CHAPTER 7

# Generation of the Initial MATSim Input

Marcel Rieser, Kai Nagel and Andreas Horni

As explained in Section 2.2, the minimal MATSim input, besides the configuration, consists of the network and population with initial plans. For illustrative scenarios, all three can be generated with a text editor. For more complicated and/or realistic scenarios, they need to be generated by other methods. People with knowledge in a scripting language may use that scripting language to generate the necessary XML files, possibly honoring the MATSim DTDs. We ourselves use Java as our scripting language for these purposes. Java is not necessarily the best choice here; this may be discussed elsewhere. We do use it, for the following reasons:

- Most of us also program MATSim extensions and these currently have to be in Java. Thus, using Java as a scripting language for initial input generation saves us the effort of becoming proficient in another programming language.
- The MATSim software, by necessity, already contains all file readers and writers for MATSim input, saving the effort of re-implementing them and one automatically moves forward with file version updates. Additionally, one can directly use the MATSim data containers.
- Once one starts writing MATSim scripts-in-Java (Section 5.1.1.4), in many situations, it makes sense to modify the input data after reading the files. The programming techniques for this are the same as for other initial input generation.

Part IV will show how initial input was generated on a practical level—discussing, e.g., the different types of original input data—for different scenarios. This section presents MATSim's technical tools for initial input generation.

**How to cite this book chapter:**
Rieser, M, Nagel, K and Horni, A. 2016. Generation of the Initial MATSim Input. In: Horni, A, Nagel, K and Axhausen, K W. (eds.) *The Multi-Agent Transport Simulation MATSim*, Pp. 61–64. London: Ubiquity Press. DOI: http://dx.doi.org/10.5334/baw.7. 

## 7.1 Coordinate Transformations in Java

Section 2.2.1.3 has given information about coordinate systems. When programming in Java and MATSim for input data generation, coordinate transformations derived from geotools (Geotools, accessed 2015) can be used. For example,

```
CoordinateTransformation ct =
  TransformationFactory.getCoordinateTransformation("WGS84", "WGS84_UTM33N");
```

would transform data given in WGS84 coordinates to data in UTM coordinates.

## 7.2 Network Generation

### *7.2.1 From OpenStreetMap*

A fairly standardized way to generate a MATSim network is from OSM data. The process (roughly) goes as follows:

1. Download the necessary `xxx.osm.pbf` file from `http://download.geofabrik.de/osm`.
2. Download a recent Osmosis build from `http://wiki.openstreetmap.org/wiki/Osmosis`.
3. The necessary command to extract the road network (approximately) is:

   ```
   java -cp osmosis.jar --rb file=xxx.osm.pbf \
      --bounding-box top=47.701 left=8.346 bottom=47.146 right=9.019 \
      completeWays=true --used-node --wb allroads.osm.pbf
   ```

   The bounding box can, e.g., be obtained from `http://www.osm.org`; it is in WGS84 coordinates.
4. It makes good sense to add the large roads of a much larger region. The necessary command (approximately) is

   ```
   java -cp osmosis.jar --rb file=xxx.osm.pbf --tf accept-ways \
      highway=motorway,motorway_link,trunk,trunk_link,primary,primary_link \
      --used-node --wb bigroads.osm.pbf
   ```

5. The two files are merged with (approximately) the following command:

   ```
   java -cp osmosis.jar --rb file=bigroads.osm.pbf --rb allroads.osm.pbf \
      --merge --wx merged-network.osm
   ```

An example script of how to convert the resulting `merged-network.osm` file into a MATSim network file can be found under `http://matsim.org/javadoc` → main distribution → `RunPNetworkGenerator` class.

### *7.2.2 From Other Sources*

Networks can also be obtained from other sources. An example script of how to convert an EMME network to MATSim can be found under `http://matsim.org/javadoc` → main distribution → `RunNetworkEmme2MatsimExample` class. A problem with EMME network files is that they use user-defined variables in non-standardized ways, meaning that each converter has to be adapted to the specific situation.

Material to read VISUM files can be found by searching for the string "visum" in the code base, but is currently not systematically maintained.

## 7.3 Initial Demand Generation

### *7.3.1 Simple Initial Demand*

A simple script to generate a population with a single synthetic person with one initial plan can be found under `http://matsim.org/javadoc` → main distribution → `RunPOnePersonPopulationGenerator`. A somewhat larger synthetic population is generated by `RunPPopulationGenerator`.

Note that coordinates in the population need to be consistent with coordinates in the network. Roughly speaking, coordinates mentioned in the population file need to be in the same range as coordinates mentioned in the network. Note that, in the examples presented here, coordinates of the network generated in Section 7.2.1 are *not* consistent with the demand generated by the `RunP*`-scripts; these need to be adapted accordingly.

### *7.3.2 Realistic Initial Demand*

A script to illustrate the generation of a more realistic population and initial demand can be found under `http://matsim.org/javadoc` → main distribution → `RunZPopulationGenerator`, generating a sample population from a census file and writing it to a file.

Here, network coordinates generated in Section 7.2.1 *are* consistent with demand generated by the `RunZ*`-script.