CHAPTER 4

More About Configuring MATSim

Andreas Horni and Kai Nagel

This chapter describes configuration options that can be used together with the three basic elements: config file, population and network. Part II discusses various options to extend MATSim beyond these three elements, sometimes using only additional files, or using additional JAR files beyond the MATSim core JAR file, by writing "scripts in Java" or by adding or replacing functionality.

MATSim writes configuration files in several locations; for example, in the logfile, in the iteration output directory, or with the `CreateFullConfig` functionality described in Section 2.1.3. As explained in Section 2.3, these files come with comments explaining configuration options. This is often the best source for configuration options.

4.1 MATSim Data Containers

4.1.1 Network

The config file section `network` specifies which network file will be used in the simulation (Section 2.1.3 and 2.2.2.2). Further configuration options, e.g., specification of time-variant networks, are presented in Section 6.1.

4.1.2 Population

The config file section `plans` specifies which population file with its day plans will be used (Section 2.1.3 and 2.2.2.3). Further configuration options, e.g., specification of arbitrary agent attributes or subpopulations, are presented in Section 6.2.

Further MATSim containers are described in Chapter 6.

How to cite this book chapter:
Horni, A and Nagel, K. 2016. More About Configuring MATSim. In: Horni, A, Nagel, K and Axhausen, K W. (eds.) *The Multi-Agent Transport Simulation MATSim*, Pp. 35–44. London: Ubiquity Press. DOI: http://dx.doi.org/10.5334/baw.4. License: CC-BY 4.0

4.2 Global Modules and Global Aspects

4.2.1 Controler

The controler is an indispensable module for running MATSim; its parameters are set in the `controler` config file section. The MATSim run's output directory, its number of iterations and the plans and events output interval can be specified here. The expected mobsim can be defined (Section 4.3). The routing algorithm is defined here by using

```
<module name="controler" >
   <param name="routingAlgorithmType" value="{Dijkstra
   | FastDijkstra | AStarLandmarks | FastAStarLandmarks}" />
    ...
</module>
```

Possibilities for extending the `Controler` functionality are given in Chapter 45.

4.2.2 Events

Events are continuously generated, reporting on all activities in the mobsim, as discussed in more detail in Section 45.2.5.

Please note that, besides these mobsim events, there is a less prominent type of events, namely `ControlerEvents`, which are created by the `Controler` to report on its current state. `ControlerEvents` are also further explained in Section 45.2.5.

4.2.3 Parallel Computing

MATSim uses multi-threading to accelerate computing speeds. Related configuration parameters can be found in several config modules; they are combined into one section here.

Global Setting The `global` section contains

```
<module name="global" >
   <param name="numberOfThreads" value="2" />
   ...
</module>
```

This number is used in several places; most importantly, innovative strategies, where multiple routing requests are distributed to multiple threads.
A good starting point is using the number of available cores.

Parallel Event Handling The config file section `parallelEventHandling` is used to define the number of threads used for event handling. As described in Waraich et al. (2009), the simulation can be substantially accelerated when using multiple threads for the events handling, which can be a bottleneck in MATSim simulation runs.

Parallel QSim The number of threads for the parallel QSim (cf. Dobler (2013)) can be configured by

```
<module name="qsim" >
   <param name="numberOfThreads" value="10" />
   ...
</module>
```

General Recommendations Generally, computations using threads are not necessarily faster with more threads, which is also true for MATSim. Some experimentation is necessary for each combination of scenario and hardware. Here are some recommendations:

- For the "global" number of threads, a good starting point is the number of available cores.
- It is no longer possible to switch off parallel event handling completely; setting it to '0' or 'null' or '1' eventually achieves the same result. Setting it to values larger than one sometimes leads to performance gains, but they are rarely significant.
- The most sensitive parameter is that for the QSim. For somewhat older hardware (e.g., Apple Macbook Pro from 2010), using all three remaining cores—in addition to the parallel event handling—led to negligible performance gains but left the machine useless for interactive tasks such as normal office work. For new hardware (e.g., Apple Macbook Pro from 2014), using six of the available eight cores for the QSim can make the mobsim more than a factor of two faster and the machine can still be used for office tasks. Experiences with older servers show that one must carefully investigate the number of threads for the mobsim, since using more threads often slows it down (Dobler, 2013). No experiences with new servers are currently available.
- HPCC (High-Performance Computing Clusters) are often available to researchers, allowing access to high-quality machines with reduced management overhead. Typically, one pays for computation time, either directly, or by a loss of priority, with an amount proportional to the reserved resources, that is, the time the job took to finish, multiplied by the number of reserved cores. In this kind of situation, the number of cores used throughout the whole process should be stable to avoid paying for unused resources. A recommendation in this case is thus to set the number of threads for the QSim to the best value (see above), say n, parallel events handling to 1, the "global" number of threads to $n + 1$, and submit the job requesting $n + 1$ cores. Also note that fewer threads are almost always better in terms of throughput. In addition, for both calibration and "what-if" scenario exploration, one typically needs to run a large number of simulations with different parameters or input data. As total RAM memory is usually not an issue on a cluster, it is often more efficient to run a large number of simulations simultaneously with a low number of threads, rather than a low number of simulations with lots of threads.

4.2.4 Global

In the config file section `global`, the simulation's random seed, the "global" number of Java threads (see Section 4.2.3) and the coordinate system (cf. Section 2.2.1) can be defined. Note that no matter if you explicitly define the random seed or not, MATSim always starts from a fixed random seed, which is either the one you define, or an internal constant. That is, if you start the same version of MATSim twice from the same config file, you will get the same sequence of random numbers, and thus exactly the same simulation. If you want to change this behavior, you need to change the random seed explicitly.

4.3 Mobility Simulations

An overview of MATSim mobility simulations is given by Dobler and Axhausen (2011).

4.3.1 QSim

The queue-based and time-step based QSim (Gawron, 1998; Simon et al., 1999; Cetin et al., 2003; Dobler and Axhausen, 2011; Dobler, 2010) is MATSim's default mobsim. Its parameters are set in the `qsim` config file section. Important parameters are: By specifying

```
<param name="numberOfThreads" value="..."/>
```

QSim can be run in parallel, see Section 4.2.3. Importantly, the `qsim` parameters

```
<param name="flowCapacityFactor" value="..." />
<param name="storageCapacityFactor" value="..." />
```

need to be set accordingly when running sample scenarios. For example, for a 10 % sample, these factors need to be 0.1.

Currently, QSim is implemented as a single-queue model (see Chapter 50). Back-propagating gaps as discussed in Section 1.3 are available experimentally (see Section 97.5) and configurable with the parameter

```
<param name="trafficDynamics" value="..." />
```

As shown in Section 4.6.1, QSim can handle multimodal scenarios.

A somewhat ancient configuration parameter is the stuck time. It determines after how many seconds of non-movement a vehicle is moved across an intersection despite violating the storage constraint of the destination link. This parameter was introduced to resolve grid-locks, i.e., geometrical arrangements where no vehicle can move any more. With the QSim model, it is possible to add vehicles beyond the storage constraint to an overcrowded link. This corresponds to maintaining a minimal flow even under very congested conditions. The default value of this parameter is set to 10, i.e., non-moving vehicles are moved forward after 10 simulation time steps of non-movement. This may seem a rather short time, but systematic investigations (unfortunately never published) have shown that the simulations become, in comparison to traffic counts data, less realistic when this parameter is increased.

4.3.2 JDEQSim

JDEQSim (Waraich et al., 2009) was used for project *KTI Frequencies* (Balmer et al., 2010). It is is a Java reimplementation of DEQSim (Waraich et al., 2009; Charypar et al., 2007b, 2009) and provides parallel event handling, but no parallel simulation (Balmer et al., 2010, p.11). Back-propagating gaps (Section 1.3) are supported, but traffic lights, public transport and within-day replanning are not.

To run JDEQSim, the parameter `mobsim` of `controler` config file section must be set to `JDEQSim` and a `jdeqsim` config file section must be provided.

4.4 Scoring

The config file section `planCalcScore` specifies the parameters used for scoring agents' plans (Section 2.1.3); parameters are explained in Chapter 3.

4.5 Replanning Strategies

Replanning strategies are the basic innovation modules available in MATSim. We do not call them *choice* modules, although they are involved in people's choice making. The choice process is performed over the iterations with an *implicit* choice set and is not based on explicit probability function drawing. One can differentiate between modules that affect the set of plans that each agent holds, and others that only select between these plans. For a detailed discussion of MATSim in choice modeling context, see Chapter 49.

All strategy modules are called by configuring the strategy module in the configuration file as shown in the following example.

```
<module name="strategy" >
   <parameterset type="strategysettings" >
      <param name="strategyName" value="ChangeLegMode" />
      <param name="weight" value="0.1" />
   </parameterset>
   <parameterset type="strategysettings" >
      <param name="strategyName" value="TimeAllocationMutator"/>
      <param name="weight" value="0.2" />
   </parameterset>
   <parameterset type="strategysettings" >
      <param name="strategyName" value="SelectExpBeta" />
      <param name="weight" value="0.7" />
   </parameterset>
</module>
```

Each module is given a weight determining the probability, by which the course of action represented by the module is taken. The strategy modules' weights are normalized, in case they do not sum to one. In this example, each agent changes her leg mode with probability 0.1 and her plan timing with probability 0.2. Otherwise, the agent chooses a plan from her set of plans according to a logit model.

By specifying the parameter `subpopulation`, replanning strategies can be applied to distinct subpopulations: e.g.,

```
<parameterset type="strategysettings" >
   <param name="strategyName" value="ChangeLegMode" />
   <param name="weight" value="0.1" />
   <param name="subpopulation" value="urbanTravelers"/>
</parameterset>
```

In older versions of the config file, you will find a deprecated configuration syntax using numbered strategy modules.

Please note that combining strategy modules that are extensions (see Section 5.1.1), like destination innovation together with public transport, may not always work as expected. Combine them with care and contact the mailing list if you are unsure.

4.5.1 Plans Generation and Removal (Choice Set Generation)

4.5.1.1 Time Innovation

Time innovation is applied by defining its parameters in the config file section `TimeAllocationMutator` and by adding

```
<param name="strategyName" value="TimeAllocationMutator" />
```

plus its weight to the strategy modules.

The module shifts activity end times randomly within a configurable range as described by Balmer et al. (2005b); Raney (2005).

4.5.1.2 Route Innovation

Route innovation is applied by adding

```
<param name="strategyName" value="ReRoute" />
```

plus its weight to the strategy modules, and by specifying the routing algorithm in the `controler` config file section (Section 4.2.1). MATSim routing is described by Lefebvre and Balmer (2007).

4.5.1.3 *Mode Innovation*

Mode innovation is applied by adding[1]

```
<param name="strategyName"
   value="{ChangeLegMode | ChangeSingleLegMode |
   SubtourModeChoice}" />
```

plus its weight to the strategy modules. In the config file, a section with one of the mode innovation strategies needs to be added, i.e.,

```
<module name="{changeLegMode | changeSingleLegMode |
subtourModeChoice}" >
   ...
</module>
```

`ChangeLegMode` randomly picks one of a person's plans and changes the mode of transport. By default, the supported modes are: driving a car and using public transport. Only one mode of transport per plan is supported. When using different modes for sub-tours on a single day, the `SubtourModeChoice` module is required. Optionally, car availability is respected. `ChangeSingleLegMode` randomly picks one of a person's plans and changes one single leg's (picked randomly) mode of transport. In contrast to `ChangeLegMode`, it allows for multiple modes in one plan. By default, supported modes are: driving a car and using public transport. Also, this module can (optionally) respect car availability.

Mode innovation is described by Rieser et al. (2009); Meister et al. (2010); Ciari et al. (2008, 2007).

4.5.1.4 *Plans Removal*

The maximum number of plans per agent is configured by the setting

```
<module name="strategy" >
   <param name="maxAgentPlanMemorySize" value="5" />
   ...
</module>
```

If an agent ends up having more plans, MATSim will start removing plans, one by one, until the maximum number of plans is reached. Plans to be removed are selected by the setting configured by

```
<module name="strategy" >
   <param name="planSelectorForRemoval" value="..." />
   ...
</module>
```

Starting with release 0.8.x, the config file comments give possible options.

This option is not yet well investigated, cf. Section 97.3. Per default, the plan with the lowest score is removed if the agent's memory is full.

4.5.2 *Plan Selection (Choice)*

Selectors and their weight are also added to the strategy modules

```
<param name="strategyName" value="KeepLastSelected | BestScore |
SelectExpBeta ChangeExpBeta | SelectRandom | SelectPathSizeLogit" />
```

[1] The names may be changed into `ChangeTripMode` and `ChangeSingleTripMode`, please keep your eyes open.

Selectors work as follows:

- `KeepLastSelected` keeps the plan selected in the previous iteration.
- `BestScore` selects the plan with the highest score from the previous iteration.
- `SelectExpBeta` performs MNL (Multinomial Logit Model) selection between plans. It can be configured by the `BrainExpBeta` parameter from the scoring config group[2] being the scale parameter in discrete choice models, as shown in Equation 49.2. We recommend keeping this parameter at its default value of 1.0.
- `ChangeExpBeta` changes to a different plan, with probability dependent on $e^{\Delta_{score}}$, where Δ_{score} is the score difference between the two plans. This will also sample from an MNL (see Sec. 47.3.2.1).
- `SelectRandom` performs random selection between the plans.
- `SelectPathSizeLogit` selects an existing plan according to the path size logit described by Frejinger and Bierlaire (2007). It can be configured by the `PathSizeLogitBeta` parameter from the scoring config group.[3] This selector has never been investigated systematically.

Note that the `BestScore` should be used with care; it tends to get stuck with sub-optimal plans. Plans badly rated due to a random fluctuation in one single iteration, e.g., a rare traffic jam, will never be tested again. Thus, we recommend using this only in conjunction with `SelectRandom`.

4.5.3 Innovation Switch-Off

For theoretical (Section 47.3.2.3) reasons, it makes sense to eventually switch off the innovative modules, thus keeping the set of plans for each agent fixed from then on. This behavior can be configured by

```
<param name="fractionOfIterationsToDisableInnovation" value="..."/>
```

It makes sense to use this together with MSA averaging of the scores (Section 3.3.4).

4.6 Other Modes than Car

The MATSim software began with the car mode of transport, since it was then the main mode in many regions. The idea of integrating other modes has always been a theme.

The following sections describe current MATSim multi-modal capabilities. The material covers not only options that can be enabled with just config options, but also gives an overview of multi-modal extensions, described in Part II of the book.

4.6.1 QSim Side

4.6.1.1 Multiple Vehicular Modes on the Same Network

The approach described so far fails as soon as more than one vehicle type is involved. Therefore, recently the ability to allow multiple modes on the same network was introduced. It is defined by the `qsim` config option of type

```
<module name="qsim">
   <param name="mainMode" value="car,truck,bicycle" />
   ...
</module>
```

[2] This is in the scoring config group for historical reasons.
[3] Also in the scoring config group for historical reasons.

This examines the plan leg mode; if that leg mode corresponds to one of the listed main modes, it will generate a vehicle for that leg and make it enter the network.

It is currently not possible to generate different vehicle types from the config alone; one either needs to provide a vehicles file (see Section 6.6 and Section 11.1), or write a script-in-Java to generate the vehicle fleet (again see Section 11.1).

4.6.1.2 *So-Called Teleportation*

All modes *not* registered with the QSim as "main modes" will be teleported. That is, the QSim will, without problems, process legs such as

```
<leg mode="pedelec" >
   <route type="generic" trav_time="00:14:44" distance="2374" />
</leg>
```

The QSim will generate a departure event (for events, see Section 2.2.3) after the end of the previous activity and an arrival event 14 minutes and 44 seconds later. The leg will be recorded with a distance of 2 374 meters. If distance is not used for scoring (cf. Chapter 3), it can also be left out of the route (the situation in most set-ups).

4.6.1.3 *Explicitly Simulated Passenger Modes*

With "driver" modes, such as car, bicycle, or also walk, travelers are also drivers, i.e., the entities making decisions about turns at intersections, as well as arrivals (or not) on links. With "passenger" modes, such as public transit or taxi, this changes; for example, the traveler boards a bus, the bus moves around in the network; the only decision the traveler has to make if she or he wants to get off or not at the current stop. The bus, in turn, is a normal participant in the corresponding traffic system, i.e., buses and taxis operate on the normal road network and can be caught in the same congestion as cars and trucks. This is exactly how it works in the MATSim QSim; taxis typically operate on the same network as cars; pt vehicles may operate on the same network if their routes are defined so that they use the same links as regular cars. In these cases, their interactions are captured by the simulation.

4.6.1.4 *Departure Handlers*

It is possible to register a separate departure handler for each mode; see Section 45.2.8 for the syntax. There are also pre-configured extensions using this approach:

- The "multimodal" contribution moves all registered modes on separate, congestion-free networks. This is better than teleportation, since the vehicles (or pedestrians) have defined positions at each point in time, meaning that they can also re-plan, e.g., re-route (see Chapter 21).
- The public transport extension moves all registered modes with specific public transit vehicles (see Chapter 16).
- The dynamic transport systems contribution will eventually be able to move a `taxicab` mode with taxis (see Chapter 23).

4.6.2 *Routing Side*

The previous Section 4.6.1 has described how the QSim handles various modes when they are requested by the plans. Correspondingly, it now needs to be considered how non-car plans, or more specifically non-car routes inside non-car legs, are generated.

4.6.2.1 *Network Modes*

The following syntax defines modes for which the router should generate network routes, i.e., routes that contain a sequence of links to follow:

```
<module name="planscalcroute" >
   <param name="networkModes" value="car, truck" />
   ...
</module>
```

The above configuration specifies that plans containing

```
<leg mode="car"...>
```

as well as

```
<leg mode="truck"...>
```

will be treated by the network router.

As of the writing of this text, the router will route all these modes on the “car” links of the network. This means that, say, denominating some links as “car only” or “truck only” will not be picked up by the current router.[4]

Note that, per the network file DTD (Document Type Description), “car” is the default mode of each link as long as long as the link’s mode field is not explicitly filled.

4.6.2.2 *Teleportation ...*

... with Teleported Mode Free Speed Factor A config entry such as

```
<module name="planscalcroute" >
   <parameterset type="teleportedModeParameters" >
      <param name="mode" value="pt" />
      <param name="teleportedModeFreespeedFactor" value="2.0" />
      <param name="teleportedModeSpeed" value="null" />
      <param name="beelineDistanceFactor" value="null" />
   </parameterset>
   ...
</module>
```

means that if the router encounters a leg with mode pt, it generates a “teleportation” route whose travel distance is the same as, and travel time is twice that of, a freespeed car route.

This models public transit, assuming it travels along roughly the same routes as a car trip, but takes twice as long (cf. Reinhold, 2006).

... with Teleported Mode Speed Setting, in the above, something like

```
   <param name="teleportedModeFreespeedFactor" value="null" />
   <param name="teleportedModeSpeed" value="4.167" />
   <param name="beelineDistanceFactor" value="1.3" />
```

will, instead, generate a teleportation route whose travel distance is 1.3 times the beeline distance, and whose travel time is that distance divided by 4.167 meters per second.

This is useful when teleported mode travel times should not change in tandem with car freespeed travel times, perhaps as a policy change result, or when teleported mode travel times are unrelated

[4] Check `https://matsim.atlassian.net/browse/MATSIM-330` for developments.

to car travel times. One disadvantage: this approach does not take obstacles like water or mountain areas, into account for the teleported modes.

4.6.2.3 Other Routing Options

It is possible to register separate routers for specific modes. This syntax is discussed in Section 45.2.7. The pre-configured extensions and contributions discussed in Section 4.6.1.4, "multimodal", public transport, taxis, come with corresponding routers.

In addition, the so-called "matrix based pt router" (Chapter 20) uses a list of transit stops and a matrix of stop-to-stop travel times and travel distances; based on this information, it computes a teleported walk leg to the next stop, another to the destination stop, and a last teleported walk leg to the final destination.

The matrix-based pt router also illustrates that, given the QSim teleportation capability, it is possible to come up with arbitrary algorithms for arbitrary modes, as long as they generate (expected) travel times and (expected) travel distances. As said earlier, the teleportation facility of the QSim will just use these two attributes at face value. Although with such an approach neither congestion nor en-route replanning are or can be included, it is flexible and allows a fully modular addition of arbitrary modes without having to interact with the QSim.

4.6.3 Scoring Side

For all modes mentioned in the plans, a corresponding scoring section must exist. See Section 3.2.1 for an example.

4.7 Observational Modules

4.7.1 Travel Time Calculator

The routing module, for example, needs travel time estimations for all network links. To keep computational effort feasible, travel time estimations need to be aggregated to time bins. Parameters of this aggregation, such as bin size, can be specified in the configuration file section `travelTimeCalculator`.

4.7.2 Link Stats

The `linkStats` config file section can specify the output interval of individual links' simulation statistics. It is configurable if the simulated volumes are written per iteration or averaged over multiple iterations. As one of their many functions, link stats are used for comparison with count values, as introduced in Section 6.3.