = log\left(\sum e^{V_{ij}}\right)$$

Where *i* represents the i^{th} individual and *j* is the index of a choice-alternative of the model; the sum in the above equation is taken over all alternatives of the model.

Pre-day has a chain of choice models. These choice models are executed one after the other in a sequence for every individual in the population. The choices made by each model cumulatively help to build a complete activity schedule for an individual.

Some of these choice models use logsums from models that are executed later along the chain of models. For example, the day-pattern binary model, which

decides whether a person must stay at home or do activities, requires logsums from the day-pattern tour and day-pattern stops models. Both of these models are executed after the day-pattern binary model.

To avoid redundant computations, we pre-compute logsums for all individuals and store them before proceeding to the generation of activity schedules. Given a population for which activity schedules are to be generated, we pre-compute all necessary logsums for each individual in the population from pertinent models and store these values in a separate table in the demand schema of Simmobility database. This process is performed by the logsum computation mode of pre-day. These pre-computed logsums are then loaded as input for the generating activity schedules in the simulation mode of pre-day. Any choice model which needs a logsum easily look-up Table 8 of pre-computed logsums.

Table 8: Logsum

column	description
person_id	id of a person in the population. This column is the primary key of the table. It is used to lookup logsum values for the person.
work	logsum value from mode or mode-destination models for work tours
education	logsum value from tour mode model for education tours
shopping	logsum value from mode-destination model for shopping tours
leisure	logsum value from mode-destination model for leisure tours
personal	logsum value from mode-destination model for personal activities tours
escortother	logsum value from mode-destination model for escort tours
dp_tour	logsum value from day-pattern tour model
dp_stop	logsum value from day-pattern stops model

4.2.3.6 Output

The logsums generated from the pre-day logsum computation mode are directly updated in the database logsum table (Table 8).

As for the pre-day simulation mode, it generates a Day Activity Schedule (DAS) table (Table 9) which is basically a list of activities made by each person in a day along with additional information for each activity to infer the trip made for that activity.

Table 9: Day Activity Schedule (DAS)

column	datatype	explanation
person_id	string	person id
tour_no	integer	
tour_type	string	Work/Education/Shop/Leisure/Personal/Escort
stop_no	integer	
stop_type	string	Work/Education/Shop/Leisure/Personal/Es-cort/Home
stop_location	integer	SimMobility node id
stop_zone	integer	zone code for TAZ chosen from Pre-day
stop_mode	string	the mode of travel opted by the person to arrive at this activity location
primary_stop	boolean	whether this stop is the primary stop of tour
arrival_time	numeric	Start time of the activity. Each of the numbers (3 to 26).(25 or 75) represent the midpoint of each half-hour window of the day starting from 3AM. E.g. 4.75 represents [4:30,4:59] AM win-dow.
departure_time	numeric	End time of the activity
prev_stop_location	integer	SimMobility node id, so we know the trip origin to arrive at this activity location
prev_stop_zone	integer	zone code for TAZ where the previous stop was located
prev_stop_depar-ture_time	numeric	End time of the last activity.

5 SOFTWARE INTERFACES

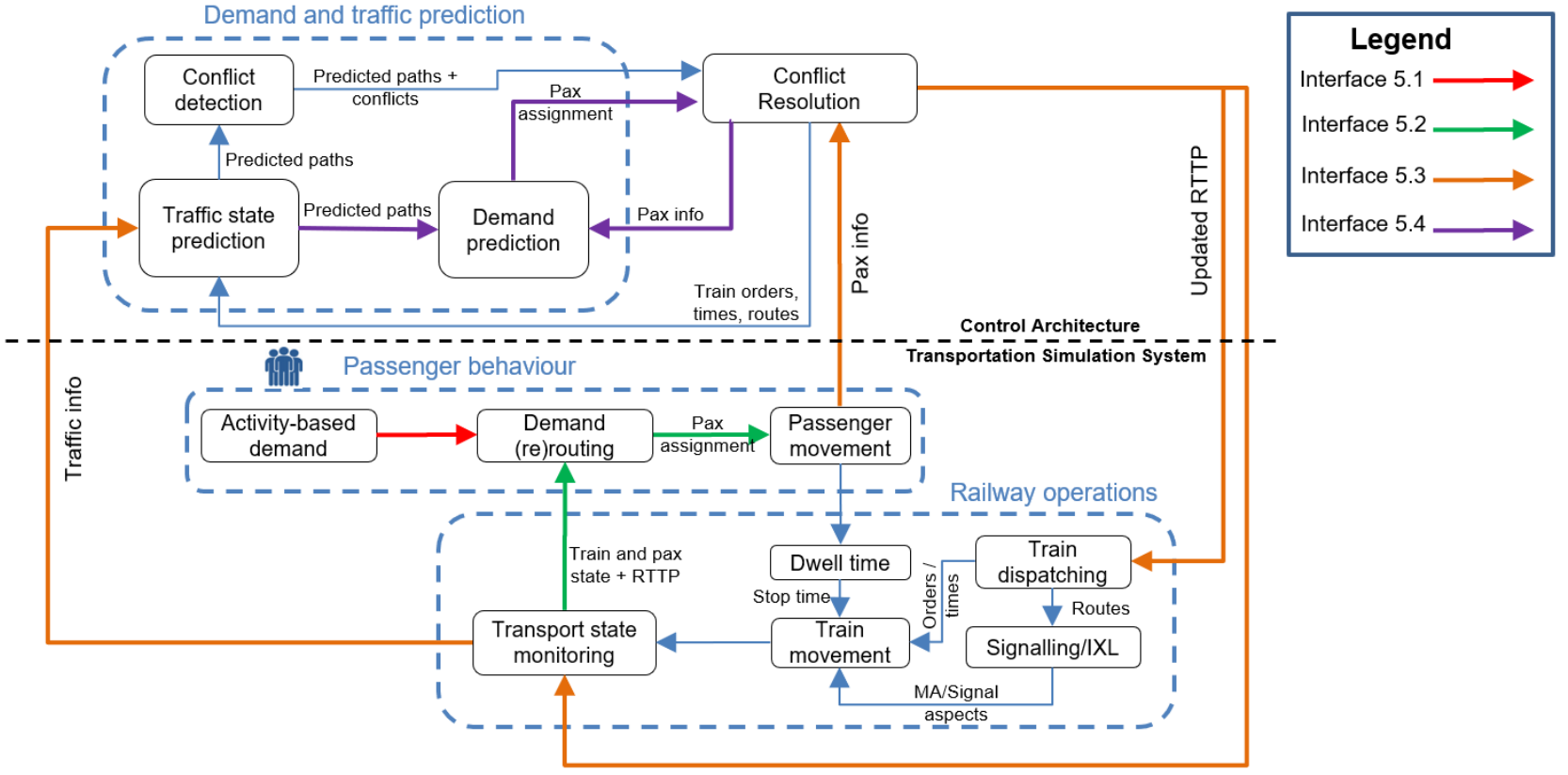

Figure 8: Software Interfaces

5.1 EGTrain and planned demand

Individuals' daily activity schedule (demand) and time-dependent origin and destination (OD) level of service matrices (rail performance) are the data interfaces between planned demand (SimMobility MT Pre-day) and EGTrain and vice-versa respectively. For the first (demand from SimMobility to EGTrain), the data is processed as origin-destination matrices, which are loaded as a passenger object in EGTrain. On the other direction, trajectories from simulated trains at the end of an EGTrain simulation are processed into zone-to-zone level of service as formatted in Table 7.

5.1.1 Module Interaction Process

Initially, the pre-day models are estimated, which consist of interconnected discrete choice models that model individuals' demand for travels as derived from their needs to perform activities, representing individuals' choices at several dimensions. They are estimated for a year of reference based on data from (i) the Danish National Travel survey (TU survey), which encompasses all travels (grouped as tours) performed by each individual during an entire day, (ii) network skims from the Danish traffic model (Landstrafikmodellen), which are level of service matrices describing car and public transport travel times and costs for different time intervals during a day of the reference year and (iii) zone level data, which includes information on area, population and number of jobs of each zone considered in the model. Based on the pre-day models estimated, the pre-day simulator (agent-based demand simulator), generates detailed activity and travel plans for each individual from a synthetic population. The output of the pre-day simulator is a database with daily activity-schedules for everyone in the synthetic population and it serves as input to EGTrain for simulating railway traffic (Table 9). If changes in rail supply are incorporated into the simulation, it will affect public transport level of service (travel times and costs), and consequently, individuals' choices. New level of service matrices can be generated for rail and aggregated by zones (python script) to serve as an input for a pre-day simulation of the travel demand for the new rail scenario. The pre-day models can also be re-estimated, according to the new rail scenarios.

5.1.2 Data Content

The data generated by the pre-day simulation that feeds the EGTrain simulation consists of detailed activity and travel plans for each individual from a synthetic population according to the pre-day models estimated. For each individual, it contains: (i) daily plan of out-of-home activities (if any), as well as their location, duration and purpose, and (ii) daily plan of travels (if any), as well as the

transport modes used to perform them and the time when they happen (Table 9).

The data from the EGTrain simulation to the pre-day simulation consists of information on travel times and costs for traveling by rail at different time intervals during the day. This information must be aggregated into zones to be used in the pre-day simulation.

5.1.3 Data Format and Exchange Interface

The pre-day simulator module receives data in the CSV format (Table 9). The new network skim matrices for rail, generated by the EGTrain simulation, must update the original skim matrix document with revised information on: rail in-vehicle travel time in hours between two zones, rail access/egress walking time in hours between two zones, rail waiting time in hour between two zones, rail costs of travels between two zones, average number of rail transfers for travelers between two zones and rail out of vehicle travel time in hours between two zones. This list may be extended later within the SortedMobility project to include level-of-service measures of interest to the self-organizing system (e.g.: reliability measures).

5.2 EGTrain and route choice

5.2.1 Module Interaction Process

EGTrain takes as inputs the pathset table and the route choice model (defined in Section 4.2.2). Interaction during simulation takes place by exchanging the simulated arrival and departure times at the Timetable points from EGTRAIN to the route choice model (Demand (re)routing in Figure 1).

The interaction diagram is illustrated in Figure 9. EGTRAIN is updating the dwell times of trains based on the Gibson dwell time model (Fernandez et al., 2008).

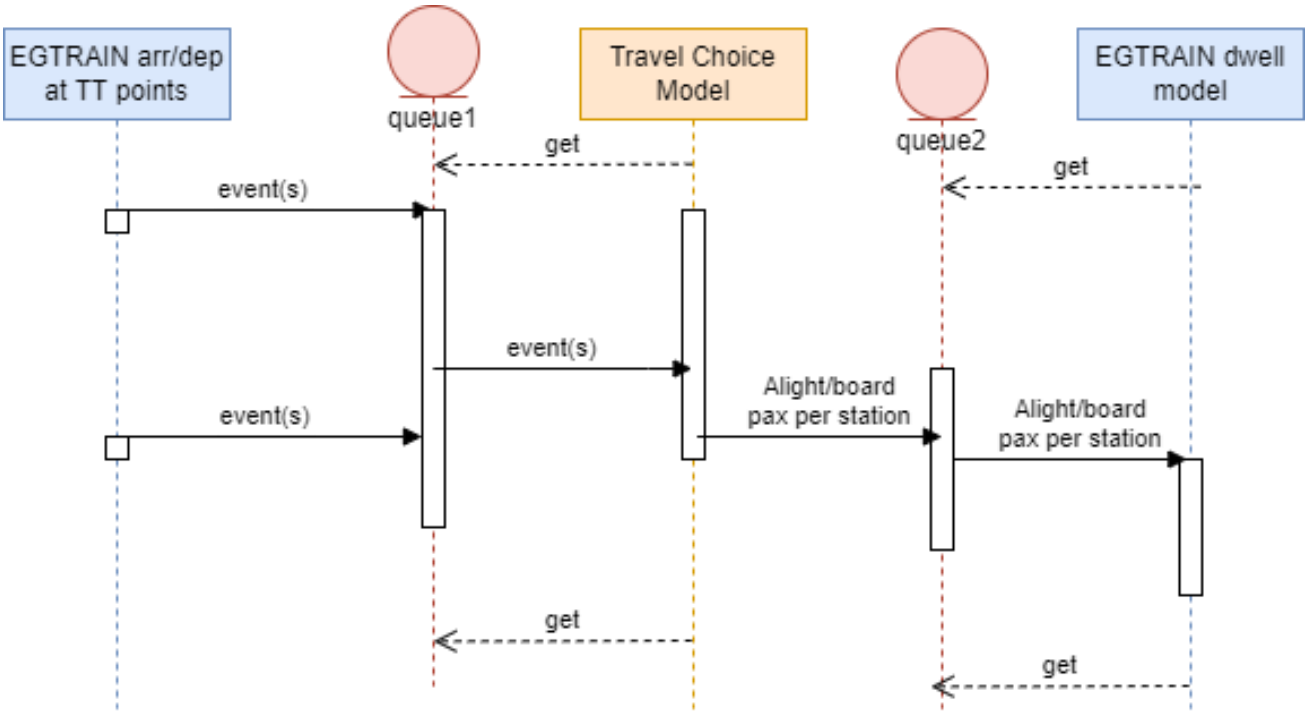

Figure 9: Integration diagram EGTrain and route choice model

5.2.2 Data Content

The interface between EGTRAIN and the route choice model allows the ex-change of two sets of information. In one direction, each passenger object in EGTRAIN may call the route-choice function at predefined triggering events. The passenger needs to pass origin, destination (stations), and (travel or re-routing) start-time to the route choice function. The latter also needs to have access to the latest RTTP. In the opposite direction, the route choice model sends the selected sequence of trains to be taken by the passenger.

5.2.3 Data Format and Exchange Interface

The data exchange is based on an XML formalisation that has been developed for representing the current traffic state and the RTTPS in the ON-TIME project (Quaglietta et al. 2016). A dedicated API will be created for the calling of the route-coice function within EGTrain.

5.3 EGTrain and control architecture

Exchanges between EGTrain and the control architecture will be exactly the same disregard the control method used: either RECIFE-MILP centralized or SORTED-MOBILITY self-organizing traffic management.

5.3.1 Module Interaction Process

The software modules will interact at the beginning of the simulation, to allow the instantiation of the control architecture. Then, throughout the simulation, the

modules will interact in a closed-loop fashion, similar to the approach introduced in the EU FP7 project Optimal Networks for Train Integration Management across Europe - ONTIME (https://cordis.europa.eu/project/id/285243). In particular, EGTrain shares information on the observed (and/or predicted) train and passenger travels. It does so either with a fixed time periodicity, or when a triggering event occurs. Based on this information, the controller makes decisions on the traffic management strategy, and it shares it with EGTrain for implementation. Such closed-loop can be deployed for any time duration, always making decisions based on the latest available information.

5.3.2 Data Content

The interface between EGTrain and the control architecture aims at sharing two types of information. On the one hand, static information concerning the railway infrastructure, planned timetable information and rolling-stock characteristics. On the other hand, dynamic information on the traffic and demand state, and on traffic management decisions to be implemented. Static information and dynamic information on traffic and demand state are sent by EGTrain and taken as input by the control architecture. Dynamic information on traffic management decisions are sent by the control architecture and used as input by EGTrain.

5.3.3 Data Format and Exchange Interface

Static information is shared in the RECIFE data format. It is an xml representation compatible with railML. It is based on three sets of data: infrastructure, timetable and rolling stock. The former includes all microscopic information on tracks. Namely, it reports all details on track detection sections, block sections, signals, stopping points and possible journeys. Journeys are sequences of block sections that can be traversed by trains. As a further specification of journeys, journey instances are possible ways to travel along a journey: depending on the used rolling-stock and stopping pattern, journey instances include information on running and clearing times on each track detection section, as well as sequences of track detection sections that are occupied by a train simultaneously, depending on the train length. The set of data on timetable includes the types and list of courses (services) that need to be performed in a day, or any time horizon. For each of them, information supplied includes the type of rolling stock, the planned entry and exit time from the infrastructure, the planned arrival and departure times at stops, and the set of journey instances that can be used to perform the course. In the timetable, planned rolling-stock re-utilizations and passenger connections are also detailed. Finally, the set of data on rolling-stock reports all information necessary to compute train motion equations, as well as data as train

lengths. A detailed description of the static data format is available at http://recife.univ-eiffel.fr/sharedData/data_format_documentation/ .

Dynamic information is shared following the data formats defined within the ONTIME project and detailed in Quaglietta et al. (2016). This format follows an xml representation as for dynamic data. It includes two main sets of data.

On the one hand, the traffic state prediction details train trips. An example is reported in Figure 10. Here, two trains (T1 and T2) are in the infrastructure considered. For each of them, the data indicate the track detection section in which the train is at the considered time and at which time the train accessed it. For trains which are expected to enter the considered infrastructure in the near future (T3 in the example), the expected entrance time is reported. This data format can be used to describe both the current traffic state or a predicted one. The predicted traffic state is the one on which decision making is based: as a few minutes are considered available to decide the traffic management strategy to implement, the initial state considered is the one expected to realize in these few minutes. In a practical deployment, the predicted one is to be used, and it can be obtained with a traffic simulator. In a laboratory environment, either the prediction is produced with a separate simulator run, or the simulator actually used to replace reality can be paused to allow decision making, and the current state can be considered equal to the traffic state. Besides the traffic state as done in the ONTIME project, in SORTEDMOBILITY we also include the demand state in this file. To do so, we add information on: origin and destination of each person's trip, start and end time of the trip, and trains used to perform it.

On the other hand, the Real Time Traffic Plan (RTTP) includes information on train entrance time on each track detection section it is planned to traverse. Entrance times are expressed in seconds. An example is reported in Figure 11. A so-called train view shows the sequence of track detection sections which compose the journey chosen for the train. An infrastructure view shows the same information from the perspective of each track detection section, making train passing orders explicit. Remark that block-sections are also mentioned in the RTTP with the name "route", following the British convention.

All the infrastructure data format described are defined for a fixed-block signalling system. In a moving-block context, in which track detection sections, block sections and signals are not defined, we adapt this definition considering an infrastructure representation based on track discretization.

```
<trafficState currentTime="2022-10-21 01:02:00.0 CET">
<trainStateInArea>
    <trainID trainNumber="T1"/>
    <trainPosition currentTrackVacancyDetectionSection="TDS013" lastOccupation-
Time="2022-10-21 01:01:30.0 CET"/>
    </trainStateInArea>
<trainStateInArea>
    <trainID trainNumber="T2"/>
    <trainPosition currentTrackVacancyDetectionSection="TDS156" lastOccupation-
Time="2022-10-21 01:01:20.0 CET"/>
  </trainStateInArea>
<trainStateOutArea>
    <trainID trainNumber="T3" expectedEntranceTime="2022-10-21 01:07:30.0 CET"/>
</trainStateOutArea>
```

Figure 10: Example of traffic state prediction (TSP)

```
<rTTP>
   <rTTPTrainView>
    <rTTPForSingleTrain journeyId="DC1D1D___CL" trainId="T004005 ">
         <tDSectionOccupation tDSectionID="TDS120" trainID="T004005" occupation-
Start=”3720” routeId="R95" trainSequenceID="0"/>
         <tDSectionOccupation tDSectionID="TDS112" trainID="T004005" occupation-
Start="4117" routeId="R81" trainSequenceID="1"/>
         <tDSectionOccupation tDSectionID="TDS111" trainID="T004005" occupation-
Start="4139" routeId="R81" trainSequenceID="2"/>
     </rTTPForSingleTrain>
    </rTTPTrainView>
   <rTTPInfrastructureView>
    <rTTPForSingleTDSection tDSectionId="TDS081">
         <tDSectionOccupation tDSectionID="TDS081" trainID="T1" occupation-
Start="3727" routeId="R50" trackSequenceID="0"/>
         <tDSectionOccupation tDSectionID="TDS081" trainID="T2" occupation-
Start="5989"routeId="R52" trackSequenceID="1"/>
         <tDSectionOccupation tDSectionID="TDS081" trainID="T8" occupation-
Start="7078" routeId="R52" trackSequenceID="2"/>
       </rTTPForSingleTDSection>
   </rTTPInfrastructureView>
 </rTTP>
```

Figure 11: Example of Real Time Traffic Plan (RTTP)

5.4 Control architecture and online prediction

The traffic control module exploits online demand prediction to make the best traffic management decisions. This holds for both RECIFE-MILP centralized and SORTEDMOBILITY self-organizing traffic management.

5.4.1 Module Interaction Process

The software modules interact at the beginning of the decision-making process and during the process itself. The control module is in charge of triggering demand prediction. At the beginning, the demand prediction based on the current TSP and RTTP (Section 5.3.3) is used to instantiate the control architecture. While decisions are made, new predictions are triggered based on new possible RTTPs to assess passenger reactions to traffic management strategies being considered. In particular, the online demand prediction module shares information about the assignment of passengers to individual trains given the timetable represented by an RTTP. Based on this information, the control module includes indicators related to the perceived level of service from the passenger's point of view in the rtRTMP (Real-time Railway Traffic Management Problem) formulation. The traffic control module includes passenger information at a high level of detail to preserve a high computational efficiency in real-time.

5.4.2 Data Content

The data sent by the control architecture to the online demand prediction module include the observed or predicted passenger state from the TSP and a train timetable. The latter is an extraction from the RTTP only including arrival and departure times at station platforms. The data returned by the online demand prediction module to the control architecture include the information on the trips of homogeneous groups of passengers (with same origin, destination and route, and similar arrival time at the origin station).

5.4.3 Data Format and Exchange Interface

The online demand prediction module uses data in the standard format General Transit Feed Specification (GTFS). In general, a GTFS feed is a collection of several CSV files. In this report, we limit the description to the files used in the online demand prediction module and its interaction with the traffic control module. For more information about the GTFS, see https://gtfs.org.

The GTFS files used in the online demand prediction module are called: routes, stops, trips, and stop_times. The files *trips* and *stop_times* are dynamic documents provided by the traffic control module. The file *trips* includes the list of all train courses (trip_id), each with the assigned route (route_id). The file *stop_times,* instead, corresponds to an extract of the RTTP which includes for

each train course: the planned time of arrival at given stop (arrival_time); planned time of departure from given stop (departure_time); unique id of the stopping location (stop_id), i.e., station or platform chosen; position of stop in sequence (stop_sequence) tied to the course id.

The static documents *routes* and *stops* include, respectively, the list of routes (sequences of portions of journey instances, using the wording of Section 4.1.1) and stops within the transport system considered. These documents are used in combination to the dynamic ones in the online demand prediction module to extrapolate and define available run-paths for passengers related to lines (routes) as well as transfers.

Therefore, the demand prediction module provides to the traffic control module the Passenger Assignment Plan (PAP) in an xml representation compatible with RECIFE data format. The PAP includes the list of homogeneous groups of passengers and the following details for each group: origin and destination station, number of passengers in the group, departure time from the origin, arrival time to destination, reference and alternative passenger journey instances (run-paths). A passenger journey instance is a sequence of courses (train services) that can be used by a passenger group to travel from origin to destination at the desired time. For each passenger journey instance, the PAP details the first and last train and the list of intermediate passenger-connections. Each passenger-connection includes the following information: name of the arriving train and departing train; name of the station where the trains are connected; minimum transfer time and maximum tolerance time, i.e., how long the departing train can/should wait at maximum for the feeder train.

References

D. McFadden. Conditional Logit Analysis of Qualitative Choice Behavior. In: Zarembka P. (Ed.), Frontiers in Econometrics, New York: Academic Press, 105–142, 1974. https://eml.berkeley.edu/reprints/mcfadden/zarembka.pdf

E. Quaglietta, P. Pellegrini, R.M.P. Goverde, T. Albrecht, B. Jaekel, G. Marlière, J. Rodriguez, T. Dollevoet, B. Ambrogio, D. Carcasole, M. Giaroli, and G. Nicholson. The ON-TIME real-time railway traffic management framework: A proof-of-concept using a scalable standardised data communication architecture. Transportation Research Part C: Emerging Technologies, 63:23–50, 2016.

E. Quaglietta. A Microscopic Simulation Model For Supporting The Design Of Railway Systems: Development And Applications. University of Naples Federico II, Italy, 2011.

I. Viegas de Lima, M. Danaf, A. Akkinepally, C. L. Azevedo, M. Ben-Akiva. Modeling Framework and Implementation of Activity- and Agent-Based Simulation: An Application to the Greater Boston Area. Transportation Research Record: Journal of the Transportation Research Board, 2672(49).

M. Adnan, F.C. Pereira, C. L. Azevedo, K. Basak, M. Lovric, S. Raveau, Y. Zhu, J. Fereira, C. Zegras, M. Ben-Akiva. SimMobility: A multi-scale integrated agent-based simulation platform. Presented at the Transportation Research Board 95th annual meeting, Washington, D.C., USA, 2016

M. Bierlaire. A short introduction to PandasBiogeme. Technical report TRANSP-OR 200605. Transport and Mobility Laboratory, ENAC, EPFL, 2020.

P. Pellegrini, G. Marlière, R. Pesenti, and J. Rodriguez. RECIFE-MILP: an effective MILP-based heuristic for the real-time railway traffic management problem. Intelligent Transportation Systems, IEEE Transactions on, 16(5):2609–2619, 2015

R. Fernandez, D. Campo, C. Swett. Data collection and calibration of passenger service time models for the TRANSANTIAGO system. Presented at the European Transport Conference, Noordwiijkerhout, Netherlands, 2008

Y. Lu, M. Adnan, K. Basak, F. C. Pereira, C. Carrion, V. H. Saber, H. Loganathan, M. Ben-Akiva. SimMobility Mid-Term Simulator: a state of the art integrated agent based demand and supply model. Presented at the Transportation Research Board 94th annual meeting, Washington, D.C., USA, 2015