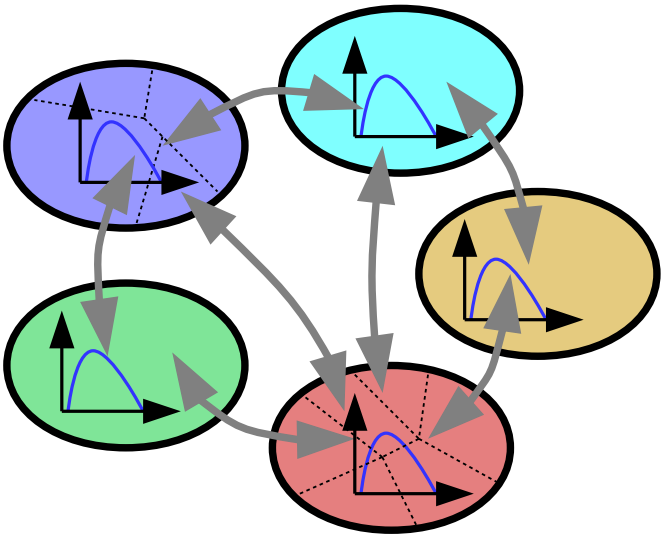
SymuRes 1.0

A Multi-Reservoir MFD-Based Traffic Simulation Platform



User Notice

Acknowledgements

This simulation platform was developed during the PhD of Guilhem Mariotte. It also received contributions from the PhD thesis of Sérgio F. A. Batista for the traffic assignment part. It is one of the outcomes of the MAGnUM project, held by Prof. Ludovic Leclercq and funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 646592).

magnum.ifsttar.fr

guilhemmariotte.com

User Notice

SymuRes 1.0

A multi-reservoir MFD-based traffic simulation platform

Guilhem Mariotte

Notice version 2020.1.0

Compiled on March 22, 2020

Contents

1.

General overview

1.1 What is SymuRes?

SymuRes is a traffic simulation platform for modeling regional urban traffic dynamics in a large metropolitan area. It is based on the concept of the Macroscopic Fundamental Diagram (MFD) that relates the vehicle space-mean speed to the density of vehicles traveling in an urban region. Unlike classical microscopic traffic simulation, this approach describes traffic states at an aggregate level: the individual position and speed of vehicles are not known, only their current regional location and the regional mean speed shared by all vehicles traveling in the same region. This platform is the outcome of the PhD thesis and publications of Guilhem Mariotte for the main part and the traffic flow solver, and of Sérgio F. A. Batista for the traffic assignment models. The relevant publications are listed below.

- Mariotte, G., Leclercq, L., Batista, S. F. A., Krug, J. & Paipuri, M. (2020). Calibration and validation of multi-reservoir MFD models: A case study in Lyon. *Transportation Research Part B: Methodological*, accepted for publication
- Mariotte, G. & Leclercq, L. (2019). Flow exchanges in multi-reservoir systems with spillbacks. *Transportation Research Part B: Methodological*, 122:327–349 https://doi.org/10.1016/j.trb.2019.02.014
- Batista, S. F. A., Leclercq, L. & Geroliminis, N. (2019). Estimation of regional trip length distributions for the calibration of the aggregated network traffic models. *Transportation Research Part B: Methodological*, 122:192–217 https://doi.org/10.1016/j.trb.2019.02.009
- Batista, S. F. A. & Leclercq, L. (2019). Regional Dynamic Traffic Assignment Framework for Macroscopic Fundamental Diagram Multi-regions Models. *Transportation Science*, 53(6):1563–1590 https://doi.org/10.1287/trsc.2019.0921
- Mariotte, G., Leclercq, L. & Laval, J. A. (2017). Macroscopic urban models: Analytical and numerical investigations of existing models. *Transportation Research Part B:*

Methodological, 101:245–267
https://doi.org/10.1016/j.trb.2017.04.002
- Mariotte, G. (2018). Dynamic Modeling of Large-Scale Urban Transportation Systems. Thèse de l'Université de Lyon, NNT: 2018LYSET010
https://tel.archives-ouvertes.fr/tel-02156187
- Batista, S. F. A. (2018). Dynamic traffic assignment for multi-regional transportation systems considering different kinds of users' behavior. Thèse de l'Université de Lyon, NNT: 2018LYSET009
https://tel.archives-ouvertes.fr/tel-02366532

1.2 Introduction of the variables used

SymuRes uses a specific aggregated network (also called *reservoir network*) to characterize traffic states. This one is defined after some key features found in real networks (topology and travelers' mobility). It consists of the following elements.

1.2.1 Reservoirs

The urban area to study is split into N_R *reservoirs*. A *reservoir* (referred to with subscript r) consists of a given urban region, defined as a connected and relatively compact set of links. As shown in Figure 1.1, every link and node belongs to a unique reservoir, except from the nodes at a border between two reservoirs that belong to both of them.

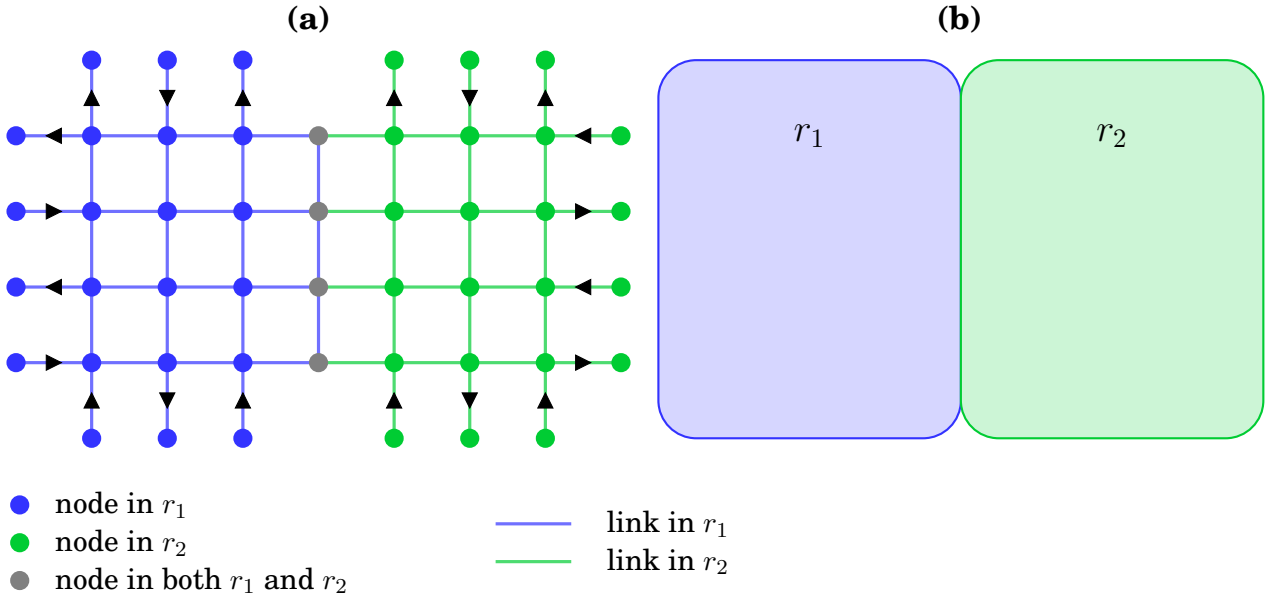


Figure 1.1. *Network split into 2 reservoirs r_1 and r_2. (a) Real network with directional links and nodes, and (b) schematic representation of the reservoirs.*

A reservoir r is an elementary block for the simulation of traffic states in SymuRes. Its state is defined by the evolution of its total accumulation $n^r(t)$ [veh], i.e. the total number of vehicles traveling in it at time t [s]. Its behavior in transient and steady state is characterized by a unique, unimodal and bell-shaped function $P^r(n^r)$ [veh.m/s]

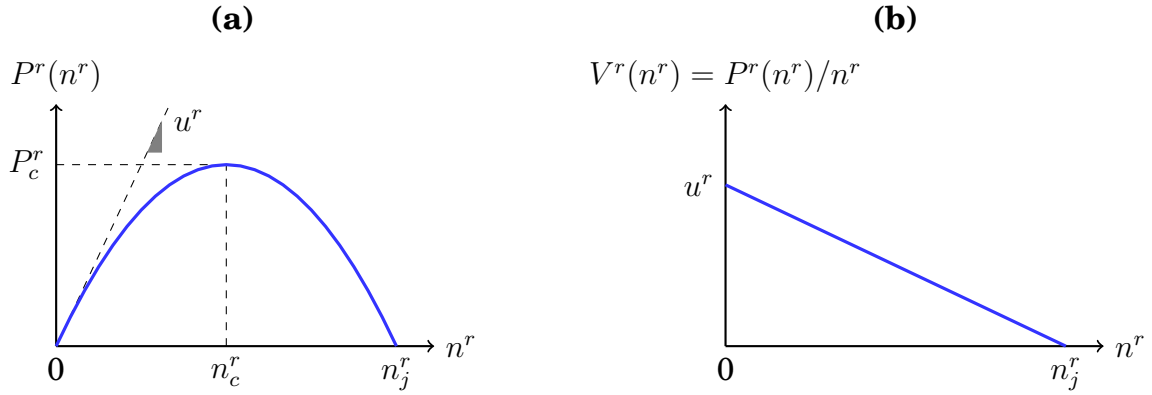


Figure 1.2. *(a) Example of a parabolic production-MFD function, and (b) its equivalent speed-MFD function.*

called production-MFD (analog to the link Fundamental Diagram concept in classical microsimulation). Alternatively, a monotonic decreasing function $V^r(n^r)$ [m/s] called speed-MFD can be used, as we have $P^r(n^r) = n^r.V^r(n^r)$. The latter represents the mean speed shared by all circulating vehicles for a given number n^r of them. Figure 1.2 displays an example of a parabolic production-MFD function, where several characteristics values are shown: the free-flow speed $u^r = V^r(0)$, the critical accumulation n_c^r for which the production $P_c^r = P^r(n_c^r)$ is maximum, and the jam or maximum accumulation n_j^r for which the speed is null (global jam or gridlock state).

1.2.2 Macroscopic nodes

The interfaces between the reservoirs and/or the outside perimeter are represented by *macroscopic nodes*. These ones ensure the incoming and outgoing transfers of vehicles in the reservoirs. A *macroscopic node* (or *node* when there is no ambiguity, referred to with m subscript) corresponds to a set of several nodes from the real network. It gathers all the flows transferring through these nodes. As illustrated in Figure 1.3, five types of macroscopic nodes are distinguished:

- *internal origin*: represent a set of nodes inside the reservoir where individual trips are created (flow generated inside the reservoir)
- *internal destination*: represent a set of nodes inside the reservoir where individual trips end (flow ended inside the reservoir)
- *external entry*: represent a set of nodes at the reservoir perimeter where individual trips come from outside
- *external exit*: represent a set of nodes at the reservoir perimeter where individual trips go to outside
- *border transfer*: represent a set of nodes at a border between two reservoirs for one way of transfer

Each reservoir has at least an internal origin and an internal destination macroscopic node. Two reservoirs are *adjacent* if they share a common border. A border is defined by at least one border transfer macroscopic node that belongs to both reservoirs, which is directional (i.e. corresponds to transfer from one reservoir to the other). For

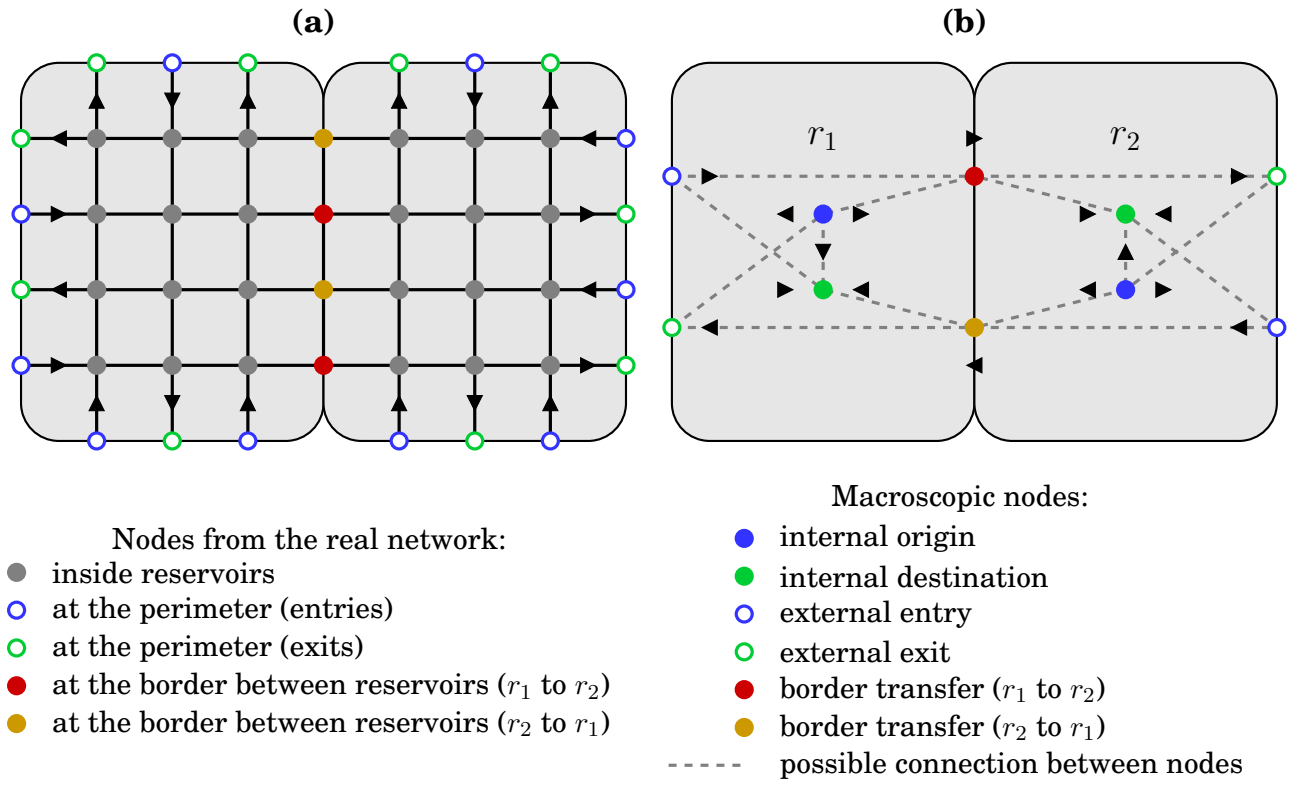


Figure 1.3. *(a) Node categories in the real network, and (b) definition of the macroscopic nodes in the reservoir network.*

transfer in the opposite way, another macroscopic node must be defined. Note that several macroscopic nodes of any type can be defined in a reservoir, providing that their definition has a topological justification. Figure 1.3(b) shows the possible connections between macroscopic nodes, which is thus a representation of the aggregated network used in SymuRes.

Each macroscopic node has a *capacity* attribute, which is the maximum admissible flow that can transfer through the macroscopic node (in vehicles per unit of time, [veh/s]). It corresponds to the sum of the capacity of all the real nodes that define the macroscopic node, i.e. the capacity of the links connected to these nodes, possibly reduced by near traffic signals. Usually, the capacity of internal origins or destinations is huge and not limiting (due to the high number of nodes defining these macroscopic nodes), but the one of external entries, exits or border nodes may be a significant constraint for flow transfers in the simulation (because represented by much less connections in the real network).

1.2.3 Macroscopic routes

The trips of vehicles in the whole urban area are described by *macroscopic routes*. A *macroscopic route* (or *route*, sometimes *path* when there is no ambiguity, referred to with subscript p) is a sequence of macroscopic nodes. It goes from an internal origin or external entry to an internal destination or external exit. It corresponds to the aggregation of individual vehicle trips that are representative of a travelers' mobility pattern.

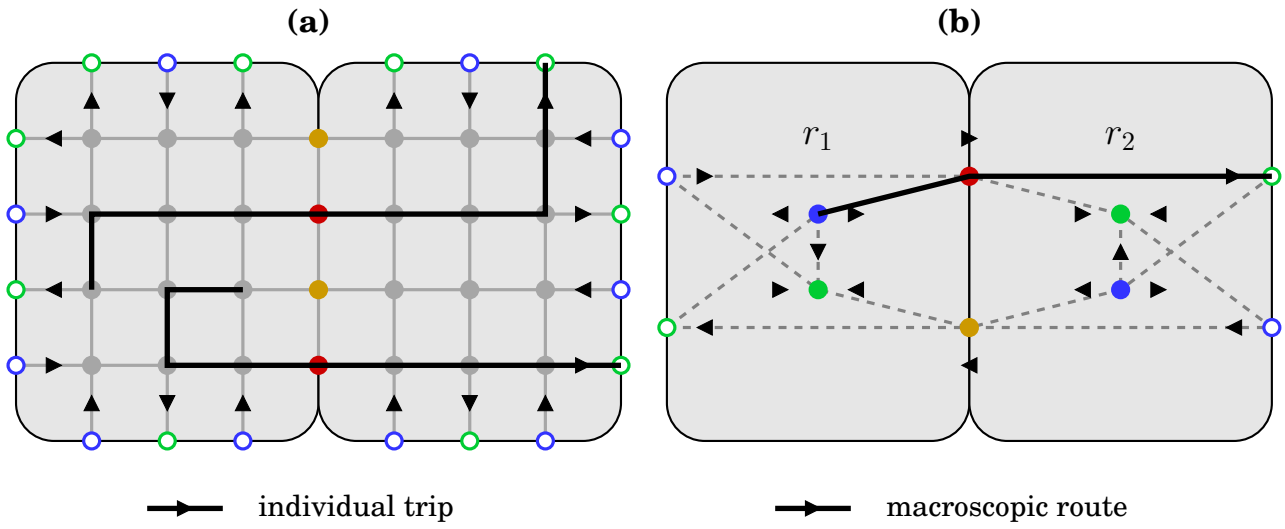


Figure 1.4. *(a) Two individual trips in the real network, and (b) their corresponding macroscopic route in the reservoir network (the legend for the nodes is the same as in Figure 1.3).*

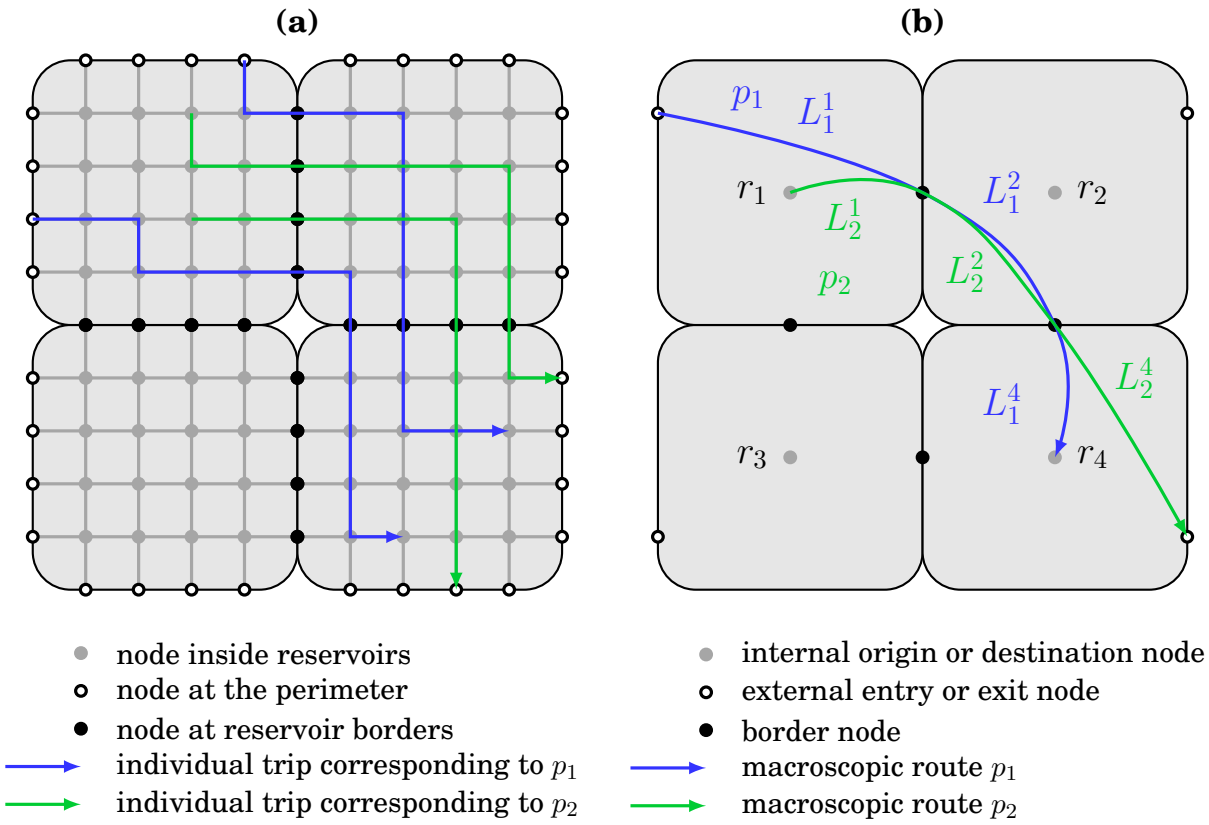


Figure 1.5. *(a) Individual trips in the real network split in 4 reservoirs, and (b) their aggregation in two macroscopic routes in the reservoir network.*

Figure 1.4 provides an example of a macroscopic route from an internal origin in r_1 to an external exit in r_2. This route is constituted by the sequence of 3 macroscopic nodes and 2 reservoirs. The node sequence of route p is written M_p while its reservoir sequence is denoted by R_p. Figure 1.5 provides another example of two macroscopic

routes p_1 and p_2, where two examples of individual trips in the real network are given for each of them. Note that both p_1 and p_2 correspond to the same sequence of reservoirs $[r_1, r_2, r_4]$, while they are still considered as two distinct macroscopic routes. As for macroscopic nodes, any macroscopic route can be defined providing that it represents a clear mobility pattern (i.e. a significant amount of individual trips). The only restriction is that repetitions in the resulting reservoir sequence are forbidden (e.g. macroscopic routes with a reservoir sequence like $[r_1, r_2, r_4, r_2]$ are not allowed).

Each macroscopic route also carries the information of the distance that has to be traveled in each reservoir of its sequence. These distances are called *trip lengths*. The trip length of route p while crossing reservoir r is denoted L_p^r [m]. The trip lengths of routes p_1 and p_2 are mentioned in Figure 1.5. The total length of route p is thus the sum of trip lengths L_p^r over the reservoirs r of the route sequence. Each reservoir dynamics is defined at the level of the macroscopic routes that cross it. The evolution of the total accumulation $n^r(t)$ is the sum of the evolution of all partial accumulations $n_p^r(t)$ [veh], where $n_p^r(t)$ is the number of vehicles in reservoir r traveling in route p.

1.3 Main features

1.3.1 Demand definition

Once an aggregated network (with reservoirs and macroscopic nodes) and a set of macroscopic routes are designed, the last step before running a simulation is the definition of demand (i.e. how users want to travel on this network). This can be done in two ways:

- The first one is the definition of a macroscopic OD (Origin-Destination) matrix: a demand profile $\lambda^{OD}(t)$ (vehicle flow, [veh/s]) is set for each macroscopic origin to each macroscopic destination. A macroscopic origin is either an internal origin or external entry macroscopic node, and a macroscopic destination is either an internal destination or external exit macroscopic node. Then a path flow distribution module is in charge of distributing these demand profiles among the possible routes. The demand profile of route p is then $\lambda_p(t) = a_p.\lambda^{OD}(t)$, where $a_p \in [0, 1]$ is the path flow or assignment coefficient of route p. The initial set of routes can be determined in different ways, depending on the initial knowledge of the network. More details are given in sections 3 and 4.

- The second way is the direct definition of demand profiles on a selection of routes. In this case, the path flow distribution module is not used.

1.3.2 Path flow distribution and network equilibrium

The path flow distribution module consists of a succession of Dynamic Traffic Assignment (DTA) loops. This module aims at reproducing travelers' route choices with different network equilibrium and behavior models, see Batista (2018), Batista & Leclercq (2019) for more details. One can choose between the following network equilibrium models:

- Deterministic User Equilibrium (DUE)

- Stochastic User Equilibrium (SUE)

And the following behavior models:

- Utility maximizer (rational user)
- Satisficing behavior (bounded rational user)
- Regret minimizer (in the sense defined by the Regret Theory)

1.3.3 Traffic flow solvers

Once all demand inputs are set, the simulation outputs provide the evolution of aggregated traffic states: accumulation, mean speed, inflow, outflow, cumulative count curves and travel time in each reservoir per route. The simulator can capture undersaturated conditions as well as propagation of congestion on specific routes. To this aim, one can choose between two different traffic flow solvers: (i) the accumulation-based model or (ii) the trip-based model. Below is the summary of their differences:

- About the representation of traffic dynamics, the accumulation-based solver deals with continuous vehicle flow and may eventually handle fractions of vehicles, whereas the trip-based solver keeps track of individual vehicle trajectories and cannot split a single vehicle into several fractions.

- About the general solver algorithm, the accumulation-based model includes a finite difference method with a fixed time step, i.e. the accumulation at the next time step is calculated with the inflow and outflow difference of the previous time step. On the other hand, the trip-based solver is an event-based scheme.

- Concerning the computational complexity, the accumulation-based solver has a complexity proportional to the number of time steps. Its computational time can be improved by increasing the time step. The number of calculations at each time step are always the same regardless of each reservoir traffic state. However, the trip-based solver complexity is proportional to the number of traveling vehicles and thus increases during congestion periods where more events are computed during a given time period. In case of very low demand flows, the trip-based solver may eventually have a faster computational time than the accumulation-based one. But very low demand flows are unlikely in large-scale simulation studies, and the trip-based complexity is generally much higher than the accumulation-based one.

- Regarding the algorithm stability, the accumulation-based solver is deterministic and stable as long as the time step does not exceed the minimum free-flow travel time in any reservoir (general stability condition for a finite difference method). It provides a smooth evolution of flows and accumulations (which may be not very consistent with usual traffic dynamics, see the discussion in Mariotte *et al.*, 2017). Because a vehicle unit cannot be split in the trip-based solver, random draws are required to arbitrate between different vehicles that want to enter or exit a reservoir at the same time. This stochastic component has a few impacts on the system stability in undersaturated conditions, but may be problematic when some reservoirs are oversaturated. In this case, instabilities come from two phenomena: first,

there are much more event conflicts than in undersaturation, because all the waiting vehicles in different reservoirs want to enter or exit as soon as possible, and second, the modeling of congestion propagation in the trip-based model may be very sensitive to numerical differences of the order of one vehicle. Actually, the modeling of spillbacks in the trip-based model is an event-based transcription of the flow merging model implemented in the accumulation-based model, where the possibility of splitting fractions of vehicles may be required to ensure the consistency of the merging scheme. When splitting vehicles is not possible as in the trip-based model, numerical errors are thus introduced.

By default, the initial simulation state is always an empty network. Although any initial state (initial accumulations) can be set for the accumulation-based model due to its memory-less property, it is difficult to begin a trip-based simulation with traveling vehicles in the network. The trip-based model does not accommodate an initial condition which is not at equilibrium (steady traffic states complying with Little's queuing formula), and even with an initial steady state condition, it is tricky to assign a remaining travel distance to each vehicle complying with this state. That is why an empty initial state is set by default. Running a short warm-up simulation period then allows to generate any network loading that one might require.

2.

Platform architecture

2.1 Modules and structures

The simulation platform consists of a standalone source code written in MATLAB™. Simulations can be run with an old version of MATLAB™ (2007b and later), but a more recent version is required for using the post-processing functions (2015b and later). All required functions and scripts are included in the platform and no additional packages nor toolboxes are needed.

The platform conception is based on an object-oriented design. A simulation uses the following objects (Matlab structures):

- **Simulation** (defined by user) is a structure that contains general information about the simulation: network name, the solver to use with its options, the simulation duration and time step.
- **Assignment** (defined by user) is a structure that contains information about the path flow distribution and the DTA convergence loop: assignment mode, assignment periods, convergence criteria, choice model type and options.
- **Reservoir** (defined by user and updated during the simulation) is an array structure that describes each reservoir: MFD parameters, connection to other reservoirs, set of routes crossing the reservoir, mean speed evolution, accumulation evolution per route crossed, inflow and outflow evolution per route crossed.
- **MacroNode** (defined by user) is an array structure that describes each macroscopic node: type, reservoir connections and capacity (maximum flow that can transfer through the node).
- **ODmacro** (defined by user) is an array structure that describes each macroscopic OD pair: origin and destination nodes, origin and destination reservoirs, corresponding set of possible routes with their trip lengths when crossing reservoirs. The original macroscopic route set is included in this structure.

- **Route** (automatically defined) is an array structure that describes each macroscopic route: reservoir and macroscopic node sequence, path flow coefficient, demand profile, evolution of travel time.

- **Vehicle** (automatically defined) is an array structure that describes vehicle trajectories for the trip-based model. It is created only when running this solver. It includes all details about each individual vehicle trajectory: macroscopic route followed, distance and time traveled, waiting time (delays) in reservoirs.

The platform is then organized in several modules that load or create these structures and interact with them:

- The **SIMULSETTINGS** module defines the simulation settings and options, via the creation of the **Simulation** and **Assignment** structures.

- The **RESDEF** module creates or loads the **Reservoir**, **MacroNode** and **ODmacro** structures.

- The **DEMDEF** module loads the aggregated demand data. Depending on the assignment mode, the temporal demand profile is either set in each origin reservoir per destination reservoir, or directly in each route.

- The **ROUTECALC** module loads the route choice set and selects some of them according to the simulation options and demand definition.

- The **ASSIGNCALC** module updates the path flow coefficients of the selected routes at the beginning of each assignment period.

- The **MFDSOLVER** module solves traffic dynamics for the current assignment period. It either corresponds to the accumulation-based model, or to the trip-based model where the additional **Vehicle** structure is created.

- The **CONVERGECALC** module tests the convergence criteria at the end of each assignment period, based on the previously simulated traffic states.

2.2 Folder organization

The folder and file organization is shown in Figure 2.1. In Matlab, the working directory must be the `SymuRes` folder, see Figure 2.1(a). The networks studied by the user are placed in the `UserNetworks` folder, where the required generic files are presented in Figure 2.1(b). These files correspond to the modules introduced earlier.

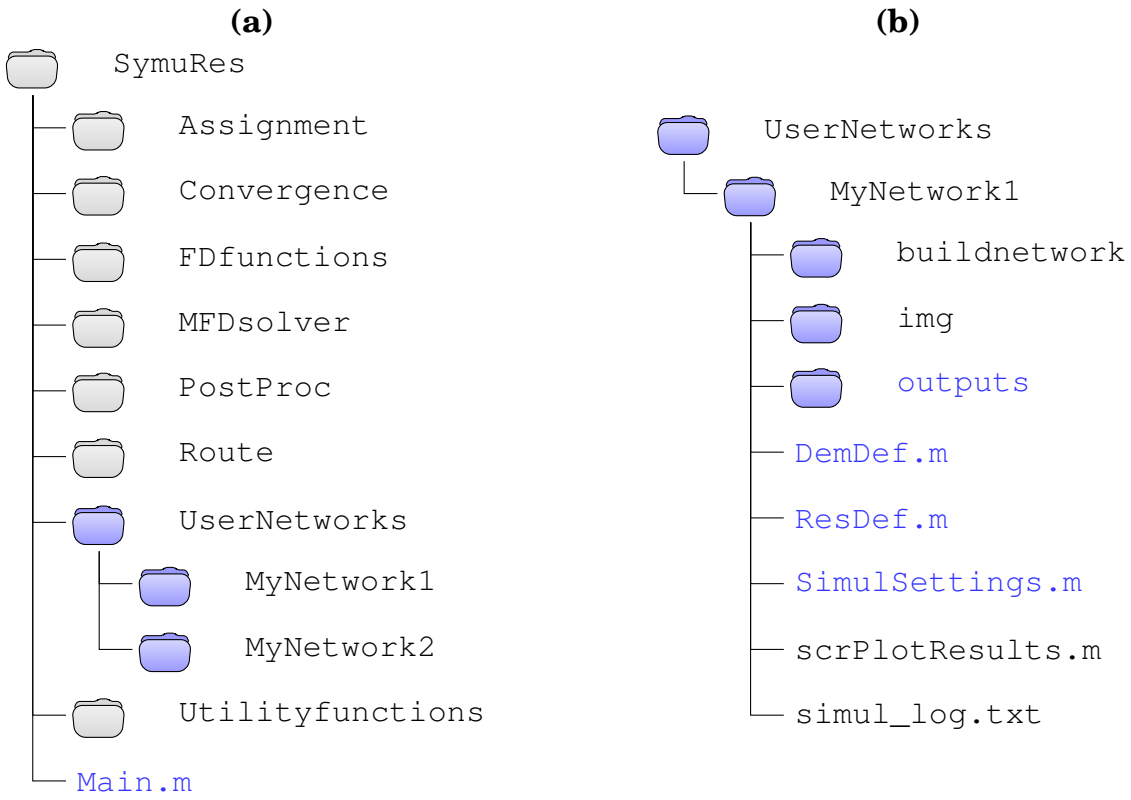


Figure 2.1. *(a)* SymuRes *main folder. The user-defined folder is highlighted in blue, where simulation input data and settings are stored. The main file to select the network and launch its simulation is also highlighted in blue. (b) User network folder, with details about a network simulation folders and files. Required elements have their names highlighted in blue, other elements are optional.*

2.3 Launching a simulation

A simulation is launched by running the **`Main.m`** file in Matlab (the working directory being the **`SymuRes`** folder). Within this file, the user must select the network, the solver, and a name for the simulation:

```
% Choice of a network defined by user
Simulation.Network = 'MyNetwork1';

% Choice of the solver
% 1: accbased / 2: tripbased
Simulation.Solver = 1;

% Simulation name
Simulation.Name = 'MyFirstSimulation';
```

Main.m

3.

Main file and DTA loop

The main file governs the simulation process by calling the different modules. Its architecture is described by the flow chart in Figure 3.1. After a network is chosen by the user, its main structures are created by the **SIMULSETTINGS**, **RESDEF** and **DEMDEF** modules. The original route set (defining all the possible routes for the simulation) is defined by the user in the **ODmacro** structure. The route set that will be actually used during the simulation is built by the **ROUTECALC** module. In **DEMDEF**, if the demand is defined in **ODmacro**, the routes are built as the k-shortest routes from the original set, based on the free-flow speeds of reservoirs to calculate travel times. Otherwise, when the demand is directly set in predefined routes in **DEMDEF**, the **Route** structure is simply built from these routes in **ROUTECALC**.

The DTA process consists of two loops to reach a given network equilibrium, as introduced earlier in section 1.3.2. The outer loop simply splits the simulation duration T_s in several periods $[0, T_1^a, T_2^a, \ldots, T_s]$, set in the **Assignment** structure. The inner loop aims at assigning flows (accumulation-based model) or vehicles (trip-based model) on the routes so that the resulting traffic states comply with a given equilibrium specified in **Assignment** for the current period. This optimization problem corresponds to the classical problem of traffic assignment, it is solved with a widely used fixed-point method, known as the Method of Successive Averages (MSA, Sheffi, 1985). The inner loop is thus also called the MSA convergence loop.

If no equilibrium is sought, the simulation can also be run without any MSA convergence loop (this is automatically the case when the routes are predefined by the user in the **DEMDEF** module). The assignment periods and the convergence option are set in the **`SimulSettings.m`** file of the network studied:

```
% Assignment by fixed periods
Assignment.Periods = [0 1800 Simulation.Duration]; % List of times
% Convergence
Assignment.Convergence = 1; % 1: Yes / 0: No
```

SimulSettings.m

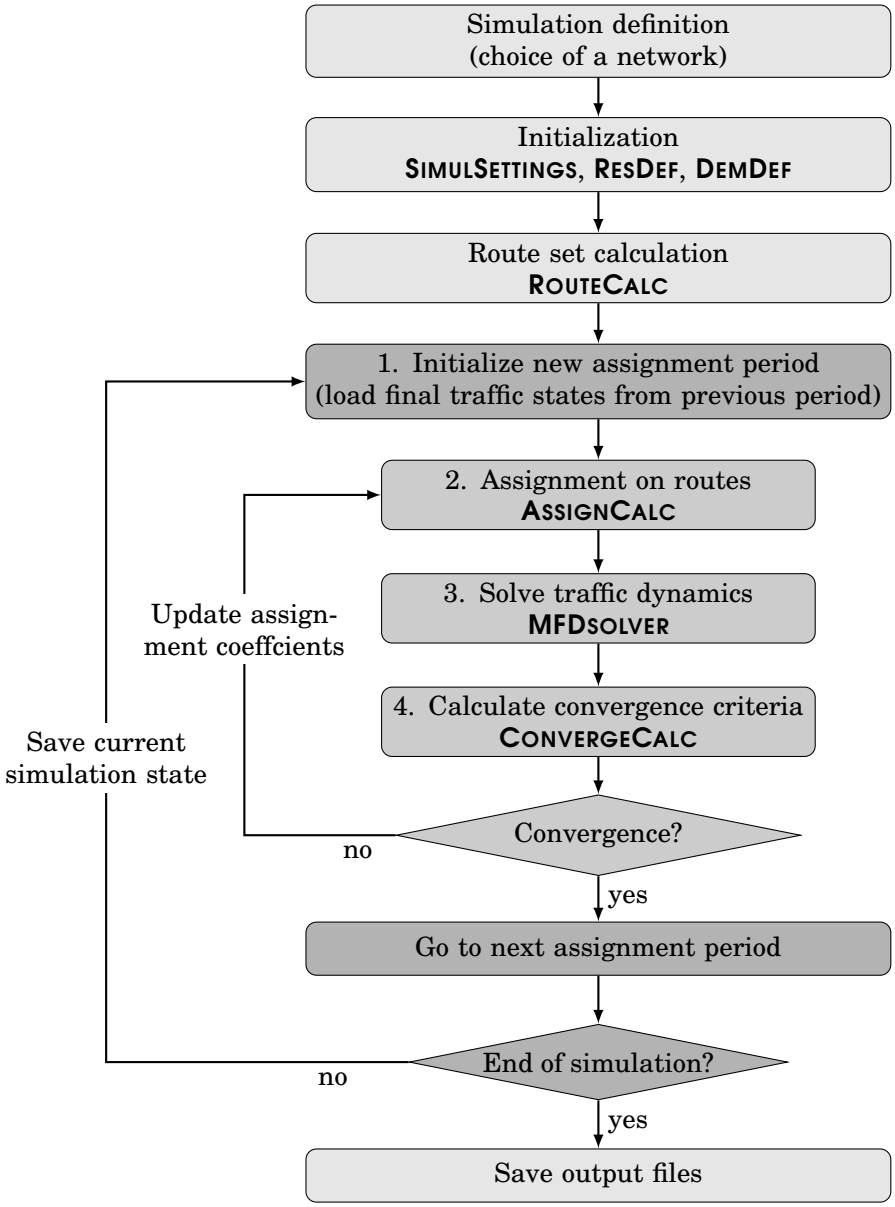


Figure 3.1. *Flow chart of the main file*

The inner MSA loop uses the **ASSIGNCALC** module to update path flow coefficients and the **CONVERGECALC** module to check convergence. To improve the convergence speed, the **ASSIGNCALC** module actually uses a modified version of MSA, the Method of Successive Weighted Averages (MSWA, Liu *et al.*, 2007). These modules were mainly developed by Sérgio F. A. Batista. The MSA loop consists of the following steps (see also Figure 3.1):

1. At the beginning of the current period, the final simulation state from the previous period is loaded (basically the **Reservoir**, **Route** and **Vehicle** structures), and it becomes the initial state of the current period. The first period is based on the simulation initial condition (empty network as explained earlier).

2. The assignment coefficients $a_{p,i}$ are calculated in the **ASSIGNCALC** module. At each

iteration i of the MSA loop, traffic outputs from the previous iteration are used to estimate the disutility of each route p, i.e. its travel time $\tilde{T}_p(t)$. However, to have a consistent definition of the source of uncertainty in the choice model, this travel time is not exactly the route travel time $T_p(t)$, derived from the route cumulative count curves. Unlike mircosimulation frameworks, in multi-reservoir systems the correlation between the routes is not only due to fluctuations in the reservoir mean speed but also to fluctuations between trip lengths inside each reservoir. Therefore, the travel time is calculated by using both mean speed and trip length data (for more details, see Batista & Leclercq, 2019). The new assignment coefficient $a^*_{p,i}$ proposed at iteration i corresponds to the probability of choosing route p for a given OD. The calculation of this probability depends on the equilibrium type. For DUE, the choice is given by the minimum of the mean travel time $\langle \tilde{T}_p \rangle_i = \sum_{r \in R_p} L^r_p / \langle v^r \rangle_i$ of all routes of a given OD, calculated for the current assignment period. For SUE where uncertainty is introduced, Monte-Carlo calculations of possible travel times are performed: the same formula is used, but with distributed values of length and speed. The distribution of trip lengths corresponds to the lengths from the micro trip set, while the distribution of speeds comes from the mean speed evolution during the last simulation iteration $i-1$ (for the first iteration, the initial mean speed values of the current period are provided). Then, the route choice for each Monte-Carlo draw is the minimum of the travel times among possible routes of each OD. The average choice among all draws gives each probability $a^*_{p,i}$ for the current iteration i of choosing route p for a given OD. Options are available in the module to select the sources of uncertainty (trip length and/or mean speed). The updated assignment coefficient $a_{p,i}$ is finally obtained with the MSWA formula (Liu *et al.*, 2007):

$$a_{p,i} = \alpha_w a^*_{p,i} + (1 - \alpha_w)\, a_{p,i-1} \tag{3.1}$$

where $\alpha_w = \frac{i^w}{\gamma_w + i^w + (i-1)^w + \dots}$. The parameters $w \geq 1$ and γ_w are set in **Assignment**.

3. The solver is run for the current period with the updated assignment coefficients that indicate the demand per route. At the end of the period, the mean speed evolution $v^r(t)$ in the reservoirs is saved for the next assignment update.

4. The convergence of the MSA is checked in the **ConvergeCalc** module, where we focus on two criteria. The first one is called the Gap criterion (Sbayti *et al.*, 2007). The Gap at iteration i corresponds to the total relative difference between the route mean travel time and the minimum travel time among all OD pairs:

$$\text{Gap}_i = \sum_{OD} \frac{1}{\langle \tilde{T}^{OD}_{\min} \rangle_i} \sum_{p \in OD} a_{p,i} \left(\langle \tilde{T}_p \rangle_i - \langle \tilde{T}^{OD}_{\min} \rangle_i \right) \tag{3.2}$$

where $\langle \tilde{T}^{OD}_{\min} \rangle_i = \min_{p \in OD} \langle \tilde{T}_p \rangle_i$. This criterion is satisfied once the Gap is lower than a given value set in **Assignment**. The second criterion is called the number of violations. It corresponds to the number of routes with the difference between old and updated coefficients $a_{p,i} - a_{p,i-1}$ greater than a given threshold set in **Assignment**. This criterion is satisfied if the proportion of routes in violation is below a given threshold. Then the convergence of the current period is achieved once one or both criteria are satisfied, depending on the options. The procedure goes back to step 2 while the convergence is not reached.

4.

Simulation pre-processing

Before running a simulation, several information and data must be provided to the platform. These data either come from preliminary studies on the real network at link-scale, or are given as direct inputs in case of an artificial reservoir network. Figure 4.1 presents the general picture of a simulation study with pre-processing. It is divided into four phases: (I) input data collection, (II) pre-processing, (III) simulation loop and (IV) post-processing.

The input data collection (I) involves several data sources, as presented below (see also Figure 4.1):

1. The real network topology data (links, nodes, signal settings, number of lanes, etc). This allows to define upper bounds for each reservoir free-flow speed, jam accumulation, border capacities, and possibly MFD capacity according to the method of Laval & Castrillon (2015). This also allows the estimation of trip lengths with the method of Batista *et al.* (2019).

2. The demand profile data (OD matrix data from preliminary studies, household surveys, local authorities, etc). It must be defined at the scale of the reservoirs or a lower scale.

3. The traffic data (loop detector data, probe vehicle data, historical mean speed data, simulated data from microsimulation, etc). This allows to define the reservoir MFD and can be use in a clustering algorithm to define the reservoirs.

4. The trip data on the real network (vehicle trajectories). It may be a part of the traffic data, if probe data is accessible. This allows to estimate trip lengths, and possibly path flow distribution.

The pre-processing phase (II) includes the following steps (see also Figure 4.1):

5. The clustering phase, which consists in partitioning the network into reservoirs. When being “traffic-oriented”, the clustering can be based on algorithms developed in Ji & Geroliminis (2012), Zhou *et al.* (2012), Saeedmanesh & Geroliminis

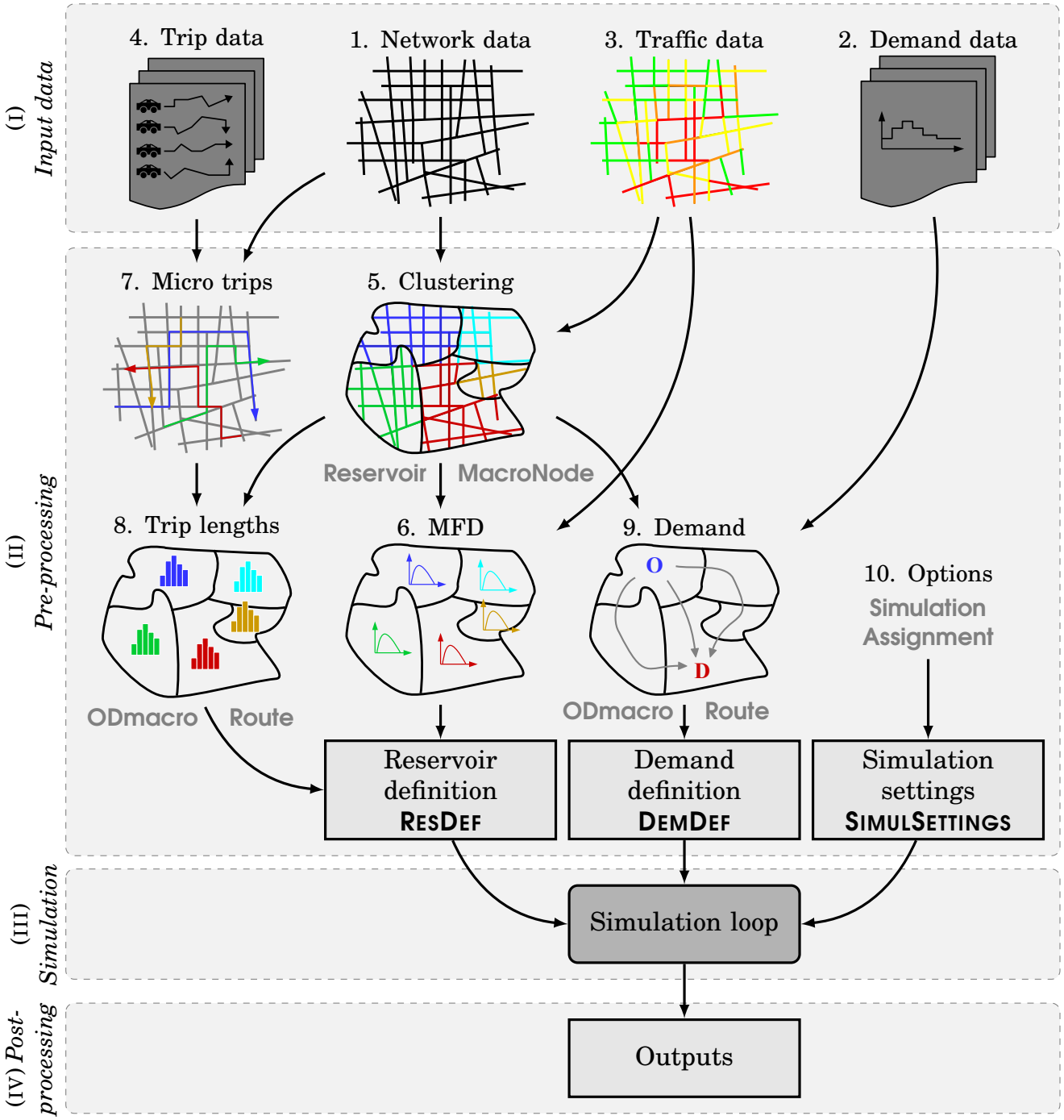


Figure 4.1. *General picture of a simulation study with pre-processing phase*

(2016), Lopez *et al.* (2017), Saeedmanesh & Geroliminis (2018). These methods are designed to ensure homogeneous traffic states in reservoirs. However, they may result in strange area shapes with a lack of connectivity between the links of each reservoir. When being "demand-oriented", the clustering uses predefined urban areas where the demand data is provided. The reservoirs are then built by aggregating these areas. In any case, the modeler's experience about the studied

network is always critical for a good clustering definition.

6. The MFD estimation phase, which consists in fitting a mathematical function to the traffic data. In practice, using a piecewise linear function is convenient, as it can adapt to many cases. Easy to fit, the parabolic model is also widely used.

7. The individual (microscopic) trip set calculation phase. It consists of assigning the trip data (if available) to the real network, or of simulating vehicle trajectories with shortest path calculations as developed in Batista *et al.* (2019).

8. The trip length estimation phase. Different aggregation methods are proposed and discussed in Batista *et al.* (2019).

9. The demand aggregation step is aimed at setting the regional OD matrix, or directly the demand per route when the routes are predefined in **DEMDEF**.

10. The simulation settings and options are specific to each study and depend on its purposes. This is detailed in the **Simulation** and **Assignment** structures.

In case of an artificial network (not based on a real city), the reservoir network is designed independently. In Figure 4.1, the input data collection phase is thus skipped, as well as the network clustering and microscopic trip set generation. Then, the MFD, trip length and demand data are set manually, or by using automated scripts to define regular network configurations. A benchmark example may be the honeycomb city model, as presented in Yildirimoglu *et al.* (2015), Ramezani *et al.* (2015).

The simulation phase (III) will be detailed further in section 7, and the post-processing phase (IV) will be illustrated with examples in section 5.

5.

Examples

5.1 Single reservoir

Available soon...

5.2 Braess network

Available soon...

5.3 Lyon 6 network

Available soon...

6.

Simulation variables

6.1 Input user-defined variables

6.1.1 Simulation structure

Simulation is a structure that contains general information about the simulation. It must have the following fields:

`Network` : ⟨*string*⟩
name of the network folder to simulate.

`Solver` : ⟨*integer:* $1 \mid 2$⟩
1 for the accumulation-based model, and 2 for the trip-based model.

`Name` : ⟨*string*⟩
name for the current simulation.

`Duration` : ⟨*double:* $T_s > 0$⟩
simulation duration [s].

`TimeStep` : ⟨*double:* $\delta t_s > 0$⟩
simulation time step [s] for the accumulation-based model (for the trip-based model, this is only used to sample the outputs when plotting them).

`MFDfct` : ⟨*function: P = func(n, param)*⟩
production-MFD function shape [veh.m/s]: $P(n^r, \mathcal{H}^r) = P^r(n^r)$ that will be used for all reservoirs, $\mathcal{H}^r$ being the list of parameters specific to reservoir r. The user may choose a built-in fundamental diagram function in the **`FDfunctions`** folder, or build his/her own.

`Entryfct` : ⟨*function: Ps = func(n, param)*⟩
entry supply function shape [veh.m/s]: $P_s(n^r, \mathcal{H}_s^r) = P_s^r(n^r)$, $\mathcal{H}_s^r$ being the list of parameters specific to reservoir r. This function is generally built with the MFD function, see the example in section 5.2 for more details.

`MergeModel` : ⟨*string: demprorata* | *endogenous* | *equiproba* | *demfifo*⟩
inflow merging model. The demand pro-rata merging (*demprorata*) is the stan-

dard model used in the literature (Yildirimoglu & Geroliminis, 2014, Ramezani *et al.*, 2015, Yildirimoglu *et al.*, 2018). The endogenous merging (*endogenous*) is the model developed in Mariotte & Leclercq (2019). The equi-probabilistic merging (*equiproba*) is a naive model mainly used for debug, it is not used in the literature. The demand FIFO merging (*demfifo*) is a demand pro-rata model complying with the FIFO rule for vehicles queuing at the reservoir entry. This is not developed in the literature yet, and it is still under debug.

`DivergeModel` : ⟨*string: maxdem* | *decrdem* | *queuedyn*⟩
outflow diverging model. The maximum outflow demand (*maxdem*) is the model adopted by Mariotte & Leclercq (2019). The decreasing outflow demand (*decrdem*) is the standard model widely used in the literature (Yildirimoglu & Geroliminis, 2014, Ramezani *et al.*, 2015, Yildirimoglu *et al.*, 2018). The reservoir queue dynamics (*queuedyn*) is the model developed in Haddad (2017), Ni & Cassidy (2019). This model is still under debug.

`TripbasedSimuFactor` : ⟨*double in* $[0, 1]$⟩
used in the trip-based model only, reduction coefficient applied to the flow of vehicles to reduce their number to improve computational time (does not change the result consistency with the original flow demand). See section 7 and chapter 6 of Mariotte (2018) for more details.

`OutfileDebug` : ⟨*boolean*⟩
if true, save all the extra structure fields created during the simulation. This is useful for debug phases only, as it leads to heavy output files.

6.1.2 Assignment structure

Assignment is a structure that contains general information about the path flow distribution and the DTA convergence loop. It must have the following fields:

`Periods` : ⟨*row vector:* $[0, T_1^a, T_2^a, \ldots, T_s]$⟩
list of times [s] that define the successive assignment periods (when a MSA loop starts), from 0 to the simulation duration T_s. Put $[0, T_s]$ for only one assignment period. See the main simulation loop described in section 3 for more details.

`PredefRoute` : ⟨*boolean*⟩
if true, this means that the routes are defined in the **DEMDEF** file with their demand profile. This corresponds to a one-step simulation without any traffic assignment and convergence loop. If false, this means that the demand is defined in the **ODmacro** structure for each macroscopic OD, and that the **Route** structure is built by the **ROUTECALC** module.

`Convergence` : ⟨*boolean*⟩
if true, perform a MSA convergence loop at each assignment period. See the main simulation loop described in section 3 for more details.

`MinimumGap` : ⟨*double* > 0⟩
default 0.01, convergence criterion for the MSA loop, minimum Gap admissible. See section 3 for the Gap definition.

`NumViolationsThreshold` : ⟨*double in* $[0, 1]$⟩
default 0.05, threshold of the relative difference between the assignment coeffi-

cients of two consecutive MSA iterations to consider that a route is in violation.

`NumViolationsTolerance` : ⟨*double in* $[0, 1]$⟩
default 0.05, convergence criterion for the MSA loop, minimum proportion admissible of routes that are considered in violation.

`MaxIteration` : ⟨*integer*⟩
default 10, maximum number of iterations for the MSA convergence loop.

`WeightMSWA` : ⟨*positive double:* $w > 0$⟩
default 2, weight parameter for the MSA scheme, based on the MSWA (Liu *et al.*, 2007).

`GammaMSWA` : ⟨*double:* $\gamma_w \geq 0$⟩
default 0, initialization parameter for the MSA scheme, based on the MSWA (Liu *et al.*, 2007).

`ConvergenceIndicatorID` : ⟨*integer:* 1 | 2 | 3⟩
global convergence criterion. Use 1 for the Gap criterion only, 2 for the number of violations only, and 3 for these two criteria together.

`NumShortestPath` : ⟨*integer*⟩
default 3, maximum number of shortest paths per macroscopic OD.

`TripLengthMethod` : ⟨*integer:* 1 | 4⟩
default 4, method ID to calculate the trip lengths in reservoirs as defined in Batista *et al.* (2019). Method 1 corresponds to the aggregation of all trip lengths at the reservoir level (one trip length per reservoir assigned to all the routes that cross this reservoir). Method 4 corresponds to the definition of trip length at the route level (multiple trip lengths per reservoir, i.e. one trip length per route portion that crosses this reservoir).

`model` : ⟨*integer:* 1 | 2 | 100 | 101 | 102 | 103 | 104⟩
traffic assignment model ID. ID from 1 to 99 are reserved for network equilibrium models. Use:

- 1 for Deterministic User Equilibrium (DUE),
- 2 for Stochastic User Equilibrium (SUE).

ID starting from 100 are reserved for user-specific models. Use:

- 100 for manual assignment (user-defined path flow coefficients),
- 101 for equi-probabilistic assignment (same flow proportion on all the routes of a given OD),
- 102 for a flow distribution at the pro-rata of the number of individual trips (micro trips) that define a macroscopic route,
- 103 for a DUE with weights on reservoirs (the higher the weight on a reservoir, the higher the path flow coefficients of the routes that cross this reservoir),
- 104 for a DUE with reservoir path penalization (apply a minimum path flow coefficient to routes that correspond to some reservoir sequences).

`Behavior` : ⟨*integer:* 1 | 2 | 3⟩
behavior model ID. Use 1 for a rational behavior (utility maximization), 2 for a bounded rational behavior (satisficing behavior), 3 for a behavior based on regret (regret minimization).

`UtilityID` : ⟨*integer:* 1 | 2 | 3 | 4⟩

utility model ID when using the assignment model SUE. Use 1 for considering uncertainties on trip lengths only, 2 for uncertainties on mean speeds only, 3 for uncertainties on both trip lengths and mean speeds, 4 for these two element plus inhomogeneities in the MFD

`ALparam` : $\langle$*double* $> 0\rangle$
default 0.5, parameter for the bounded rational behavior, relative value for the indifference band (aspiration level).

`parameters` : $\langle$*row vector*$\rangle$
list of parameters for SUE (number of draws for Monte-Carlo simulations). See the example files.

`ManualCoefficients` : $\langle N_P$*-size row vector*$\rangle$
list of the path flow coefficients per route, N_P being the number of routes, for model 100. A first one-loop simulation must be run to get the list of routes, as in the assignment mode the **Route** structure is not known before the simulation.

`ReservoirWeight` : $\langle N_R$*-size row vector*$\rangle$
list of weights (> 1) per reservoir, N_R being the number of reservoirs, for model 103.

`PenalizedResPath` : $\langle$*cell list of row vectors:* $\{[r_1, r_2], [r_1, r_3, r_2], \dots\}\rangle$
list of reservoir sequences that are penalized, for model 104.

`PenalizedAssignCoeff` : $\langle$*double in* $[0, 1]\rangle$
maximum path flow coefficient that is assigned to the routes that correspond to the reservoir sequences given in the above-mentioned **`PenalizedResPath`**.

6.1.3 Reservoir structure

Reservoir is an array structure (ID from 1 to **`NumRes`**: $r \in \{1, \dots, N_R\}$) that describes each reservoir. It must have the following fields:

`AdjacentRes` : $\langle$*row vector:* $[r_1, r_2, r_3, \dots]\rangle$
list of ID of adjacent reservoirs.

`FreeflowSpeed` : $\langle$*double:* $u^r > 0\rangle$
MFD free-flow speed [m/s].

`MaxProd` : $\langle$*double:* $P_c^r > 0\rangle$
MFD maximum production [veh.m/s].

`MaxAcc` : $\langle$*double:* $n_j^r > 0\rangle$
MFD jam accumulation [veh].

`CritAcc` : $\langle$*double:* $n_c^r > 0\rangle$
MFD critical accumulation [veh].

`MFDfctparam` : $\langle$*vector or matrix:* $\mathcal{H}^r\rangle$
parameters of the MFD function (defined in the **Simulation** structure).

`Entryfctparam` : $\langle$*vector or matrix:* $\mathcal{H}_s^r\rangle$
parameters of the entry supply function (defined in the **Simulation** structure).

`Centroid` (optional): $\langle$*2-size row vector:* $[x, y]\rangle$
abscissa and ordinate of the reservoir center (for plotting purpose).

`BorderPoints` (optional): $\langle$*2-by-N matrix:* $\begin{bmatrix} x_1 & x_2 & \cdots \\ y_1 & y_2 & \cdots \end{bmatrix}\rangle$

list of abscissa (row 1) and ordinates (row 2) of the N points delimiting the borders of the reservoir (for plotting purpose).

6.1.4 MacroNode structure

MacroNode is an array structure (ID from 1 to **`NumMacroNodes`**: $m \in \{1, \ldots, N_M\}$) that describes each macroscopic node. It must have the following fields:

`Type` : ⟨*string: origin* | *destination* | *externalentry* | *externalexit* | *border*⟩
type of the macroscopic node, see section 1.

`ResID` : ⟨*integer:* r_1⟩ or ⟨*2-size row vector:* $[r_1, r_2]$⟩
ID of the reservoir where the node belongs; or transfer from reservoir r_1 to reservoir r_2 in case of a border node.

`Capacity` : ⟨*structure*⟩
evolution of the node capacity (maximum flow that can transfer through the node) with the fields:

- **`Time`** : ⟨*row vector:* $[0, t_1, t_2, \ldots]$⟩
  times at which the capacity changes [s]. The last time must be lesser or equal to the simulation time T_s.
- **`Data`** : ⟨*row vector:* $[C_m(0), C_m(t_1), C_m(t_2), \ldots]$⟩
  same size as **`Time`**, values of the capacity [veh/s].

`Coord` (optional): ⟨*2-size row vector:* $[x, y]$⟩
abscissa and ordinate of the node.

6.1.5 ODmacro structure

ODmacro is an array structure (ID from 1 to **`NumODmacro`**: $OD \in \{1, \ldots, N_{OD}\}$) that describes each macroscopic OD pair. It must have the following fields:

`OriginID` : ⟨*integer*⟩
origin ID for the **ODmacro** structure.

`DestinationID` : ⟨*integer*⟩
destination ID for the **ODmacro** structure.

`ResOriginID` : ⟨*integer*⟩
origin reservoir ID.

`ResDestinationID` : ⟨*integer*⟩
destination reservoir ID.

`NodeOriginID` : ⟨*integer*⟩
origin macroscopic node ID.

`NodeDestinationID` : ⟨*integer*⟩
destination macroscopic node ID.

`NumPossibleRoutes` : ⟨*integer*⟩
total number of routes associated to the OD.

`PossibleRoute` : ⟨*array structure*⟩
set of routes associated to the OD, of size **`NumPossibleRoutes`**. The routes actually used during the simulation are only a subset of these routes, either automati-

cally selected in the **RouteCalc** module (the k-shortest ones), or manually selected by the user in the **DemDef** module. It must have the fields:

ResPath : ⟨*row vector:* $[r_1, r_2, r_3, \dots]$⟩
list of ID of the successive reservoirs of the route.

NodePath : ⟨*row vector:* $[m_1, m_2, m_3, \dots]$⟩
list of ID of the successive macroscopic nodes of the route, its size is the size of **ResPath** plus 1.

TripLengths : ⟨*row vector:* $[L_p^{r_1}, L_p^{r_2}, L_p^{r_3}, \dots]$⟩
same size as **ResPath**, list of trip lengths [m] in the reservoirs crossed by the current route p.

NumMicroTrips : ⟨*integer*⟩
number of real or simulated individual vehicle trajectories represented by the macroscopic route. This value is useful for the assignment model 102 only.

Demand : ⟨*array structure*⟩
set of demand profiles for different trip categories or modes. This is to enable future developments, but for now, only a single category of "individual car trip" is used. It must have the fields:

Purpose : ⟨*string: cartrip*⟩
for now, default car trip for all demand profiles.

Time : ⟨*row vector:* $[0, t_1, t_2, \dots]$⟩
times at which the demand changes [s]. The last time must be lesser or equal to the simulation time T_s.

Data : ⟨*row vector:* $[\lambda^{OD}(0), \lambda^{OD}(t_1), \lambda^{OD}(t_2), \dots]$⟩
same size as **Time**, values of the demand [veh/s].

6.1.6 Route structure

Route is an array structure (ID from 1 to **NumRoutes**: $p \in \{1, \dots, N_P\}$) that describes each macroscopic route. In the **DemDef** module, if the routes are not predefined this structure is created in the **RouteCalc** module. Otherwise, if the routes are predefined, it must have the following fields:

ODmacroID : ⟨*integer*⟩
ID (in **ODmacro**) of the OD associated to the route.

ResPath : ⟨*row vector:* $R_p = [r_1, r_2, r_3, \dots]$⟩
list of ID of the successive reservoirs of the route p.

NodePath : ⟨*row vector:* $M_p = [m_1, m_2, m_3, \dots]$⟩
list of ID of the successive macroscopic nodes of the route p, its size is the size of **ResPath** plus 1.

TripLengths : ⟨*row vector:* $[L_p^{r_1}, L_p^{r_2}, L_p^{r_3}, \dots]$⟩
same size as **ResPath**, list of trip lengths [m] in the reservoirs crossed by the route p.

Demand0 : ⟨*array structure*⟩
set of demand profiles for different trip categories or modes. This is to enable future

developments, but for now, only a single category of "individual car trip" is used. It must have the fields:

`Purpose` : ⟨*string: cartrip*⟩
for now, default car trip for all demand profiles.

`Time` : ⟨*row vector:* $[0, t_1, t_2, \dots]$⟩
times at which the demand changes [s]. The last time must be lesser or equal to the simulation time T_s.

`Data` : ⟨*row vector:* $[\lambda_p(0), \lambda_p(t_1), \lambda_p(t_2), \dots]$⟩
same size as **`Time`**, values of the demand [veh/s].

Note that this field is written **`Demand0`** as the field **`Demand`** is reserved for the simulation solvers.

6.2 Output variables

After a simulation, some structures are updated or created with the simulation results. Their new fields are introduced below. These ones are added in the **MFDSOLVER** module or in post-processing. We only present here the most essential ones, as some others (sometimes obsolete) can be also added in post-processing. The fields useful in the solver algorithms only are introduced later in section 7. These ones are only saved in the different structures when the **`OutfileDebug`** is set to 1 in **Simulation**.

The notation convention uses the term "ID" whenever an integer or list of integers refers to the absolute indexing of an element (reservoir ID $r \in \{1, \dots, N_R\}$, route ID $p \in \{1, \dots, N_P\}$, etc). It uses the term "Index" whenever an integer or a list of integers refers to a relative indexing depending on the current element (e.g. index i_p of a route p in the list of routes crossing a given reservoir r). The evolution of traffic states is given for each time step of the simulation (index from 1 to **`NumTimes`**: $i_t \in \{1, \dots, N_T\}$).

6.2.1 Simulation structure

`Time` : ⟨N_T*-size row vector:* t⟩
simulation time [s], sampled with the simulation time step δt_s.

6.2.2 Reservoir structure

`RoutesID` : ⟨N*-size row vector:* $\mathcal{P}^r = [p_1, p_2, p_3, \dots]$⟩
list of ID of the routes crossing or originating in the reservoir. The order of routes is arbitrary but defines other fields below.

`TripLengthPerRoute` : ⟨N*-size row vector:* $[L^r_{p_1}, L^r_{p_2}, L^r_{p_3}, \dots]$⟩
same size as **`RoutesID`**, trip length portion [m] in the reservoir r of the routes in **`RoutesID`**. The order follows the order of the routes in **`RoutesID`**.

`RoutesPathIndex` : ⟨N*-size row vector:* $[i^r_1, i^r_2, i^r_3, \dots]$⟩
same size as **`RoutesID`**, index of the reservoir r in the reservoir sequence of the routes in **`RoutesID`**. The order follows the order of the routes in **`RoutesID`**.

`OriRoutesIndex` : ⟨*row vector:* $[i_1, i_2, i_3, \dots]$⟩
indexes in **`RoutesID`** of the routes in $\mathcal{P}^r_{\text{int}}$, i.e. that originate in the reservoir.

`DestRoutesIndex` : ⟨*row vector:* $[i_1, i_2, i_3, \dots]$⟩
indexes in **`RoutesID`** of the routes that end in the reservoir.

`EntryRoutesIndex` : ⟨*row vector:* $[i_1, i_2, i_3, \dots]$⟩
indexes in **`RoutesID`** of the routes in $\mathcal{P}^r_{\text{ext}}$, i.e. that enter the reservoir from its borders.

`ExitRoutesIndex` : ⟨*row vector:* $[i_1, i_2, i_3, \dots]$⟩
indexes in **`RoutesID`** of the routes that exit the reservoir through its borders.

`MacroNodesID` : ⟨*row vector:* $\mathcal{M}^r = [m_1, m_2, m_3, \dots]$⟩
list of ID of the macroscopic nodes within or at the borders of the reservoir.

`NodeRoutesIndex` : ⟨*cell list of row vectors:* $\{[i_1, i_2], [i_1, i_2, i_3], \dots\}$⟩
same size as **`MacroNodesID`**, indexes in **`RoutesID`** of the routes that goes through the macroscopic nodes of the reservoir. The order follows the order of the nodes in **`MacroNodesID`**.

`RoutesNodeID` : ⟨*2-by-*N *matrix:* $\begin{bmatrix} m_1^{\text{in}} & m_2^{\text{in}} & \dots \\ m_1^{\text{out}} & m_2^{\text{out}} & \dots \end{bmatrix}$⟩
entry node ID when originating or entering the reservoir (row 1), and exit node ID when ending or exiting the reservoir (row 2) for the N routes in **`RoutesID`**.

`OriNodesIndex` : ⟨*row vector:* $[i_1, i_2, i_3, \dots]$⟩
indexes in **`MacroNodesID`** of the nodes in $\mathcal{M}^r_{\text{int}}$, i.e. that are origin nodes in the reservoir (at which the routes in $\mathcal{P}^r_{\text{int}}$ start).

`DestNodesIndex` : ⟨*row vector:* $[i_1, i_2, i_3, \dots]$⟩
indexes in **`MacroNodesID`** of the nodes that are destination nodes in the reservoir.

`EntryNodesIndex` : ⟨*row vector:* $[i_1, i_2, i_3, \dots]$⟩
indexes in **`MacroNodesID`** of the nodes in $\mathcal{M}^r_{\text{ext}}$, i.e. that are entry nodes in the reservoir (through which the routes in $\mathcal{P}^r_{\text{ext}}$ enter).

`ExitNodesIndex` : ⟨*row vector:* $[i_1, i_2, i_3, \dots]$⟩
indexes in **`MacroNodesID`** of the nodes that are exit nodes in the reservoir.

`MeanSpeed` : ⟨N_T*-size row vector:* $v^r(t)$⟩
reservoir mean speed at each time step [m/s].

`AvgTripLength` : ⟨N_T*-size row vector:* $L^r(t)$⟩
reservoir dynamic average trip length at each time step [m].

`Acc` : ⟨N_T*-size row vector:* $n^r(t)$⟩
reservoir total accumulation at each time step [veh].

`Inflow` : ⟨N_T*-size row vector:* $q^r_{\text{in}}(t)$⟩
reservoir total inflow at each time step [veh/s].

`Outflow` : ⟨N_T*-size row vector:* $q^r_{\text{out}}(t)$⟩
reservoir total outflow at each time step [veh/s].

`Nin` : ⟨N_T*-size row vector:* $N^r_{\text{in}}(t)$⟩
reservoir entry cumulative count at each time step [veh].

`Nout` : ⟨N_T*-size row vector:* $N^r_{\text{out}}(t)$⟩
reservoir exit cumulative count at each time step [veh].

`AccPerRoute` : ⟨N*-by-*N_T *matrix:* $n^r_p(t)$⟩
partial accumulation per route in **`RoutesID`** at each time step [veh]. The row order follows the order of the routes in **`RoutesID`**.

`AccCircuPerRoute` : ⟨N*-by-*N_T *matrix:* $n^r_{\text{circu},p}(t)$⟩

partial circulating accumulation (not queuing for transferring to other reservoirs) per route in **RoutesID** at each time step [veh]. The row order follows the order of the routes in **RoutesID**. This field is only useful for the "queuedyn" diverge model in the accumulation-based model (**DivergeModel** in the **Simulation** structure). In other models, it equals **AccPerRoute** by default.

AccQueuePerRoute : $\langle N\text{-}by\text{-}N_T$ *matrix:* $n^r_{\text{queue},p}(t)\rangle$
partial queuing accumulation (vehicles waiting for transferring to other reservoirs) per route in **RoutesID** at each time step [veh]. The row order follows the order of the routes in **RoutesID**. This field is only useful for the "queuedyn" diverge model in the accumulation-based model (**DivergeModel** in the **Simulation** structure). In other models, it is null by default.

InflowPerRoute : $\langle N\text{-}by\text{-}N_T$ *matrix:* $q^r_{\text{in},p}(t)\rangle$
partial inflow per route in **RoutesID** at each time step [veh/s]. The row order follows the order of the routes in **RoutesID**.

OutflowPerRoute : $\langle N\text{-}by\text{-}N_T$ *matrix:* $q^r_{\text{out},p}(t)\rangle$
partial outflow per route in **RoutesID** at each time step [veh/s]. The row order follows the order of the routes in **RoutesID**.

OutflowCircuPerRoute : $\langle N\text{-}by\text{-}N_T$ *matrix:* $q^r_{\text{circu},p}(t)\rangle$
partial circulating outflow (flow into the queue for transferring to other reservoirs) per route in **RoutesID** at each time step [veh/s]. The row order follows the order of the routes in **RoutesID**. This field is only useful for the "queuedyn" diverge model in the accumulation-based model (**DivergeModel** in the **Simulation** structure). In other models, it equals **OutflowPerRoute** by default.

NinPerRoute : $\langle N\text{-}by\text{-}N_T$ *matrix:* $N^r_{\text{in},p}(t)\rangle$
partial entry cumulative count per route in **RoutesID** at each time step [veh]. The row order follows the order of the routes in **RoutesID**.

NoutPerRoute : $\langle N\text{-}by\text{-}N_T$ *matrix:* $N^r_{\text{out},p}(t)\rangle$
partial exit cumulative count per route in **RoutesID** at each time step [veh]. The row order follows the order of the routes in **RoutesID**.

NoutCircuPerRoute : $\langle N\text{-}by\text{-}N_T$ *matrix:* $N^r_{\text{circu},p}(t)\rangle$
partial exit cumulative count of the circulating outflow per route in **RoutesID** at each time step [veh]. The row order follows the order of the routes in **RoutesID**. This field is only useful for the "queuedyn" diverge model in the accumulation-based model (**DivergeModel** in the **Simulation** structure). In other models, it equals **NoutPerRoute** by default.

6.2.3 Route structure

ResOriginID : ⟨*integer*⟩
origin reservoir ID.

ResDestinationID : ⟨*integer*⟩
destination reservoir ID.

NodeOriginID : ⟨*integer*⟩
origin macroscopic node ID.

NodeDestinationID : ⟨*integer*⟩

destination macroscopic node ID.

ResRouteIndex : $\langle N_R$*-size row vector*$\rangle$
list of indexes of the reservoirs in the route p when this route cross the reservoirs. The order of indexes follows the ID of reservoirs: $1, 2, \ldots, N_R$. A default 0 index is given when the route does not cross a reservoir.

Length : $\langle$*double* $> 0\rangle$
route total length [m].

TravelTime : $\langle N_T$*-size row vector:* $T_p(t)\rangle$
route experienced travel time at each time step [s]. $T_p(t)$ is the time spent by a user exiting the last reservoir of the route at t.

TotalTime : $\langle$*double:* $\langle T_p\rangle_i > 0\rangle$
route average travel time [s] over the current assignment period, for the current MSA loop iteration i.

FreeFlowTravelTime : $\langle$*double* $> 0\rangle$
route free-flow travel time [s], calculated as the sum of its trip lengths over the corresponding reservoir free-flow speed: $\sum_{r\in R_p} L_p^r/u^r$.

OldTT : $\langle$*double:* $\langle T_p\rangle_{i-1} > 0\rangle$
route average travel time [s] over the current assignment period, for the previous MSA loop iteration $i-1$. Used for convergence calculations only.

AssignCoeff : $\langle$*double* $a_{p,i} \in [0, 1]\rangle$
assignment coefficient of the current MSA loop iteration i of the current assignment period.

ListAssignCoeff : $\langle$*row vector:* $[a_p^1, a_p^2, \ldots]\rangle$
list of assignment coefficients of all assignment periods.

Demand : $\langle N_T$*-size row vector:* $\lambda_p(t)\rangle$
route demand at each time step [veh/s].

NVehicles : $\langle$*double* $> 0\rangle$
number of vehicles created during the simulation to travel on the route, calculated as the integration of the demand profile over the simulation duration: $\int_0^{T_s} \lambda_p(t)dt$.

6.2.4 Vehicle structure

Vehicle is an array structure (ID from 1 to **NumVeh**: $u \in \{1, \ldots, N_U\}$) that describes each vehicle. This structure is automatically created and used in the trip-based model only. It has the following fields:

RouteID : $\langle$*integer:* $p\rangle$
ID of the route on which the vehicle is traveling.

CreationTime : $\langle$*double:* $t^u_{\text{crea}} > 0\rangle$
creation time [s] of the vehicle according to the route demand profile.

EntryTimes : $\langle N$*-size row vector:* $T^u_{\text{in}} = [t_1, t_2, \ldots]\rangle$
list of entry times [s] in the successive reservoirs of the route p assigned to the vehicle. Same size N as the reservoir sequence R_p.

ExitTimes : $\langle N$*-size row vector:* $T^u_{\text{out}} = [t_1, t_2, \ldots]\rangle$
list of exit times [s] in the successive reservoirs of the route p assigned to the vehicle. Same size N as the reservoir sequence R_p.

`TripLength` : $\langle N$*-size row vector:* $L^u = [L_p^{r_1}, L_p^{r_2}, \dots] \rangle$
list of trip lengths [m] in the successive reservoirs of the route p assigned to the vehicle. Same size N as the reservoir sequence R_p.

`TraveledDistance` : $\langle N$*-size row vector:* $[d_1, d_2, \dots] \rangle$
list of distances traveled by the vehicle [m] in the successive reservoirs of the route p assigned to the vehicle. Same size N as the reservoir sequence R_p. These distances are higher than the corresponding trip lengths in congestion, as they are still incremented when the vehicle is waiting for transferring to another reservoir. The excess of distance traveled is artificial and may only serve as an indicator, as the actual distance traveled is the trip length.

`WaitingTimes` : $\langle N$*-size row vector:* $[\Delta t_1, \Delta t_2, \dots] \rangle$
list of waiting times [s] in the successive reservoirs of the route p assigned to the vehicle. Same size N as the reservoir sequence R_p. A waiting time corresponds to the difference between the actual entry time into the next reservoir of the route and the time at which the vehicle completed its trip length in the current reservoir.

`Purpose` : $\langle$*string: cartrip*$\rangle$
for now, default demand type (individual car trip) for all vehicles.

7.

Simulation solvers

7.1 Accumulation-based model

The accumulation-based solver is described by the pseudo-code in algorithms 1 and 2. Below are the additional notations used throughout the algorithm:

- In the **Reservoir** structure, $r \in \{1, \ldots, N_R\}$:
  - set of routes $\mathcal{P}^r = \mathcal{P}^r_{\text{int}} \oplus \mathcal{P}^r_{\text{ext}}$ in reservoir r, with $\mathcal{P}^r_{\text{int}}$ the set of routes originating in r, and $\mathcal{P}^r_{\text{ext}}$ the set of routes entering r from its borders
  - set of macroscopic nodes $\mathcal{M}^r$ in reservoir r, with $\mathcal{M}^r_{\text{int}} \subset \mathcal{M}^r$ the set of origin macroscopic nodes in r (at which the routes in $\mathcal{P}^r_{\text{int}}$ start), and $\mathcal{M}^r_{\text{ext}} \subset \mathcal{M}^r$ the set of entry border macroscopic nodes in r (through which the routes in $\mathcal{P}^r_{\text{ext}}$ enter)
  - modified entry production supply $P^r_{s,\text{ext}}$ [veh.m/s] accounting for internal trip generation in reservoir r
  - average trip length L^r_{ext} [m] corresponding to the routes originating outside reservoir r only
  - queuing vehicles $n^r_{\text{ext},p}$ [veh] at the beginning of route $p \in \mathcal{P}^r_{\text{ext}}$ (external entry)
  - outflow demand O^r_p [veh/s] and outflow supply μ^r_p [veh/s] per route $p \in \mathcal{P}^r$ to exit reservoir r
  - inflow demand λ^r_p [veh/s], inflow supply I^r_p [veh/s] and inflow merge coefficient α^r_p [-] per route $p \in \mathcal{P}^r$ to enter reservoir r
- In the **Route** structure, $p \in \{1, \ldots, N_P\}$:
  - origin reservoir $R_p[1]$, destination reservoir $R_p[end]$
  - origin node $M_p[1]$, destination node $M_p[end]$
  - previous reservoir $p^-(r)$ and next reservoir $p^+(r)$ for a given reservoir $r \in R_p$

The traffic flows are solved for a given assignment period, from its initial time t_0 to its final time t_f. The solver takes as inputs the initial accumulations $n^r_p(t_0) = n^r_{\text{circu},p}(t_0) +$

$n^r_{\text{queue},p}(t_0)$ and the route demand profiles $\lambda_p(t)$, $t_0 \leq t \leq t_f$. It returns as outputs the evolution of accumulations $n^r_p(t)$, inflows $q^r_{\text{in},p}(t)$ and outflows $q^r_{\text{out},p}(t)$, $t_0 \leq t \leq t_f$. Then, the mean speed $v^r(t)$, cumulative count curves $N^r_{\text{in},p}(t)$ and $N^r_{\text{out},p}(t)$, and experienced travel times $T_p(t)$ are calculated as follows:

$$\forall r \in \{1, \dots, N_R\},\ v^r(t) = \frac{P^r(n^r(t))}{n^r(t)};\quad n^r(t) = \sum_{p \in \mathcal{P}^r} n^r_p(t) \tag{7.1}$$

$$\forall r \in \{1, \dots, N_R\}, p \in \mathcal{P}^r,\ N^r_{\text{in},p}(t) = \int_{t_0}^{t} q^r_{\text{in},p}(\tau)d\tau + N^r_{\text{in},p}(t_0) \tag{7.2}$$

$$N^r_{\text{out},p}(t) = \int_{t_0}^{t} q^r_{\text{out},p}(\tau)d\tau \tag{7.3}$$

$$\forall p \in \{1, \dots, N_P\},\ T_p(t) = t - [N^{R_p[1]}_{\text{in},p}]^{-1}\left(N^{R_p[end]}_{\text{out},p}(t)\right) \tag{7.4}$$

where $[\cdot]^{-1}$ is the function inverse operator.

The $Merge()$ algorithm used to calculate inflow supply in reservoirs was proposed in Leclercq & Becarie (2012) and consists in an extension of the fair merge of Daganzo (1995). It ensures that the total available capacity is always fully used when only some of the merging flows are limited. It is described by the pseudo-code in algorithm 3. Note that it can be applied to either flow or production values.

Algorithm 1. *Accumulation-based solver (1/2)*

Input: reservoir initial accumulation $n_p^r(t_0) = n_{\text{circu},p}^r(t_0) + n_{\text{queue},p}^r(t_0)$ per route, route demand profile $\lambda_p(t)$, initial time t_0, final time t_f and timestep δt_s

Output: reservoir accumulation $n_p^r(t) = n_{\text{circu},p}^r(t) + n_{\text{queue},p}^r(t)$, inflow $q_{\text{in},p}^r(t)$ and outflow $q_{\text{out},p}^r(t)$ per route

for $t = t_0$ **to** t_f **by** δt_s **do**

  for $r = 1$ **to** N_R **do**

    Queue factor: $Q_F = 1 - \sum_{p\in\mathcal{P}^r} n_{\text{queue},p}^r(t_0)/n_j^r$ (queuedyn diverge)

    Production modification: $P^{r*} = Q_F P^r(n^r(t));\ P_c^{r*} = Q_F P_c^r$

    Outflow demand:

    $\forall p \in \mathcal{P}^r, q_{\text{circu},p}^r(t) = \frac{n_p^r(t)}{n^r(t)} \frac{P^{r*}}{L_p^r}$

    $$\begin{cases} O_p^r = q_{\text{circu},p}^r(t)\ (n \le n_c^r);\ O_p^r = \frac{n_p^r(t)}{n^r(t)} \frac{P_c^{r*}}{L_p^r}\ (n > n_c^r) & \text{if maxdem diverge} \\ O_p^r = q_{\text{circu},p}^r(t)\ (n_{\text{queue},p}^r(t) = 0);\ O_p^r = C_m(t)\ (n_{\text{queue},p}^r(t) > 0) & \text{if queuedyn diverge} \\ \quad m = \text{exit node of } p \text{ in } r & \\ O_p^r = q_{\text{circu},p}^r(t) & \text{if decrdem diverge} \end{cases}$$

    $\forall p \in \mathcal{P}^r, q_{\text{circu},p}^r(t) = O_p^r$

    Production supply: $P_{s,\text{ext}}^r = P_s^r(n^r(t)) - \sum_{p\in\mathcal{P}_{\text{int}}^r} L_p^r \lambda_p(t)$

    Average trip length: $L_{\text{ext}}^r = \sum_{p\in\mathcal{P}_{\text{ext}}^r} n_p^r(t) / \sum_{p\in\mathcal{P}_{\text{ext}}^r} n_p^r(t)/L_p^r$

  end for

  for $r = 1$ **to** N_R **do**

    Inflow demand:

    $\forall p \in \mathcal{P}^r$, **if** $r = R_p[1]$ **then** $\lambda_p^r = \lambda_p(t)\ (n_{\text{ext},p}^r = 0);\ \lambda_p^r = C_{M_p[1]}(t)\ (n_{\text{ext},p}^r > 0)$ **else** $\lambda_p^r = O_p^{p^-(r)}$

    Merging coefficients:

    $$\forall p \in \mathcal{P}_{\text{ext}}^r, \begin{cases} \alpha_p^r = n_p^r(t)/\sum_{k\in\mathcal{P}_{\text{ext}}^r} n_k^r(t) & \text{if endogenous merge} \\ \alpha_p^r = \lambda_p^r/\sum_{k\in\mathcal{P}_{\text{ext}}^r} \lambda_k^r & \text{if demprorata merge} \\ \alpha_p^r = 1/|\mathcal{P}_{\text{ext}}^r| & \text{if equiproba merge} \end{cases}$$

    Border inflow supply:

    $$\forall m \in \mathcal{M}_{\text{ext}}^r, \{I_p^{r*}\}_{p\in\mathcal{P}^m} = \text{Merge}\left(\{\lambda_p^r\}_{p\in\mathcal{P}^m}, \left\{\frac{\alpha_p^r}{\sum_{k\in\mathcal{P}^m}\alpha_k^r}\right\}_{p\in\mathcal{P}^m}, C_m(t)\right)$$

    Reservoir inflow supply:

    $$\begin{cases} \{L_p^r I_p^r\}_{p\in\mathcal{P}_{\text{ext}}^r} = \text{Merge}\left(\{L_p^r I_p^{r*}\}_{p\in\mathcal{P}_{\text{ext}}^r}, \{\alpha_p^r\}_{p\in\mathcal{P}_{\text{ext}}^r}, P_{s,\text{ext}}^r\right) & \text{if endogenous merge} \\ \{I_p^r\}_{p\in\mathcal{P}_{\text{ext}}^r} = \text{Merge}\left(\{I_p^{r*}\}_{p\in\mathcal{P}_{\text{ext}}^r}, \{\alpha_p^r\}_{p\in\mathcal{P}_{\text{ext}}^r}, \frac{P_{s,\text{ext}}^r}{L_{\text{ext}}^r}\right) & \text{otherwise} \end{cases}$$

  end for

Algorithm 2. *Accumulation-based solver (2/2)*

(for loop)

for $r = 1$ **to** N_R **do**

Outflow supply:

$\forall p \in \mathcal{P}^r$, **if** $r = R_p[end]$ **then** $\mu_p^r = C_{M_p[end]}(t)$ **else** $\mu_p^r = I_p^{p^+(r)}$

Effective outflow: $\forall p \in \mathcal{P}^r$,

$$\begin{cases} q_{\text{out},p}^r(t) = \frac{n_p^r(t)}{n_k^r(t)} \frac{L_k^r}{L_p^r} \min\left[O_k^r; \mu_k^r\right]; \ k = \arg\min_{p \in \mathcal{P}^r} \frac{\mu_p^r}{O_p^r} & \text{if maxdem diverge} \\ q_{\text{out},p}^r(t) = \min\left[O_p^r; \mu_p^r\right] & \text{otherwise} \end{cases}$$

Effective inflow:

$\forall p \in \mathcal{P}_{\text{int}}^r, q_{\text{in},p}^r(t) = \lambda_p(t)$

$\forall p \in \mathcal{P}_{\text{ext}}^r$, **if** $r = R_p[1]$ **then** $q_{\text{in},p}^r(t) = I_p^r$ **else** $q_{\text{in},p}^r(t) = q_{\text{out},p}^{p^-(r)}(t)$

External queue update:

$\forall p \in \mathcal{P}_{\text{ext}}^r$, **if** $r = R_p[1]$ **then** $n_{\text{ext},p}^r = n_{\text{ext},p}^r + \delta t_s \left(\lambda_p(t) - q_{\text{in},p}^r(t)\right)$

Accumulation update: $\forall p \in \mathcal{P}^r$,

$$\begin{cases} n_{\text{circu},p}^r(t+\delta t_s) = n_{\text{circu},p}^r(t) + \delta t_s \left(q_{\text{in},p}^r(t) - q_{\text{circu},p}^r(t)\right) & \text{if queuedyn diverge} \\ n_{\text{queue},p}^r(t+\delta t_s) = n_{\text{queue},p}^r(t) + \delta t_s \left(q_{\text{circu},p}^r(t) - q_{\text{out},p}^r(t)\right) & \\ n_{\text{circu},p}^r(t+\delta t_s) = n_{\text{circu},p}^r(t) + \delta t_s \left(q_{\text{in},p}^r(t) - q_{\text{out},p}^r(t)\right) & \text{otherwise} \\ n_{\text{queue},p}^r(t+\delta t_s) = 0 & \end{cases}$$

$$n_p^r(t+\delta t_s) = n_{\text{circu},p}^r(t+\delta t_s) + n_{\text{queue},p}^r(t+\delta t_s)$$

end for

end for

Algorithm 3. *Fair merge with multiple incoming flows*

Function *Merge*$(\{\Lambda_i\}_{1\leq i\leq M}, \{\alpha_i\}_{1\leq i\leq M}, C)$

Input: set of M incoming demand flows (resp. productions) $\{\Lambda_i\}_{1\leq i\leq M}$ with respective merge coefficients $\{\alpha_i\}_{1\leq i\leq M}$ towards a unique entry with flow (resp. production) capacity C

Output: resulting effective inflows (resp. entering productions) $\{Q_i\}_{1\leq i\leq M}$

Initialization:
set of unserved flows: $U = \{1, \ldots, M\}$
sum of all coefficients in U: $\alpha_U = 1$
total inflow already served: $Q_F = 0$

while $U \neq \emptyset$ **do**
  set $U' = \emptyset;\ \alpha'_U = 0;\ Q'_F = 0$
  for $i \in U$ **do**
    if $\Lambda_i < \alpha_i/\alpha_U(C - Q_F)$ **then**
      demand i is served: $Q_i = \Lambda_i$
      $Q'_F = Q'_F + Q_i$
    else
      demand i is not served: $Q_i = \alpha_i/\alpha_U(C - Q_F)$
      $U' = \{U', i\}$
      $\alpha'_U = \alpha'_U + \alpha_i$
    end if
  end for
  update $U = U';\ \alpha_U = \alpha'_U;\ Q_F = Q_F + Q'_F$
  if $\sum_{i=1}^{M} Q_i = C$ **then**
    stop the procedure by setting $U = \emptyset$
  end if
end while

end function

7.2 Trip-based model

The trip-based solver is described by the pseudo-code in algorithms 4 and 5. Below are the additional notations used throughout the algorithm (other notations from the accumulation-based solver are also used):

- In the **Reservoir** structure, $r \in \{1, \dots, N_R\}$:
  - entry demand time $t^r_{\text{in},d,p}$ and supply time $t^r_{\text{in},s,p}$ per route $p \in \mathcal{P}^r$
  - exit demand time $t^r_{\text{out},d,p}$ and supply time $t^r_{\text{out},s,p}$ per route $p \in \mathcal{P}^r$
  - last entry time $t^r_{\text{last in},p}$ and last exit time $t^r_{\text{last out},p}$ per route $p \in \mathcal{P}^r$
  - estimated inflow demand $q^r_{\text{in},d,p}$ per route $p \in \mathcal{P}^r$
  - possible next entry time t^r_{in} and next exit time t^r_{out}
  - list of traveling vehicles per route U^r_p sorted by remaining travel distance, $p \in \mathcal{P}^r$
  - traveled distance during the elapsed time Δt for all the vehicles in r: TD^r
- In the **Vehicle** structure, $u \in \{1, \dots, N_U\}$:
  - current traveled distance TD^u
  - current traveled time TT^u

Like the accumulation-based model, the traffic flows are solved for a given assignment period, from its initial time t_0 to its final time t_f. The solver takes as inputs the initial accumulations $n^r_p(t_0)$, cumulative count curves $N^r_{\text{in},p}(t_0)$ and $N^r_{\text{out},p}(t_0)$, and the route demand profiles $\lambda_p(t)$, $t_0 \leq t \leq t_f$. It also takes the beginning trajectory of vehicles that are already traveling: the list of trip lengths L^u, the completed entry times T^u_{in} and exit times T^u_{out}. It returns as outputs the evolution of accumulations $n^r_p(t)$, cumulative count curves $N^r_{\text{in},p}(t)$ and $N^r_{\text{out},p}(t)$, mean speed $v^r(t)$ and travel times $T_p(t)$, $t_0 \leq t \leq t_f$. Then, the inflows $q^r_{\text{in},p}(t)$ and outflows $q^r_{\text{out},p}(t)$ are calculated as follows:

$$\forall r \in \{1, \dots, N_R\}, p \in \mathcal{P}^r, \quad q^r_{\text{in},p}(t) = \frac{dN^r_{\text{in},p}}{dt} \tag{7.5}$$

$$q^r_{\text{out},p}(t) = \frac{dN^r_{\text{out},p}}{dt} \tag{7.6}$$

In practice, the derivatives are not computed on the direct outputs from the trip-based solver to avoid high oscillations in flow. The cumulative count curves are thus smoothed and filtered over a fixed number of consecutive vehicles before the numerical derivative is applied.

The $MergeTime()$ algorithm used to calculate entry supply times in reservoirs corresponds to the adaptation of the function in algorithm 3 used in the accumulation-based solver. It is described by the pseudo-code in algorithm 6.

One of the drawbacks of the trip-based solver is its high complexity compared to the accumulation-based one when the demand is high (numerous vehicles created). Moreover, while the computational time of the latter can be improved by increasing the time step (up to the numerical stability limit of this type of finite difference scheme), nothing similar can be done to reduce the number of calculations in the trip-based solver.

However, if the user is only interested in the evolution of aggregated variables (accumulation, flow, mean speed, etc), and not in the detail of each individual vehicle trajectory, reducing the computational burden can be achieved by reducing the demand during the simulation, which will create less vehicles. The resulting accumulations and flows are then scaled up by the inverse of the demand reduction factor. For the whole simulation to be consistent, other boundary conditions (supply limitations) must also be down-scaled accordingly, as well as each reservoir MFD. This method of demand down-scaling relies on the following equivalence, presented here with a single conservation equation (see also chap. 6 of Mariotte, 2018):

$$\frac{dn}{dt} = \lambda(t) - \frac{P(n(t))}{L} \iff \frac{d(S_F n)}{dt} = S_F\lambda(t) - \frac{P_{SF}(S_F n(t))}{L} \tag{7.7}$$

where S_F is the scaling factor applied to the demand $\lambda(t)$, the MFD $P(n)$ and other boundary conditions. If $S_F < 1$, solving the right-hand equation is less costly in the trip-based solver because the demand is reduced. The down-scaled MFD $P_{SF}(n)$ corresponds to the following transformation:

$$P_{SF}(n) = S_F P(n/S_F) \tag{7.8}$$

When $P(n)$ is approximated by a set of N_b linear branches such that $P(n) = \min_{1\leq i\leq N_b}[w_i n + P_{c,i}]$, this transformation can be easily applied:

$$P_{SF}(n) = \min_{1\leq i\leq N_b}[w_i n + S_F P_{c,i}] \tag{7.9}$$

The actual accumulations and cumulative curves are then calculated by multiplying the outputs from the trip-based solver by $1/S_F$. The simulation scaling factor S_F is defined by the field **`TripbasedSimuFactor`** of the **Simulation** structure.

Algorithm 4. *Trip-based solver (1/2)*

Input: reservoir initial accumulation $n_p^r(t_0)$, cumulative curves $N_{\text{in},p}^r(t_0)$ and $N_{\text{out},p}^r(t_0)$ per route, route demand profile $\lambda_p(t)$, vehicle beginning trajectory L^u, T_{in}^u and T_{out}^u, initial time t_0, final time t_f

Output: reservoir accumulation $n_p^r(t)$, cumulative curves $N_{\text{in},p}^r(t)$ and $N_{\text{out},p}^r(t)$ per route, mean speed $v^r(t)$, route travel time $T_p(t)$, vehicle trajectory L^u, T_{in}^u and T_{out}^u

Current time: $t = t_0$; elapsed time: $\Delta t = 0$; current vehicle: u_0

while $t < t_f$ **do**

Set by default: $\forall r \in \{1, \dots, N_R\}, p \in \mathcal{P}^r, n_p^r(t) = n_p^r(t - \Delta t);\ N_{\text{in},p}^r(t) = N_{\text{in},p}^r(t - \Delta t);\ N_{\text{out},p}^r(t) = N_{\text{out},p}^r(t - \Delta t);\ v^r(t) = v^r(t - \Delta t)$

Update traveled distances and times:

for $r = 1$ **to** N_R **do**

$TD^r = \Delta t.v^r(t)$

$\forall p \in \mathcal{P}^r, \forall u \in U_p^r,\ TD^u = TD^u + TD^r;\ TT^u = TT^u + \Delta t$

end for

Current vehicle u_0 in reservoir r_0, at index i_0 on route p_0 ($r_0 = R_{p_0}[i_0]$), set of reservoirs where accumulation changed: $\mathcal{SR} = \emptyset$

Update entering and/or exiting reservoir information:

if *vehicle* u_0 *exits* r_0 **then**

$T_{\text{out}}^{u_0}[i_0] = t$

$n_{p_0}^{r_0}(t) = n_{p_0}^{r_0}(t) - 1;\ N_{\text{out},p_0}^{r_0}(t) = N_{\text{out},p_0}^{r_0}(t) + 1;\ v^{r_0}(t) = P_{SF}^{r_0}(n^{r_0}(t))/n^{r_0}(t)$

$\mathcal{SR} = \{\mathcal{SR}, r_0\}$

remove u_0 from the list $U_{p_0}^{r_0}$

end if

if *vehicle* u_0 *enters* r_0 **or** *enters next reservoir* $p_0^+(r_0)$ *of the route* p_0 **then**

if *vehicle enters* $p_0^+(r_0)$ **then** $r_0 = p^+(r_0), i_0 = i_0 + 1$

$T_{\text{in}}^{u_0}[i_0] = t$

$n_{p_0}^{r_0}(t) = n_{p_0}^{r_0}(t) + 1;\ N_{\text{in},p_0}^{r_0}(t) = N_{\text{in},p_0}^{r_0}(t) + 1;\ v^{r_0}(t) = P_{SF}^{r_0}(n^{r_0}(t))/n^{r_0}(t)$

$\mathcal{SR} = \{\mathcal{SR}, r_0\}$

append u_0 to the list $U_{p_0}^{r_0}$ and sort the list by remaining travel distance $L^u[i] - TD^u, u \in U_{p_0}^{r_0}$

else

vehicle u_0 completed its trip

route travel time: $T_{p_0}(t) = T_{\text{out}}^{u_0}[end] - T_{\text{in}}^{u_0}[1]$

end if

Reservoir exit demand times:

for $r \in \mathcal{SR}$ **do**

for $p \in \mathcal{P}^r$ **do**

first vehicle to exit: $u_0 = U_p^r[1]$ at reservoir index i_0

if $n^r(t) > n_c^r$ **and** *maxdem diverge* **then** $TD^{u_0} = L^{u_0}[i_0]$

$t_{\text{out},d,p}^r = t + (L^{u_0}[i_0] - TD^{u_0})/v^r(t)$

end for

end for

Algorithm 5. *Trip-based solver (2/2)*

(while loop)

Reservoir entry demand times and estimated inflow demand:

for $r \in \mathcal{SR}$ **do**

$\forall p \in \mathcal{P}^r$, **if** $r = R_p[1]$ **then** $t^r_{\text{in},d,p} = t^r_{\text{last in},p} + 1/(S_F \lambda_p(t))$ **else** $t^r_{\text{in},d,p} = t^{p^-(r)}_{\text{out},d,p}$

$\forall p \in \mathcal{P}^r$, **if** $r = R_p[1]$ **then** $q^r_{\text{in},d,p} = S_F \lambda_p(t)$ **else**

$q^r_{\text{in},d,p} = 1/\left(t^{p^-(r)}_{\text{out},d,p} - t^{p^-(r)}_{\text{last out},p}\right)$

end for

Reservoir entry supply times:

for $r \in \mathcal{SR}$ **do**

merging coefficients:

$$\forall p \in \mathcal{P}^r_{\text{ext}}, \begin{cases} \alpha^r_p = n^r_p(t)/\sum_{k\in\mathcal{P}^r_{\text{ext}}} n^r_k(t) & \text{if endogenous merge} \\ \alpha^r_p = q^r_{\text{in},d,p}/\sum_{k\in\mathcal{P}^r_{\text{ext}}} q^r_{\text{in},d,k} & \text{if demprorata merge} \\ \alpha^r_p = 1/|\mathcal{P}^r_{\text{ext}}| & \text{if equiproba merge} \end{cases}$$

modification of entry demand times due to border supply:

$\forall m \in \mathcal{M}^r_{\text{ext}}, \{t^{r*}_{\text{in},d,p}\}_{p\in\mathcal{P}^m} =$

$$\text{MergeTime}\left(\{t^r_{\text{in},d,p}\}_{p\in\mathcal{P}^m}, \left\{\frac{\alpha^r_p}{\sum_{k\in\mathcal{P}^m}\alpha_k}\right\}_{p\in\mathcal{P}^m}, \{1\}_{p\in\mathcal{P}^m}, \{t^r_{\text{last in},p}\}_{p\in\mathcal{P}^m}, S_F C_m(t)\right)$$

$\forall m \in \mathcal{M}^r_{\text{ext}}, p \in \mathcal{P}^m, t^r_{\text{in},d,p} = \max\left[t^r_{\text{in},d,p}; t^{r*}_{\text{in},d,p}\right]$

production supply: $P^r_{s,\text{ext}} = S_F P^r_s(n^r(t)) - \sum_{p\in\mathcal{P}^r_{\text{int}}} L^r_p S_F \lambda_p(t)$

average trip length: $L^r_{\text{ext}} = \sum_{p\in\mathcal{P}^r_{\text{ext}}} n^r_p(t)/\sum_{p\in\mathcal{P}^r_{\text{ext}}} n^r_p(t)/L^r_p$

entry supply times: $\{t^r_{\text{in},s,p}\}_{p\in\mathcal{P}^r_{\text{ext}}} =$

$$\begin{cases} \text{MergeTime}\left(\{t^r_{\text{in},d,p}\}_{p\in\mathcal{P}^r_{\text{ext}}}, \{\alpha^r_p\}_{p\in\mathcal{P}^r_{\text{ext}}}, \{L^r_p\}_{p\in\mathcal{P}^r_{\text{ext}}}, \{t^r_{\text{last in},p}\}_{p\in\mathcal{P}^r_{\text{ext}}}, P^r_{s,\text{ext}}\right) \\ \quad \text{if endogenous merge} \\ \text{MergeTime}\left(\{t^r_{\text{in},d,p}\}_{p\in\mathcal{P}^r_{\text{ext}}}, \{\alpha^r_p\}_{p\in\mathcal{P}^r_{\text{ext}}}, \{1\}_{p\in\mathcal{P}^r_{\text{ext}}}, \{t^r_{\text{last in},p}\}_{p\in\mathcal{P}^r_{\text{ext}}}, \frac{P^r_{s,\text{ext}}}{L^r_{\text{ext}}}\right) \\ \quad \text{if demprorata merge} \end{cases}$$

$\{t^r_{\text{in},s,p}\}_{p\in\mathcal{P}^r_{\text{int}}} = \{t^r_{\text{in},d,p}\}_{p\in\mathcal{P}^r_{\text{int}}}$

end for

Reservoir exit supply times: $\forall r \in \{1,\dots,N_R\}$,

$\forall p \in \mathcal{P}^r$, **if** $r = R_p[end]$ **then** $t^r_{\text{out},s,p} = t^r_{\text{last out},p} + 1/(S_F C_{M_p[end]}(t))$ **else**

$t^r_{\text{out},s,p} = t^{p^+(r)}_{\text{in},s,p}$

Next possible entry time (route beginning): $\forall r \in \{1,\dots,N_R\}$,

$t^r_{\text{in}} = \min_{p\in\mathcal{P}^r/r=R_p[1]}\left[\max[t^r_{\text{in},d,p}; t^r_{\text{in},s,p}]\right]$

$t_{\text{in}} = \min_{1\le r\le N_R} t^r_{\text{in}}$

Next possible exit time: $\forall r \in \{1,\dots,N_R\}$,

$$\begin{cases} t^r_{\text{out}} = \max[t^r_{\text{out},d,k}; t^r_{\text{out},s,k}];\ k = \arg\min_{p\in\mathcal{P}^r} t^r_{\text{out},d,p} & \text{if maxdem diverge} \\ t^r_{\text{out}} = \min_{p\in\mathcal{P}^r}\left[\max[t^r_{\text{out},d,p}; t^r_{\text{out},s,p}]\right] & \text{if decrdem diverge} \end{cases}$$

$t_{\text{out}} = \min_{1\le r\le N_R} t^r_{\text{out}}$

Next event time: $t_{\text{event}} = \min[t_{\text{in}}; t_{\text{out}}]$

Save corresponding vehicle u_0

Elapsed time update: $\Delta t = t_{\text{event}} - t$; time update: $t = t_{\text{event}}$

end while

Algorithm 6. *Fair merge (entry supply times) with multiple incoming flows*

Function *MergeTime*$(\{T_{d,i}\}_{1\leq i\leq M}, \{\alpha_i\}_{1\leq i\leq M}, \{L_i\}_{1\leq i\leq M}, \{T_{\text{last},i}\}_{1\leq i\leq M}, C)$

Input: set of M incoming flows with entry demand times $\{T_{d,i}\}_{1\leq i\leq M}$ with respective merge coefficients $\{\alpha_i\}_{1\leq i\leq M}$, trip lengths $\{L_i\}_{1\leq i\leq M}$ and last entry times $\{T_{\text{last},i}\}_{1\leq i\leq M}$ towards a unique entry with flow (resp. production) capacity C. If C is in flow units, put by default $\forall i \in \{1,\dots,M\}, L_i = 1$ (no need for the trip length input)

Output: resulting entry supply times $\{T_{s,i}\}_{1\leq i\leq M}$

Initialization:
set of unserved flows: $U = \{1,\dots,M\}$
sum of all coefficients in U: $\alpha_U = 1$
total inflow already served: $Q_F = 0$

while $U \neq \emptyset$ **do**
  set $U' = \emptyset;\ \alpha'_U = 0;\ Q'_F = 0$
  for $i \in U$ **do**
    Inflow supply for demand i: $Q_{s,i} = \alpha_i/\alpha_U(C - Q_F)/L_i$
    Entry supply time for demand i: $T_{s,i} = T_{\text{last},i} + 1/Q_{s,i}$
    Effective flow for demand i: $Q_i = L_i Q_{s,i}$
    if $T_{d,i} \geq T_{s,i}$ **then**
      demand i is served
      $Q'_F = Q'_F + Q_i$
    else
      demand i is not served
      $U' = \{U', i\}$
      $\alpha'_U = \alpha'_U + \alpha_i$
    end if
  end for
  update $U = U';\ \alpha_U = \alpha'_U;\ Q_F = Q_F + Q'_F$
  if $\sum_{i=1}^{M} Q_i = C$ **then**
    stop the procedure by setting: $U = \emptyset$
  end if
end while
end function

Bibliography

Batista, S. F. A. (2018). *Dynamic traffic assignment for multi-regional transportation systems considering different kinds of users' behavior*. Phd thesis, University of Lyon. NNT:2018LYSET009.

Batista, S. F. A. & Leclercq, L. (2019). Regional dynamic traffic assignment framework for macroscopic fundamental diagram multi-regions models. *Transportation Science*, 53(6):1563–1590.

Batista, S. F. A., Leclercq, L. & Geroliminis, N. (2019). Estimation of regional trip length distributions for the calibration of the aggregated network traffic models. *Transportation Research Part B: Methodological*, 122:192–217.

Daganzo, C. F. (1995). The cell transmission model, part ii: Network traffic. *Transportation Research Part B: Methodological*, 29(2):79–93.

Haddad, J. (2017). Optimal perimeter control synthesis for two urban regions with aggregate boundary queue dynamics. *Transportation Research Part B: Methodological*, 96:1–25.

Ji, Y. & Geroliminis, N. (2012). On the spatial partitioning of urban transportation networks. *Transportation Research Part B: Methodological*, 46(10):1639–1656.

Laval, J. A. & Castrillon, F. (2015). Stochastic approximations for the macroscopic fundamental diagram of urban networks. *Transportation Research Part B: Methodological*, 81(3):904–916.

Leclercq, L. & Becarie, C. (2012). Meso lighthill-whitham and richards model designed for network applications. In *Transportation Research Board 91st Annual Meeting*, 12-0387. Washington DC.

Liu, H. X., He, X. & He, B. (2007). Method of successive weighted averages (mswa) and self-regulated averaging schemes for solving stochastic user equilibrium problem. *Networks and Spatial Economics*, 9(4):485.

Lopez, C., Leclercq, L., Krishnakumari, P., Chiabaut, N. & van Lint, H. (2017). Revealing the day-to-day regularity of urban congestion patterns with 3d speed maps. *Scientific Reports*, 7(14029):1–11.

Mariotte, G. (2018). *Dynamic Modeling of Large-Scale Urban Transportation Systems*. Phd thesis, University of Lyon. NNT:2018LYSET010.

Mariotte, G. & Leclercq, L. (2019). Flow exchanges in multi-reservoir systems with spillbacks. *Transportation Research Part B: Methodological*, 122:327–349.

Mariotte, G., Leclercq, L. & Laval, J. A. (2017). Macroscopic urban dynamics: Analytical and numerical comparisons of existing models. *Transportation Research Part B: Methodological*, 101:245–267.

Ni, W. & Cassidy, M. (2019). City-wide traffic control: Modeling impacts of cordon queues. *Transportation Research Part C: Emerging Technologies*, in press.

Ramezani, M., Haddad, J. & Geroliminis, N. (2015). Dynamics of heterogeneity in urban networks: aggregated traffic modeling and hierarchical control. *Transportation Research Part B: Methodological*, 74:1–19.

Saeedmanesh, M. & Geroliminis, N. (2016). Clustering of heterogeneous networks with directional flows based on "snake" similarities. *Transportation Research Part B: Methodological*, 91:250–269.

Saeedmanesh, M. & Geroliminis, N. (2018). Exact formulation of homogeneous and compact-shaped partitioning in large-scale heterogeneous traffic networks. In *Transportation Research Board 97th Annual Meeting*, 18-06566. Washington DC.

Sbayti, H., Lu, C.-C. & Mahmassani, H. S. (2007). Efficient implementation of method of successive averages in simulation-based dynamic traffic assignment models for large-scale network applications. *Transportation Research Record: Journal of the Transportation Research Board*, 2029:22–30.

Sheffi, Y. (1985). *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*, chap. Stochastic User Equilibrium, pages 322–331. Prentice-Hall Inc, Englewood Cliffs, New Jersey, USA.

Yildirimoglu, M. & Geroliminis, N. (2014). Approximating dynamic equilibrium conditions with macroscopic fundamental diagrams. *Transportation Research Part B: Methodological*, 70:186–200.

Yildirimoglu, M., Ramezani, M. & Geroliminis, N. (2015). Equilibrium analysis and route guidance in large-scale networks with mfd dynamics. *Transportation Research Part C: Emerging Technologies*, 59:404–420.

Yildirimoglu, M., Sirmatel, I. I. & Geroliminis, N. (2018). Hierarchical control of heterogeneous large-scale urban road networks via path assignment and regional route guidance. *Transportation Research Part B: Methodological*, 118:106–123.

Zhou, Z., Lin, S. & Xi, Y. (2012). A dynamic network partition method for heterogenous urban traffic networks. In *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pages 820–825.