MATSim User's Guide

Generated 2013-03-15T13:50:39+01:00
from matsim.org/docs/userguide

Contents

User's Guide

The "tutorial" section contains "reduced" information about how to find your way into matsim.

This "user's guide" section contains additional information, concentrating on features and details that are not explained in the tutorials. Clearly, there may be overlap.

Features

The following list shows the key features of MATSim:

Fast Dynamic and Agent-Based Traffic Simulation
In many cases, MATSim only takes a couple of minutes for a single simulation of a complete day of traffic. This includes the completely time-dynamic simulation of motorized individual traffic as well as the handling of agents using other modes of transport.

Supports Large Scenarios
MATSim is able to simulate scenarios with several millions agents or network with hundreds of thousands of streeets. All you need is a current, fast desktop computer with plenty of memory. Additionally, MATSim allows you to only simulate a certain percentage of the traffic, speeding up the simulation even more while reducing memory consumption, and still generate useful results.

Sophisticated Interactive Visualizer
Forget aggregated results! MATSim provides a fast Visualizer that can display the location of each agent in the simulation and what it is currently doing. It can even connect to a running simulation, allowing interactively querying agents' states, visualizing agents' routes or perform live analyses of the network state.

Versatile Analyses and Simulation Output
During the simulation, MATSim collects several key values from the simulation and outputs them to give you a quick overview of the current state of the simulation. Among other results, it can compare the simulated traffic to real world data from counting stations, displaying the results interactively in Google Earth. Additionally, MATSim provides detailed output from the traffic microsimulation, which can easily be parsed by other applications to create your own special analyses.

Modular Approach
MATSim allows for easy replacement or addition of functionality. This allows you to add your own algorithms for agent-behavior and plug them into MATSim, or use your own transport simulation while using MATSim's replanning features.

Open Source
MATSim is distributed under the Gnu Public License (GPL), which means that MATSim can be downloaded and used free of charge. Additionally, you get the complete Source Code which you may modify within certain constraints (see the license for more details). Written in Java, MATSim runs on all major operating systems, including Linux, Windows and Mac OS X.

Active Development and Versatile Usage of MATSim
Researchers from several locations are currently working on MATSim. Core development takes place at the Berlin Institute of Technology (TU Berlin), the Swiss Federal Institute of Technology (ETH) in Zurich, as well as in a start-up founded by two former PhD students. Additional development (as far as we are aware of) currently takes place in South Africa, Germany (Munich, Karlsruhe) as well as other places around the world. This distribution of development ensures that MATSim not only works for one scenario/context, but can be adapted to many different scenarios.

System Requirements

Software

MATSim runs on any machine that has the Java Platform, Standard Edition (SE) 6 or newer installed (commonly referred to as "Java 6" or newer).

Hardware

Smaller scenarios (e.g. the examples included in the tutorials, 5%- or 10%-samples of large scenarios) can be run on common desktop or laptop computers.

To simulate large scenarios (several hundreds of thousands of agents, networks with ten-thousands of links and nodes), high end computers with a large amount of memory (RAM) may be required to keep the agents' data in memory. The description of agents' plans and the simulation output can take several Gigabytes of hard disk space. To store the data for several scenarios and / or output of simulation runs, large amounts of disk space may thus be needed. MATSim can read and write compressed files to reduce the amount of required disk space, but this aspect still shouldn't be underestimated. MATSim can make use of multiple CPUs or CPU cores that share common memory ("shared memory machine") during the replanning-phase.

Running large scenarios for a high number of iterations can take several hours, up to a few days. Thus it may be advisable to have a dedicated machine running MATSim if you plan to simulate many different scenarios.

Recommendations

- To try MATSim out:
  Any modern laptop or desktop computer with 1GB RAM and 500MB free disk space should be suitable.
- To run a large scenario (100 000+ agents, networks with 50 000+ links):
  A high-end desktop computer with at least 4GB RAM and 200 GB free disk space.
- To run many large scenarios, so they can be compared against each other:
  Multiple high-end desktop computers or servers with at least 4GB RAM that share a common storage disk (at least 1TB).

The high numbers for free disk space result from the fact that the simulation writes quite a lot of data to the disk during a run. For analysis, usually only the last version of the data is required, and data from earlier iterations can be deleted, freeing space up again.

What we use

Currently, we simulate most of our scenarios on machines with 8 or 16 GB RAM, having 2 dual-core processors. The amount of memory allows us to run 2 scenarios at the same time on the machines. A RAID array is used as storage backend, offering about 4 TB of hard disk space. This huge disk space is able to store the results of hundreds of simulations and will suit us for the next few years. Computers and RAID are regular components used in data centers, usually available at moderate prices.

Terminology

In many cases, MATSim uses a terminology that is different from the mainstream terminology. In most cases, the reason is that the concepts are only similar, but not identic, ~~and we wanted to avoid the confusion of using the same term for aspects that are similar but not identical. The following attempts some commented approximate "translations" from more standard teminology to MATSim terminology.~~

They are now on separate web pages per issue in order to facilitate comments. kai, dec'10

Choice set --> "plan set" of an agent

Choice set --> "plan set" of an agent

Comments: During MATSim iterations, ~~agent~~ accumulate plans. This can be interpreted as building a choice set over time. ~~A problem is that the process that generates the choice set at this point is not systematic.~~

Possible future developments: Once it has been made explicit that "plans generation" means "choice set generation", the terminology may be made standard.

Choice set generation --> Time mutation/re-route/... ; "innovation"

Choice set generation --> Time mutation/re-route/... ; "innovation"

Comments: As said above, the set of MATSim plans can be seen as this agent's choice set. MATSim generates new plans "on-the-fly", i.e. while the simulation is running. We sometimes call this "innovation", since agents create new plans (= add entries to the choice set), rather than choosing between existing plans.

Choice set generation, choice --> plan selection, replanning

choice set generation, choice --> plan selection, replanning

It seems as if replanning is only used in Matsim for "generating a new plan alternative" but not for "selecting an existing alternative". It took me some time and some misunderstandings with Yu to figure this out. It appears more plausible to me to phrase it as "replanning = {choice, choice set generation/update}". Gunnar

The above needs to be discussed; in my understanding replanning indeed includes choice. Kai

Convergence --> learning rate

Convergence --> learning rate

Scores in matsim are computed as score_new = (1-alpha) * score_old + alpha * score_sim, where score_sim is the score that is obtained from the execution of the plans (= network loading).

Mu (logit model scaling factor) --> beta_brain

mu (logit model scaling factor) --> beta_brain

Matsim scoring function = beta_brain * \sum_i beta_i * attribute_i

Typical formulation = \mu * \sum_i ...

For estimation, \mu and the \beta_i are not independently identifiable. For simulation, they are hence somewhat arbitrary. Since this is arbitrary anyway, beta_brain should be at least set to one and not to two. The re-labeling into "mu" would not only be more consistent with common language, it would also avoid the "brain"-notion, which I find irritating. Gunnar

I agree. "beta_brain" and "2" are there for historical reasons, and api changes are always difficult. Kai

Multinomial logit --> ExpBetaPlanSelector

Multinomial logit --> ExpBetaPlanSelector

Comments:

- The main problem is that one needs to keep in mind how the choice set is constructed (see above).
- In most simulations, we use ExpBetaPlanChanger instead, which is a Metropolis Monte Carlo variant of making multinomial logit draws

Possible future developments: None of this is ideal, since, after the introduction of a policy, it is not clear which behavioral switches are due to the policy, and which are due to sampling. In theory, one should have unbiased samples before and after the introduction of the policy, but at this point this is not implemented and it is also computationally considerably more expensive than what is done now.

Network loading --> mobsim, mobility simulation, physical simulation

network loading --> mobsim, mobility simulation, physical simulation

Comments: The standard terminology has the "network loading" on the "supply side". In my (KN's) view, the "simulation of the physical system" is not the supply side, but what in economics is called "technology". This can for example be seen in the fact that "lane changing" is part of the mobsim, but this is, in my view, not a "supply side" aspect.

Possible future developments: May switch to "network loading" if there is agreement that this is a better name.

Stationary --> relaxed

stationary --> relaxed

Comments: "stationary" means that the probability distribution does not shift any more. However, as long as "innovation" is still switched in on MATSim (new routes, new times, ...), the result is not truly stationary. Thus we avoid the word. If innovation is switched off, the result is indeed a ~~statinary~~ process, but limited to the set of plans that every agent has at that point in time.

Possible future developments: not clear. Minimally, publications should be precise.

Configuration:

```
<module name="strategy" >
        <!-- iteration after which module will be disabled.  most useful
for ``innovative'' strategies (new routes, new times, ...) -->
        <param name="ModuleDisableAfterIteration_1" value="null" />
        <param name="ModuleDisableAfterIteration_2" value="950" />

        <!-- probability that a strategy is applied to a given person.
despite its name, this really is a ``weight'' -->
        <param name="ModuleProbability_1" value="0.9" />
        <param name="ModuleProbability_2" value="0.1" />

        <!-- name of strategy (if not full class name, resolved in
StrategyManagerConfigLoader) -->
        <param name="Module_1" value="ChangeExpBeta" />
        <param name="Module_2" value="ReRoute" />

        <!-- maximum number of plans per agent.  ``0'' means ``infinity''.
Currently (2010), ``5'' is a good number -->
        <param name="maxAgentPlanMemorySize" value="4" />
</module>
```

The above means:

- StrategyModule "ReRoute" (= innovative Module, produces plans with new routes) is switched off after iteration 950.
- StrategyModule "ChangeExpBeta" (= non-innovative Module, switches between existing plans) is never switched off.
- If an agent ever ends up with more than 4 plans, plans are deleted until ~~she~~ is back to 4 plans. (Deletion goes via a "PlanSelectorForRemoval", which affects the choice set,

and thus more thought needs to go into this. Currently, the plan with the worst score is removed.)

Utility <--> score

utility <--> score

At least when using random utility models (such as multinomial logit aka ExpBeta...), the score has the same function as the deterministic utility.

Building new scenarios

Starting a new scenario (our term for the application of MATSim to a region/area) can appear quite cumbersome at the first ~~glace,~~ as a lot of data preparation may be required. In any case required for a new scenario are:

- description of the **network**
- description of the travel **demand** (synthetic **population**)

Given these two data items, you can already start building your own scenario. The "Learning MATSim in 3 days"-Tutorial gives you an introduction on how to build your own scenario.

Import from VISUM

See here for javadoc, and here for code.

Programming

In many cases, using pre-configured software is not possible because there are just too many possibilities of how input could look like. Although matsim is not there yet, these should be api-only use cases, i.e. they should only use the "stable" api. Therefore, the following are under the api-users section of the documentation:

- Additional information about network generation is here.
- additional information about initial demand generation is here.

Information concerning specific scenarios

The following ~~sections contains~~ various information for building a scenario that presumably goes beyond what is in the tutorial.

Coordinate Systems in MATSim

For some operations, MATSim must know about the coordinate system your data is in. For example, ~~If~~ you want to generate kml-Output for Counts-Validation, MATSim has to convert the coordinates in your network to WGS84, the coordinate system used by Google Earth.

Specifying the Coordinate System used

You can specify the coordinate system in the config-file:

```
<module name="global">
  <param name="coordinateSystem" value="CH1903LV1903" />
</module>
```

The value specified for the coordinateSystem parameter can be:

- The short-name of a coordinate system known to MATSim. We define names for coordinate systems we use regularly in our work. These names are currently defined in TransformationFactory. The short-name `Atlantis` stands for an artificial coordinate system which maps our examples without relation to the real world somewhere in to the Atlantic ocean.
- Well-Known-Text (WKT) description of a coordinate system as they are supported by Geotools. This variant is not very readable, but allows one to experiment also in regions where MATSim does not provide a short-name for. Examples of WKT can be found in the MATSim-class MGC (in the `transformations` map).

Notes about Coordinate Systems

As the distance calculation in WGS84-coordinates (or any spherical coordinates) is rather complex (a simple Pythagoras is not enough), we advise people to use a Cartesian coordinate systems, preferable where one unit corresponds to one meter. Using such a coordinate system is a pre-requisit if one wants to use the optimized A*Landmarks-Router in MATSim.

Using MATSim for Switzerland

General remarks

This is material that was in the tutorial "Learning MATSim in 3 days". We moved it to ~~here.~~ but did not check the content.

A. Horni: Shortly, there will be a working paper describing the usage of MATSim for Switzerland., 02.05.2011

Some data sources

- Microcensus (BfS): www.bfs.admin.ch/bfs/portal/de/index/themen/11/07/01/02/01.html
- Census (BfS): www.bfs.admin.ch/bfs/portal/en/index/infothek/erhebungen__quellen/blank/blank/vz/uebersicht.html
- ASTRA traffic counts: www.astra.admin.ch/verkehrsdaten/00299/00303/index.html
- Business census (BfS): www.bfs.admin.ch/bfs/portal/en/index/infothek/erhebungen__quellen/blank/blank/bz/01.html
- Border crossing traffic (IVT, BfS): www.bfs.admin.ch/bfs/portal/de/index/themen/11/07/04/blank/01/01.html

Using MATSim from Urbansim (for the PSRC region)

Deprecated (and soon to be removed). See here instead.

~~Check out the opus source tree from www.urbansim.org . (The source tree is the tree containing directories such as opus_core or opus_gui.)~~

~~In this source tree, there should be a directory opus_matsim .~~

~~There is documentation in opus_matsim/docs .~~

Config(uration) options

This section describes possible settings of the MATSim configuration file. The section is probably incomplete since documentation always lags behind the actual code; please also look at the config dumps including their internal comments in the log files.

Strategy Modules

These are modules that can be used via the syntax

```
<module name="strategy" >
        <param name="ModuleProbability_1" value="0.1" />
        <param name="Module_1" value="ChangeLegMode" />
         <param name="ModuleProbability_2" value="0.1" />
         <param name="Module_2" value="TimeAllocationMutator" />
</module>
```

Strategy modules are numbered. Also, each module is given a weight which determines the probability by which the course of action represented by the module is taken. In this example, each person stands a chance of 1/2 that their transport mode is changed, and a chance of 1/2 that their time allocation is changed. (The weights are renormalized so that they add up to one.)

A strategy module is, in the code, always a combination of a plan selector and zero or more strategy module elements. There are two cases, which are handled differently:

- If there are zero strategy module elements, the chosen plan is made "selected" for the person, and the method returns.
- If there is at least one strategy module element, the chosen plan is copied, that copy is added to the persons's set of plan, and the new plan is made "selected". That new plan is then given to the strategy module elements for modification. These latter strategy

modules, with at least one strategy module element, are sometimes called "innovative".

The strategy modules that are understood by MATSim are defined in the class StrategyManagerConfigLoader. In addition, you can program your own strategy modules; see tutorial.programming in matsim/src/main/java for examples.

Unfortunately, the naming in the code is different from the naming in the config file:

- "strategy" in config file --> StrategyManager (or "set of strategies") in code
- "strategy module" in config file --> PlanStrategy in code
- There is a PlanStrategyModule in the code; it corresponds to what was called strategy module element in the description above.

It is not clear which combinations of these modules can be used together. Depending on required features, special variants sometimes need to be used. This has not yet been sorted out. Also see here.

BestScore. Status: works

Pure plan selecting (i.e. non-innovative) strategy module.

Will select the plan with the highest score. The score will be updated after execution of the mobsim.

Disadvantage: Will never try again plans that obtained a bad score from a fluctuation (e.g. a rare traffic jam). It is therefore recommended to either use this in conjunction with a small probability for RandomPlanSelector, or to use ChangeExpBeta.

ChangeExpBeta. Status: works. RECOMMENDED!

KeepLastSelected. Status: works

SelectExpBeta. Status: works

Multinomial logit model choice between plans.

The scores are taken as utilities; the betaBrain parameter from the config file is taken as the scale parameter. As equation:

```
p_i = exp( beta_brain * score_i) / sum_j exp( beta_brain * score_j )
```

SelectRandom

Divider: Pure plan selectors above, innovative strategy modules below.

ReRoute. Status: nearly indispensable

Maintainer: Marcel Rieser

All routes of a plan are recomputed.

The module is called by inserting the following lines into the "strategy" module:

```
<module name="strategy" >
        <param name="ModuleProbability_XXX" value="0.1" />
        <param name="Module_XXX" value="ReRoute" />
                ...
</module>
```

The corresponding configuration module unfortunately has a different name:

```
<module name="planscalcroute" >
        <param name="beelineDistanceFactor" value="1.3" />
        <param name="bikeSpeed" value="4.166666666666667" />
        <param name="ptSpeedFactor" value="2.0" />
        <param name="undefinedModeSpeed" value="13.88888888888889" />
        <param name="walkSpeed" value="0.8333333333333333" />
</module>
```

This works pretty reliably for car.

It also works for other modes, as "pseudo"-mode, in the following way:

- Travel times for these other modes are not obtained from true routing on the corresponding network, but by some estimates. These are configured by the parameters above, but ~~no guarantee~~ that they work consistently.
- The mobsim will not execute such routes on the network, but "teleport" them.
- The scoring works quite normally, since it just takes the time from leg start to leg end by mode.

It is possible to route such legs on the network, by using a different router.

It is *not* possible to "physically" execute a leg in the mobsim if it has not been routed before. That is, the capability of the router needs to be >= the capability of the mobsim. (Makes sense, if one thinks about it.)

TimeAllocationMutator. Status: works for vsp and ivt

Simple module that shifts activity end times randomly. ("Good" time shifts will be selected through the matsim plans selection mechanism.)

The maximum extent of the shifts can be configured; see the config section of the log file. It is, as of now (may'10), not possible to add a comment to that parameter.

The usage of the module is configured in the "strategy" section.

ChangeSingleLegMode. Status: works

Maintainer: Marcel Rieser

This replanning module randomly picks one of the plans of a person and changes the mode of transport of **one single leg**. The leg is picked randomly. For changing the mode of transport for all legs use ChangeLegMode. In contrast to ChangeLegMode, ChangeSingleLegMode allows for multiple modes in one plan. By default, the supported modes are driving a car and using public transport. Also, this module is able to (optionally) respect car-availability.

Note that the configuration is done by <module name="changeLegMode"> and not by <module name="changeSingleLegMode">. The replanning module is configured like this using the very same configuration module as ChangeLegMode:

```
<module name="changeLegMode">
   <param name="modes" value="car,pt,bike,walk" />
   <param name="ignoreCarAvailability" value="false" />
</module>
```

Add the module to the replanning strategy like this:

```
<param name="Module_X" value="ChangeSingleLegMode" />
<param name="ModuleProbability_X" value="0.1" />
```

Replace the 'X' with the number you assign to this module. For some more details on the syntax of this section, see here.

By default, the simulation will handle legs with modes different from "car" by using a delayed teleportation. If another behavior is requested (e.g. detailed simulation of public transport), this needs to be manually configured for the simulation.

ChangeLegMode. Status: works

Maintainer: Michael Zilske

This replanning module randomly picks one of the plans of a person and changes its 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. For using different modes for sub-tours on a single day see the "SubtourModeChoice" module. Also, this module is able to (optionally) respect car-availability.

The replanning module is configured like this, where the value parameter lists the modes of transport from which the module randomly chooses:

```
<module name="changeLegMode">
   <param name="modes" value="car,pt,bike,walk" />
   <param name="ignoreCarAvailability" value="false" />
</module>
```

Add the module to the replanning strategy like this:

```
<param name="Module_X" value="ChangeLegMode" />
<param name="ModuleProbability_X" value="0.1" />
```

Replace the 'X' with the number you assign to this module. For some more details on the syntax of this section, see here.

By default, the simulation will handle legs with modes different from "car" by using a delayed teleportation. If another behavior is requested (e.g. detailed simulation of public transport), this needs to be manually configured for the simulation.

Reference

M. Rieser, D. Grether, K. Nagel; **Adding mode choice to a multi-agent transport simulation**; TRB'09

LocationChoice. Status: ready

Maintenance and Questions

A. Horni, IVT (horni_at_IVT.baug.ethz.ch)

Javadoc

www.matsim.org/javadoc/org/matsim/locationchoice/package-summary.html

Config Parameters

www.matsim.org/javadoc/org/matsim/locationchoice/package-summary.html#locationchoice_parameters

Status

Ready

In Brief

MATSim provides destination choice based on three different basic concepts. First, random search can be applied. Second, local search implemented in the time geography framework is available [1]. Third, the most recent module ~~ist~~ best response and includes random error terms to make MATSim fully compatible with discrete choice theory [2]. The authors recommend to use this recent module in general. Random search should be utilized for algorithmic comparative investigations only.

The time geography module provides the possibility to take into account spatial competition in the activity location infrastructure (see Figure 3). It is planned-after a thorough calibration- to integrate spatial competition in the best response version.

Estimation of a MATSim destination choice utility function for Switzerland is in development [3].

Calling the Location Choice Strategy

The strategy module in the config file needs to be extended as follows:

```
<module name="strategy">
    ...
    <param name="ModuleProbability_X" value="0.0 < double <=1.0" />
    <param name="Module_X" value="LocationChoice" />
    ...
</module>
```

I. Random Search

Due to slow convergence, this approach is only useful for very small scenarios.

II. Local Search With Time Geography

The MATSim local search destination choice module is based on Hägerstrand's time geography. That is, in every replanning step locations are chosen within the region restrained by travel time budgets as defined by the time allocation module (see Figure 1 and Figure 2). Within this region the choice is performed based on the MATSim utility function.

In more detail, the following procedure is iteratively applied. An approximate choice set of locations is built to begin with, where the constructing of this set is initially based on an initial global travel speed assumption (*recursionTravelSpeed*). After tentatively choosing one location from this approximate set, the actual accessibility in terms of travel time is checked. If the location is not accessible it is rejected, the initial travel time is adapted according to the *recursionTravelSpeedChange* parameter in the configuration file and a next trial is started. After a certain number of failed trials to find an accessible location (*maxRecursions*), the choice is made from the universal choice set.

The parameters *recursionTravelSpeed*, *recursionTravelSpeedChange* and *maxRecursions* are explained in Section Parameters

III. Best Response Including Random Error Terms

The parameters *scaleEpsShopping* and *scaleEpsLeisure* correspond to $f_{Shopping}$ and $f_{Leisure}$ in [2]. *probChoiceSetSize* is Φ. epsilonDistribution, *tt_approximationLevel*, *maxDistanceEpsilon*, and *probChoiceExponent* are explained in in Section Parameters.

The parameters *scaleEpsShopping* and *scaleEpsLeisure* can be calibrated, based on e.g., travel distance distributions as described in [2].

NOTE: This variant will NOT work as described in Ref. [2] when configuring it as described above. Additionally, the scoring function needs to be modified. As of now, there does not seem to be a way to achieve this without some Java programming. kai, jan'13

Spatial Competition: Facility Load Penalty Computation

Similar to route and time choice being influenced by the competition in transport infrastructure it can be expected that competition in activities infrastructure has an effect on destination choice. Consequently, the utility of performing an activity is dependend on the actual load of the activities infrastructure at least for some activities such as e.g., grocery shopping (e.g.; searching for a parking space or waiting time at cash points etc.). In MATSim spatial competition is taken into account, which has shown to reduce the number of implausibly overcrowded locations (see Figure 3 below).

The score for perfoming an activity is calculated as follows:

score = (1- fp) * score_without_penalty

fp = Max(0.5, fcrf)

fcrf = restraintFcnFactor * [(facility load) / (facility capacity)] ^ *restraintFcnExp*

The parameters *restraintFcnFactor* and *restraintFcnExp* are explained in the section Parameters

Literature

[1] Horni, A., D.M. Scott, M. Balmer and K.W. Axhausen (2009) Location choice modeling for shopping and leisure activities with MATSim: Combining micro-simulation and time geography, *Transportation Research Record*, **2135**, 87-95.

[2] Horni, A., K. Nagel and K.W. Axhausen (2011) High-Resolution Destination Choice in Agent-Based Demand Models, *Arbeitsberichte Verkehrs- und Raumplanung*, **682**, IVT, ETH Zürich, Zürich.

[3] Horni, A., D. Charypar and K.W. Axhausen (2011) Empirically approaching destination choice set formation, paper presented at the *90th Annual Meeting of the Transportation Research Board*, Washington, D.C., January 2011.

Figures

Figure 1:

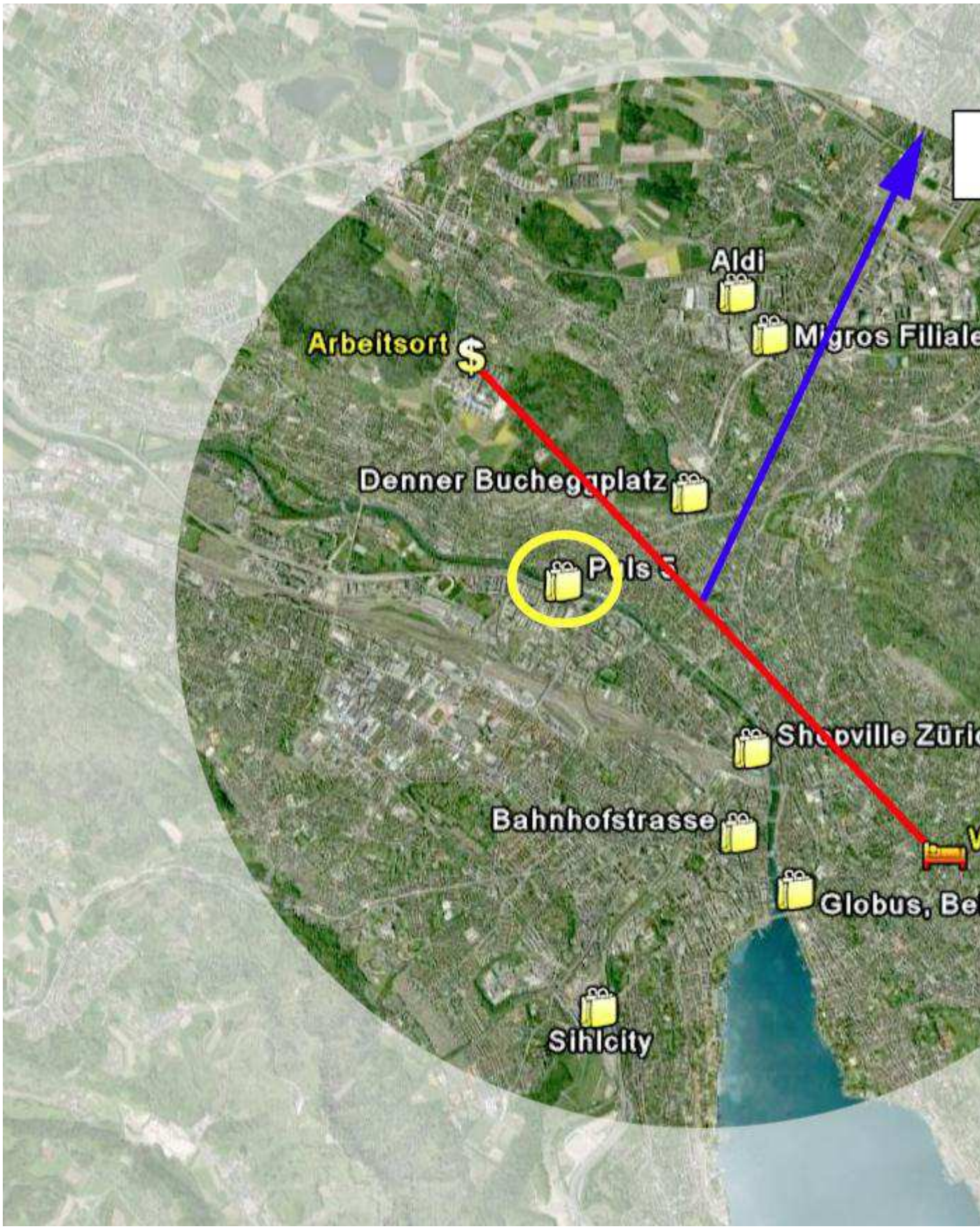

Aldi
Arbeitsort
Denner Buchegg
Bahnhofstrasse
Sihlcity

Figure 2:

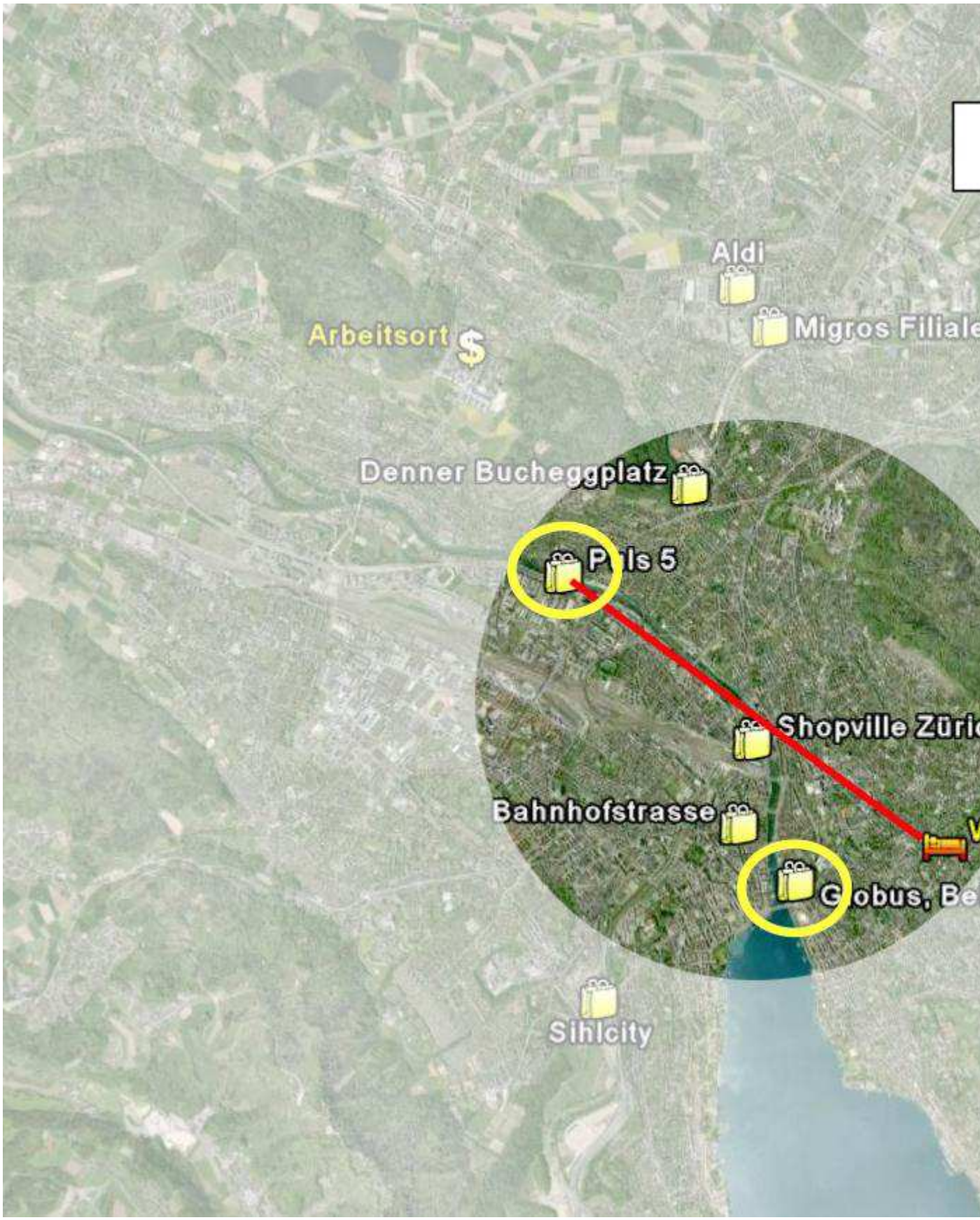
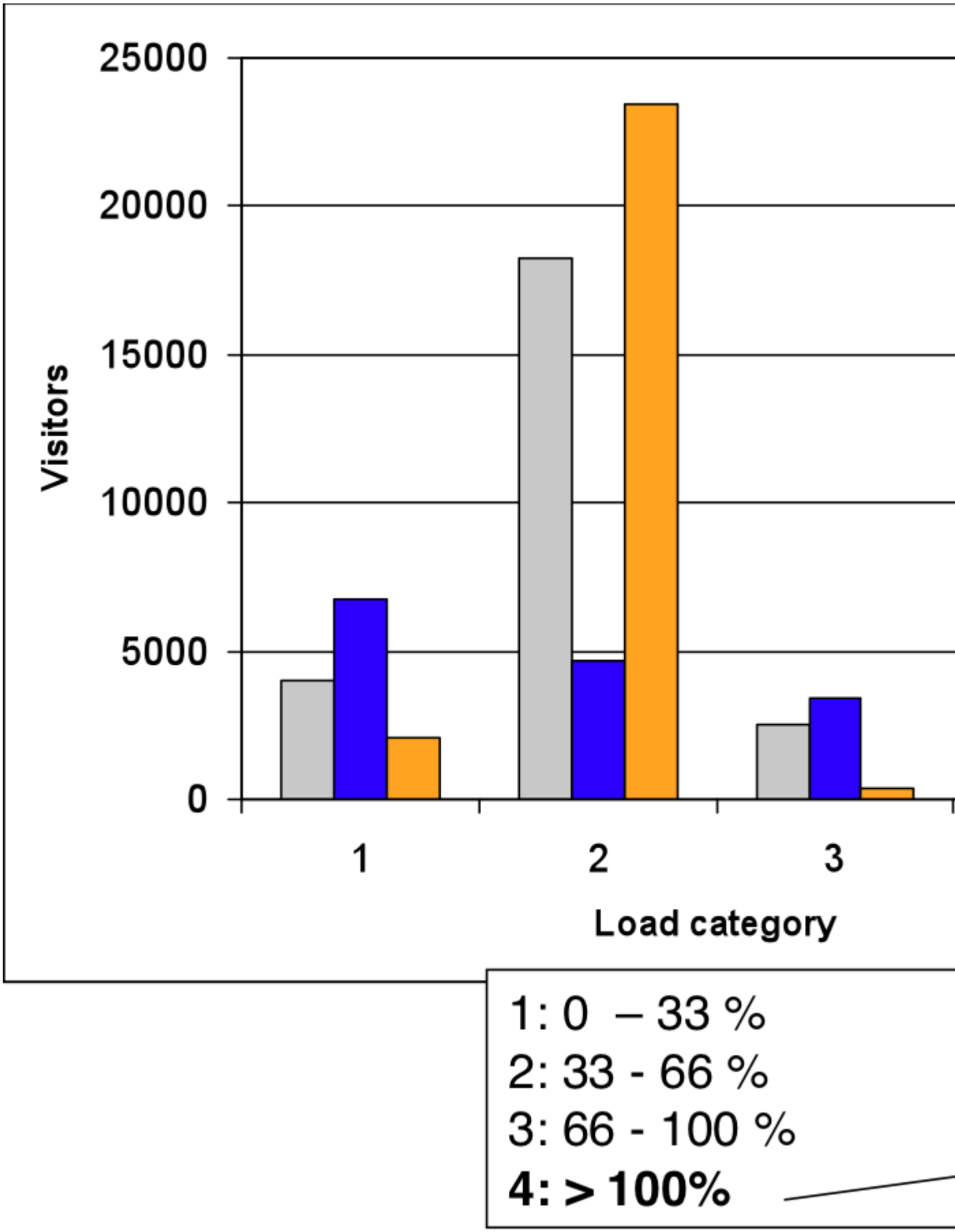

Aldi
Migros Filiale
Arbeitsort
Denner Bucheggplatz
Shopville Züri
Bahnhofstrasse
Sihlcity

Figure 3:


25000
20000
15000
10000
5000
0
Visitors
1
2
3
Load category
1: 0 – 33 %
2: 33 - 66 %
3: 66 - 100 %
4: > 100%

Planomat. Status: works for ivt

Maintenance

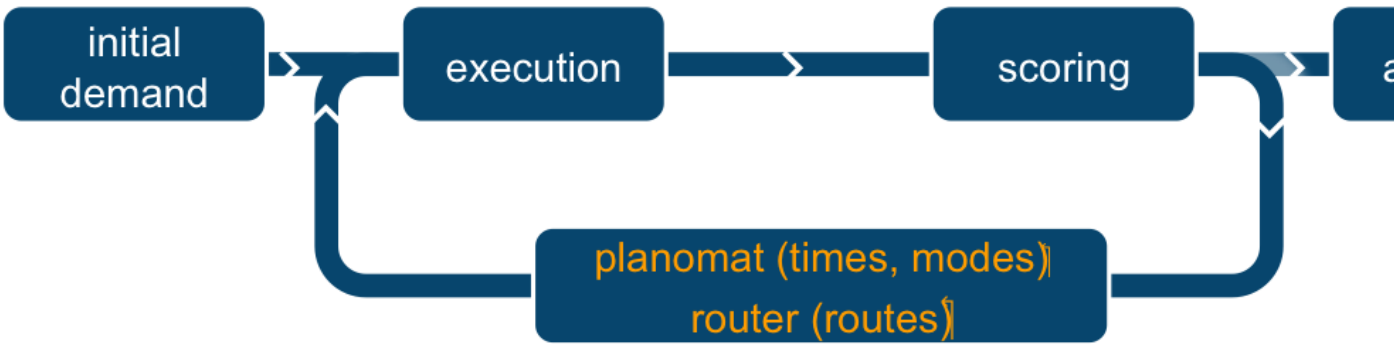
For questions, please contact:

Konrad Meister
IVT, ETH Zurich
email: meister (at) ivt.baug.ethz.ch

Description

The planomat is a stochastic best-reply replanning module for the activity durations and the departure times, as well as the mode choice on subtour level. On the one hand, this module replaces a simple random mutation module for departure times (the Time allocation mutator) with a genetic algorithm (GA) for daily plan optimization. This leads to a convergence of the MATSim learning framework within several dozens of iterations, independent of the departure times and activity durations in the initial demand (Meister et al., 2006). The choice of a GA makes it possible to include more plan variables into the optimization procedure. Currently it is possible to modify the mode of each subtour of the activity plan is varied besides the time information (unpublished, experiments documented in Meister, 2008). The mode choice set for each subtour may include modes whose travel times/costs can be estimated from events (typically "car") or directly from a routing algorithm, when events are not available, or the transport mode is not dynamic (typically "pt", "bike", "walk").

The optimal timing of the daily activity plan depends mainly on the travel times which the agents expects for a given leg. Just like in the router, and for reasons of simplicity, a travel time approximation based on global knowledge is assumed. The travel time will be estimated for the route that is given in the activity plan, which remains unchanged. For the “virtual” assessment of other modes of a given leg, the free-speed route for the respective mode is determined. The actual travel time on this route may still be time-dependent, as for the mode "car" in the application presented here.



The planomat replanning module was used in a large-scale study of agent-based travel demand optimization (Balmer et al., 2010).

Usage in MATSim

In order to use the planomat as a replanning module, activate it in the *strategy* section of the config xml file:

```
<module name="strategy">
  ...
  <param name="ModuleProbability_n" value="0.1" />
  <param name="Module_n" value="Planomat" />
  ...
</module>
```

- Replace `n` with the consecutive number of replanning/selection modules in the *strategy* section.

Without further settings, only time information (departure times, activity durations) will be modified and optimized. This is the default behavior, leg mode information will not be touched. In order to switch on mode choice optimization on the subtour-level, indicate the mode choice set in a separate config section with the name *planomat*:

```
<module name="planomat">
  ...
  <param name="possibleModes" value="car,pt" />
  ...
</module>
```

- The transport mode identifiers are determined by the MATSim API.
- The resulting plan will contain legs only with modes from the specified choice set. Any leg mode information that was previously present is deleted. For example, when the initial demand contained legs with mode "bike", this mode will not be present in the plan after it was processed by the planomat replanning module with the above setting.
- The same transport mode will be chosen for every leg of a subtour. See the algorithm for subtour identification here. Note that, up to now, mode choice does not interact between subtours. This might generate physically unfeasible mode chains (imagine a car leg from a location where the vehicle is not available). An algorithm for *feasible mode chain analysis* has been implemented (see here), but not yet integrated with the processing of the mode choice set.

The behavior of the planomat module can be controlled with several other configuration parameters. For example, set the maximum number of GA generations to a lower value of 10 than the default value of 100:

```
<module name="planomat">
  ...
  <param name="jgapMaxGenerations" value="10" />
  ...
</module>
```

For additional config parameters, their respective default values and the possible value ranges, please refer to the code at
`org.matsim.core.config.groups.PlanomatConfigGroup.class`.

References

see hyperlinks in description.

SubtourModeChoice. Status: probably works

Maintainer: Michael Zilske

In contrast to "ChangeLegMode", which changes *all* legs of a plan to a different mode, this module changes the modes of sub-tours separately.

For example, somebody might take the car to work, walk to lunch and back, and take the car back home.

"chainBasedModes" means modes where a vehicle (car, bicycle, ...) is parked and in consequence needs to be picked up again.

```
<module name="subtourModeChoice" >
        <param name="chainBasedModes" value="car, bike" />
        <param name="modes" value="car, bike, pt, walk" />
</module>
```

The module is called by inserting the following lines into the "strategy" module:

```
<module name="strategy" >
        <param name="ModuleProbability_XXX" value="0.1" />
        <param name="Module_XXX" value="SubtourModeChoice" />
         ...
</module>
```

For modes other than car, travel time and travel distance are computed according to some heuristics, which are configured in the router.

Combination of strategy modules

It is not clear which combinations of these modules can be used together. Depending on required features, special variants sometimes need to be used. This has not yet been sorted out.

The following table tries to give an overview, but it is an old table that has not been maintained (table status 2011; this sentence written 2012).

Choice dimension	Default Strategy	Transit	Transit &

			Parking
departure time choice	TimeAllocationMutator	TransitTimeAllocationMutator	?
route choice	ReRoute	ReRoute	?
mode choice (all legs get same mode)	ChangeLegMode	TransitChangeLegMode	?
mode choice (each leg can have a different mode)	ChangeSingleLegMode	TransitChangeSingleLegMode	?
mode choice (subtour-based)	SubtourModeChoice	TransitSubtourModeChoice	?
location choice	LocationChoice	?	?

Legend:

- n/a means this choice dimension is not supported/available for the specified feature
- ? means there is no known implementation available

Other configurable modules

Modules are loosely defined by their corresponding entry in the config file.

They are also sorted in the same sequence (which is done by the machine, not by content).

Note that individual config options are often explained inside the config section of the log file.

Config file modules that just define files/directories are, as a tendency, not explained here.

Note that strategy modules (such as ReRoute, Planomat) are described in a separate section.

Maintainers are mentioned as far as possible, but they are *not* responsible for answering arbitrary service requests.

"JDEQSim". Status: works for ivt

Maintainer: Rashid Waraich

Overview

JDEQSim (Java Deterministic Event Driven Queue Based Simulation) has the following properties and features:

- it is based on a discrete event simulation model
- traffic simulation is based on a queue model for streets (FIFO: first in first out)
- deadlock prevention is achieved by squeezing vehicles
- gaps generated at front of queue propagate backwards with a speed called 'gapTravelSpeed' resulting in a more realistic traffic model

Usage

Insert a new module called 'JDEQSim' into the config XML file. All parameters are optional and have default values (shown below), ~~never the less~~ it could be helpful to know their meaning and physical units.

```
<module name="JDEQSim">
        <param name="endTime" value="00:00:00"   />
        <param name="flowCapacityFactor" value="1.0"   />
        <param name="storageCapacityFactor" value="1.0"   />
        <param name="minimumInFlowCapacity" value="1800"   />
        <param name="carSize" value="7.5"   />
        <param name="gapTravelSpeed" value="15.0"   />
        <param name="squeezeTime" value="1800"   />
</module>
```

The mobsim type now also needs to be defined in the controler section of the config file. See comments in config dumps in logfiles.

The 'endTime' defines the time of the last event of the simulation. If it is set to '00:00:00', no end time is defined and the simulation will stop, when the last event of the simulation has been processed. The (scaling) parameters 'flowCapacityFactor' and 'storageCapacityFactor' can be used as with mobSim and have no unit. The 'minimumInFlowCapacity' defines for all roads the minimum number of cars, which could enter the road per hour, for the congestion less case. The 'carSize' parameter allows to set the size of a car in meters. The 'gapTravelSpeed' parameter defines the speed of gaps in [m/s]. Finally the 'squeezeTime' is used for deadlock prevention and defines, how long a car should wait at maximum for entering the next road before deadlock prevention is turned on (unit: seconds).

The 'minimumInFlowCapacity' is a parameter, which was not published in the C++ DEQSim, but only used interally and was hardcoded to the value 1800 vehicles per hour. This value was estimated from literature assuming that independently from the speed limit of a road the minimum interval between two vehicles is 2 seconds (inverse of 1800 vehicles per hour). This factor does not need to be changed, when the 'flowCapacityFactor' is changed, as the scaling is automatically done internally. The reason for publishing this factor is to make it possible for users to adapt this factor, if they want to use a different minium inflow capacity based on their model estimations.

Hints

- ~~If the module 'JDEQSim' is present all parameters from module 'simulation' are ignored.~~ [[Given that the type of the mobsim now needs to be specified in the controler section of the config file, this works differently now. kai, apr'11]]
- You might consider turning on the module 'parallelEventHandling' when using JDEQSim, as often JDEQSim can make much better use of this module than QueueSim (as JDEQSim is faster).
- Use the the following controller for running the java DEQSim micro-simulation: org.matsim.controler.Controler (and not

org.matsim.mobsim.cppdeqsim.DEQSimControler, which is used for C++ DEQSim)

- If you are getting lots of breakdowns, consider using smaller squeezeTime (e.g. 10 seconds or lower)

Requirements for the Plans XML File

- For each person the 'end_time' of the first act must be defined ('dur' is ignored).
- For the other acts of a person either 'dur' or 'end_time' needs to be defined
- If both 'dur' and 'end_time' are defined, then only the one which occurs earlier is considered

Details of the Model

Difference between MobSim and JDEQSim

- QueueSim uses a simulation approach called 'fixed-increment time advance' instead of 'next-event time advance', which makes it much slower than JDEQSim for high resolution networks.
- JDEQSim allows squeezing of vehicles to resolve possible deadlocks. Deadlock prevention in QueueSim is (traditionally) dealt with by removing vehicles from the network or squeezing.
- JDEQSim models gap travel times more realistically than QueueSim, where this feature is missing.

Further Reading

This implementation is based on the micro-simulation described in the following paper:

Charypar, D., K. Nagel and K.W. Axhausen (2007) An event-driven queue-based microsimulation of traffic flow, *Transportation Research Record*, **2003**, 35-40.Order here.

Some Java specific implementation aspects and performance tests of JDEQSim and parallelEventHandling are described in the following paper:

Waraich, R., D. Charypar, M. Balmer and K.W. Axhausen (2009) Performance improvements for large scale traffic simulation in MATSim, paper presented at the *9th Swiss Transport Research Conference*, Ascona, September 2009. Download from here.

"controler". Status: indispensable

Maintenance and Questions

Marcel Rieser, senozon AG (rieser_at_senozon.com)

Javadoc

www.matsim.org/javadoc/org/matsim/core/controler/package-summary.html

Config Parameters

www.matsim.org/javadoc/org/matsim/core/controler/package-summary.html#controler_parameters

In Brief

Central module to run matsim. Specifies, for example, the number of iterations.

Notes

See here for some instructions how to use an external executable as mobsim.

"counts". Status: works for vsp and ivt

Maintenance and Questions:

A. Horni, IVT (horni_at_IVT.baug.ethz.ch)

Javadoc:

www.matsim.org/javadoc/org/matsim/counts/package-summary.html

Config Parameters

www.matsim.org/javadoc/org/matsim/counts/package-summary.html#counts_parameters

In Brief:

MATSim can compare the simulated traffic volumes to traffic counts from the real world. Counts is the module that allows to

- read some external file with traffic flow counts
- compare them automatically to the counts generated inside the matsim simulation
- submit the result to a kmz file which can be displayed inside google earth

There is a feature to re-scale the counts before comparison (for example if you are running the simulations with a 10% sample).

Comparison Data

Prepare a file containing the real-world traffic counts. The file, e.g. named counts.xml, must follow the xml-format defined in counts_v1.xsd. An example of such a file can be found in MATSim at examples/equil/counts100.xml.

The file contains the following information:

- For each link in the network for which traffic count information is available, a count-element must exist. The count-element specifies the link it refers to in its attribute `loc_id`. In addition, an optional `cs_id` can be stored that may, for example, refer to the original id of the counting station (for tracking back the origin of the data).
- In each count-element, 1 to 24 volume-elements can appear. Each volume-element contains the measured traffic count (attribute "`val`") for an hour of the day (attribute "`h`", numbered from 1 to 24; 1 = 00:00-00:59, 2 = 01:00-01:59, etc). It is not necessary that traffic counts are available for all 24 hours of a day.

Enabling Comparison in Configuration File

Add the following lines to your configuration file:

```
<module name="counts">
  <param name="inputCountsFile" value="/path/to/counts.xml" />
  <param name="outputformat" value="txt,html,kml" />
</module>
```

The comparison is automatically generated every 10th iteration. Generated output is located in the output-directory of the iteration (usually something like `output/ITERS/it.10/`).

Configuring the Counts Comparison

The counts-module offers the following config-parameters:

- `<param name="outputformat" value="txt,html,kml" />`
  The output format specifies in which format the comparison results are written to disk. It can be any combination of `txt`, html and `kml`. Multiple formats can be specified separated by commas. `txt` writes simple text-tables containing the values to a file. It is most useful to create custom graphs, e.g. in Excel. `html` creates a directory containing several html files, allowing to browse the results interactively. `kml` creates a file to be displayed in Google Earth. This last option only works if the correct coordinate system is set.
- `<param name="countsScaleFactor" value="1.0" />`
  If you only simulate a sample of your population, the simulated traffic volumes are likely lower than the real-world traffic counts. In order to allow useful comparison,

one can specify a factor by which the simulated traffic volumes are multiplied. For example, if you simulate a 25% sample of your full population, specify a countsScaleFactor of 4.

- `<param name="distanceFilterCenterNode" value="2386" />`
  `<param name="distanceFiler" value="30000.0" />`
  If the traffic counts cover a larger area than the area being simulated, the traffic counts outside your area will result in a bad comparison. Instead of removing the traffic counts from the counts.xml, you can specify a filter to only include some traffic counts from the file in the comparison. To activate the filter, specify the id of a node that acts as the center of a circle. The circle has the radius specified in "`distanceFilter`", the unit being the same unit as the length of links (i.e. usually meters).

"evacuation"(-ivt). Status: ??

Evacuation code used by IVT; please note that IVT and VSP use different evacuation codes.

Maintained by C. Dobler.

I (kn) don't know how this works

"evacuation"(-vsp). Status: works if you know what you are doing

(Note that VSP and IVT use different evacuation packages.)

Maintenance and Questions

G. Lämmel, TU Berlin

Javadoc

http://www.matsim.org/javadoc/org/matsim/evacuation/package-summary.html

Config Parameters

http://www.matsim.org/javadoc/org/matsim/evacuation/package-summary.html#evacuation_parameters

In Brief

I (kn) can't say how this works. There is, at this point, neither documentation nor funding.

"facilities". Status: "user" version work in progress

Maintainer: Andreas Horni

One may, or may not, use a separate file that contains "facilities" – essentially some kind of land use information.

The prototype for this is fairly old. But the final design is somewhat different, and has not been fully executed. So I (kn) do not know if this can currently be used as a non-developer.

"global". Status: indispensable

"Maintainer": Marcel Rieser

"Global" information. Arguably should be merged with "controler" section.

"households". Status: probably ready but nowhere used

Maintainer: Christoph Dobler

An option to read a households file into matsim.

I (kn) don't know the exact status.

"network" (time dependent). Status: works for vsp

Maintenance: G. Lämmel, VSP

MATSim provides the opportunity to model time dependent aspects of the network explicitly. For each link in the network basic parameters (i.e. freespeed, number of lanes and flow capacity) can be varied over the time. ~~So it is possible to model accidents or the like~~, One particular area for this technique is the modeling of evacuation scenarios.
~~In the case of an evacuation simulation~~, the network has time dependent attributes. For instance, large-scale inundations or conflagrations do not cover all the endangered area at once.
In MATSim this time varying aspects are modeled as network change events. A network change event modifies parameters of links in the network at predefined time steps. The network change events have to be provided in a XML file to MATSim.

A sample network change event XML file could look like:

```
<?xml version="1.0" encoding="UTF-8"?>
<networkChangeEvents xmlns="http://www.matsim.org/files/dtd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.matsim.org/files/dtd
http://www.matsim.org/files/dtd/networkChangeEvents.xsd">
  <networkChangeEvent startTime="03:06:00">
```

```
        <link refId="12487"/>
        <link refId="12489"/>
        <link refId="12491"/>
        <freespeed type="absolute" value="0.0"/>
  </networkChangeEvent>
</networkChangeEvents>
```

This change event would set the freespeed of the links `12487, 12489, 12491` to 0 m/s at `03:06` am (all values have to be provided in SI units). These values are valid until the next network change event (if there is any) changes the freespeed of link `12487, 12489, 12491` again. In this example the freespeed would be set to an absolute value. It is also possible to take the old `freespeed` value and multiply it by a factor. For dividing the old `freespeed` value by 2, the corresponding line of the network change event XML file would look like:

```
<freespeed type="scaleFactor" value="0.5"/>
```

Besides changing the `freespeed`, one could also change the number of `lanes`:

```
<lane type="absolute" value="2.0"/>
```

Or the flow capacity:

```
<flowCapacity type="absolute" value="0.0"/>
```

To make use of the network change events one has to define ~~it~~ in the MATSim config file. Therefore the following two lines have to be added in the network section of the config file:

```
<param name="timeVariantNetwork" value="true" />
<param name="inputChangeEventsFile" value="path_to_ change_events_file" />
```

Now one has just to start the controller with this config file and the network change events will be applied automatically.

It seems that the ``absolute" version of this module was never tested (and may not work) with freespeeds other than zero. kai, oct'10

"parallelEventHandling". Status: works for ivt and vsp

Maintainer: Rashid Waraich

see details here.

"planCalcScore". Status: nearly indispensible

Maintainer: Marcel Rieser

This module contains the definitions for the utility function.

Some help for it should be in the tutorials.

There is also some description in the "scoring function" section of the documentation.

"qsim" (parallel version). Status: looks promising

Responsible: C. Dobler, IVT

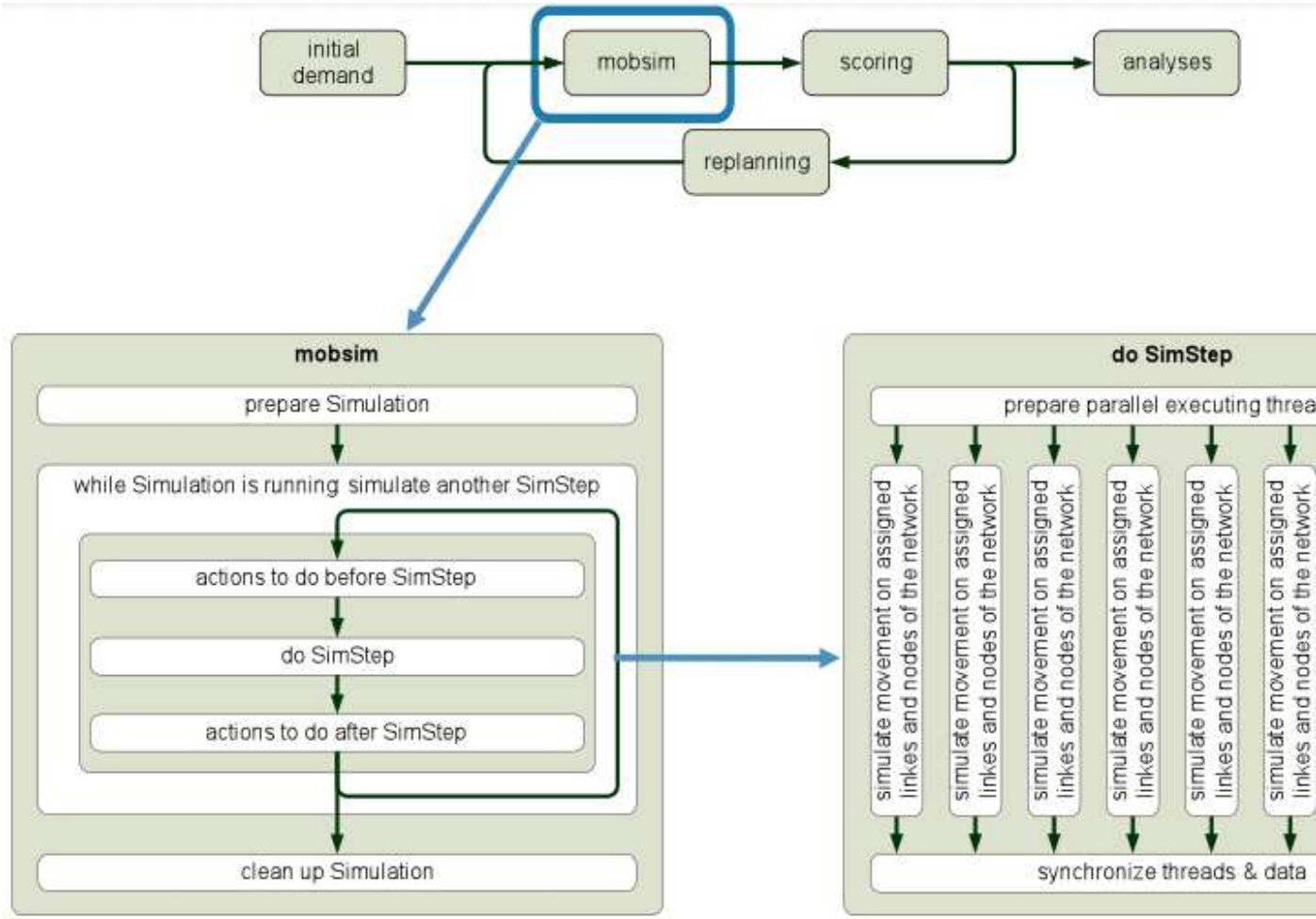
Analysis of performance and structure of (non parallel) QueueSim shows:

- Simulation of movement on links and over nodes is most time consuming.
- Within a timestep actions on nodes and links can be simulated on parallel threads with low additional synchronization effort.

The parallel QueueSim is based on the existing QueueSim and can be used by just adding a new parameter to a scenario configuration file (see below).

First performance measurements show promising results.

Working paper will be published in Q2 2010.



The config option presumably is:

```
<module name="qsim">
   ...
   <param name="numberOfThreads" value="5"/>
</module>
```

Any number of threads larger than one triggers the use of the parallel version. (??)

"qsim". Status: works but is not very transparent

~~If you do ***not*** put a "qsim" section into the config file, the system will use the default "simulation" (look there).~~

"qsim" is what we use for new features such as public transit or signalsystems. "New features" implies "unstable". Use only if you have to.

Also see www.matsim.org/javadoc/org/matsim/ptproject/qsim/package-summary.html

Some calibration hints, especially when the main mode is not "car"

The (exit) flow capacity of a link is:

```
capacity_value_of_link / capacity_period_of network * flow_capacity_factor
```

where

- the capacity value of the link is given by the link entry in the network file
- the capacity period of the network is given at the beginning of the "links" section in the network file. Normally set to one hour
- the flow capacity factor is given in the qsim config group

The storage capacity of a link is:

```
(length_of_link * number_of_lanes_of_link / effective_cell_size) *
storage_capacity_factor
```

where

- the length of the link is given by the link entry in the network file
- the number of lanes of the link is given by the link entry in the network file
- the effective cell size is given at the beginning of the "links" section in the network file. Normally set to 7.5m
- the storage capacity factor is given in the qsim config group
- There is also an effective lane width, also at the beginning of the "links" section in the network file, normally set to 3.75m. See below for its use.

This is most useful if you have something else than cars, for example pedestrians. Let us assume an effective lane with of 0.4m and an effective cell size also of 0.4m. This would lead to a maximum density of 0.4*0.4=0.16persons/m^2, not totally unrealistic.

If, now, a link has an area of 200m^2 and a length of 50m, then it would obtain

```
number_of_lanes = area / length / effective_lane_width = 200 / 50 / 0.4 =
10
```

Note that, in the end, the lane width is not used by the dynamics; all the meaning is subsumed in the number of lanes. The storage capacity comes out as

```
storage_capacity = number_of_lanes * length / effective_cell_size
```

in the above example

```
= 10 * 50 / 0.4 = 1250 .
```

This is, naturally, the same as dividing the 200m^2 of the link by the 0.16persons/m^2.

The effective lane width might be used by the visualization (unclear if this is the case).

"roadpricing". Status: works for vsp

Maintainer: Michael Zilske

The roadpricing module provides functionality to simulate different road-pricing scenarios in MATSim.

Documentation can be found at:

- http://ci.matsim.org:8080/job/MATSim_contrib_M2/org.matsim.contrib$roadpricing/javadoc/?

Publications using this module:

- https://svn.vsp.tu-berlin.de/repos/public-svn/publications/vspwp/2007/07-14/
- https://svn.vsp.tu-berlin.de/repos/public-svn/publications/vspwp/2008/08-01/
- https://svn.vsp.tu-berlin.de/repos/public-svn/publications/vspwp/2008/08-08/
- https://svn.vsp.tu-berlin.de/repos/public-svn/publications/vspwp/2010/10-03/

"signalsystems". Status: works for vsp

Maintainer: Dominik Grether

The signal systems module provides functionality to simulate traffic lights with MATSim. It is recommended to use a nightly build that is younger than 04-19-2011, i.e. revision 15081.

The starting point of the documentation is

- http://www.matsim.org/javadoc/org/matsim/signalsystems/package-summary.html

Note that there are links to continuative documentation at the bottom of the package-summary.html www page.

Nightly builds younger than 04-21-2011 contain a tutorial that shows you how to set up a traffic light scenario. The network and traffic light configuration of the turorial is shown in the slides attached to this page. The network and code can be found in the folder tutorial/unsupported/example90TrafficLights in the nightly build. The code examples are divided into several classes:

- CreateSimpleTrafficSignalScenario.java: Uses traffic signals without lanes and creates the traffic lights at nodes 3, 4, 7 and 8.
- CreateTrafficSignalScenarioWithLanes.java: Uses traffic signals with lanes and creates the traffic lights at nodes 2 and 5.

Publications using this module:

- https://svn.vsp.tu-berlin.de/repos/public-svn/publications/vspwp/2008/08-24/
- https://svn.vsp.tu-berlin.de/repos/public-svn/publications/vspwp/2011/11-12/
- https://svn.vsp.tu-berlin.de/repos/public-svn/publications/vspwp/2011/11-08/

This documentation is missing an explanation of the "lanes" option. Please ask if you need this (separate "lanes" for separate turning movements).

Attachment	Size
signals_tutorial.pdf	77.72 KB

"simulation". Status: should work

This was essentially the production of the queue simulation until Nov/2010. The "qsim" was then forked out for further development. Unfortunately, this fork was done somewhat too late, so "simulation" is not exactly the stable version that was used over many years, but something that is already somewhat modified, and was not used very much after that. (Please let us know if you have problems.)

Note that you will get a "simulation" section in the log file even if you have selected a different mobsim (such as qsim or jdqsim).

There used to be an option to start an external mobsim. This still seems to be there but the syntax is a bit awkward:

```
<module name="controler" >
   ...
```

```
   <param name="mobsim" value="null" />
</module>
<module name="simulation" >
   <param name="externalExe" value="<path-to-executable>" />
</module>
```

I.e. you need to specify that you are *not* using the (queue)Simulation, but then set a parameter inside the (queue)Simulation config block.

"strategy". Status: indispensable

Maintainer: Marcel Rieser ("core")

See here.

"transit" (public transport). Status: works

A public transport system is simulated and integrated on a fine scale with both the traffic simulation and the behavior of the artificial population.

Agents who use transit determine a route to their destination based on the transit schedule. Transit vehicles are moved on the road network in accordance with the traffic flow model, i.e. they may get stuck in congestion and fail to keep their schedule. Agents getting on and off transit vehicles cause realistic delays.

A transport mode decision model is implemented which allows agents to switch their choice of driving a car or using transit based on the relative utility of the two modes. The disutility of travel time, which this model takes into account, is based on actual travel times taken from the simulation.

See the tutorial. This requires quite some additional input.

Reference

M. Rieser, K. Nagel; **Combined agent-based simulation of private car traffic and transit**; IATBR 2009

"travelTimeCalculator". Status: nearly indispensable

Maintainer: Marcel Rieser ("core")

"router" and "travelTimeCalculator" are separate in matsim, so that they can be configured separately. They refer to each other, though.

"vehicles". Status: probably reads the file correctly, but does nothing else

Maintainer: Michael Zilske (within limits of DFG/MUC project; possibly pt project)

"vspExperimental". Status: used by VSP

This section defines switches that are used at VSP or when collaborating with VSP. ~~There~~ are experimental and may we withdrawn without notice.

vsp defaults

I (kn) am in the process of defining some defaults that everybody at VSP should be using. These can be switched on by:

```
<module name="vspExperimental" >
   <param name="vspDefaultsCheckingLevel" value="abort" />
   ...
</module>
```

This will make the code abort when these defaults are violated.

The number of vsp defaults will grow over time. This may have the effect that some config file that used to be working for you in the past may not work any more after an svn update. I will try to communicate such changes, but will sometimes fail to do so. In any case, if you encounter an abort because of a vsp defaults violation, please

- check what is causing the problem, and
- enter the relevant config setting into all config files that you use.

If you think that you cannot live with these settings, please talk to me.

Since those settings involve all aspects of matsim, they may often be irrelevant to you. Please set them anyway~~s~~.

0 deprecated modules

Deprecated Modules

"world"

Dismantler: Michael Zilske

This was an attempt to integrate GIS functionality into matsim.

Has been superceeded by calls to geotools. Please use geotools functionality. Look under the demand generation tutorials for getting some ideas.

World also provides datastructures to assign facilities to links, and links to zones, etc. This functionality is mostly used in initial demand modelling, but is not very straight-forwardly implemented. Should be replaced in the future with some kind of "~~mappig~~ manager" to manage the mappings between different MATSim objects, like facilities, links, etc.

The MATSim default scoring function (= utility function)

This section contains information that pertains to the so-called "Charypar-Nagel scoring function".

Calibration of the scoring function

Simplified version

The simplified version assumes that all activities operate near their typical duration. In this case (see here), one can approximate the marginal utility of activity duration (i.e. the marginal utility if the sum of all activities is extended by that amount of time) by beta_perf .

Now let us consider the typical changes (of the Vickrey scenario). Note that in the Vickrey scenario, the meaning of the marginal utility of arriving earlier means the marginal contribution assuming that the travel time remains the same. We will assume that activities are ended by the endtime attribute, not by the duration attribute.

Travel takes longer (by amount deltaTtime)

In this situation, the activity that follows the trip is cut short by deltaTtime. We thus have the following (linearized) modifications of the utility:

- Travel takes longer by deltaTtime; the utility change is beta_travel * deltaTtime . Note that beta_travel typically is negative.
- The following activity is shortened by deltaTtime; the (linearized) utility change is - beta_perf * deltaTtime . beta_perf is typically positive, so the contribution is negative.

Overall: The (linearized) utility change caused by longer travel is

```
( - beta_perf + beta_travel ) * deltaTtime
```

Traveller increases arriving early (by amount deltaEtime)

In that situation, the traveller will "do nothing" between the arrival and the opening time of the activity. That is, the amount of time that the traveller is doing nothing is now increased by deltaEtime. Consistent with the meaning of the Vickrey parameter "marginal utility of arriving early", we assume that the travel time is the same compared to the later arrival. This

means that the preceeding activity was cut shorter by deltaTtime. We thus have the following (linearized) modifications of the utility:

- The preceeding activity is shortened by deltaTtime; the (linearized) utility change is - beta_perf * deltaEtime . beta_perf is typically positive, so the contribution is negative.

There are no other contributions, since the time between the arrival and the opening time prodices neither positive nor negative utility contributions. Overall: The (linearized) utility change caused by arriving early is

```
( - beta_perf ) * deltaEtime
```

That is, as long as there are no additional utilities or disutilities of waiting, the marginal utility of performing can be approximated by the marginal utility of schedule delay early.

Traveller increases arriving late (by amount deltaLtime)

In this situation, we have the following (linearized) modifications of the utility:

- The preceeding activity is extended by deltaLtime; the (linearized) utility change is beta_perf * deltaLtime . beta_perf is typically positive, so the contribution is positive.
- The following activity is shortened by deltaLtime; the (linearized) utility change is - beta_perf * deltaLtime . beta_perf is typically positive, so the contribution is negative (and exactly cancels the previous contribution).
- Arriving late is increased by deltaLtime; the (exact) utility change is beta_late * deltaLtime . beta_late is typically negative, so the contribution is negative.

Overall: The (linearized) utility change caused by increasing the amount of arriving late is

```
beta_late * deltaLtime
```

Overall

Overall, calibration of the Charypar-Nagel scoring function is best done as follows:

- **Run a survey and estimate logit models that include penalties for travelling (by mode), schedule delay early, and schedule delay late.**
- **The marginal utility of schedule delay early from the logit model, multiplied by minus one, results in the MATSim beta_perf.** Since the marginal utility of schedule delay early is typically negative, beta_perf is thus typically positive. This is the marginal opportunity cost of time. A useful interpretation is that this is the difference between "leisure" and "doing nothing".
- **The marginal utility of travelling from the logit model, *plus beta_perf,* results in the MATSim beta_trav (by mode).** That is, the MATSim beta_trav is an *additional utility offset* when compared to doing nothing. Since driving can well be seen as more positive than doing nothing (e.g. because of making phone calls, listening to music, enjoying to drive), the MATSim beta_trav can well be positive.
  (Note that this has still nothing to do with "positive values of travel time", e.g. by Susan Handy. Those positive values imply that the additional utility offset over-compensates the marginal opportunity cost of time. In other words, "time spent driving home" is (to an extent) seen more positive than "being at home".)

- **The marginal utility of being late from the logit model results in the MATSim beta_late.**
- Note that you also need reasonable values for opening time, latest arrival time, and closing time, in order to achieve that the schedule delay cost mechanics works in MATSim. This is quite clear if you think about it; nevertheless, it has been forgotten uncountable times (in particular in studies that start from trips, not from full daily plans).

Without schedule delay

If you intend to run MATSim without time adaptation (TimeAllocationMutator), these things are not that critical. In that situation, you just need to make sure that - beta_perf + beta_trav matches your marginal utility of travel time differences. An easy way in our view is:

- Set beta_car to zero (i.e. assume that driving is as good or bad as doing nothing).
- Set beta_perf to the estimated marginal utility of travel time savings (make sure you get the sign right; beta_perf should be positive).
- Set (say) beta_pt to your estimated marginal utility of travel time savings (should be positive) *minus* the MATSim beta_perf. The result may be positive (implying that spending time using the mode is better than doing nothing) or negative (implying that spending time using the mode is worse than doing nothing).

Note that even without time adaptation, beta_late may still have an influence if you have set the latest arrival times for some activities.

Full version

"Full version" would imply that we could calibrate the MATSim parameters also for situations where the actual activity durations are far from their "typical" values. This could happen for two reaons:

- There are too many activities that need to be squeezed into a day. A possible interpretation would be that beta_perf corresponds to the marginal utility of additional leisure time on, say, sundays, but the weekday activites cannot be shifted to sundays.
- There are too many activities that need to be squeezed into certain time periods, say between day care opening and closing, or into typical business hours.

Both of these interpretations make sense (in my view) and should be investigated for MATSim. Presumably, there is already general research; it would then be necessary to bring that research and the MATSim formulation together.

Default values for the Charypar-Nagel scoring function

As explained here, the MATSim scoring function has, under some circumstances (actual durations near "typical" durations"), some similarity to the Vickrey scenario.

The "typical" parameters of the Vickrey scenario are beta_early=-6, beta_travel=-12, and beta_late=-18.

For MATSim, as explained here, this translates into beta_perf=6, beta_travel=-6, and beta_late=-18. These are the parameters that were, for a lack of estimated parameters, introduced into (the precursor of) MATSim approximately in 2006.

These parameters are multiplied with the beta_brain parameter, which can be seen as a separately configurable logit scale parameter. A useful setting for this parameter was determined via systematic tests concerning the stability of the iterations, see here.

As a next step, an infrastructure to compare MATSim simulations with real world traffic counts was set up. Only after that infrastructure was there, an attempt to calibrate the MATSim parameters from a survey was made. This is documented here, unfortunately in German. Two results were

- The estimated parameters all have the same order of magnitude as the MATSim default parameters (the "Vickrey" parameters).
- The results with respect to traffic counts were not considerably different from before.

Interpretation of the logarithmic "utility of performing"

The so-called "Charypar-Nagel scoring function" is used in many MATSim studies. It is called that way because there is an ancient paper where this scoring function was introduced.

It uses a logarithmic utility of time for activities: U = beta * t_x * ln(x/t_0) . I sometimes call t_x the "typical duration".

The first derivative of U is beta at the typical duration:

- `dU/dx = beta * t_x / x`
- `dU/dx(x=t_x) = beta`

Interpretation: marginal utility of duration at "typical duration" is indep of activity type. (*)

The second derivative of U at the typical duration is

- `d^2U/dx^2 = - beta * t_x / x^2`
- `d^2U/dx^2(x=t_x) = - beta / t_x`

An important consequence of this is that there is no separate free parameter to calibrate the curvature (= 2nd derivative) at the typical duration: beta needs to be the same across all activities, and t_x is given by (*).

A second consequence is that t_0 is largely irrelevant. It shifts the function up and down, i.e. it determines how much you lose if you drop an activity completely.

In the original paper (and in most of MATSim), t_0 is set to t_x * exp(-10h/t_x) . This has the (intended) consequence that all activities have the same utility contribution at their typical duration:

```
U = beta * t_x * ln( x / t_x / exp(-10h/t_x) ) = beta * t_x * [ ln( x/t_x ) + 10h/t_x ]
```

which is, at x=t_x: = beta * t_x * [0 + 10h/t_x] = beta * 10h . With our usual beta = 6Eu/h, this results in 60Eu per activity.

The slope at U=0, i.e. at x=t_0, is

```
(beta * t_x / t_x) * exp( 10h/t_x) = beta * exp( 10h/t_x )
```

which *de*creases with increasing t_x. This means that activities with larger typical duration are *easier* to drop completely.

In the end, this makes sense: Since the additional score of any activity is the same, the score *per time* is smallest for activities with long typical durations. Therefore, it makes sense to drop them first.

But practically, this is probably not desired behavior, since it would first drop the home activity from a daily plan.

Overall, therefore: ***In my opinion, the current utility function does not work for activity dropping.***

An alternative, never tested since activity dropping was never tested with this utl fct, would to to recognize that U'(t_0) = beta * t_x / t_0 , i.e. *increasing slope* with *decreasing* t_0. That is, high priority activities should have t_0 such that t_x/t_0 is large (large slope = hard to drop). Activities of the same priority should have t_0 such that t_x/t_0 is the same between those activities. Overall, something like

```
weight \propto t_x/t_0
```

or

```
t_0 \propto t_x/weight
```

where large weight implies a large importance of the activity.

This was, as said, never tried, since activity dropping was never systematically tried. It also does not fix the problem, discussed later, that different activities might have different resistance against making them shorter; since this is U'', this is -beta/t_x with the above utl fct: activities are shortened proportional to their typical duration.

To make matters worse, there is currently the convention that negative values of U are set to zero. This is done since we need useable values for negative durations (since they may

happen at the "stitching together" of the last to the first activity of a day), and if we give those a "very negative" score, then the utl at t=0 cannot be even smaller than this.

This has, however, the unfortunate consequence that the "drift direction" of the adaptive algorithm, once an activity duration has gone below t_0, goes to zero duration.

Outlook: What would we want for our next generation utl function? Some wishes from my perspective:

- Curvature at typical duration can be calibrated
- Slope at U=0 can be calibrated
- Utl function extends in meaningful way to negative durations (this would fix the arbitrary handling that we currently employ)

In my view, a polynomial of second degree would be worth trying. As usual, there are several ways to set this up. One way is to expand around the typical duration:

```
U(t_x + eps) = U(t_x) + eps * U'(t_x) + eps^2 * U''(t_x)/2
```

or

```
U(x) = U(t_x) + (x-t_x) * U'(t_x) + (x-t_x)^2 * U''(t_x)/2
```

with t_x = typical duration, U'(t_x) = beta = marg utl at typ dur, U''(t_x) = curvature at typ dur ("priority"), and U(t_x) = "base value of act" (which could be something like beta*t_x).

Another way (having the parabola going through (0,0)) would be

```
U(x) = - a x ( x - c ) = - a x^2 + a c x
U'(x) = - 2 a x + a c
prio = U'(x=0) = a c , i.e. c = prio/a .
beta = U'(x=t_x) = - 2 a t_x + prio, i.e. a = (prio - beta)/2t_x
```

There is other work (e.g. by Joh) that should be looked at.

Visualization and analysis

Somewhat historically, there was

- The so-called OTFVis for visualization.
- Self-written so-called events handlers for analysis.

In the meantime,

- senozon has written a commercial tool for visualization called via. That tool also does an increasing number of analysis tasks.

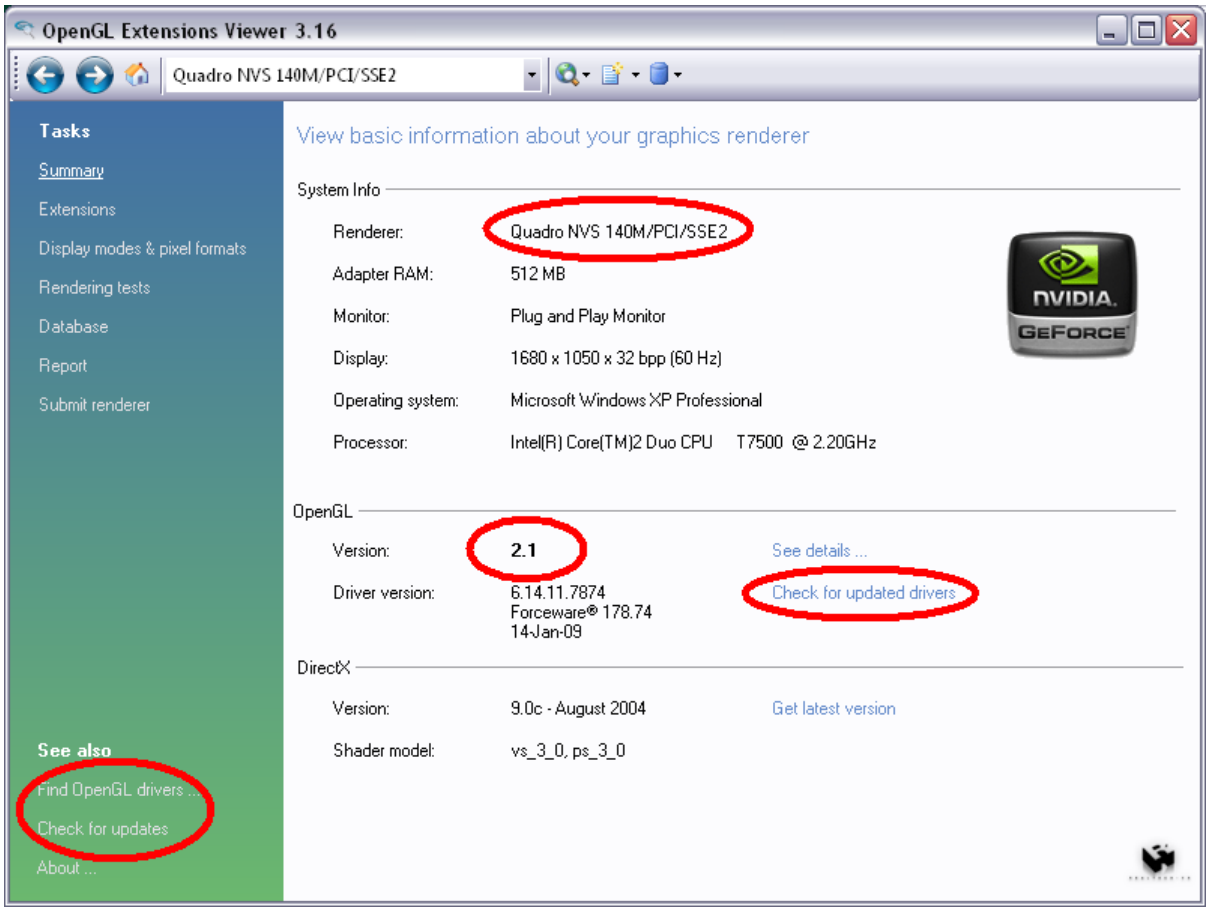
Using OTFVis

This documentation is about the version of OTFVis included in the MATSim release 0.4.x or earlier. To use OTFVis in any later release (or in the current development version), refer to the new documentation at extensions/otfvis.

OTFVis is a visualizer for MATSim. It can be used to replay snapshots of simulations, or run a simulation and interact with it. The visualizer makes use of hardware acceleration (Open GL) and is thus also suitable for visualizing large scenarios. For the hardware acceleration to work, (i) the OpenGL graphic card driver installed on you machine must be at least of **version 2.0** and (ii) **native libraries** are required, which must be correctly set up.

Check and Update Graphic Card Driver

Either you use check and update mechanims / software already installed (e.g. NVIDIA software, ATI update manager, etc...) or download and install OpenGL Extension Viewer. After starting this little tool, it ~~show~~ all necessary information abour your graphic card including OpenGL version. Please be sure that at least **OpenGL version 2.0** is installed. Otherwise try to find approriate driver updates of your graphic card (the read circles in the Figure below shows the important ~~featrues~~ / information).



Starting the Visualizer

The main class for the visualizer is org.matsim.run.OTFVis. If it is invoked without any arguments, a short description is displayed how the visualizer can be used. The different ways to start OTFVis will be described in more details below.

The visualizer may require a lot of memory, it is thus advised to start it with the corresponding Java options:

```
java -Xmx500m -cp MATSim.jar org.matsim.run.OTFVis arguments
```

This sets the memory limit for Java to 500 MB.

In addition, OTFVis requires native libraries to provide the required hardware acceleration. Native libraries contain source code that can only run on some specific machines and operating systems. We provide the native libraries for the most common operating systems in `libs/jogl-1.1.1`. You may also check for the newest versions of the native libraries at the Java OpenGL website. There are several ways to include the native library when starting OTFVis:

- specify on the command line. Example:
- `java -Djava.library.path=libs/jogl-1.1.1/jogl-1.1.1-linux-amd64/lib/ -cp MATSim.jar org.matsim.run.OTFVis arguments`
-

- install the corresponding native library as a java extension. The location where the library must be installed differs from one operating system to ~~the next one~~. Please check the documentation at java.sun.com for general information about Java Extensions or at Apple for Macintosh-specific installation instructions.

- specify the native library location in Eclipse. If you use the Eclipse IDE for MATSim development, you can specify in the buildpath-settings which native library should be used. Open the Project Properties in Eclipse, and go to Java Build Path > Libraries. Select the jogl.jar in the list of libraries, and expand its display. There should be an entry "Native libary location" that you can set to the corresponding library for your operating system.
- I (KN) had also success with putting it somewhere and then adding the path to the library path:
- `export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${CORRECT_PATH}/libs/jogl-1.1.1/jogl-1.1.1-macosx-universal/lib  # linux`
- `export DYLD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${CORRECT_PATH}/libs/jogl-1.1.1/jogl-1.1.1-macosx-universal/lib  # mac os x`

Creating snapshots (mvi-files) from Events

Use the following arguments:

```
-convert event-file network-file mvi-file [snapshot-period]
```

to record a snapshot of all vehicles' positions every snapshot-period seconds, based on the events and network given in the corresponding files.

Example call:

```
java -cp MATSim.jar org.matsim.run.OTFVis -convert output/50.events.txt.gz
input/network.xml.gz output/50.visualization.mvi 300
```

This will create a snapshot of every 5th minute and store it in the file `output/50.visualization.mvi`.

Displaying MATSim Visualization Snapshots (mvi-files)

Just pass the file as first argument. Example call:

```
java -cp MATSim.jar org.matsim.run.OTFVis output/0.visualization.mvi
```

Displaying TRANSIMS Vehicle files

For reasons of backward compatibility, OTFVis can display vehicles files traditionally generated by TRANSIMS. As the vehicle file does not include any network information, the network must be passed as well. Example call:

```
java -cp MATSim.jar org.matsim.run.OTFVis output/0.T.veh
input/network.xml.gz
```

Displaying MATSim Network files

OTFVis can ~~display just a network~~. This is useful when building a scenario, and a network converted from other data must be inspected. Example call:

```
java -cp MATSim.jar org.matsim.run.OTFVis input/network.xml.gz
```

(Note: Currently only available in Nightly Builds since revision r5821)

Start Interactive Simulation

OTFVis can directly start a simulation and visualize it in real time. As in that case, all data (esp. the population) is loaded into memory, interactive queries about agents and link states can be issued from the visualizer. To start OTFVis in this interactive, live mode, just pass it the config-file you would otherwise pass to the Controler:

```
java -cp MATSim.jar org.matsim.run.OTFVis input/config.xml
```

Please note that this will require even more resources (memory, cpu-speed) than only running the simulation with the Controler.

Running ~~OTFVis~~ from within a windows systems

As shown by the Nightly Builds tutorial OTFVis and other classes can be run by using the command line or a shell script respectively. As the Unix based way is already described by the tutorial, this is about the windows user.

The windows command line call looks similiar to the Unix based one. Finally, you should end with something like that

```
java -Xmx1500m -Djava.library.path=libs/jogl-1.1.1/jogl-1.1.1-windows-
i586/lib -cp MATSim_r6508.jar org.matsim.run.OTFVis %*
```

which can be saved as a *.bat file, e.g. otfvis.bat. Please note that the example is based on the assumption that otfvis.bat is saved in the same folder as the matsim.jar and the libs folder. The placeholder %* will be substitued by the parameters you've specified when calling otfvis.bat from the command line, e.g.

```
otfvis.bat -convert event-file network-file mvi-file
```

To call the OTFVis from any folder, put the otfvis.bat into your PATH environment.

If your are more familiar with the point and click behaviour of win systems, you can create a shortcut pointing to your otfvis.bat.

1. By putting it on your desktop, you can drop any file on it, to call OTFVis with the file dropped, e.g. a network.

2. Move the shortcut to your SendTo folder and rename it to something like OTFVis.lnk. Depending on the system you use, the SendTo folder should be located in your home directory. Now you can start the OTFVis by rightclicking at any file within

your system, e.g. rightclick a mvi-file, from the context menu select SendTo -> OTFVis.

Using senozon via

See here on the senozon website.

On the relation between senozon (commercial), senzon via (commercial), MATSim (open source) and MATSim OTFVis (open source):

- Historically, MATSim is open source. An important reason for this was that multiple teams contribute, and we wanted to make progress rather than sorting out the intellectual property.
- However, this community is unable to provide support for any and all requests that may come up. As a result, the commercial company senozon was founded, which provides commercial support for such situations.
- Senozon also helps significantly with the development and maintenance of the MATSim core. The open source community and senozon have a shared interest in a functional and robust MATSim core: Both our academic research and the senozon commercial success depend on this.
- In addition, senozon has developed the MATSim visualization and analysis software via. OTFVis remains available but maintenance is limited. In particular, please understand that we are unable to provide support for specific hardware configurations or specific query requests.

Events analysis

In order to write MATSim events handlers, some amount of Java programming is necessary. Material can thus found in the api-users section of the documentation, see here.

Using MATSim Extensions

Introduction

The default MATSim releases contain all the functionality typically used to model agent behavior and simulate traffic. But sometimes, this just is not enough. The MATSim Extensions provide additional functionality for specific tasks, and can be used along MATSim. MATSim Extensions gives an overview of the currently available extensions. Please note that these extensions are usually provided and maintained by single persons from the community, and thus long-term support may vary from the default MATSim release.

Downloading Extensions

All extensions come as a compressed zip-file. You can either download the last stable release of an extension to be used together with the stable release of MATSim, or you can download a so-called "nightly build"—an automatically created, but untested and probably unstable version of the extension.

- You can download the stable releases of extensions from SourceForge.
- Likely unstable nightly builds can be downloaded from our nightly builds directory.
- Make sure to also download MATSim itself. The extensions cannot be used without MATSim.

Using Extensions on the Command Line

Once you've downloaded an extension and MATSim, unzip the extension and place the extension's directory inside the MATSim directory, next to the `libs` directory. The file/directory structure should look similar to the following example:

```
matsim/
+ MATSim.jar
+ libs/
| + <lots of .jar files>
+ extension1/
| + extension1.jar
+extension2/
| + extension2.jar
| + libs/       <-- not all extensions contain additional libs
| | + <one or more .jar files>
```

Then, start your simulation with the extension.jar-file on the classpath along the MATSim jar-file, e.g:

```
java -Xmx512m -cp
MATSim.jar:extension1/extension1.jar:extension2/extension2.jar
org.matsim.run.Controler myConfig.xml
```

On Windows, use `;` instead of `:` to separate the different jar-files.

Using Extensions in Eclipse

Unzip the downloaded extension and place the extension's directory in your eclipse project. Then, add the extension's jar-file to the `Java Build Path` in Eclipse's `Project Settings`.

Documentation about Specific Extensions

Extensions are developed and documented by their maintainers. Not all extensions are listed below; see the list of available extensions for their description and documentation.

GTFS2TransitSchedule

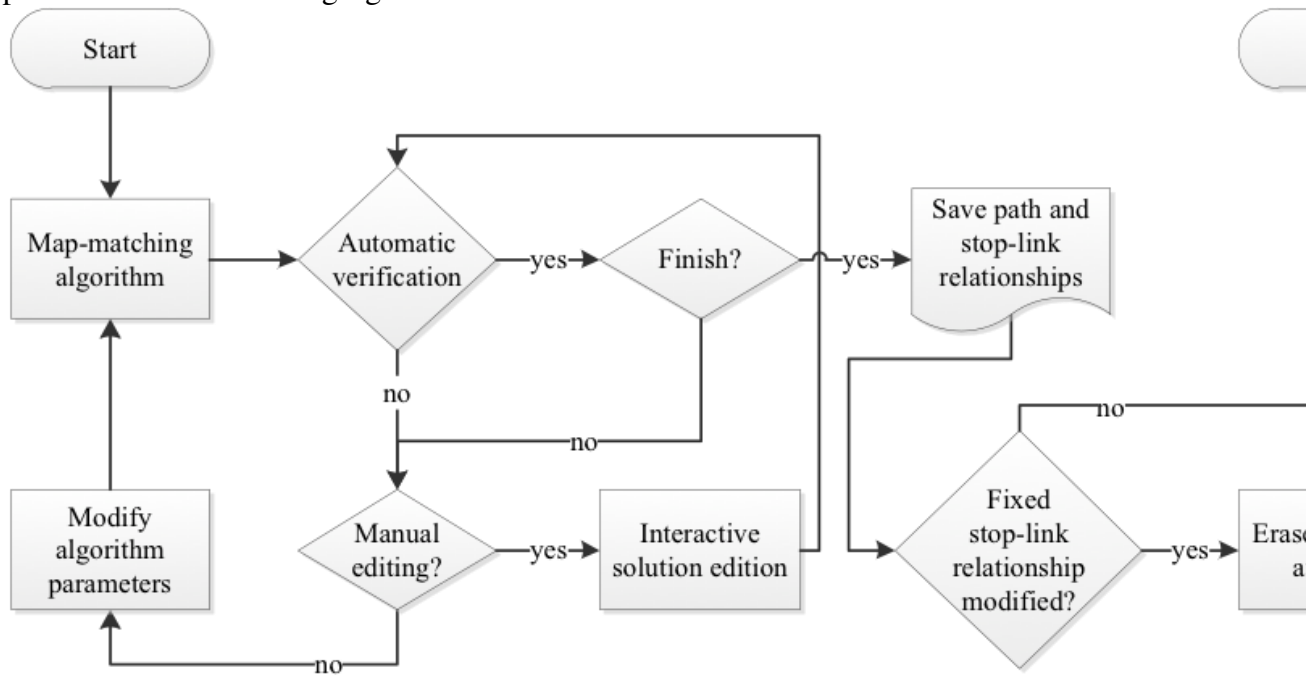
This guide whill show you how to convert GTFS data to a MATSim Transit Schedule

Automatic conversion

- Put the set of GTFS files of each public transport system in a different folder of your file system.
- Create a java program that constructs an object of the class `GGTFS2MATSimTransitSchedule` in the package `GTFS2PTSchedule` which is part of this extension. For this you need to specify:

1. An array of folders (`File` class) where your public transport system specifications are located.
2. An array of Strings representing the network modes correspondent to each public transport defined in b) (e. g. "`car`" for buses, "`rail`" for metro).
3. The MATSim network object with the nodes in latitude and longitude coordinates (`WGS84`).
4. An array of Strings with the names of the calendar services that are desired (e. g. "`weekday`", "`daily`"). Remember that MATSim only simulates one day, but the GTFS files specify routes for many calendar days or dates.
5. The desired output coordinates system

- Call the method `TransitSchedule getTransitSchedule()`. Then, each route of each given public transit systems will be processed with the semi-automatic procedure presented in the following figure.



Manual correction of automatic conversion

For the manual editing process one can visualize, edit and verify the solution for each route:

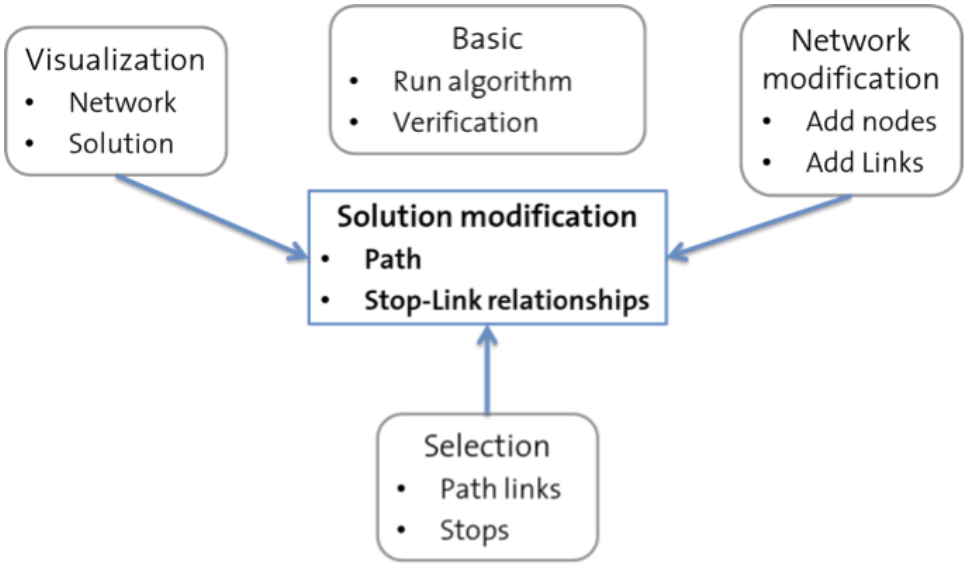
- Visualization: A navigation network is displayed, including all relevant information for working with one single route. This includes the route's profile, the given sequence of GPS points and its current solution (path and stop-link relationships). Selected

elements are drawn in a different color. All is displayed in a bi-dimensional interactive way with refresh of the cursor location in the working coordinates, and panning, zoom and view-all options.

- Selection: Different options for selecting elements of the solution or elements from the network are provided. It is possible to select the nearest link (solution or network), nearest node (network) or nearest stop (solution) to a point indicated by the user. ~~When~~ a stop that has a stop-link relationship already, the corresponding link is selected as well. If a link of the solution path is selected and it does not have a subsequent link connected, a new one from the network is selected with one click; the selected link is the one with the most similar angle than the line defined by the end node of the initial link and a point indicated by the user.
- Solution modification: The first link of the sequence can be added selecting any link of the network. If a link of the solution path does not have a subsequent link connected, it is possible to add one according to the selection function described in (b). If there are two subsequent links in the solution that are not connected (a gap), a subsequence that connects the mentioned links is added, using the shortest path algorithm, with the current parameters. Furthermore, selecting one link of the path, it is possible to delete it, or to delete all the links before or after it. Finally, stop-link relationships can be modified selecting both elements. If the modified relationship was fixed, the user is prevented because the tool erase the solutions of the routes to which the selected stop belongs.
- Network modification: New nodes to the road network can be added. In addition, with any node selected, it is possible to add a new link selecting the end node.

Hints and interaction details:

- It is necessary to pass the verification process ("Is OK") for saving a route result
- The routes are saved in temporal files located in the ./data/paths/ folder relative to the program location.
- Panning and zoom functions are provided dragging the mouse and moving the mouse wheel.
- View all function is provided typing the "v" key
- Up and down keys allow to select the next or previous link of the path.
- "<" or ">" keys allow to select the previous or next stop



Saving the converted data

Finally, after the semi-automatic process, the Transit Schedule object is returned and the network object is modified (splitting, new nodes and links, and modes of the links). One can save in a XML the `TransitSchedule` object constructing a `TransitScheduleWriter` object, and the modified network with a `NetworkWriter`.

MATSim4UrbanSim

Guide on UrbanSim usage of the travel model plug-in

The current travel model plug-in implementation is applicable for the Brussels zone, Zurich parcel, PSRC (Puget Sound Region Council) parcel and Seattle parcel UrbanSim application.

Note that some of the instructions may change since the travel model plug-in is still under development.

1 Prerequisites

You must have installed UrbanSim, before getting started with the travel model plug-in. The following provides an entry point to install UrbanSim and continues with installation instructions for additional software packages required by the travel model plug-in.

Hints for installing UrbanSim

To install UrbanSim please follow the UrbanSim Downloads and Installation Instructions on http://urbansim.org/Download/.

When installing OPUS and UrbanSim manually you can get the installation instructions in "Downloading Sample Data and Source Code" on: http://urbansim.org/Download/DownloadingSampleDataAndSourceCode.

Windows users using the installer are getting the source code and data automatically.

Finally make sure that all UrbanSim environment variables, meaning OPUS_HOME, OPUS_DATA_PATH and PYTHONPATH, are set as described in the installation instructions, see http://urbansim.org/Download/SixtyFourBitMachines for Windows, http://urbansim.org/Download/MacintoshInstallation for Mac or http://urbansim.org/Download/LinuxInstallation for Linux.

Note: For using the travel model plug-in it is sufficient only to install the required Python packages, i. e. numpy, scipy, lxml, sqlalchemy and elixir.

MATSim4UrbanSim prerequisites

In addition to the UrbanSim installation the MATSim travel model plug-in requires the following software installed:

- Java JDK 1.6 or newer: Download and install the newest version of the Java SE Development Kit (JDK) for your operating system from

http://www.oracle.com/technetwork/java/javase/downloads/index.html. Make sure adding Java's /bin directory to the PATH environment variable. Click here to test if java is already installed on your computer.

- Python XML Schema Bindings (PyXB): Download the PyXB 1.1.3 distribution file from http://sourceforge.net/projects/pyxb/files/pyxb/1.1.3/ and extract it to a convenient place.

  Open a command prompt (Windows) or a terminal (Mac, Linux). To install PyXB (this may requires administrator or root privileges) go into the extracted directory and type

  ```
  python setup.py install
  ```

2 MATSim4UrbanSim installation

This section describes how to install MATSim4UrbanSim.

Automatic installation

Make sure to have the Python lib directory included to the PYTHONPATH. This is the directory that contains the site-package directory that is already included in the PYTHONPATH. The lib directory should be something like

`C:\Python2.6\Lib\` for Windows,

`/Library/Frameworks/Python.framework/Versions/2.6/lib/` for Mac or

`/usr/lib/python2.6/` for Linux (Ubuntu).

To install MATSim4UrbanSim open command prompt (Windows) or a terminal (Mac, Linux) and navigate to opus_matsim/configs in the opus source directory. Than type

```
python install_matsim4urbansim.py
```

This creates the subdirectories matsim4opus/jar, loads required MATSim executables and libraries and configures them. After the installation the file/directory structure should look something like Figure 1.

To test whether the MATSim4UrbanSim installation was successful follow the instructions described in Section 2.3.

Note: The installer will replace jar-files and libraries from a previous MATSim4UrbanSim installation.

Manual installation

In case that the automatic installation does not work follow these instructions:

- Create the following directory structure in OPUS_HOME:
  OPUS_HOME/matsim4opus/jar
- Download the following files from http://www.matsim.org/files/builds/ into the jar directory:

    - MATSim_rXXXXX.jar (where XXXXX refers the current revision)
    - MATSim_libs.zip
    - matsim4urbansim-X.X.X-SNAPSHOT-rXXXXX.zip (where XXXXX refers the current revision)
- Rename MATSim_rXXXXX.jar into "matsim.jar".
- Extract the zip files MATSim_libs.zip and matsim4urbansim-X.X.X-SNAPSHOT-rXXXXX.zip. After that the zip files can be removed.
- Rename the directory matsim4urbansim-X.X.X-SNAPSHOT-rXXXXX into "contrib". Than navigate into contrib and rename the jar-file matsim4urbansim-X.X.X-SNAPSHOT.jar into "matsim4urbansim.jar".

Be careful when renaming files or directories (i. e. make sure that everything is written in lower case and check for spelling errors). After the installation the file/directory structure in the matsim4opus directory should look something like Figure 1. To test whether the MATSim4UrbanSim installation was successful follow the instructions described in Section 2.3.

Test your MATSim4UrbanSim installation

To test your installation open a command prompt (Windows) or a terminal (Mac, Linux) and navigate to opus_matsim/tests in the opus source directory (PYTHONPATH). Then type

```
python travel_model_test.py
```

This starts a test scenario. If the test completes without errors, your travel model plug-in should be working.

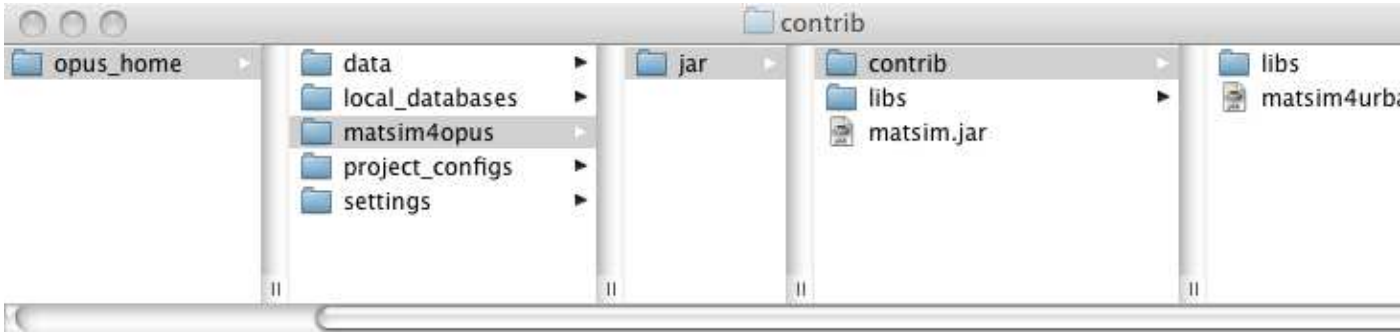


Figure 1: After the installation the matsim4opus directory should contain the depicted files and subdirectories.

3 Using MATSim for UrbanSim

This section aims to explain the MATSim travel model plug-in at the example of the Seattle_parcel scenario.

MATSim Data Requirements

MATSim related input-files are:

- a road network in MATSim format (mandatory)
- a plans file (optional)

These files are not included in the UrbanSim base_year_data and must be added manually. To store these files create the red marked folder structure as shown in Figure 2. This means store network files in the "matsim/network" folder and the plans-files in the "matsim/plans" folder.

For the current Seattle parcel example scenario you can download the zipped matsim folder here. Unzip it into your OPUS_DATA/seattle_parcel/base_year_data/2000/ directory. After that your seattle_parcel base_year_data should look like Figure 2.

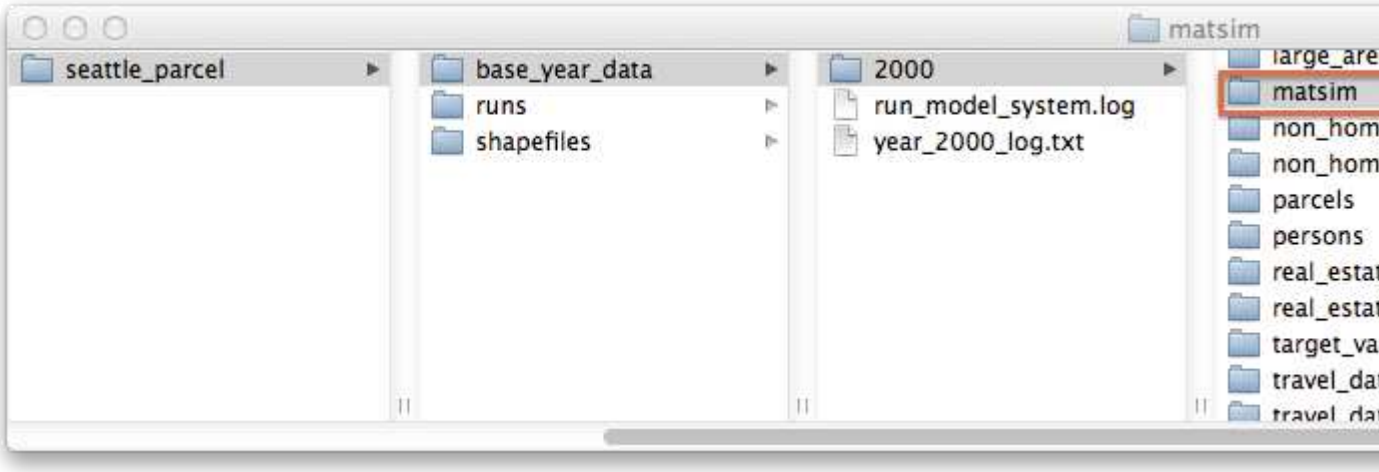


Figure 2: Store MATSim related files like road networks and plans-files directly in the base_year_data folder of the corresponding UrbanSim application.

UrbanSim Data Requirements

In order to create input data for MATSim UrbanSim requires certain data sets and attributes to reflect where a person lives and works.

For UrbanSim parcel applications the following data sets and attributes (in parenthesis) are required:

- persons (person_id, household_id, job_id)
- households (household_id, building_id)
- jobs (job_id, building_id)
- buildings (building_id, parcel_id)
- parcels (parcel_id, x_coord_sp, y_coord_sp, zone_id)
- zones (zone_id)

For UrbanSim zone applications the following data sets and attributes (in parenthesis) are required:

- persons (person_id, household_id, job_id)
- households (household_id, zone_id)
- jobs (job_id, zone_id)
- zones (zone_id, xcoord, ycoord)

Travel Model Configuration Options

In a recent effort a set of MATSim configuration parameters are embedded into the travel_parameter_configuration section of the UrbanSim configuration. This allows to configure MATSim in the OPUS GUI under the Models tab as shown in Figure 3. The travel model conguration section consists of a few lines of XML code and can be added into existing UrbanSim configurations. A sample configuration for Seattle parcel including the travel model configuration section can be downloaded here.

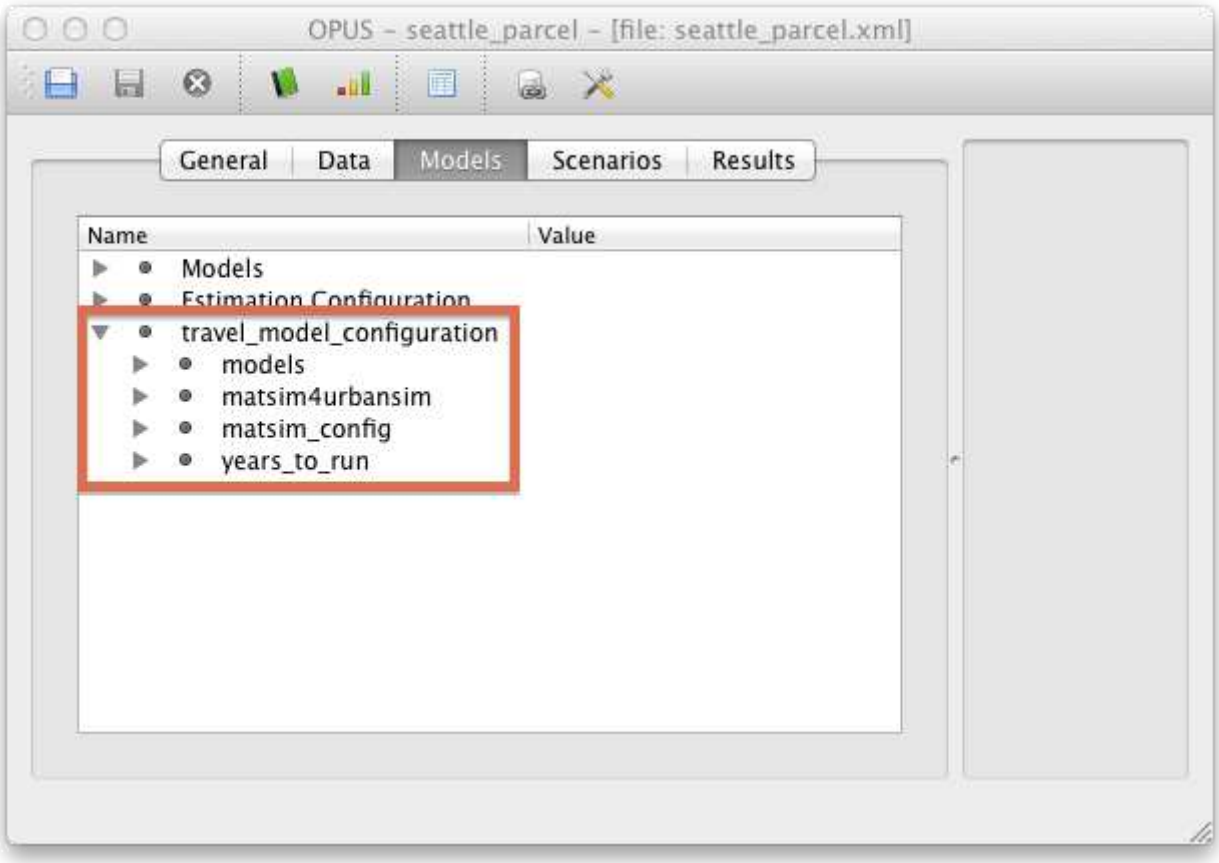


Figure 3: MATSim4UrbanSim configuration in OPUS GUI

The following explains step by step the MATSim configuration options provided by the OPUS GUI.

Launch the OPUS GUI and open the Seattle_parcel sample configuration (download see above). Switch to the Models tab to get to the travel model configuration section as shown in Figure 4. The following subsections are available:

Models:

The models section contains five models that couple MATSim with UrbanSim:

1. **get_cache_data_into_matsim_parcel** generates the MATSim input for UrbanSim parcel applications
2. **get_cache_data_into_matsim_zone** generates the MATSim input for UrbanSim zone applications
3. **run_travel_model_parcel** executes MATSim for UrbanSim parcel applications.
4. **run_travel_model_zone** executes MATSim for UrbanSim zone applications.

5. **get_matsim_data_into_cache** imports the results of the traffic simulation for the next UrbanSim iteration (no distinction between parcel and zone here).

To run MATSim for parcel applications enable the models **1**, **3** and **5**. For UrbanSim zone applications use the models **2, 4** and **5**.

MATSim4UrbanSim:

- **population_sampling_rate**: The population sampling rate determines the percentage of considered travellers for a MATSim run. For instance 0.01 means that only 1% of travellers are considered for the traffic simulation. This option allows to speed up computations on the MATSim side by using low sampling rates, e.g. for testing purposes.
  Note that low sampling rates cause some peculiarities in terms of realism. In this situation results are useful for sketch planning only, not for quantitative analysis. Higher sampling rates need more ram, hard drive space and computation time.
- **matsim_controler**: This determines which access- and accessibility measures to perform in MATSim and accordingly which UrbanSim data sets and attributes to update. Following options are ~~are~~ available:
  - **zone_to_zone_impedance**: This returns an origin-destination-matrix (OD-matrix) compromising car (congested and free speed), bicycle and walk travel times at the mornig peak hours and vehicle trips for each pair of zones. The resulting OD-matrix is imported into the "travel_data" data set in UrbanSim.
  - **agent_performance**: The agent performance feedback contains the individual travel performances of MATSim agents including congested car travel times and travel distances for commuting from home to work and back.
    Note: To update the travel performances of all UrbanSim persons use a full population_sample_rate, i.e. the population_sampling_rate must be 1. The resulting values are imported into the "persons" data set in UrbanSim.
  - **zone_based_accessibility**: This measures the accessibility to work places at the zone-level for the modes car (free speed and congested), bicycle and walk. Such accessibility values are attached (or updated) to the zones data set in UrbanSim. The resulting accessibilities are imported into the "zones" data set in UrbanSim.
  - **cell_based_accessibility:** This measures the accessibility to work places at the parcel-level for the modes car (free speed and congested), bicycle and walk. The resulting accessibilities are imported into the "parcels" data set in UrbanSim.
- **controler_parameter**: This section is for configuring the cell_based_accessibility measure:
  - **cell_size**: This parameter sets the cell size (in meter) and thus the resolution of the cell_based_accessibility measure, see Figure 4. Short side lengths lead to higher resolutions, but also to longer computation times.
  - **shape_file** (optional): To speed up accessibility computation the exact shape, i.e. a boundary like in Figure 4, of the study area can be provided as shape file (optional). Make sure that the shape-file is consistent with the UrbanSim coordinates.
  - **bounding_box** (optional): To speed up accessibility computation the study area can be defined by a bounding box giving the most top, left, right and bottom coordinates (optional). Make sure that these are consistent with the UrbanSim coordinates.

To use the bounding box enable the "use_bounding_box" option and put the coordinates into the corresponding fields, e.g. put the most top coordinate into the "baounding_box_top" field.

By default neither a shape file nor a bounding box is needed. In this case MATSim takes the road network to determine the study area, which could need more ram and computation time for accessibility computation.

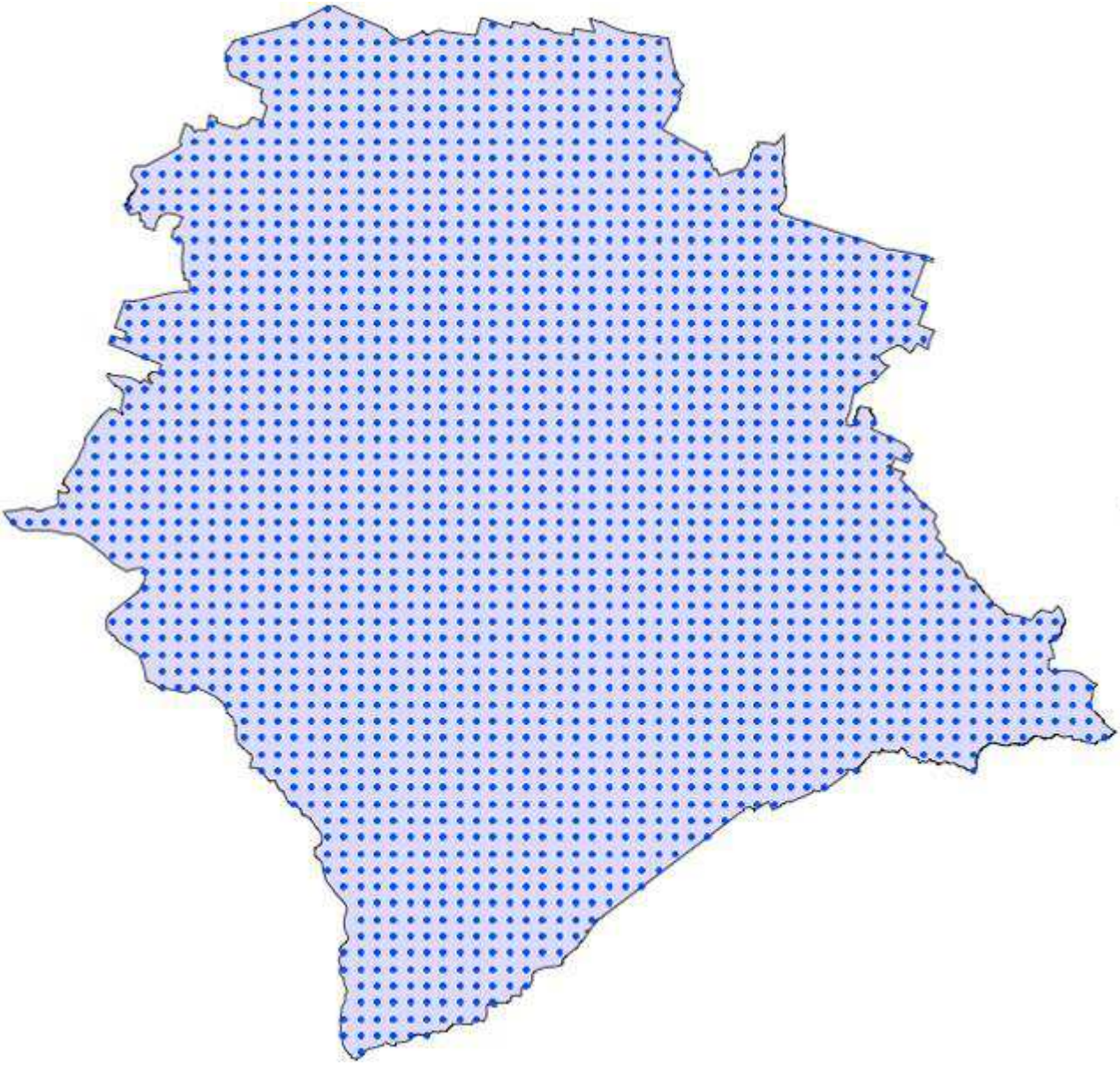


Figure 4: The figure visualizes how the study area (blue area) is subdivided into cells of configurable size by using the "cell_size" parameter. The left illustration has a side length of 200 meter (cell_size=200), the right illustration has a side length of 400 meter (cell_size=400). The blue dots are the corresponding cell centroids, which serve as measuring points for the accessibility computation.

- **accessibility_parameter**: At this point the marginal utilities e.g. for different transport modes can be configured (for calibration instructions see here):

- **accessibility_destination_sampling_rate**: This determines the percentage of considered opportunities, currently work places, for the accessibility computation. A value of 1 is recommended.
- **use_MATSim_parameter**: Enables MATSim default settings for the following parameter:
  - **use_logit_scale_parameter_from_MATSim**: This sets the logit scale parameter to a MATSim default value (currently this is 1.0)
  - **use_car_parameter_from_MATSim**: This sets the marginal utility for traveling by car ($beta_{tt,car}$) to a MATSim default value (currently -12 utils/h). Otherwise the car_parameter settings (see below) are used.
  - **use_walk_parameter_from_MATSim**: This sets the marginal utility for traveling on foot ($beta_{tt,walk}$) to a MATSim default value (currently -12 utils/h). Otherwise the walk_parameter settings (see below) are used.
  - **use_raw_sums_without_ln**: If enabeld the summation of the term $exp(V_{ik})$ is computed, i.e. accessibility is computed as $A_i:=sum_k(exp(V_{ik}))$ for all opportunities k.
- **logit_scale_parameter**: Set a custom value for the logit scale parameter. Make sure that "use_logit_scale_parameter_from_MATSim" is disabled!
- **car_parameter** and **walk_parameter**: This allows to configure the disutility of traveling $V_{ik,mode}$ for a given mode (car, bicycle, walk) and thus to configure the accessibility measurement. $V_{ik,mode}$ is composed as follows:

  $V_{ik,mode}:= V_{ik,tt,mode} + V_{ik,tt^2,mode} + V_{ik,ln(tt),mode} + V_{ik,td,mode} + V_{ik,td^2,mode} + V_{ik,ln(td),mode} + V_{ik,m,mode} + V_{ik,m^2,mode} + V_{ik,ln(m),mode}$

  where "tt" are travel times, "td" are travel distances and "m" are monetary costs.

  Each summand $V_{ik,xx,mode}$ consists of the following contributions, see Figure 5:
  1. The disutility of travel of reaching the transport network from origin *i*. It is assumed that opprotunities (e.g. work places) can only be reached via the transport network.
  2. The disutiliy of travel on the network towards opportunity *k*
  3. The disutility of travel reaching opportunity *k* from the transport network

As a result the disutility $V_{ik,xx,mode}$ is composed as follows:

$V_{ik,xx,mode}:= beta_{xx,wlk} * xx_{wlk,gap,i} + beta_{xx,mode} * XX_{mode} + beta_{xx,wlk} * xx_{wlk,gap,k}$

where xx refers to the travel costs (tt, tt^2,ln(tt), td, td^2,ln(td), m, m^2,ln(m)) and $beta_{xx,wlk}$ and $beta_{,xx,mode}$ are marginal utilities that convert the given travel cost into utils.

Setting the marginal utility $beta_{xx}$to zero removes the corresponding summand from the equation. In order to use your own $V_{ik,mode}$ make sure that the corresponding switches ("use_car_parameter_from_MATSim" and/or "use_walk_parameter_from_MATSim") are disabled.

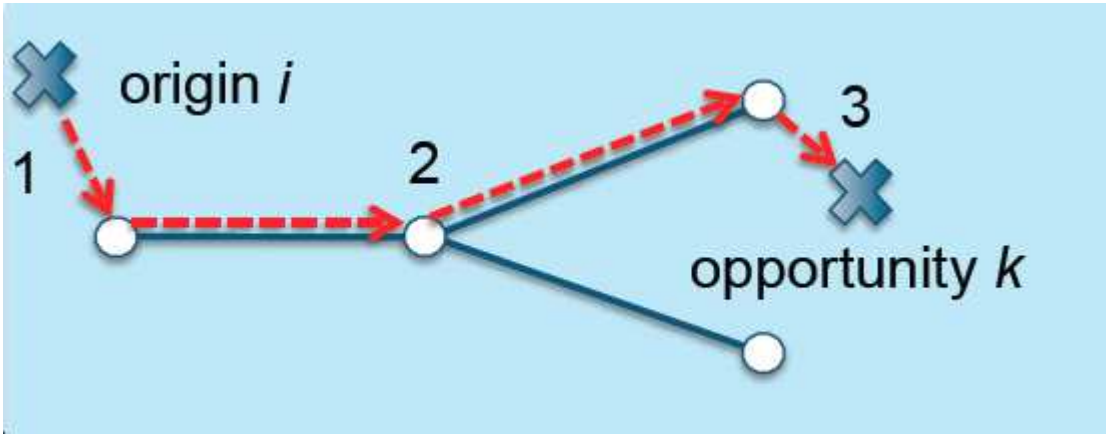


Figure 5: The composition of the disutility $V_{ik,xx,mode}$ consists of three parts: the cost (1) to reach network from i, (2) the cost on the network and (3) the costs to reach the opportunity *k* from the network.

- **random_location_distribution_radius_for urbansim_zone**: This option is only relevant for UrbanSim zone applications. It randomly distributes persons living in a certain zone within a given radius (in meter) around the zone centroid. See also section 4 "Additional MATSim4UrbanSim Parameters" for an alternative distribution of persons.

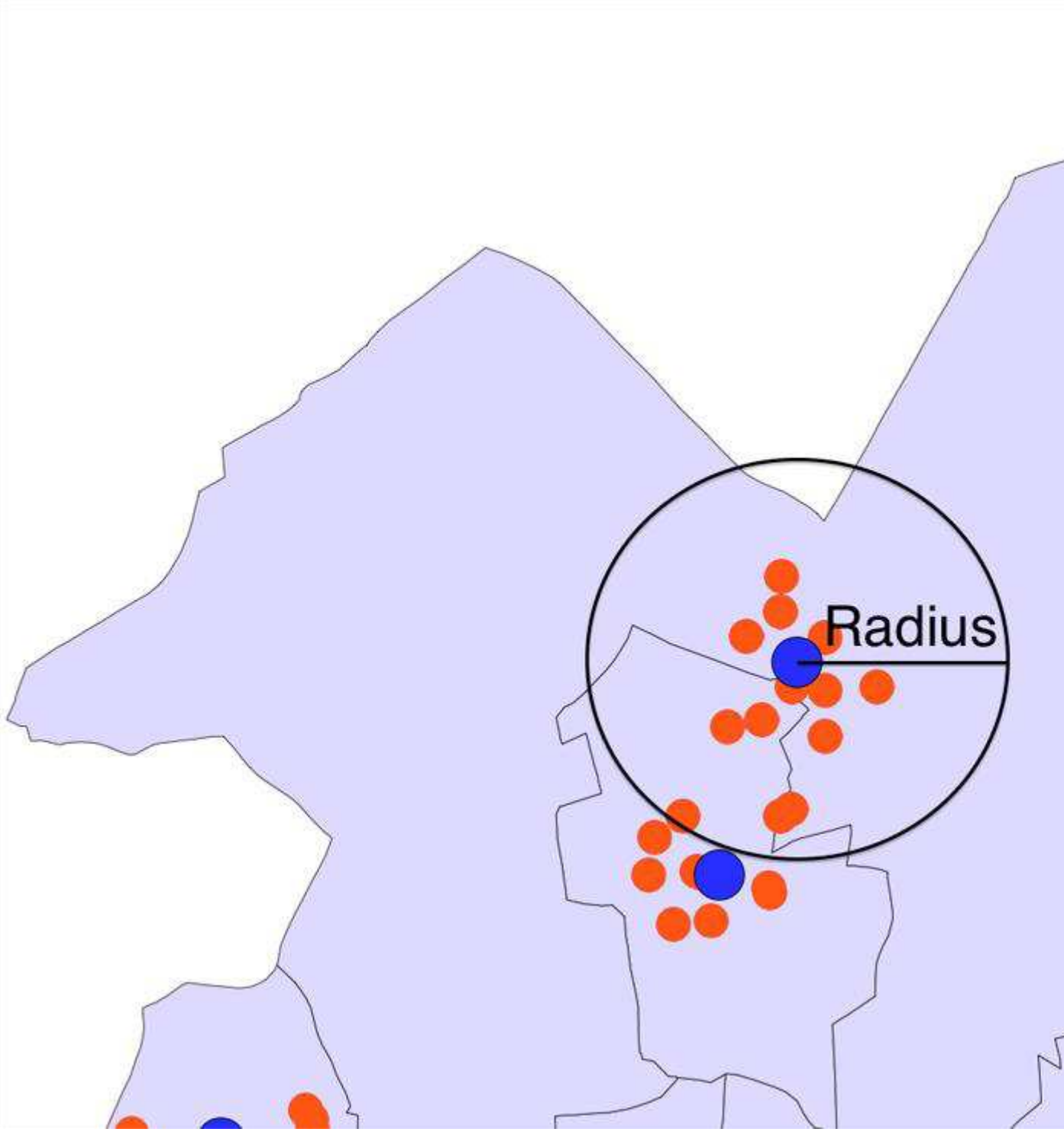


Figure 6: The random_location_distribution_radius_ for_urbansim_zone parameter randomly distributes persons (red dots) living in a certain zone within a given radius around the zone centroid (blue dot).

MATSim Config:

This section contains parameters for the configuration of the traffic model.

- **common**: The common subsection provides some basic configuration options for MATSim

- **external_MATSim_config**: This allows to integrate an external MATSim configuration for a specified UrbanSim year by setting the relative path to the separate configuration file, which must be located in the OPUS HOME directory.
  Note that the following parameter settings from the external conguration are overwritten: population sampling rate, network, last iteration, input plans file, plan calc score and strategy parameters. To use the network from the external conguration leave the matsim_network_file field empty (next item).

  In order to set an external configuration file add the following lines in the "travel_model_configuration > matsim_config > common" section:
  - `<external_matsim_config type="dictionary">`
  - `  <matsim_config name="2001" type="file">ralative_path/to/external_matsim_config.xml</matsim_config>`
  - `</external_matsim_config>`
  - **matsim_network_file**: This points, as the name implies, to a road network in MATSim format. A relative path (located in the OPUS_HOME directory) to the network file is expected.
  - **last_iteration**: This gives the number of MATSim iterations to perform. In MATSim iterations start at zero.
  - **input_start_plans_file**: This gives the path to a “relaxed” plans file from which MATSim starts ("warm start"). It allows MATSim to recycle agent decisions like route and departure times from a previous run to speed up computing time. If this is not set, MATSim will construct its initial plans file purely from UrbanSim input, and take much longer to relax ("cold start"). See below (section 5) how to create a plans file.
  - **hot_start_plans_file**: To speed up computing times for traffic simulation it would be desireable to reuse the output plans file of one MATSim run for one UbanSim year as input for a MATSim run of a following UrbanSim year. This describes "hot start" (as opposed to "warm start") in MATSim. At this point one can specify a location where MATSim should store the plans file. If no location is provided but an "input_start_plans_file" is given MATSim has "warm start", otherwise MATSim has a "cold start".
  - **backup**: If enabled the following files are saved after each MATSim run: the MATSim configuration, the final plans file, the MATSim input files for UrbanSim and some output files to visualize accessibility via R. These files are stored using the follwing folder structure: OPUS_HOME/matsim4opus/backup/runXXXX, where XXXX refers to the current UrbanSim simulation year.
- **plan_calc_score**: This specifies the following activity constraints:
  - **home_activity_typical_duration**: Typical home activity duration (in seconds)
  - **work_activity_typical_duration**: Typical work activity duration (in seconds)
  - **work_activity_opening_time**: The earliest time where a work activity can be started (in seconds)in seconds
  - **work_activity_latest_start_time**: The latest time to start a work activity (in seconds)
- **strategy**:
  - **max_agent_plan_memory_size**: This gives the number of plans per agent, where 0 means infinity. A plans size of 5 is recommended.
  - **time_allocation_mutator_probability**: Probability* that an agent obtains new activity starting and end times

- **reroute_dijkstra_probability**: Probability* that an agent obtains a new route
- **change_exp_beta_probability**: Probability* that an agent switches between existing plans

*) despite its name, this really is a "weight"

Years_To_Run:

This defines the years in which the travel model should run. In order to add additional years add the following lines in the "travel_model_configuration > years_to_run" section:

```
<run_description type="directory">
        <year type="integer">2002</year>
</run_description>
```

4 Additional MATSim4UrbanSim Parameters

Some MATSim4UrbanSim configuration parameters are only availabe via the standard MATSim configuration, see Figure 7. A sample MATSim configuration containing ~~olny~~ these relevant parameters for MATSim4UrbanSim can be downloaded here. The following modules and parameters are available:

MATSim4UrbanSimParameter:

This module provides the following parameter:

- **timeOfDay**: Specify the time of day (in seconds) for which the zone2zone impedance matrix and accessibilites should be calculated ~~be calculated~~. By default this is set to 8am (28800 sec).
- **urbanSimZoneShapefileLocationDistribution**: This option is only relevant for UrbanSim zone applications. This randomly distributes persons living in a certain zone within the zone boundaries provided by zone shape file, see Figure 8. Enter the path to a zones shape file here. Note: This deactivates random_location_distribution_radius_for urbansim_zone (see above).
- **usePtStops**: This is a switch to enable a MATSim4UrbanSim specific pseudo pt based on a given csv input file provided at the 'ptStops' parameter (see next).
- **ptStops**: This parameter expects a csv input file providing a pt stop id and a x and y coordinate. The csv files needs a header indicating the cooresponding columns by "id" (for the pt stop id), "x" and "y" for the coordinates. A sample file to illustrate the format can be found here.
- **useTravelTimesAndDistances**: This is a switch to initialize the MASim4UrbanSim specific pseudo pt by the given pt travel times and distances provided at the parameters 'ptTravelTimes' and 'ptTravelDistances' (see next). This requires the 'usePtStops' to be TRUE and a ptStop input file provided at 'ptStops' parameter.
- **ptTravelTimes**: This parameter expects an input file providing an origin and destination ptStop id, which is consistent with the ptStop id provided at 'ptStops', and the corrosponding travel time in minutes. The input file can be in VISUM format (e.g. *.JRT) or just a text file (*.txt) with space separated values in the following order: origin ptStop id, destination ptStop id and travel times in minutes. A sample file illustrating format can be found here.
- **ptTravelDistances**: This parameter expects an input file providing an origin and destination ptStop id, which is consistent with the ptStop id provided at 'ptStops', and

the corrosponding travel distances in meter. The input file can be in VISUM format (e.g. *.JRD) or just a text file (*.txt) with space separated values in the following order: origin ptStop id, destination ptStop id and travel distances in meter. A sample file illustrating format can be found here.

- **betaBikeXXX parameter**: This allows to configure the disutility of traveling for travelling by bicycle. For more information see "car_parameter and walk_parameter" above**.**
- **betaPtXXX parameter**: This allows to configure the disutility of traveling for travelling by pseudo pt. For more information see "car_parameter and walk_parameter" above**.**

ChangeLegMode and Strategy:

A full description for the changeLegMode module is given here. Basically the changeLegMode module defines the transport modes that can be used by MATSim agents. Currently MATSim4UrbanSim supports car, pt, bike (bicycle) and walk. In order to allow MATSim agents to switch between these modes either the "ChangeLegMode" or "ChangeSingleLegMode" module must be set in the strategy module, a comprehensive description is given here.

Note: When using MATSim4UrbanSim make sure that any strategy defined in the standard MATSim configuration has an "index" >= 4 (ModuleProbability_index, Module_index). Otherwise these strategies are overwritten by the strategies that are configurable via the OPUS GUI, see above "strategy".

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE config SYSTEM "http://www.matsim.org/files/dtd/config_v1.dtd">
<config>

<!-- ====================================================================== -->

    <module name="matsim4urbansimParameter" >

        <!-- Specify the time of day (in seconds) for which the zone2zone impedance matrix and access
        <param name="timeOfDay" value="28800" />

        <!-- Set shape file with zones. Activates the location distribution of persons for UrbanSim Z
        <param name="urbanSimZoneShapefileLocationDistribution" value="/full path to/zone.shp" />

        <!-- Boolean. "true": MATSim expects a ptStops input file and runs the pseudo pt based on the
        <param name="usePtStops" value="true" />
        <!-- Set a path to pt stop csv file ingluding columns for "id", "x"- and "y"-coordinate -->
        <param name="ptStops" value="/full path to/ptStopsInputFile.csv" />

        <!-- Boolean. "true": MATSim expects both an input file for ptTravelTimes and ptTravelDistanc
        <param name="useTravelTimesAndDistances" value="true" />
        <!-- Set a path to pt travel times file (in VISUM JRT or TXT format) including columns for fr
        <param name="ptTravelTimes" value="/full path to/ptTravelTimes.JRT" />
        <!-- Set a path to pt travel distances file (in VISUM JRD or TXT format) including columns fo
        <param name="ptTravelDistances" value="/full path ot/ptTravelDistances.JRD" />

        <!-- marginal utilities for bicycle travel times in accessibility calculation -->
        <param name="betaBikeTravelTime" value="-12." />
        <param name="betaBikeTravelTimePower2" value="0." />
        <param name="betaBikeLnTravelTime" value="0." />
        <!-- marginal utilities for bicycle travel distances in accessibility calculation -->
        <param name="betaBikeTravelDistance" value="0." />
        <param name="betaBikeTravelDistancePower2" value="0." />
        <param name="betaBikeLnTravelDistance" value="0." />
        <!-- marginal utilities for bicycle monetary costs in accessibility calculation -->
        <param name="betaBikeTravelCost" value="0." />
        <param name="betaBikeTravelCostPower2" value="0." />
        <param name="betaBikeLnTravelCost" value="0." />

         <!-- marginal utilities for pt travel times in accessibility calculation -->
        <param name="betaPtTravelTime" value="-12." />
        <param name="betaPtTravelTimePower2" value="0." />
        <param name="betaPtLnTravelTime" value="0." />
        <!-- marginal utilities for pt travel distances in accessibility calculation -->
        <param name="betaPtTravelDistance" value="0." />
        <param name="betaPtTravelDistancePower2" value="0." />
        <param name="betaPtLnTravelDistance" value="0." />
        <!-- marginal utilities for pt monetary costs in accessibility calculation -->
        <param name="betaPtTravelCost" value="0." />
        <param name="betaPtTravelCostPower2" value="0." />
        <param name="betaPtLnTravelCost" value="0." />

    </module>

<!-- ====================================================================== -->

    <module name="changeLegMode" >
        <param name="modes" value="car,pt" />
    </module>

<!-- ====================================================================== -->

    <module name="strategy" >

        <!-- probability that a strategy is applied to a given a person.  despite its name, this rea
        <param name="ModuleProbability_4" value="0.1" />

        <!-- name of strategy (if not full class name, resolved in StrategyManagerConfigLoader) -->
        <param name="Module_4" value="ChangeLegMode" />

    </module>

<!-- ====================================================================== -->

</config>
```

Figure 7: Some addidional configuration settings for MATSim4UrbanSim are only available/configurable via the standard MATSim configuration, which are depicted in this illustration.

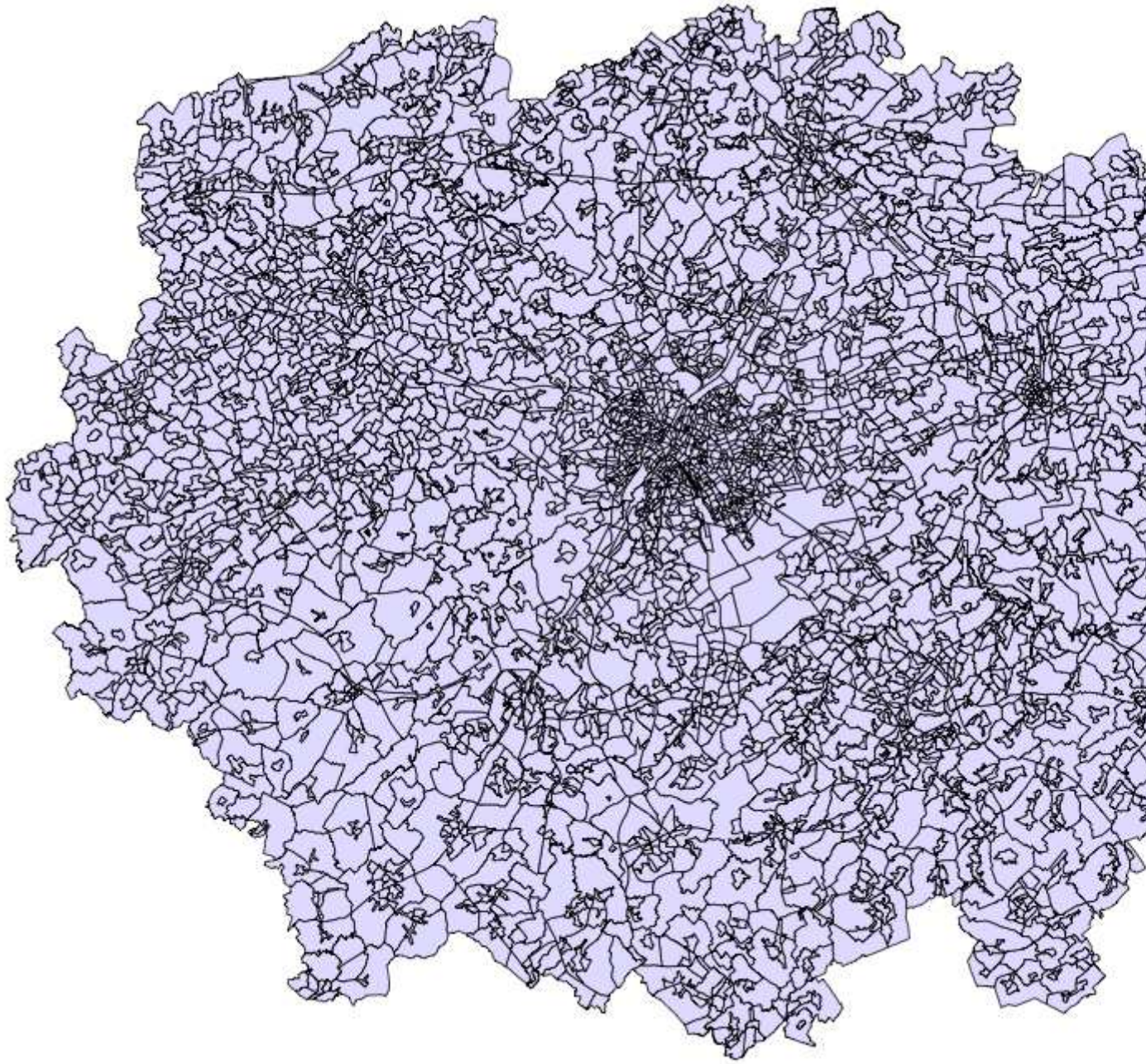


Figure 8: A zone shape file at the example of the greater Brussels area.

5 Create An Initial Plans File

This section describes how to generate an initial plans file for warm start. This example based on the Seattle parcel configuration, which can be downloaded in section 3.

1. Launch the OPUS GUI and open the sample Seattle parcel configuration.
2. Switch to the Models tab and set the last iteration to 100" or higher
3. Note: Leaving the population sample rate at 0.01 will generate a 1% plans file

4. Switch to the Scenarios tab and right-click on "Seattle_baseline" and then on "Run this Scenario" to start the simulation
5. When the simulation finished find the plans file at "OPUS_HOME/matsim4opus/output/ITERS/it.100/100.plans.xml.gz" and move it to a convenient place, e.g. into "OPUS_HOME/data/seattle_parcel/base_year_data/2000/matsim/plans"
6. Finally enter the relative path to the plans-file, i.e. data/seattle_parcel/base_year_data/2000/matsim/plans/100.plans.xml.gz, into the **input_start_plans_file** field in the OPUS GUI under "travel model conguration > matsim_config > common" to start MATSim in warm start mode next time.

6 Travel Model Visualization

There are at least two options to visualize the traffic in MATSim:

1. OTFVis
2. Senozon via

7 Limitations

The Java virtual machine (VM) can't allocate more than 1.5 GB on Windows systems, no matter how much RAM is available in your computer. For this reason the travel model plug-in runs MATSim with 1.5 GB on Windows and with 4 GB on Mac and Linux systems by default. This may cause longer computing times on Windows computers.

MATSim4UrbanSim (~~Frist~~ Release)

The current user guide can be found here !!!

Guide on UrbanSim usage of the travel model plug-in

The current travel model plug-in implementation is applicable for PSRC (Puget Sound Region Council) parcel, Seattle parcel and Zurich parcel scenario.

Note that some of the instructions may change since the travel model plug-in is still under development.

1 Prerequisites

You must have installed UrbanSim, before getting started with the travel model plug-in. The following provides an entry point to install UrbanSim and continues with installation instructions for additional software packages required by the travel model plug-in.

Hints for installing UrbanSim

To install UrbanSim please follow the UrbanSim Downloads and Installation Instructions on http://urbansim.org/Download/.

When installing OPUS and UrbanSim manually (Windows users may require an additional svn client, e. g. totoisesvn, which is available for free at http://tortoisesvn.tigris.org/) please make sure to get the source code for the stable release or the latest development version as described in Downloading Sample Data and Source Code on: http://urbansim.org/Download/DownloadingSampleDataAndSourceCode.

Windows users using the installer are getting the source code and data automatically.

Finally make sure that all UrbanSim environment variables, meaning OPUS_HOME, OPUS_DATA_PATH and PYTHONPATH, are set as described in the installation instructions, see http://urbansim.org/Download/SixtyFourBitMachines for Windows, http://urbansim.org/Download/MacintoshInstallation for Mac or http://urbansim.org/Download/LinuxInstallation for Linux.

Note: For using the travel model plug-in it is sufficient only to install the required Python packages, i. e. numpy, scipy, lxml, sqlalchemy and elixir.

MATSim4UrbanSim prerequisites

In addition to the UrbanSim installation the MATSim travel model plug-in requires the following software installed:

- Java JDK 1.6 or newer: Download and install the newest version of the Java SE Development Kit (JDK) for your operating system from http://www.oracle.com/technetwork/java/javase/downloads/index.html. Make sure adding Java's /bin directory to the PATH environment variable. Click here to test if java is already installed on your computer.
- Python XML Schema Bindings (PyXB): Download the PyXB distribution file from http://sourceforge.net/projects/pyxb/ and extract it to a convenient place. Windows users may use Win-Rar to extract tar or gz files, which is available for free at www.win-rar.com.

  Open a command prompt (Windows) or a terminal (Mac, Linux). To install PyXB (this may requires administrator or root privileges) go into the extracted directory and type

  ```
  python setup.py install
  ```

2 MATSim4UrbanSim installation

This section describes how to install MATSim4UrbanSim.

Automatic installation

Make sure to have the Python lib directory included to the PYTHONPATH. This is the directory that contains the site-package directory that is already included in the PYTHONPATH. The lib directory should be something like

`C:\Python2.6\Lib\` for Windows,

`/Library/Frameworks/Python.framework/Versions/2.6/lib/` for Mac or

`/usr/lib/python2.6/` for Linux (Ubuntu).

To install MATSim4UrbanSim open command prompt (Windows) or a terminal (Mac, Linux) and navigate to opus_matsim/configs in the opus source directory. Than type

```
python install_matsim4urbansim.py
```

This creates the subdirectories matsim4opus/jar, loads required MATSim executables and libraries and configures them. After the installation the file/directory structure should look something like Figure 1.

To test whether the MATSim4UrbanSim installation was successful follow the instructions described in Section 2.3.

Note: The installer will replace jar-files and libraries from a previous MATSim4UrbanSim installation.

Manual installation

In case that the automatic installation does not work follow these instructions:

- Create the following directory structure in OPUS_HOME: OPUS_HOME/matsim4opus/jar
- Download the following files from http://www.matsim.org/files/builds/ into the jar directory:
  - MATSim_rXXXXX.jar (where XXXXX refers the current revision)
  - MATSim_libs.zip
  - matsim4urbansim-X.X.X-SNAPSHOT-rXXXXX.zip (where XXXXX refers the current revision)
- Rename MATSim_rXXXXX.jar into "matsim.jar".
- Extract the zip files MATSim_libs.zip and matsim4urbansim-X.X.X-SNAPSHOT-rXXXXX.zip. After that the zip files can be removed.
- Rename the directory matsim4urbansim-X.X.X-SNAPSHOT-rXXXXX into "contrib". Than navigate into contrib and rename the jar-file matsim4urbansim-X.X.X-SNAPSHOT.jar into "matsim4urbansim.jar".

Be careful when renaming files or directories (i. e. make sure that everything is written in lower case and check for spelling errors). After the installation the file/directory structure in the matsim4opus directory should look something like Figure 1. To test whether the MATSim4UrbanSim installation was successful follow the instructions described in Section 2.3.

Test your MATSim4UrbanSim installation

To test your installation open a command prompt (Windows) or a terminal (Mac, Linux) and navigate to opus_matsim/tests in the opus source directory (PYTHONPATH). Then type

```
python travel_model_test.py
```

This starts a test scenario. If the test completes without errors, your travel model plug-in should be working.

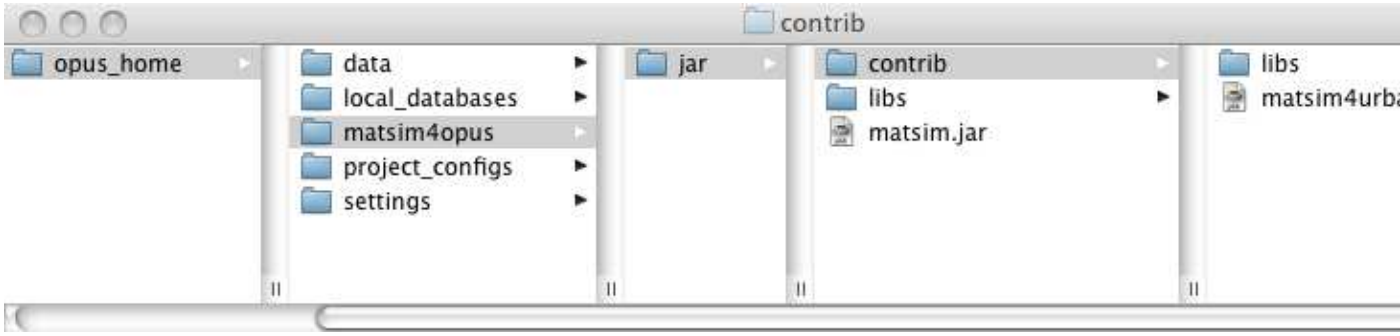


Figure 1: After the installation the matsim4opus directory should contain the depicted files and subdirectories.

3 Using MATSim for UrbanSim

This section aims to explain the MATSim travel model plug-in at the example of the Seattle_parcel scenario. **In order to run Seattle parcel with MATSim the following steps are necessary:** Download the matsim_seattle.zip and unzip the file into your OPUS_DATA/seattle_parcel/base_year/2000/ directory. After that your seattle_parcel base year cache should look like Figure 2. The zip-file contains a road network and a plans-file, used by MATSim. These steps also apply for the PSRC scenario using matsim_psrc.zip file.

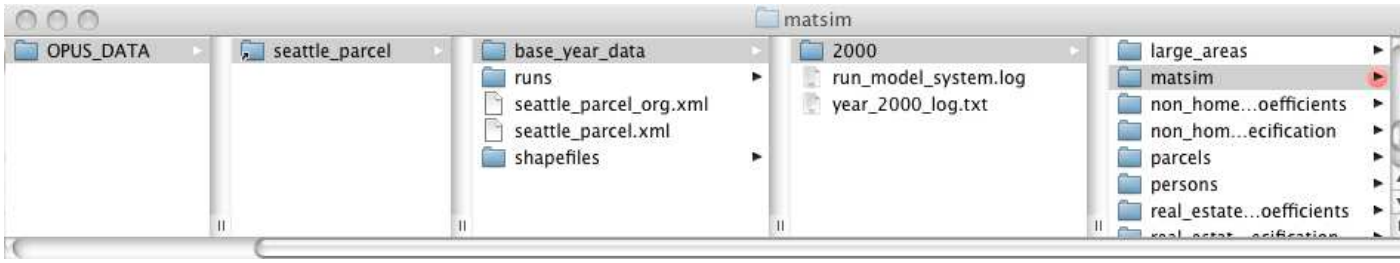


Figure 2: Put the unzipped "matsim" directory, including a road network and a plans-file, into your seattle_parcel base year cache in order to run MATSim as a travel model for the Seattle_parcel scenario.

In a recent effort the MATSim configuration is embedded into UrbanSim. This enables the MATSim (travel model) plug-in in UrbanSim and provides all necessary or basic options to run MATSim via the OPUS GUI. To enable the MATSim plug-in requires the travel model configuration section in the UrbanSim configuration as depicted in Figure 3.

Sample configurations, including the travel model configuration section, can be found for the Seattle_parcel (seattle_parcel.xml) and PSRC_parcel (psrc_parcel.xml) scenario in the opus source directory at opus_matsim/configs.

Windows users will need to replace slashes in paths by backslashes, e.g. replace "data/seattle_parcel/base_year_data/2000/matsim/network/psrc.xml.gz" by "data\seattle_parcel\base_year_data\2000\matsim\network\psrc.xml.gz".

```
<travel_model_configuration type="dictionary">
  <models type="selectable_list">
    <selectable name="opus_matsim.models.get_cache_data_into_matsim" type="selectable">True</selectable>
    <selectable name="opus_matsim.models.run_travel_model" type="selectable">True</selectable>
    <selectable name="opus_matsim.models.get_matsim_data_into_cache" type="selectable">True</selectable>
  </models>
  <matsim4urbansim type="dictionary">
    <sampling_rate type="float">0.01</sampling_rate>
  </matsim4urbansim>
  <matsim_config type="dictionary">
    <common type="dictionary">
      <matsim_network_file type="file">data/seattle_parcel/base_year_data/2000/matsim/network/psrc.xml.gz</mats
      <last_iteration type="integer">1</last_iteration>
    </common>
  </matsim_config>
  <years_to_run key_name="year" type="category_with_special_keys">
    <run_description type="dictionary">
      <year type="integer">2001</year>
    </run_description>
  </years_to_run>
</travel_model_configuration>
```

Figure 3: Adding the travel_model_configuration section into an UrbanSim configuration enables the MATSim plug-in and configuration options within OPUS GUI.

Travel Model configuration options

This sections explains step by step the MATSim configuration options provided by the travel model plug-in.

Launch the OPUS GUI and open the Seattle_parcel sample configuration located at opus_matsim/config/seattle_parcel.xml in opus source directory. Switch to the Models tab to get to the travel model configuration section as shown in Figure 4. The following options are available:

- **Models**: The models section contains three models integrating MATSim into UrbanSim:
  - Get_cache_data_into_matsim generates input data for MATSim and stores it a specified location.
  - Run_travel_model executes MATSim.
  - Get_matsim_data_into_cache imports the results of the traffic simulation for the next UrbanSim iteration.
  - By default all models are enabled. Only disable models if you know what you are doing.

- **MATSim4UrbanSim**: This section contains options concerning the interaction of both simulation models MATSim and UrbanSim.
  - The sampling_rate determines the percentage of considered travellers for a MATSim run. 0.01 means that only one percent of travelers are considered for the traffic simulation. This option allows to speed up computations on the MATSim side, e. g. during testing a scenario.
  - Note that low sampling rates cause some peculiarities in terms of realism. In this situation results are useful for sketch planning only, not for quantitative analysis. Higher sampling rates need more ram and hard drive space.
- **MATSim Config**: The common subsection provides some basic configuration option for MATSim.

    - The matsim_network_file points, as the name implies, to a road network in MATSim format. This expects a relative path to the network file located in the OPUS_HOME directory.
    - Determine the number of MATSim iterations with the item last iteration. In MATSim iterations start at zero.

- **Years_To_Run**: This defines the years in which the travel model should run.

  Adding additional years requires to edit the configuration file directly, e. g. with a xml editor, within the year_to_run section in the travel_model_configuration. Make sure that each year you are adding is surrounded by the run_description tags like this:

```
<run_description type="directory">

        <year type="integer">2002</year>

</run_description>
```

All configuration options can be easily edited in the OPUS GUI by clicking on the value or check box on the right hand side.

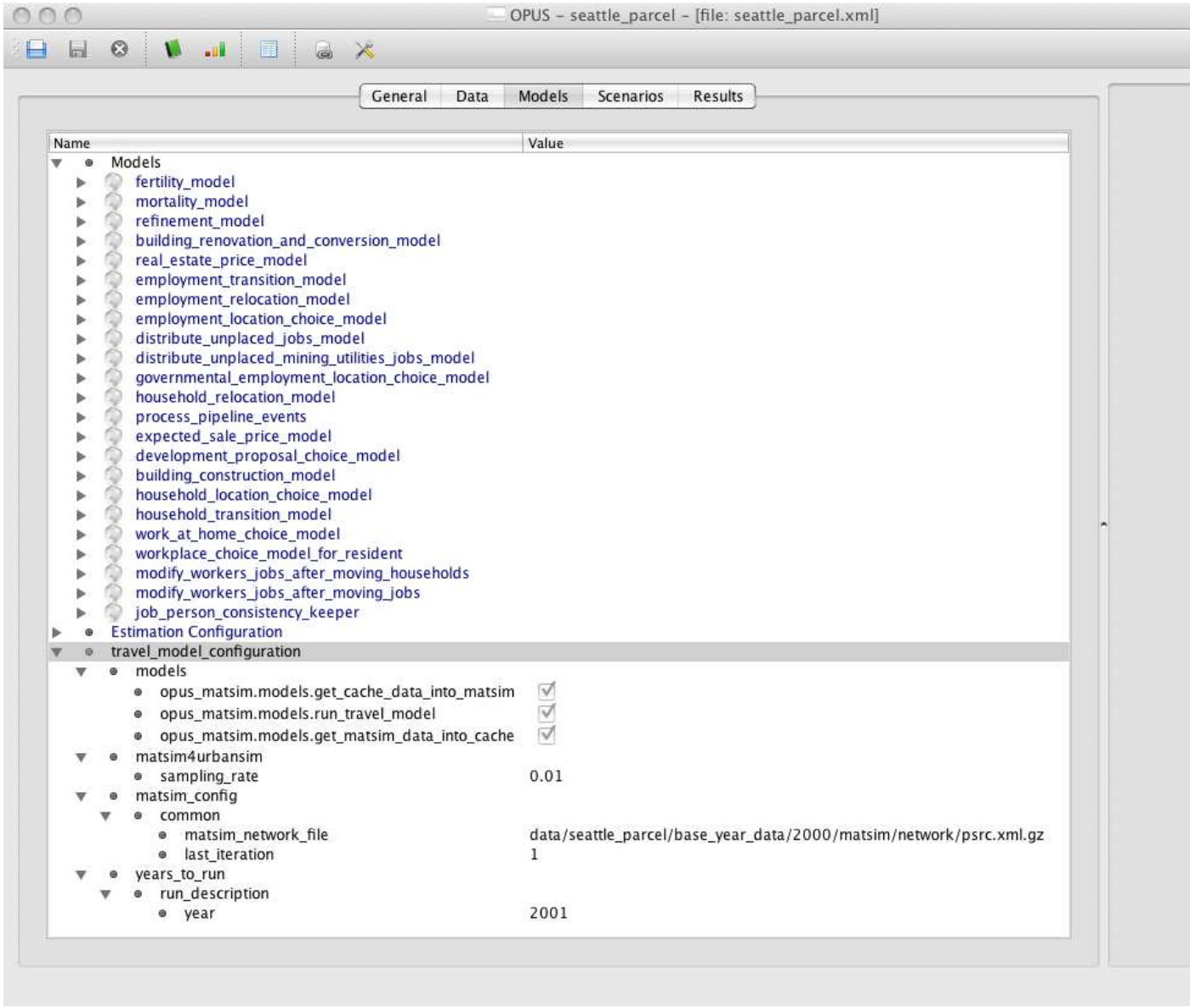


Figure 4: Configuring MATSim via the travel model configuration section in OPUS GUI.

4 Travel Model Visualization

There are at least two options to visualize the traffic in MATSim:

1. OTFVis
2. Senozon via

5 Limitations

The Java virtual machine (VM) can't allocate more than 1.5 GB on Windows systems, no matter how much RAM is available in your computer. For this reason the travel model plug-in runs MATSim with 1.5 GB on Windows and with 2 GB on Mac and Linux systems by default. This may cause longer computing times on Windows computers.

The MATSim network for the Brussels application

Note:

The current MATSim network for the Brussels case study can be downloaded here! In the following it is described step by step how this network is created by using Open Street Map (OSM).

Step 1:

The network for the Brussels case study is composed of separate OSM networks for Belgium and its bordering regions in the Netherlands, Germany, Luxembourg and France. The following OSM files are used, which are available at http://download.geofabrik.de/osm/europe/:

- alsace.osm.bz2
- belgium.osm.bz2
- champagne-ardenne.osm.bz2
- lorraine.osm.bz2
- luxembourg.osm.bz2
- netherlands.osm.bz2
- nord-pas-de-calais.osm.bz2
- nordrhein-westfalen.osm.bz2
- picardie.osm.bz2
- rheinland-pfalz.osm.bz2
- saarland.osm.bz2

Step 2:

These OSM files are now merged to a coherent OSM network in a two step process using the java command line application Osmosis. A deteiled manual for Osmosis can be found at http://wiki.openstreetmap.org/wiki/Osmosis.

- In the first step each OSM network is treated separately on the command line as follows:

```
osmosis --rx xxx.osm.bz2 --lp interval=60 --bb top=51.671 left=2.177
bottom=49.402 right=6.764 completeWays=yes --tf accept-ways highway=
motorway,motorway_link,trunk,trunk_link,primary,primary_link,secondary
, tertiary,minor,unclassified,residential,living_street --tf reject-
relations --used-node --wx xxx_filtered.osm.bz2
```

  This modifies each network file as follows:

  - The network links and nodes that are located in the area of interest are extracted. This area, defined by a rectangle containing Belgium and its bordering regions.
  - Links and nodes that do not correspond to one of the following road types in OSM, for instance links and nodes belonging railways, are removed: motorway,

trunk, primary, secondary, tertiary, minor, unclassified, residential and living street.

- In the second step, the modified networks are merged on the command line as follows:

```
osmosis --rx xxx_filtered.osm.bz2 --lp interval=30 --m --wx
belgium_incl_borderArea.osm
```

At this point we have one merged OSM network "belgium_incl_borderArea.osm" for our area of interest.

Step 3:

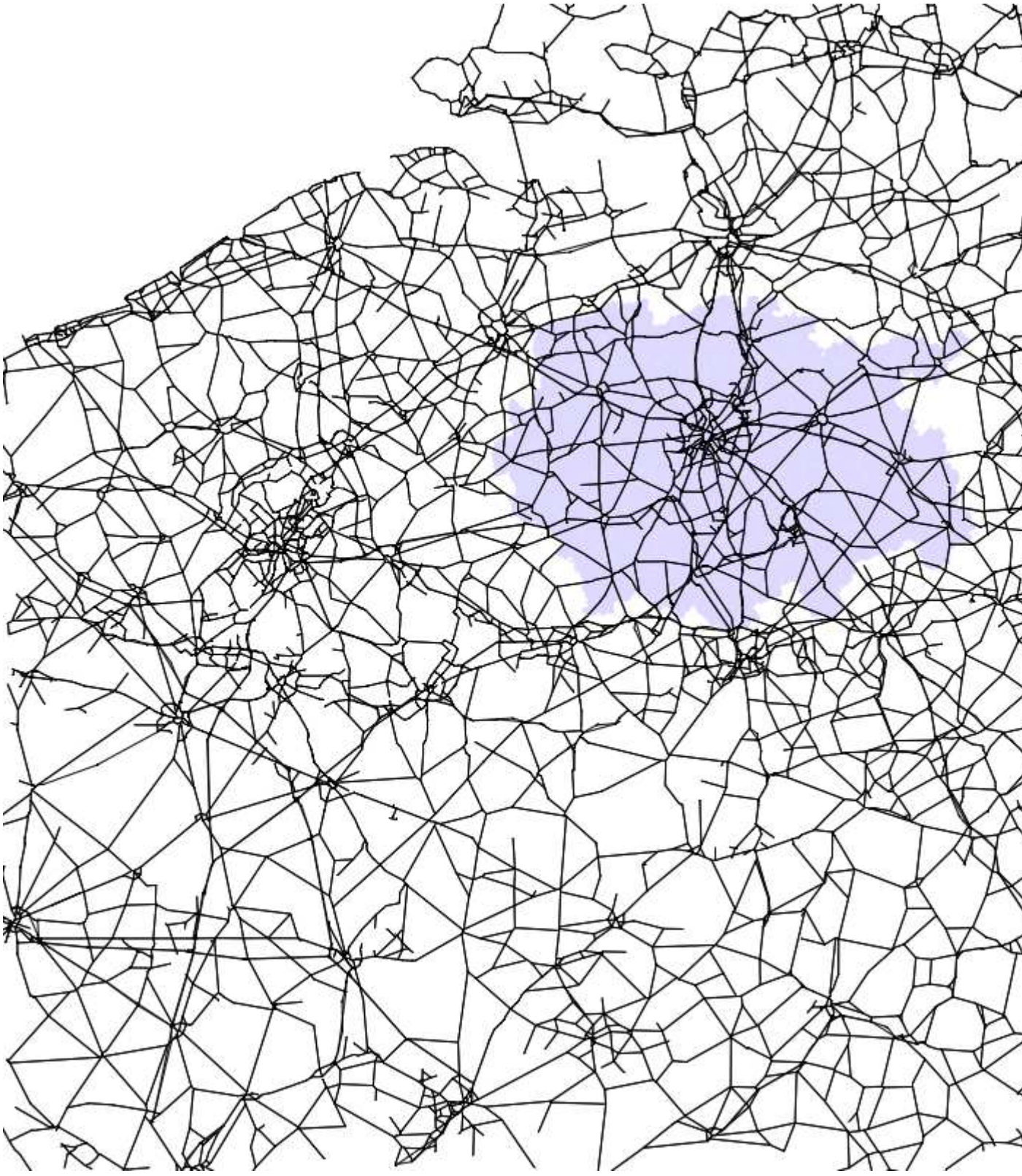
The merged OSM network is converted into MATSim format. To be consistent with the UrbanSim coordinates in the Brussels case study the MATSim default network coordinates are transformed using the *Belge Lambert 72* projection. One hierarchy level is used to avoid side effects of having different network densities within and around the study area. The used hierachy level includes, in OSM terms, links and nodes of secondary roads or greater.

Finally the MATSim network is “cleaned” and “simplified”. Cleaning means, that only links that can be reached by other links are kept in the network. Simplifying means, that a set of links that belong to a road are merged into one single link. The resulting network is depicted below, where the study area is highlighted in blue.

The Java code for the conversion is given in here:

```
Scenario sc = ScenarioUtils.createScenario(ConfigUtils.createConfig());
// creating an empty matsim network
Network network = sc.getNetwork();
// using the Belge Lambert 72 projection for the matsim network
CoordinateTransformation ct = TransformationFactory
                .getCoordinateTransformation(TransformationFactory.WGS84,
                        "EPSG:31300");
OsmNetworkReader osmReader = new OsmNetworkReader(network, ct);

osmReader.setKeepPaths(false);
osmReader.setScaleMaxSpeed(true);

// this layer covers the whole area, Belgium and bordering areas
// including OSM secondary roads or greater
osmReader.setHierarchyLayer(51.671, 2.177, 49.402, 6.764, 4);

// converting the merged OSM network into matsim format
osmReader.parse(INFILE);
new NetworkWriter(network).write(OUTFILE);
// writing out a cleaned matsim network and loading it
// into the scenario
new NetworkCleaner().run(OUTFILE, OUTFILE.split(".xml")[0] +
"_clean.xml.gz");
Scenario scenario = (ScenarioImpl) ScenarioUtils
                .createScenario(ConfigUtils.createConfig());
new MatsimNetworkReader(scenario).readFile(OUTFILE.split(".xml")[0] +
"_clean.xml.gz");
network = (NetworkImpl) scenario.getNetwork();

// simplifying the cleaned network
NetworkSimplifier simplifier = new NetworkSimplifier();
```

```
Set<Integer> nodeTypess2merge = new HashSet<Integer>();
nodeTypess2merge.add(new Integer(4));
nodeTypess2merge.add(new Integer(5));
simplifier.setNodesToMerge(nodeTypess2merge);
simplifier.run(network);
new NetworkWriter(network).write(OUTFILE.split(".xml")[0] +
"_clean_simple.xml.gz");
```



OTFVis

OTFVis is a visualizer for MATSim. It can be used to replay snapshots of simulations, or run a simulation and interact with it. The visualizer makes use of hardware acceleration (OpenGL) and is thus also suitable for visualizing large scenarios. If you have problems running OTFVis, make sure to check your Graphics Card is able to support OTFVis.

Download / Requirements

To use OTFVis, you need MATSim as well as the OTFVis extension. The OTFVis extension is not yet available as an official release, so the following documentation will use a nightly build of it.

- Download a current nightly build of MATSim, the MATSim libraries and OTFVis from our nightly build download page.
- Unzip the MATSim libs
- Unzip the OTFVis Extension

You should now have: the MATSim jar, the libs directory, and the otfvis directory next to each other.

Starting the Visualizer

The main class for the visualizer is `org.matsim.contrib.otfvis.OTFVis`. The different ways to start OTFVis will be described in more details below.

The visualizer may require a lot of memory, it is thus advised to start it with the corresponding Java options, e.g. with 500MB:

```
java -Xmx500m -cp
MATSim.jar:otfvis/otfvis.jar org.matsim.contrib.otfvis.OTFVis arguments
```

If you're on Windows, use `;` instead of `:` to separate the jar files from each other. Also, depending on the version you downloaded, you might have to adapt the file and directory names a little bit.

Creating snapshots (mvi-files) from Events

Use the following arguments:

```
-convert event-file network-file mvi-file [snapshot-period]
```

to record a snapshot of all vehicles' positions every snapshot-period seconds, based on the events and network given in the corresponding files.

Example call:

```
java -cp MATSim.jar:otfvis/otfvis.jar org.matsim.contrib.otfvis.OTFVis -
convert output/50.events.txt.gz input/network.xml.gz
output/50.visualization.mvi 300
```

This will create a snapshot of every 5th minute and store it in the file `output/50.visualization.mvi`.

Displaying MATSim Visualization Snapshots (mvi-files)

Just pass the file as first argument. Example call:

```
java -cp MATSim.jar:otfvis/otfvis.jar org.matsim.contrib.otfvis.OTFVis
output/0.visualization.mvi
```

Displaying TRANSIMS Vehicle files

For reasons of backward compatibility, OTFVis can display vehicles files traditionally generated by TRANSIMS. As the vehicle file does not include any network information, the network must be passed as well. Example call:

```
java -cp MATSim.jar:otfvis/otfvis.jar org.matsim.contrib.otfvis.OTFVis
output/0.T.veh input/network.xml.gz
```

Displaying MATSim Network files

OTFVis can display just a network. This is useful when building a scenario, and a network converted from other data must be inspected. Example call:

```
java -cp MATSim.jar:otfvis/otfvis.jar org.matsim.contrib.otfvis.OTFVis
input/network.xml.gz
```

(Note: Currently only available in Nightly Builds since revision r5821)

Start Interactive Simulation

OTFVis can directly start a simulation and visualize it in real time. As in that case, all data (esp. the population) is loaded into memory, interactive queries about agents and link states can be issued from the visualizer. To start OTFVis in this interactive, live mode, just pass it the config-file you would otherwise pass to the Controler:

```
java -cp MATSim.jar:otfvis/otfvis.jar org.matsim.contrib.otfvis.OTFVis
input/config.xml
```

Please note that this will require even more resources (memory, cpu-speed) than only running the simulation with the Controler.

Running OTFVis from within a windows systems

As shown by the Nightly Builds tutorial OTFVis and other classes can be run by using the command line or a shell script respectively. As the Unix based way is already described by the tutorial, this is about the windows user.

The windows command line call looks similiar to the Unix based one. Finally, you should end with something like that

```
java -Xmx1500m -cp
MATSim.jar:otfvis/otfvis.jar org.matsim.contrib.otfvis.OTFVis %*
```

which can be saved as a *.bat file, e.g. otfvis.bat. Please note that the example is based on the assumption that otfvis.bat is saved in the same folder as the matsim.jar and the libs folder. The placeholder %* will be substitued by the parameters you've specified when calling otfvis.bat from the command line, e.g.

```
otfvis.bat -convert event-file network-file mvi-file
```

To call the OTFVis from any folder, put the otfvis.bat into your PATH environment.

If your are more familiar with the point and click behaviour of win systems, you can create a shortcut pointing to your otfvis.bat.

1. By putting it on your desktop, you can drop any file on it, to call OTFVis with the file dropped, e.g. a network.

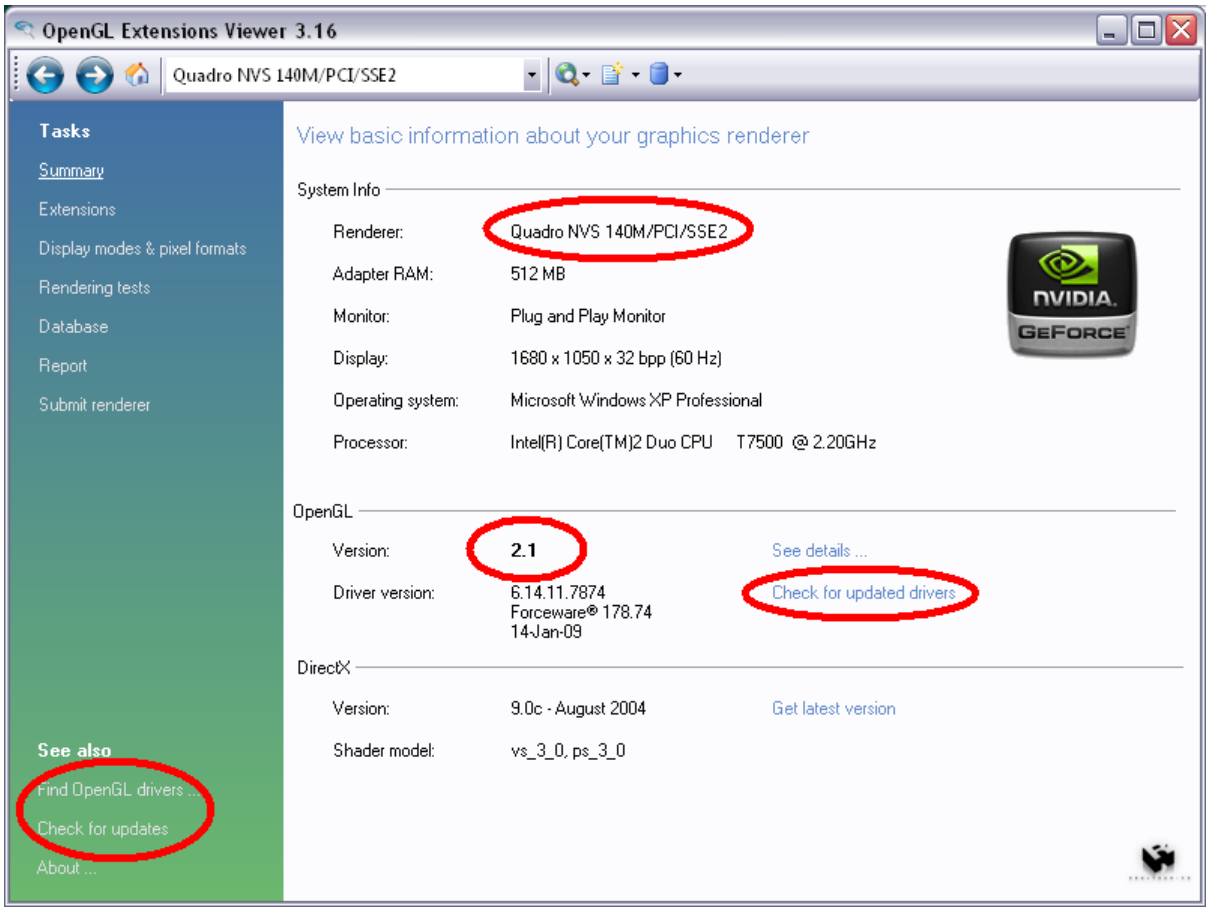
2. Move the shortcut to your SendTo folder and rename it to something like OTFVis.lnk. Depending on the system you use, the SendTo folder should be located in your home directory. Now you can start the OTFVis by rightclicking at any file within your system, e.g. rightclick a mvi-file, from the context menu select SendTo -> OTFVis.

OpenGL Requirements

For the hardware acceleration to work, (i) the OpenGL graphic card driver installed on your machine must be at least of version 2.0 and (ii) native libraries are required, which must be correctly set up.

Check and Update Graphic Card Driver

Either you use check and update mechanims / software already installed (e.g. NVIDIA software, ATI update manager, etc...) or download and install OpenGL Extension Viewer. After starting this little tool, it show all necessary information abour your graphic card including OpenGL version. Please be sure that at least **OpenGL version 2.0** is installed. Otherwise try to find approriate driver updates of your graphic card (the read circles in the ~~F~~Figure below shows the important ~~featrues~~ / information).



Transportation Energy Simulation Framework (transEnergySim)

[module is still under construction]

In this MATSim contribution a framework to simulate a whole range of tranportation related energy scenarios is implemented. The focus is on electric and plug-in hybrid electric vehicles. This contributio is being built and updated as part of the PhD of Rashid A. Waraich (waraich at ivt.baug.ethz.ch). As this is an open source project, which encourages contribution by others, there are also modules, which have been ~~contriubted~~ by the following people: Dr. Matthias D. Galus, Gil Georges ?, Marina ?, Zain?, Raffaela?, etc.?

The following modules should be available soon/ are planned:

- Inductive charging along roads
- Charging at activity location
- Several charging schemes including smart charging ("smart grid")
- V2G
- General energy flow model
- buy/sell of electricity price over market

These features require several basic constructs, which will also be documented soon here:

- vehicle fleet definition model
- vehicle energy consumption model
- charging infrastructure definition model
- output modules
- more to come here...

If you want to contribute with a new module to the framework (e.g. for charging, energy consumption, emissions, etc.), please contact Rashid A. Waraich (waraich at ivt.baug.ethz.ch).

Applications

TODO: Describe ARTEMIS and THELMA project here with figures and references.

TODO: also add work of stella, zain, raffaela, marina, etc. here.

Emissions

Modules for green house gas and other emissions coming soon here.

Hints and Pitfalls

ParallelEventHandling

Many of the modules for keeping track of charging and energy consumption are based on event handlers. In order to avoid raceconditions and accessing the same data unsynchronized (e.g. vehicle trying to charge before the energy consumption is updated), we advise to use the EventHandlerGroup class (or inherit from it, if needed). Using this class, you can clearly control, which thing happens first, e.g. energy consumption updated first and trying to charge happening afterwards (also important for road charging). For an example, see the InductiveChargingController.

At the moment, the it does not seem to be a performance issue, to group several modules together (forming effectivly one event handler). But if there are concerns about this, a synchronized version could be provided in future.

Inductive Charging

Charging along Roads

For Inductive charging along roads the InductiveStreetCharger Module can be used, which is based on event handlers. An example controller, which allows both stationary charging at activitiy locations and charging at roads is called InductiveChargingController. General help

regarding how to configure the controller, can be found in a test of the controller and the documentation of the different modules, which are used in that controller.

(TODO: add links to the code/test cases)

Stationary Inductive Charging

This is currently not distinguished separatly from stationary charging with a plug, although it might be in future.

Stationary Charging

For stationary charing, at the moment the following modules are available:

ChargingUponArrival: Vehicles with a state of charge (SOC) smaller than the "usable battery size" start charging immediatly opon arrival at a location. TODO: descibe, how to filter the location, where vehicle can be charged.

Vehicle Energy Consumption Models

Each vehicle in the vehicle has an Energy consumption model attached to it, based on which vehicle energy consumption is logged for each street. Furthermore for electric and plug-in hybrid electric vehicles (EV/PHEV), this module also updates the state of charge (SOC) of the vehicles.

A couple of models are available for use, of which many have been contributed by the respective authors of the models. If you want to contribute a new model, please drop an email to waraich@ivt.baug.ethz.ch.

Electric Vehicle

PHEV

Galus Model

TODO: also show shape of curve!

Conventional Vehicle

(no model available at the moment)

Visualizations

TODO: emissions map, charging acts, power load per link, etc.

networkEditor

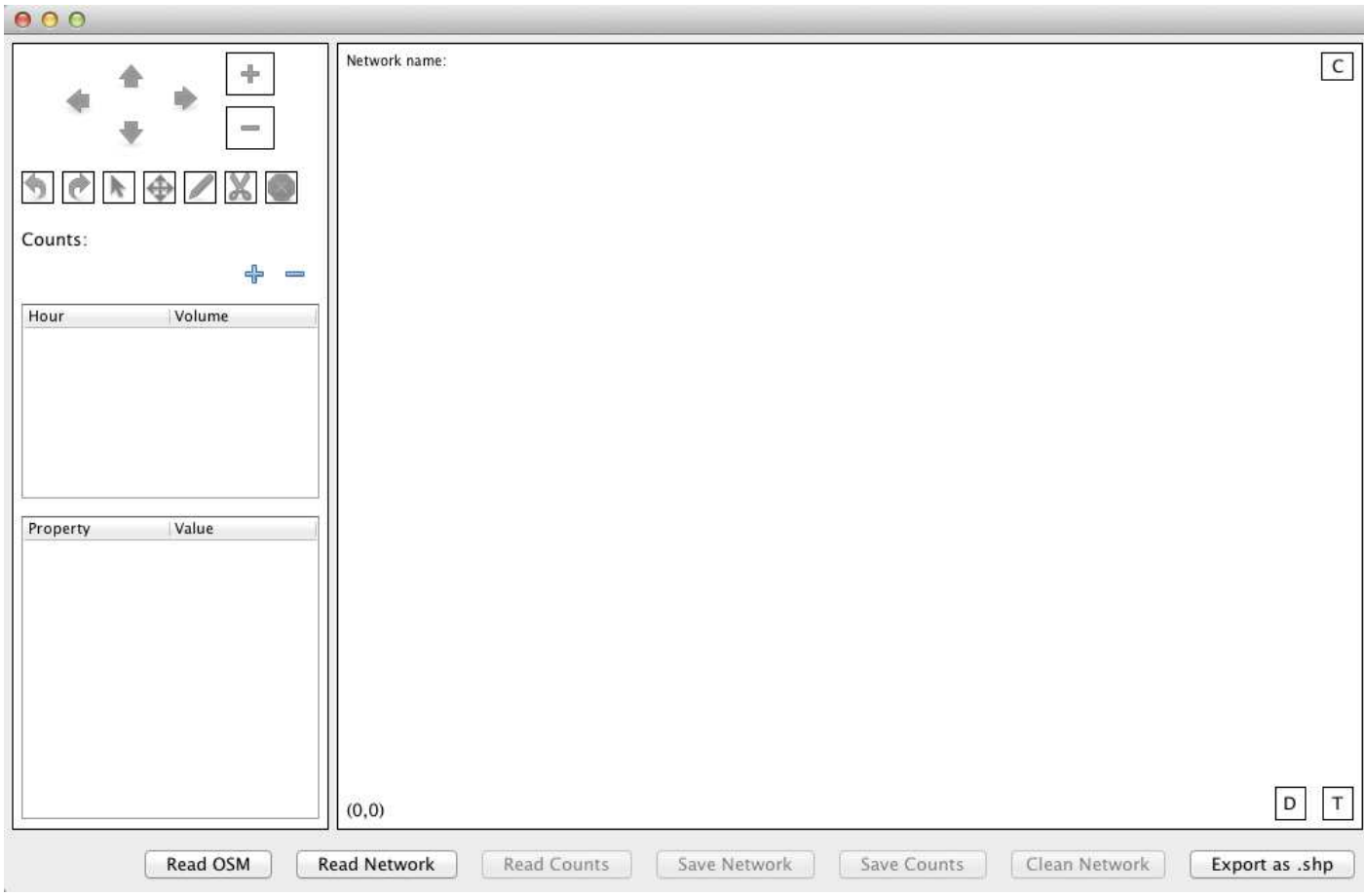
Starting the network editor

If you're working with a release, make sure to have MATSim and the networkEditor extension downloaded.

- **If you use Eclipse to run MATSim**
  Make sure that you have MATSim correctly added to your project's build path. Add the jar file for the networkEditor the same way to your project's build path. Then, start the class `org.matsim.contrib.networkEditor.run.NetworkEditor`.
- **If you run MATSim on a shell / command line**
  Make sure you have MATSim correctly downloaded and ready for use. Add the networkEditor extension next to the MATSim jar file and the `libs` directory. Then, use the following command to start the network editor:
  ```
  java -Xmx512m -cp networkEditor/networkEditor.jar:matsim.jar \
    org.matsim.contrib.networkEditor.run.NetworkEditor
  ```
  Note: On Windows, use `;` instead of `:` to separate the two jar files.

Depending on the size of the network you want to edit, make sure the editor has enough memory by increasing the memory limit (e.g. "`-Xmx1500m`" instead of only "`-Xmx512m`").

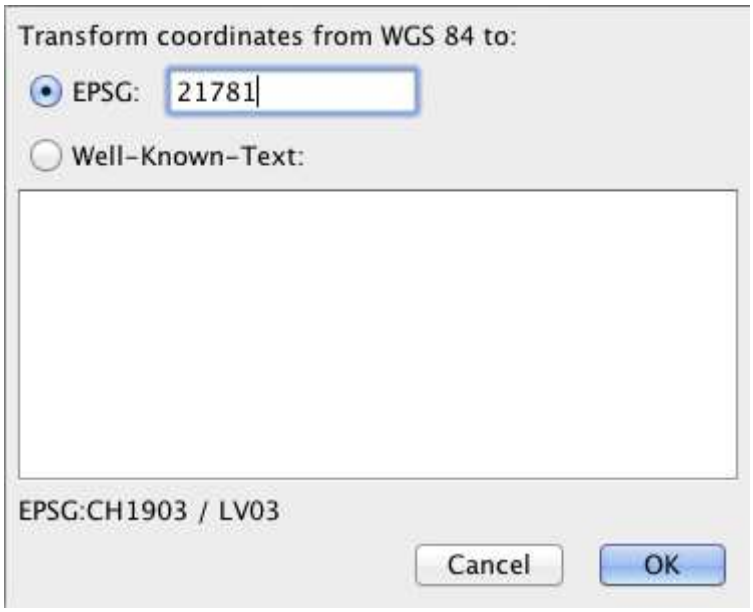
If the application correctly starts, you should see window similar to the one below:



Loading network data

To load an existing MATSim network, click on the button "Read Network" at the bottom of the window and select the network you want to load.

Alternatively, you can import data from OpenStreetMap (OSM) and automatically convert it into a network. For this, first download the osm data for the region you're interested in. The tutorial contains information on how you can download the required data from OSM. Once you have a *.osm file containing the data for your region, click on the button "Read OSM" and select the file. When loading the data, the following window will open:

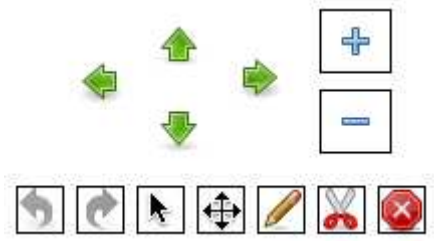


The original data from OSM contains coordinates in the WGS 84 coordinate reference system. WGS 84 is impractical to work with in MATSim, so we need to convert the coordinates into another coordinate reference system (CRS). In the example above, we convert it to the Swiss national coordinate reference system. To find the corresponding EPSG codes, or the Well-Known-Text description of your CRS, have a look at http://spatialreference.org/ and search for your CRS.

Click OK to close the dialog once you have the correct CRS set. The OSM data will now be converted and the network displayed afterwards in the editor. Especially for larger osm files, have some patience for the conversion process. It may take some while to convert the data and finally show the network.

Editing network

Once a network is loaded, most of the tools in the upper left corner of the window become active:



The 4 green arrows are to move the network around in the editor view. The blue + and – are for zooming in and out. In the row below, the buttons have the following functionality:

- undo / redo: revert and redo changes you did to the network
- Selection: select a link or a node by clicking on them in the editor view. click and drag to select multiple elements. If you have a single link selected, you can change some of the link's attributes in the panel on the lower left of the window.
- move view: click and drag the network to move the network in the editor view

- add link: click in a place to start a new link, click a second time to end the link at that place. If the clicks are close to a node, the link will start/end at that node, otherwise a new node will be created.
- divide a link: select a link first, then use the scissors to cut the link into two. A new node will be added at the location of the click, and the original link will be splitted into two parts.
- delete a link: select a link first, then click this button to delete the selected link. If you clicked on the button by accident, use the undo-button.

Saving a network

To save a network in MATSim's format, click on the button "Save Network". Alternatively, you can export the network as a Shape file that can be used by allmost all GIS applications for visualization purposes (but not necessarily for network operations that some GIS applications offer). Just click on "Export as Shp" to export the network as a Shape file.

Working with counts data

Once you have a network loaded, you can optionally also load some existing counts for this network. If you just converted your network from OSM data, you likely won't have any such file. In that case, you can directly start to add counts where you want. Select a link by clicking on it. If the link already has counts associated, they will be displayed in the panel on the left side of the window. Click the + there to add count values, even if no counts exist yet for this link.

After you're done with the counts editing, save the counts to a file by clicking on "Save Counts".

Policy Measures

A discussion of policy measures that can be investigated with matsim is under matsim.org/policy-measures . It is not in the user section but in the developer section of the documentation since, at this point, many of those measures need additional coding. Clearly, something like adding or removing lanes or links can be investigated without any coding.