CHAPTER 2

Let's Get Started

Marcel Rieser, Andreas Horni and Kai Nagel

This chapter explains how to set up and run MATSim and describes the requirements for building a basic scenario. Updated information may be available from `http://matsim.org`, in particular from `http://matsim.org/docs`.

Getting the source code into different computing environments and extending MATSim through the API is described in Part II, Chapter 45.

2.1 Running MATSim

2.1.1 Setting Up MATSim

To run MATSim, you must install the Java SE (Java Standard Edition) that complies with the appropriate MATSim version. At this time, this is Java SE 7.

Download of the release You also need the official *MATSim release,* a zip file (usually designated with the version number `matsim-yy.yy.yy.zip`), that includes everything required to run it. It can be downloaded following the "release" link under `http://matsim.org/downloads`. Unzip results in the **MATSim directory tree**. Continue with Section 2.1.2.

The MATSim directory tree on the web If you want to look at the development version, or look at things without downloading and installing a zip file: On GitHub, the root of the **MATSim directory tree** (i.e., excluding so-called contribs and playgrounds) is at `https://github.com/matsim-org/matsim/tree/master/matsim`.

How to cite this book chapter:
Rieser, M, Horni, A and Nagel, K. 2016. Let's Get Started. In: Horni, A, Nagel, K and Axhausen, K W. (eds.) *The Multi-Agent Transport Simulation MATSim*, Pp. 9–22. London: Ubiquity Press. DOI: http://dx.doi.org/10.5334/baw.2. License: CC-BY 4.0

Download of nightly builds If you prefer to use the more up-to-date, but less stable, *nightly builds,* you should download, via the same URL (Uniform Resource Locator) `http://matsim.org/downloads`,

- the MATSim JAR (Java ARchive) file (usually tagged with the revision number `MATSim_ryyyy.jar`), and
- the required external libraries (`MATSim_libs.zip`). Unzipping this collection of 3rd-party libraries, you should then get a directory `libs`, with several JAR files inside. If the directory `libs` is in the same directory as the MATSim JAR file, the libraries are found automatically and do not have to be added manually to the `classpath`.

Maven A relatively new feature is that one can use MATSim as an Apache Maven plugin; both release versions and snapshots are available. See again `http://matsim.org/downloads` for more information. For someone who has used Apache Maven before, this is probably the best option. In this case, one may use the simple Java programming approach of Section 5.1.1.4 to get started.

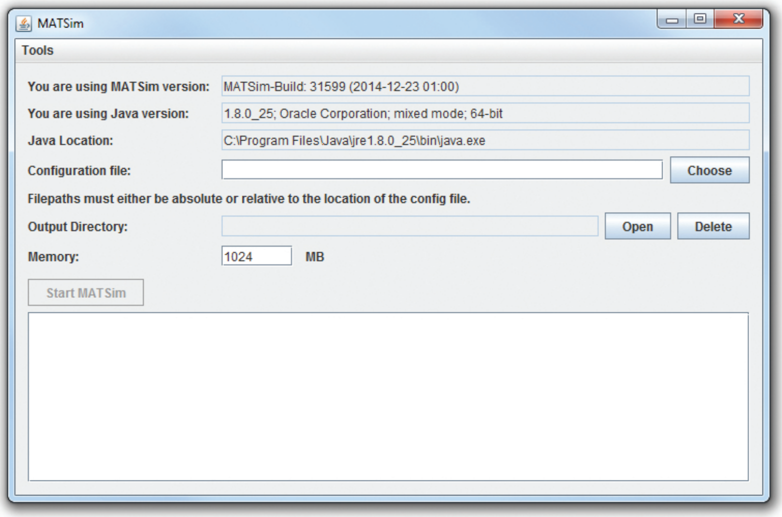
2.1.2 Running MATSim

When this book was written, only the nightly built MATSim JAR file could be started by double-clicking. A minimal GUI (Graphical User Interface), as shown in Figure 2.1, opens and the MATSim run can be configured and started. This feature will appear in the releases, starting with version 0.8.

For the release 0.7, MATSim does not provide a GUI; thus, you must be able to handle and access a command line tool. In Linux or Mac OS X, this is typically a Terminal application; in Microsoft Windows, the Power Shell or Command Prompt. At the command prompt type the following command in one line, but substitute the correct paths:

On Linux or Mac OS X, something like:

```
java -Xmx512m -cp /path/to/matsim.jar org.matsim.run.Controler /path/
   to/config.xml
```



You are using MATSim version: MATSim-Build: 31599 (2014-12-23 01:00)
You are using Java version: 1.8.0_25; Oracle Corporation; mixed mode; 64-bit
Java Location: C:\Program Files\Java\jre1.8.0_25\bin\java.exe
Configuration file:
Filepaths must either be absolute or relative to the location of the config file.
Output Directory:
Memory: 1024 MB

Figure 2.1: Minimal MATSim GUI.

On Windows, an example command could be:

```
java -Xmx512m -cp C:\MATSim\matsim.jar org.matsim.run.Controler
   C:\MATSim\input\config.xml
```

Such a command consists of multiple parts:

- `java` tells the system that you want to run Java.
- `-Xmx512m` tells Java that it should use up to 512 MB (Megabyte) of memory. This is typically enough to run the small examples. For larger scenarios, you might need more memory, e.g., `-Xmx3g` would allow Java to use up to 3 GB (Gigabyte) of RAM (Random Access Memory).
- `-cp /path/to/matsim.jar` tells Java where to find the MATSim code.
- `org.matsim.run.Controler` specifies which class (think of an "entry point") should be run. In most cases, the default MATSim `Controler` is the class you will need to run simulations.
- `/path/to/config.xml` tells MATSim which config file is to be used.

2.1.3 Configuring MATSim

MATSim is configured in the config file, building the connection between the user and MATSim and containing a settings list that influences how the simulation behaves.

All configuration parameters are simple pairs of a parameter `name` and a parameter `value`. The parameters are grouped into logical groups; one group has settings related to the `Controler`, like the number of iterations, or another group has settings for the mobsim, e.g., end time of the mobsim. As shown in Chapter 5, numerous MATSim modules can be added to MATSim and configured by specifying the respective configuration file section.

The list of available parameters and valid parameter values may vary from release to release. Although we try to keep this stable, software changes, mainly new features, may cause settings to change. For a list of all available settings available with the version you are working with, run the following command:

```
java -cp /path/to/matsim.jar org.matsim.run.CreateFullConfig fullConfig.xml
```

This command will create a new config file `fullConfig.xml`, containing all available parameters, along with their default values and often an explanatory comment, making it easy to see what settings are available. To use and modify specific settings, lines with their corresponding parameters can be copied to the config file, specific to the scenario to be simulated, and the parameter values can be modified in that file. See `http://matsim.org/javadoc` → main distribution → `CreateFullConfig` for more information.

A fairly minimal config file contains the following information:

```
<module name="network">
   <param name="inputNetworkFile" value="<path-to-network-file>" />
</module>

<module name="plans">
   <param name="inputPlansFile" value="<path-to-plans-file" />
</module>

<module name="controler">
   <param name="firstIteration" value="0" />
   <param name="lastIteration" value="0" />
</module>

<module name="planCalcScore" >
   <parameterset type="activityParams" >
```

```
      <param name="activityType" value="h" />
      <param name="typicalDuration" value="12:00:00" />
   </parameterset>
   <parameterset type="activityParams" >
      <param name="activityType" value="w" />
      <param name="typicalDuration" value="08:00:00" />
   </parameterset>
</module>
```

For a working example, see the MATSim directory tree (cf. 2.1.1) under `examples/tutorial/config/example1-config.xml`.

In the example, supply is provided by the network and demand by the plans file. Typical input data is described in Section 2.2.2. The specification that the first and last iteration are the same, means that no replanning of the demand is performed. What *is* executed is the mobsim (Figure 1.1), followed by each executed plan's performance scoring. To function, the scoring needs to know, from the config file, all activity types used in the plans and the typical duration for each activity type.

Further configuration possibilities are described in Chapter 4.

2.2 Building and Running a Basic Scenario

This section provides information on typical input data files used for a MATSim experiment, as well as the standard output files generated. It presents a minimal example scenario and briefly explains units, conventions and coordinate systems used in MATSim. Then, hints on practical data requirements are provided.

2.2.1 Units, Conventions, and Coordinate Systems

2.2.1.1 Units

MATSim tries to make few assumptions about actual units, but it is sometimes necessary for certain estimates. In general, MATSim expects similar types of variables (e.g., all distances) to be in the same unit wherever they are used. In the following short overview, the most important (expected) units are listed.

Distance Distance units are for example used in links' length. They should be specified in the same unit the coordinate system uses, allowing MATSim to calculate beeline distances. As the much used UTM (Universal Transverse Mercator) projected coordinate systems (see Section 2.2.1.3) use meters as the unit of distance, this is the most commonly used distance unit in MATSim.

Time MATSim supports an hour:minute:second notation in several places, but internally, it uses seconds as the default time unit. This implies, for example, that link speeds must be specified in distance per second, typically meters per second. One notable exception to this rule are scoring parameters, where MATSim expects values per hour.

Money Money is unit-free. Units are implicitly given by the marginal utility of money (cf. Equation (3.4) below). Thus, when one moves from Germany to Switzerland, the parameter β_c must be changed from "utility per Euro" to "utility per Swiss Franc".

2.2.1.2 Conventions

MATSim uses IDs intensely. These identifiers can be arbitrary strings, with the following exceptions: IDs should not contain any whitespace characters (incl. tabs, new lines, etc.) or commas, semicolons, etc., because those characters are typically used for separating different IDs from each other on IDs lists.

2.2.1.3 Coordinate Systems

Preparing Your Data in the Appropriate Coordinate System In several input files, you need to specify coordinates, e.g., for network nodes. We strongly advise not to use WGS84 coordinates (i.e., GPS (Global Positioning System) coordinates), or any other spherical coordinates (coordinates ranging from −180 to +180 in west-east direction and from −90 to +90 in south-north direction). MATSim has to calculate distances between two points in several sections of the code. Calculation of distances between spherical coordinates is very complex and potentially slow. Instead, MATSim uses the simple Pythagoras theorem, but this requires Cartesian coordinate system coordinates. Thus, we emphatically recommend using a Cartesian coordinate system along with MATSim, preferably one where the distance unit corresponds to one meter.

Many countries and regions have custom coordinate systems defined, optimized for local usage. It might be best to ask GIS (Geographic Information System) specialists in your region of interest for the most commonly used coordinate system there and use that for your data.

If you have no information about what coordinate system is used in your region, it might be best to use the UTM coordinate system. This system divides the world into multiple bands, each six degrees wide, and separated into a northern and southern part, which it calls UTM zones. For each zone, an optimized coordinate system is defined. Choose the UTM zone for your region (Wikipedia has a good map showing the zones) and use its coordinate system.

Telling MATSim About Your Coordinate System For some operations, MATSim must know the coordinate system where your data is located. For example, some analyses may create output to be visualized in Google Earth or by QGIS (Quantum GIS). The coordinate system used by your data can be specified in the config file:

```
<module name="global">
   <param name="coordinateSystem" value="EPSG:32608" />
</module>
```

This allows MATSim to work with your coordinates and convert them whenever needed.

You have multiple ways to specify the coordinate system you use. The easiest one is to use the so-called "EPSG (European Petroleum Survey Group) codes". Most of the commonly used coordinate systems have been standardized and numbered. The EPSG code identifies a coordinate system and can be directly used by MATSim. To find the correct EPSG code for your coordinate system (e.g., for one of the UTM zones), the website `http://www.spatialreference.org` is extremely useful. Search on this website for your coordinate system, e.g., for "WGS 84 / UTM Zone 8N" (for the northern-hemisphere UTM Zone 8), to find a list of matching coordinate systems along with their EPSG codes (in this case `EPSG:32608`).

As an alternative, MATSim can also parse the description of a coordinate system in the WKT (Well-Known Text) format.

2.2.2 Typical Input Data

Minimally, MATSim needs the files

- `config.xml`, containing the configuration options for MATSim and presented above in Section 2.1.3,
- `network.xml`, with the description of the (road) network, and
- `population.xml`, providing information about travel demand, i.e., list of agents and their day plans.

Thus, `population.xml` and `network.xml` might get quite large. To save space, MATSim supports reading and writing data in a compressed format. MATSim uses GZIP-compression for this. Thus, many file names have the additional suffix `.gz`, as in `population.xml.gz`. MATSim acknowledges whether files are compressed, or should be written compressed, based on file name.

2.2.2.1 An Outlook on Extending MATSim in Part II of this Book

Chapter 7 provides some information about MATSim's technical tools for initial input generation. With the basic setting, MATSim agents perform their activities on a specific link. If further information about activity locations needs to be specified, this can be carried out with facilities described in Section 6.4. Further, for the *simulation* of public transport, the base scenario must be extended by additional files as shown in Section 16.4.1 and Chapter 16. Count data are a common evaluation measure in transport planning. In MATSim, count data can be provided for the simulation, as shown in Section 6.3.

In more detail, the network and population files resemble the following; for the config file, see Section 2.1.3 above.

2.2.2.2 `network.xml`

Network is the infrastructure on which agents (or vehicles) can move around. The network consists of nodes and links (in graph theory, typically called vertices and edges). A simple network description in MATSim's XML (Extensible Markup Language) data format could contain approximately the following information:

```
<network name="example network">
   <nodes>
      <node id="1" x="0.0" y="0.0"/>
      <node id="2" x="1000.0" y="0.0"/>
      <node id="3" x="1000.0" y="1000.0"/>
   </nodes>
   <links>
      <link id="1" from="1" to="2" length="3000.00" capacity="3600"
            freespeed="27.78" permlanes="2" modes="car" />
      <link id="2" from="2" to="3" length="4000.00" capacity="1800"
            freespeed="27.78" permlanes="1" modes="car" />
      <link id="3" from="3" to="2" length="4000.00" capacity="1800"
            freespeed="27.78" permlanes="1" modes="car" />
      <link id="4" from="3" to="1" length="6000.00" capacity="3600"
            freespeed="27.78" permlanes="2" modes="car" />
   </links>
</network>
```

For a working example, check the `examples/equil` directory in the MATSim directory tree (cf. Section 2.1.1).

Each element has an identifier `id`. Nodes are described by an `x` and a `y` coordinate value (also see Sections 2.2.1.3 and 7.1). Links have more features; the `from` and `to` attributes reference nodes and describe network geometry. Additional attributes describe traffic-related link aspects:

- The `length` of the link, typically in meters (see Section 2.2.1).
- The flow `capacity` of the link, i.e., number of vehicles that traverse the link, typically in vehicles per hour.
- The `freespeed` is the maximum speed that vehicles are allowed to travel along the link, typically in meters per second.
- The number of lanes (`permlanes`) available in the direction specified by the 'from' and 'to' nodes.
- The list of `modes` allowed on the link. This is a comma-separated list, e.g., `modes="car, bike, taxi"`.

All links are uni-directional. If a road can be traveled in both directions, two links must be defined with alternating `to` and `from` attributes (see links with id 2 and 3 in the listing above).

2.2.2.3 `population.xml`

File Format MATSim travel demand is described by the agents' day plans. The full set of agents is also called the population, hence the file name `population.xml`. Alternatively, `plans.xml` is also commonly used in MATSim, as the population file essentially contains a list of day plans.

The population contains the data in a hierarchical structure, as shown in the following example. This example illustrates the data structure; minimal input files need less information, as illustrated later.

```
<population>
   <person id="1">
      <plan selected="yes" score="93.2987721">
         <act type="home" link="1" end_time="07:16:23" />
         <leg mode="car">
            <route type="links">1 2 3</route>
         </leg>
         <act type="work" link="3" end_time="17:38:34" />
         <leg mode="car">
            <route type="links">3 1</route>
         </leg>
         <act type="home" link="1" />
      </plan>
   </person>
   <person id="2">
      <plan selected="yes" score="144.39002">
         ...
      </plan>
   </person>
</population>
```

For a working example, check the `examples/equil` directory in the MATSim directory tree (cf. Section 2.1.1).

The population contains a list of persons, each person contains a list of plans, and each plan contains a list of activities and legs.

Exactly one plan per person is marked as selected. Each agent's selected plan is executed by the mobility simulation. During the replanning stage, a different plan might become selected. A plan can contain a score as attribute. The score is calculated and stored in the plan after its execution by the mobility simulation during the scoring stage.

The list of activities and legs in each plan describe each agent's planned actions. Activities are assigned a type and typically have—except for the last activity in a day plan—a defined end time. There are some exceptions where activities have a duration instead of an end time. Such activities are often automatically generated by routing algorithms and are not described in this book. To describe the location where an activity takes place, the activity is either assigned a coordinate by giving it an `x` and `y` attribute value, or it has a link assigned, describing from which link the activity can be reached. Because the simulation requires a link attribute, `Controler` calculates the nearest link for a given coordinate when the link attribute is missing.

A leg describes how an agent plans to travel from one location to the next; each leg must have a transport mode assigned. Optionally, legs may have an attribute, `trav_time`, describing the expected travel time for the leg. For a leg to be simulated, it must contain a route. The format of a route depends on the mode of a leg. For car legs, the route lists the links the agent has to traverse in the given order, while for transit legs, information about stop locations and expected transit services are stored. MATSim automatically computes initial routes for initial plans that do not contain them.

An agent starts a leg directly after the previous activity (or leg) has ended. The handling of the agent in the mobsim depends on the mode. By default, car and transit legs are well-supported by the mobsim. If the mobsim encounters a mode it does not know, it defaults to teleportation. In this case, an agent is removed from the simulated reality and re-inserted at its target location after the leg's expected travel time has passed.

A Minimal Population File The population data format is one of the most central data structures in MATSim and might appear a bit overwhelming at first. Luckily, to get started, it is only necessary to know a small subset. A population file needs, approximately, only the following information:

```
<population>
   <person id="1">
      <plan>
         <act type="home" x="5.0" y="8.0" end_time="08:00:00" />
         <leg mode="car" />
         <act type="work" x="1500.0" y="890.0" end_time="17:30:00" />
         <leg mode="car" />
         <act type="home" x="5.0" y="8.0" />
      </plan>
   </person>
   <person id="2">
      ...
   </person>
</population>
```

For a working example, check the `examples/equil` directory in the MATSim directory tree (cf. Section 2.1.1).

The following items can be used for simplification:

- Each person needs exactly one plan.
- The plan does not have to be selected or have a score.
- Activities can be located just by their coordinates.
- Activities should have a somewhat reasonable end-time.
- Legs need only a mode, no routes.

When a simulation is started, MATSim's `Controler` will load such a file and then automatically assign the link nearest to each activity and calculate a suitable route for each leg. This makes it easy to get started quickly.

2.2.3 Typical Output Data

MATSim creates output data that can be used to analyze results as well as to monitor the current simulation setup progress. Some of the files summarize a complete MATSim run, while others are created for a specific iteration only. The first type of files goes directly to the output folder's top level, which can be specified in the `controler` section of the config file. The other files are stored in iteration-specific folders `ITERS/it.{iteration number}`, which are continuously created in the output folder. For some files (typically for large ones, such as population), the output frequency can be specified in the config file. They then go only to the respective iteration folders. The files summarizing the complete MATSim run are built 'on the fly', i.e., after every iteration, currently computed iteration values are stored, allowing continuous monitoring of the run. Some files are created by default (such as the score statistics files); others need to be triggered by a respective configuration file section (such as count data files).

The following output files are continuously built up to summarize the complete run.

Log File: During a MATSim run, a log file is printed containing information you might need later for your analyses, or in case a run has crashed.

Warnings and Errors Log File: Sometimes, MATSim identifies problems in the simulation or its configuration; it will then write warning and error messages to the log file. Because the log file contains so much information, these warnings can be overlooked. For this reason, a separate log file is generated in the run output directory, containing only warnings and error messages. It is important to check this file during/after a run for possible problems.

Score Statistics: Score statistics are available as a picture (`scorestats.png`), as well as a text file (`scorestats.txt`). They show the average best, worst, executed and overall average of all agents' plans for every iteration. An example score plot is shown in Figure 1.2.

Leg Travel Distance Statistics: Leg travel distance statistics (files `traveldistancestats.png` and `traveldistancestats.txt`) are comparable to score statistics, but instead, they plot travel distance.

Stopwatch: The stopwatch file (`stopwatch.txt`) contains the computer time (so-called wall clock time) of actions like replanning or the execution of the mobsim for every iteration. This data is helpful for computational performance analyses, e.g., how long does replanning take compared to the mobility simulation?

The following output files are created for specific iterations:

Events: Every action in the simulation is recorded as a MATSim event, be it an activity start or change of network link; see Fig. 2.2. Each event possesses one or multiple attributes. By default, the time when the event occurred is included. Additionally, information like the ID of the agent triggering the event, or the link ID where the event occurred, could be included. The events file is an important base for post-analyses, like the visualizers. Events are discussed in detail in Section 45.2.5.

Plans: At configurable iterations, the current state of the population, with the agents' plans, is printed. The final iteration's plans are also generated on the top level of the output folder.

Leg Histogram: In every iteration, a leg histogram is plotted. A leg histogram depicts the number of agents arriving, departing or en route, per time unit. Histograms are created for each transport mode and for the sum of all transport modes. Each file starts with the iteration number and ends with the transport mode (e.g., `1.legHistogram_car.png` or `1.legHistogram_all.png`). A text file is also created (e.g., `1.legHistogram.txt`), containing the data for all transport modes.

Trip Durations: For each iteration, a trip durations text file (e.g., `1.tripdurations.txt`), listing number of trips and their durations, on a time bin level for each activity pair (e.g., from work to home or from home to shopping), is produced.

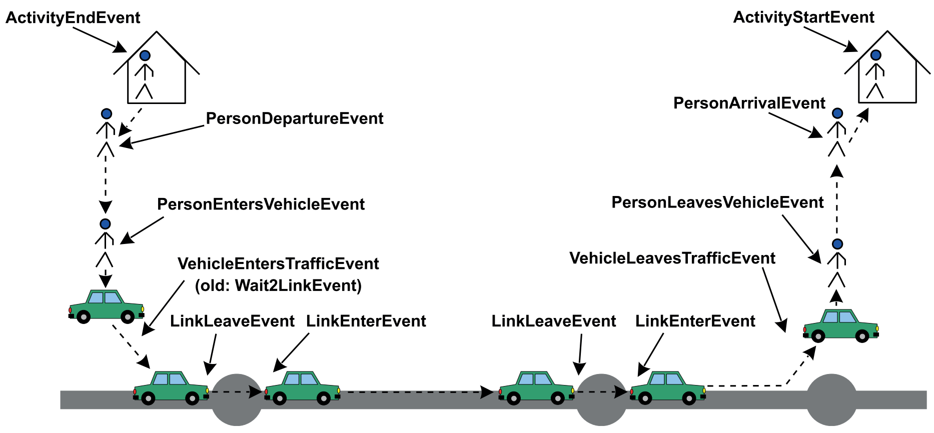


Figure 2.2: Mobsim events.

Link Stats: In each iteration, a link stats file containing hourly count values and travel times on every network link is printed. Link stats are particularly important for comparison with real-world count data, as introduced in Section 6.3.

2.2.4 An Example Scenario

The MATSim release is shipped with an example scenario named `equil` in the folder `examples/equil`, containing these files: `config.xml`, `network.xml`, `plans100.xml`, and `plans2000.xml.gz`, containing, respectively, 100 and 2000 persons with their day plans, using car mode only. A tiny population containing only 2 persons (`plans2.xml`), one using public transport, the other using car mode, is also provided. An example for count data is also found in the folder (`counts100.xml`).

In addition, there is also a file with 100 *trips* (`plans100trips.xml`), i.e., demand going only from one location to another, using a `dummy` activity type at each end. This is provided to show that MATSim can also be run as a fully trip-based approach, without considering any activities. Clearly, it loses some of its expressiveness, but the basic concepts, including route and even departure time adaptation, still work in exactly the same way.

The scenario network is shown in Figure 2.3.

The following lines explain the scenario by discussing the most important sections from the config file `config.xml`.

"strategy" section of the config file As shown in the config file excerpt below, this scenario uses replanning. 10 % of the agents reroute their current route (module `ReRoute`). The remaining 90 % select their highest score plan for re-execution in the current iteration (module `BestScore`). Plans are deleted from the agent's memory if it is full, defined by `maxAgentPlanMemorySize`. By default, the plan with the lowest score is removed; this is configurable and currently being researched (see Section 97.3).

```
<module name="strategy">
   <param name="maxAgentPlanMemorySize" value="5" />
      <!-- 0 means unlimited -->

   <parameterset type="strategysettings" >
      <param name="strategyName" value="ReRoute" />
      <param name="weight" value="0.1" />
   </parameterset>
```

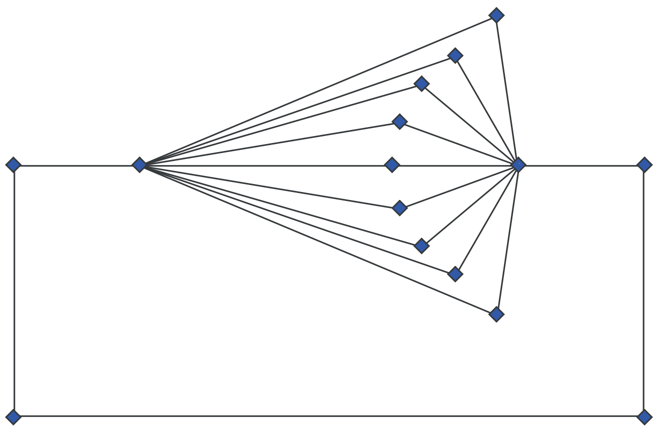


Figure 2.3: Equil scenario network.

```
   <parameterset type="strategysettings" >
      <param name="strategyName" value="BestScore" />
      <param name="weight" value="0.9" />
   </parameterset>

</module>
```

"planCalcScore" section of the config file The section `planCalcScore` defines parameters used for scoring, explained in Chapter 3. As seen in the example, two activity types, `h` (home) and `w` (work), are specified. All activity types contained in the population file (cf. Section 2.2.2.3) must be defined in the `planCalcScore` section of the config file.

```
<module name="planCalcScore" >
   <parameterset type="activityParams" >
      <param name="activityType" value="h" />
      <param name="typicalDuration" value="12:00:00" />
   </parameterset>
   <parameterset type="activityParams" >
      <param name="activityType" value="w" />
      <param name="typicalDuration" value="08:00:00" />
   </parameterset>
</module>
```

"controler" section of the config file The scenario is run for 10 iterations, writes the output files to `./output/equil` (Section 2.2.3) and uses QSim as the mobsim (more on mobsims in Section 1.3, 4.3 and 11).

```
<module name="controler">
   <param name="outputDirectory" value="./output/equil" />
   <param name="lastIteration" value="10" />
   <param name="mobsim" value="qsim" />
</module>
```

Visualization Simulation results can be visualized with Via (Chapter 33) or OTFVis (On The Fly Visualizer) (Chapter 34).

2.2.5 Data Requirements

2.2.5.1 Population and Activity Schedules

Demand estimation is an important component of MATSim. That means that, in theory, only demand components that do *not* change from one simulated average working day to the next need to be provided to MATSim. Examples are: population and its residential and working locations. In practice, however, MATSim is not yet prepared to endogenously model complete travel demand. Sequence and preferred durations of activities, for example, must be provided as input. As a result, all travel demand choices not covered by the MATSim loop have to be exogenously estimated.

For population generation, two possibilities exist: the comfortable way is to translate a full population census and the slightly more demanding way is to generate a synthetic population (e.g., Guo and Bhat, 2007), based on sample or structure surveys. For MATSim, both methods have been used based on e.g., Swiss Federal Statistical Office (BFS) (2000) and Müller (2011a).

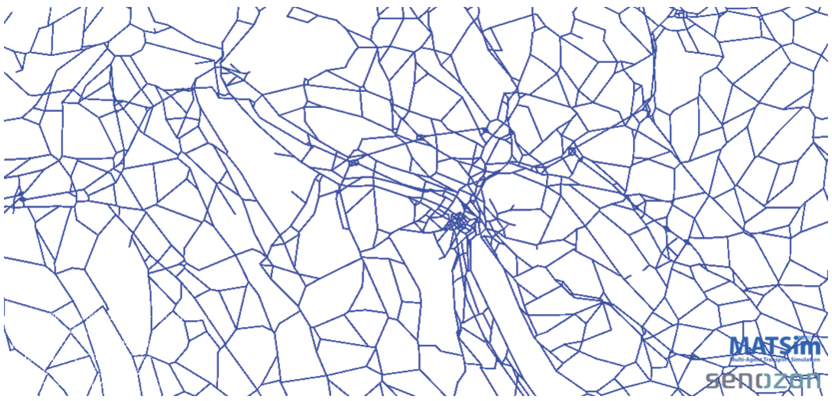
Travel demand is usually derived from surveys: for Switzerland, from the microcensus (Swiss Federal Statistical Office (BFS), 2006). Newer data sources, such as GPS or smartphone travel diaries, are currently being investigated (e.g., Zilske and Nagel, 2015).

A critical topic in demand and population generation is workplace assignment, as commuting traffic is still a major issue, particularly during peak hours. Switzerland's full census work location was surveyed at municipality level. Such comfortable data bases are rare, however.

Having generated the residential population of the study area, additional demand components might be necessary, for example, cross-border and freight traffic. As these components often cannot be endogenously modeled, MATSim offers the feature to handle different subpopulations differently (Section 4.5). One can specify that border-crossing agents, for example, are not allowed to make destination choices within the study area, or that freight agents are not allowed to change their delivery activity to a leisure activity.

2.2.5.2 Network

In simulation practice, two different network types are used: planning networks and navigation networks (compare Swiss examples in Figure 2.4(a) and Figure 2.4(b) for the Zürich region). The former are leaner and often serve as initial explorative simulation runs, while the latter are often used for policy runs, usually offering far more details, such as bike and even pedestrian links. Data are available from official sources like federal offices, free sources, such as OSM (OpenStreetMap), and commercial sources, including navigation network providers.


(a) Planning network.

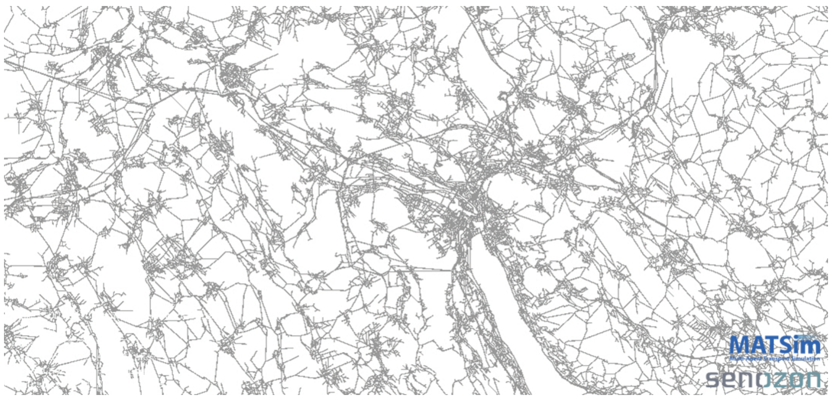

(b) Navigation network..

Figure 2.4: Zürich networks

2.2.6 *Example Scenario Input Data*

Some example scenarios are included in the MATSim main distribution, in the directory "examples".

More pre-packaged scenarios can be found under `http://www.matsim.org/datasets`.

2.3 MATSim Survival Guide

There are many options and possibilities available with MATSim, and finding them can be a daunting exercise. Here are a couple of recommendations, derived from our own frequent use of the system.

1. *Always start with and test a small example.*
2. *Always test large scenarios with one percent runs first (e.g., a randomly drawn subsample of your initial demand).* The MATSim GUI (Figure 2.1) allows creating sample populations with the command `Tools...Create Sample Population`.
   As described in Section 4.3, this requires adaptation of parameters, in particular, the mobsim's `flowCapacityFactor` and `storageCapacityFactor` factors. As shown in Part II, Section 6.3, sample scenarios also require parameter adaption for count data comparisons.
3. *If your set-up does not work any more,* immediately *go back to a working version and proceed from there in small steps.*
4. *Check* `logfileWarningErrors.log`.
5. *Check the comments that are attached to the config file options.*
   One finds them in the file `output_config.xml.gz`, or near the beginning of `logfile.log`.
6. *Try setting as few config file options as possible.*
   This has two advantages: (i) Except for the deliberately set options, your simulation will move along with changed MATSim defaults, and thus with what the community currently considers the best configuration. (ii) You will not be affected by changes in the config file syntax as long as they are different from your own settings.
7. *Search for documentation via* `http://matsim.org/javadoc`.
8. *Search for the latest tutorial via* `http://matsim.org/docs`.