Integration of agent-based transport and land use models

Thomas W. Nicolai*[1] and Kai Nagel†[1]

[1]Transport Systems Planning and Transport Telematics (VSP), Berlin Institute of Technology (TU Berlin), Berlin, Germany

December 10, 2013

1 Introduction

The interaction between land-use and transport plays an important role as mentioned in Ch. **??** of this Handbook. Land-use transport interaction (LUTI) models denote models that combine land use models and transport models with feedback mechanisms between them. Most urban simulation models such as UrbanSim, DELTA, CUFM, MUSSA, POLIS or RURBAN are not modeling transport themselves; instead they rely on interaction with external transport models (Wegener 2004).

The present chapter focuses on the integration of the extensible, microscopic, agent-based urban land use model UrbanSim with the agent-based transport simulation model MATSim ("Multi-Agent Transport Simulation"; Balmer et al. 2005, Raney & Nagel 2006). To utilize the agent-based modeling technique, both models are directly linked at the person level. The goal is to provide improved access and accessibility indicators as feedback from the travel model in order to enhance UrbanSim sub-models reflecting the decisions of households, businesses, and developers.

A challenge for that software integration was that UrbanSim is written in Python and heavily using C/C++ libraries, while MATSim is written in Java. It is clear that many ways of coupling two pieces of software are technically feasible, but the question was which of these approaches might have a chance to remain robustly functional beyond the end of the project without additional maintenance. Thus, the first objective was to investigate how a *robust* coupling of the two software packages at the agent level could be achieved.

*nicolai@vsp.tu-berlin.de
†Correspondance: nagel@vsp.tu-berlin.de

Travel models typically return zone-to-zone impedance matrices, containing, say, travel time or travel distance, to UrbanSim. A problem with zone-to-zone matrices is that they grow quadratically in the number of zones, which for computational performance reasons puts a limit on the number of zones that can be used and thus on the spatial resolution. Thus, another objective was to investigate if other results of the travel model, in particular accessibility measures, could be computed and provided at the UrbanSim parcel level.

A final question was if the number of iterations, inherent in traffic assignment models and thus also in MATSim, could be reduced by re-using information from previous runs. This is called the "warm" and "hot" start capability.

This chapter first summarizes the simulation approach of MATSim (Sec. 2). Sec. 3 describes MATSim extensions that were used in the case studies. In Sec. 4 the integration approach is described, illustrating the feedback mechanism between both frameworks and describing the data requirements. The access and accessibility indicators provided by MATSim are explained in Sec. 5. Sec. 6 shows, as illustration, the accessibility consequences of a cordon toll for Brussels, obtained during work on the SustainCity Brussels case study. Computational issues are briefly discussed in Sec. 7. The chapter is concluded by a discussion (Sec. 8).

2 MATSim

MATSim (Balmer et al. 2005, Raney & Nagel 2006, Balmer et al. 2009) is a disaggregated, agent-based transport model that is designed to simulate several million travellers (agents) individually for large real-world transport scenarios. Advantages over traditional static assignment include: time-dependent congestion, time-dependent mode choice, the option to accelerate computations by running small samples. Compared to dynamic traffic assignment (DTA; Carey & Watling 2003), the main difference is that MATSim looks at full daily plans whereas DTA looks at trips. This gives MATSim more expressiveness with respect to individual replanning which goes beyond the trip, such as mode choice, moving the whole daily plan forward or backward in time, or activity location choice. In addition to this higher expressiveness, MATSim is often also considerably faster, at the expense of an often less realistic representation of traffic dynamics. MATSim has been applied to large scale scenarios in Zurich, Berlin and many other cities (Balmer 2007).

2.1 MATSim process structure

The general MATSim process structure consists of the following parts (Fig. 1; Balmer et al. 2009):

Initial demand: MATSim requires the physical infrastructure, determined by the road network and facilities (i.e. activity locations like home, work, shopping or leisure) and the population including the demand of each individual person.

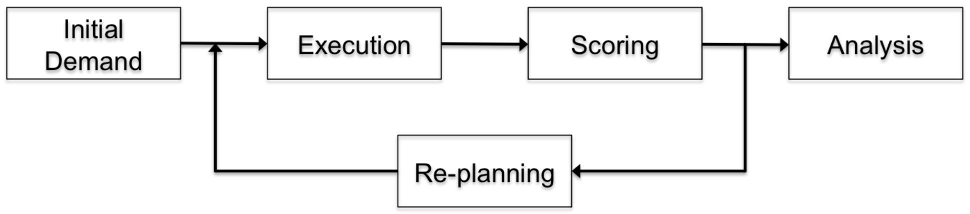


Figure 1: Process structure of MATSim

The initial demand for each agent is usually generated based on microcensus and/or survey data. Initial mode choice is either based on data or heuristic assumptions; initial routes are usually the fastest paths on an empty network.

Iterative demand optimization: In an iterative demand optimization process the demand for each individual agent is improved. It takes into account physical constraints, e.g. the road network, and the interaction between the agents. The optimization process consists of an iteration cycle with three main steps, "Execution", "Scoring" and "Re-planning", which are explained below in more detail.

Analysis: Finally, the simulation results such as the resulting population and demand and the traffic conditions on the network can be used for post-process analysis.

2.2 Agents and plans

Each person or traveller in MATSim is modeled as an individual agent. The demand of an agent is called plan (Balmer et al. 2009). A plan encodes the daily routine of an agent. It contains the agents travel schedule including its intended activities and routing decisions between the activity locations (Balmer et al. 2005). Moreover, a plan captures (i) the order, type, location, duration as well as other time constraints for every activity and (ii) the selected mode, route and expected departure and travel times of each leg (Balmer et al. 2005). A leg describes a part of a trip that uses exactly one transport mode (Balmer et al. 2005). An example plan is illustrated and explained in Fig. 2.

2.3 Iterative demand optimization process

As stated before, the simulation takes the representation of the infrastructure and the population including their daily plans as input (Balmer et al. 2009). Balmer et al. (2005), Raney & Nagel (2006), Balmer et al. (2009) explain the simulation process. It consists of an iterative loop with three main steps, see Fig. 1. These three steps are summarized in the following:

```
<person id="123456" employed="yes">
    <plan score="128.122" selected="yes">
        <act type="home" link="1000" x="100.0" y="100.0" end_time="07:00:00" />
        <leg mode="car" dep_time="07:00:00" trav_time="00:20:00">
            <route type="links"> 1000 1001 1002 1003 </route>
        </leg>
        <act type="work" link="1003" x="200.0" y="200.0" end_time="15:20:00" />
        <leg mode="car" dep_time="15:20:00" trav_time="00:20:00">
            <route type="links"> 1003 1004 1005 1000 </route>
        </leg>
        <act type="home" link="1000" x="100.0" y="100.0" />
    </plan>
</person>
```

Figure 2: This illustrates the demand or plan of a fictive MATSim agent. The agent with the id 123456 intends to leave home (located at link 1000) at 07:00 to go to work. The selected route consists of four links. The expected travel time by car takes 20 minutes. At 15:20, the agent intends to travel back home, which is expected to take again 20 minutes.

Execution: The traffic flow simulation executes the selected plans of all agents simultaneously on the road network. At this stage agents are interacting with the physical environment and with other agents. The traffic flow simulation is implemented as a queue simulation, see Gawron (1998), Cetin et al. (2003) for more details. It has the advantage that it is computationally fast while still resolving each vehicle individually.

Scoring: All executed plans are scored by a utility function that determines the performance of each plan. The utility or scoring function is discussed in more detail below.

Re-planning: In this step, some agents obtain new plans (choice set extension), while all others choose between existing plans, typically based on a logit model. The choice set extension is implemented in a modular way so that different choice dimensions can be addressed. The modules used in conjunction with the Brussels case study (Ch. **??** of this book) are:

Time allocation module: This module generates a new plan by taking an existing plan and randomly changing the planned ending times of activities (Balmer et al. 2005).

Router module: This module generates a new plan by taking an existing plan and recomputing all routes. The router uses the updated generalized costs for each link from the last traffic flow iteration. The router is a time-dependent best path algorithm (Lefebvre & Balmer 2007), where best path is defined as the one with the least negative utility (Balmer et al. 2009).

Single trip mode switch module: This module generates a new plan by taking an existing plan, randomly selecting a trip, and switching its mode to a randomly selected alternative mode. Other modules which take, say, subtours into account, are available (see Rieser et al. 2013).

The number of modules is not limited to those mentioned here. Every module is assigned a weight that determines the probability with which a module is applied to an agent. If one of these modules makes the choice set of an agent larger than some configurable limit, one of the plans, typically the one with the worst score, is removed.

The repetition of the iteration cycle coupled with the agent memory, i.e. the capability to remember more than one plan per agent, enables agents to improve their plans over several iterations (Balmer et al. 2005). The iteration cycle continues until a "relaxed" state of the system has been reached. MATSim currently does not use any quantitative measure of when this state is reached; usually the iteration cycle is repeated until the outcome is stable (Balmer et al. 2005). For simple situations (only two modes, only the car mode explicitly simulated) it is our experience that about 100 iterations are sufficient to obtain useful results (Nagel 2008, 2011, Röder et al. 2013). Under certain conditions, the result is a probabilistic integer version of the stochastic user equilibrium (SUE) (Nagel & Flötteröd 2012).

2.4 Evaluation of the performance of a plan with the scoring function

The utility of an executed plan is computed as (Charypar & Nagel 2005):

$$V_p = \sum_{i=1}^{n} \left(V_{perf,i} + V_{late,i} + V_{tr,i} \right) , \tag{1}$$

where V_p is the accumulated utility for a given plan p with n activities, $V_{perf,i}$ is the utility for performing activity i, $V_{late,i}$ is the disutility (negative utility) of being late at activity i, and $V_{tr,i}$ is a penalty for traveling from activity i to activity $i+1$. Plans are assumed to wrap around a 24-hr day, for that reason, the last activity is assumed to be the same as the first, and in consequence there are as many trips as there are activities.

The utility for performing an activity has a logarithmic form and is defined as:

$$V_{perf,i}(t_{perf,i}) = \beta_{perf} \cdot t_{*,i} \cdot \ln \left(\frac{t_{perf,i}}{t_{0,i}} \right) \tag{2}$$

Here t_{perf} is the actually performed duration of activity i, t_* is the "typical" duration of activity i, β_{perf} (positive) gives the marginal utility of any activity at its typical duration. $t_{0,i}$ has no effect as long as activity chains remain fixed throughout the iterations, which will be assumed for all MATSim runs in this book. More details are provided by Rieser et al. (2013).

The disutility for being late is $V_{late,i}(t_{late,i}) = \beta_{late} \cdot t_{late,i}$, where β_{late} (normally negative) is the marginal utility for being late while $t_{late,i}$ gives the amount of time of being late at activity i.

The penalty for traveling is given by

$$V_{tr,i} = \beta_{tr,mode} \cdot t_i + \beta_{td} \cdot d_i + \beta_m \cdot m_i, \tag{3}$$

where t_i is the travel time, d_i the distance, m_i the change of the monetary position, and the different β_x are the usual pre-factors. In MATSim, m is negative when money is taken away from the traveler, so β_m is typically positive. β_{td} is typically negative. β_{tr} can be positive or negative, see next.

The *effective* disutility for traveling is Eq. (3) plus the lost utility of time as a resource (opportunity cost of time). The marginal utility of time as a resource can be understood as the marginal loss of utility when the activity which follows the trip is shortened. This is computed as

$$\frac{\partial V_{perf}}{\partial t_{perf}} = \beta_{perf} \cdot t_* \cdot \frac{1}{t_{perf}} \approx \beta_{perf} \ , \tag{4}$$

where the approximation holds when the actual duration, t_{perf}, is close to the typical duration, t_*. That is, shortening an activity by Δt is penalized approximately by $-\beta_{perf} \cdot \Delta t$. If that time is spent travelling, Eq. (3) comes on top.

In consequence, the marginal utility of travel time savings, $mUTTS$, is, within the approximation of Eq. (4), equal to $\beta_{perf} - \beta_{tr,mode}$, and a monetary value of travel time savings is given by

$$VTTS = \frac{\beta_{perf} - \beta_{tr,mode}}{\beta_m} \ . \tag{5}$$

Hints for calibration based on those insights are given in the MATSim user guide (Rieser et al. 2013).

2.5 An illustration

Let us illustrate the MATSim iteration process. Initially, a synthetic traveler will have only one plan, say by car. This plan will be repeatedly executed in the traffic flow simulation.

Eventually, the synthetic traveler will be selected for re-planning, say for re-routing. The previous plan will be copied, all routes will be re-computed based on the previous iteration's travel times, the new plan will be made "selected", thus executed in the next traffic flow simulation, and scored based on its performance. The next couple of iterations, this synthetic traveler will choose between these two plans, based on a logit model using the scores of these two plans.

Eventually, the synthetic traveler will again be selected for re-planning, say this time for single trip mode switch. The original plan will be copied, one trip will be randomly selected, and its mode will be switched to a randomly selected alternative mode (here: pt). That trip will be routed if necessary, the plan will be made "selected", thus executed in the next traffic flow simulation, and scored based on its performance. The next couple of iterations, this synthetic traveler will choose between these three plans, based on a logit model using the scores of these three plans.

In consequence, if the pt plan receives a much larger score than the two car plans, it will be selected with a much higher probability. Conversely, if the car plans have higher scores, then they will be selected with higher probability. Since for sufficiently many iterations, eventually all travelers will have all combinations of modes in their choice set, choice is entirely given by the calibration of the scoring function. This statement does even hold (approximately) when the choice set is limited in size, since only options with bad scores, which thus carry a low probability in terms of the logit model, are removed.

3 Special MATSim features used for the case studies

3.1 Public transit

MATSim provides several ways to simulate public transport (pt). The conceptually most straightforward, but also computationally and in terms of data procurement most demanding variant is the direct microscopic simulation, where every public transit vehicle is simulated individually according to its schedule and operating rules (Rieser & Nagel 2009, Rieser 2010). It includes information about public transit lines, their routes, the travel time between stops and the time of departure at the start of the route. Data in the GTFS format (General Transit Feed Specification, GTFS www pages 2012) can be converted into MATSim transit schedules (MATSim extensions www page 2013, under "GTFS2TransitSchedule").

A simple model, sometimes called "pseudo pt" (Grether et al. 2009, Rieser & Nagel 2009), estimates travel times by assuming that the travel distance between activity locations is composed of the beeline distance multiplied with a configurable beeline distance factor, often 1.3. The pt speed is configurable as well. The resulting travel times are determined by:

$$tt_{pt} = \frac{BeelineDistance * BeelineDistanceFactor}{PtSpeed} \tag{6}$$

Alternatively, one can multiply the car free speed travel time with a configurable factor in order to obtain the public transit travel time. This approach models public transport continuously and without capacity constraints and works without any knowledge about the public transport service in an area.

Within the project, an additional approach was implemented, called "matrix based pt", also without capacity constraints, but taking into account access and egress to pt stops and possible other details of the pt system. Given an input table with pt stops and the associated coordinates, a matrix is generated including travel times and travel distances for any pair of stops. Travel times are determined based on Eq. 6. Travel distances are given by the numerator of Eq. 6, i.e. beeline distance between two stops multiplied by the beeline distance factor. The difference to the plain "pseudo pt" is that access and agress to the pt system are now modelled much more realistically. The stop-to-stop impedance matrix can also be taken from an existing VISUM (PTV AG 2009*b*,*a*) model or some other equivalent source, thereby taking connectivity characteristics of the pt system beyond its stop locations into account.

In both cases traveling by public transport is executed as so-called teleportation in MATSim, i.e. simulation of pt is not performed on the physical road network. The total travel time is composed of:

$$t_{pt,ik} = t_{wlk,gap,i} + t_{pt,matrix} + t_{wlk,gap,k} \ , \tag{7}$$

where $t_{wlk,gap,i}$ and $t_{wlk,gap,k}$ are the travel times on foot to overcome the gap between the activity locations i and k respectively to their nearest pt stop based on the beeline distance, and $t_{pt,matrix}$ is the travel time between the two stops, given by the matrix. The total travel distances are determined analogously:

$$d_{pt,ik} = d_{gap,i} + d_{pt,matrix} + d_{gap,k} \tag{8}$$

Here, $d_{gap,i}$ and $d_{gap,k}$ are the beeline distances between location i and k respectively to their nearest pt stop, and $d_{pt,matrix}$ is the travel distance between the two stops from the matrix. The pt travel disutility is computed on those combined travel times and travel distances; separate marginal disutilities for the walk and the pt parts are thus (unfortunately) currently not possible. – For additional information, see the MATSim extensions www page (2013) under "matrix based pt router".

3.2 Road pricing

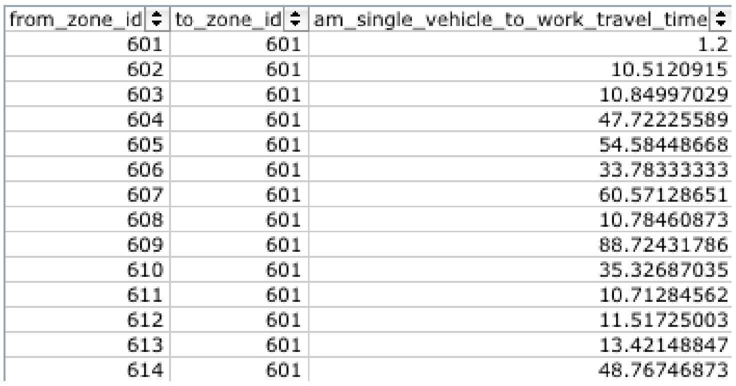
MATSim is capable of simulating different toll schemes such as distance or cordon toll (Nagel et al. 2008, Kickhöfer et al. 2011, Kickhöfer & Nagel 2013). Tolls can be limited to a part of the network. They can also be made time-dependent, i.e. the amount agents have to pay for the toll can vary depending on the time-of-day. – For additional information, see the MATSim extensions www page (2013).

4 Integration approach to use MATSim as a travel model plugin to UrbanSim

In the past, some efforts towards integrating UrbanSim with external travel models like EMME (Babin et al. 1982) or VISUM (PTV AG 2009*a*,*b*) have been made.

Both EMME and VISUM are traditional assignment models using origin-destination matrices (OD-matrices) as inputs (Ortúzar & Willumsen 2001). Thus, they do not make use of the disaggregated nature of UrbanSim. In this situation it seems quite natural to link UrbanSim with an agent-based travel model like MATSim directly at the agent level, by directly feeding location and socio-economic characteristics of individual residents and firms from the land use model to the travel model and then having the travel model return updated accessibility measures back to the land use model.

By default, the feedback from external travel models to UrbanSim is a zone-to-zone impedance matrix including generalized costs of travel between any given pair of zones (Fig. 3). This is an $n \times n$ matrix, where n is the number of zones. UrbanSim uses this matrix as input for various models, as explained in Ch. **??**. Such matrices are growing quadratically with the number of zones and thus quickly become very large. This puts limits both on the spatial resolution that can be used with these zones, and on the number of attributes which can be returned from the travel model to UrbanSim. As one of the objectives of the present study, it makes therefore sense to search for alternative measures to feedback from a travel model.



from_zone_id	to_zone_id	am_single_vehicle_to_work_travel_time
601	601	1.2
602	601	10.5120915
603	601	10.84997029
604	601	47.72225589
605	601	54.58448668
606	601	33.78333333
607	601	60.57128651
608	601	10.78460873
609	601	88.72431786
610	601	35.32687035
611	601	10.71284562
612	601	11.51725003
613	601	13.42148847
614	601	48.76746873

Figure 3: Conventional zone-to-zone impedance matrix including travel times from any origin zone “from_zone_id” to any destination zone “to_zone_id”.

4.1 MATSim4UrbanSim at a glance

The interaction between MATSim and UrbanSim (Nicolai et al. 2011) is a bi-directional relationship as depicted in Fig. 4. It consists of three main steps:

1. When UrbanSim iterates in annual steps, it calls MATSim from time to time, at most once per annual step, and passes a path to the traffic network data together with the persons and jobs data tables as input. The tables include attributes such as the person id as well as the residence and job location of each individual person in UrbanSim.

2. Given the input tables from UrbanSim, MATSim generates the traffic assignment, as discussed in Sec. 2. Finally, it returns the resulting access and accessibility indicators, discussed below in Sec. 5. Then MATSim terminates.

3. Once MATSim has terminated, UrbanSim takes back control. It reads the indicators and updates the data store for the next UrbanSim iteration. UrbanSim models that make use of the updated traffic conditions are for example the Household and Employment Location Choice Models as well as the Real Estate Price Model, presented in Sec. **??**. In fact, arbitrary UrbanSim models can make use of the travel model output, since the data is available as a regular UrbanSim table.

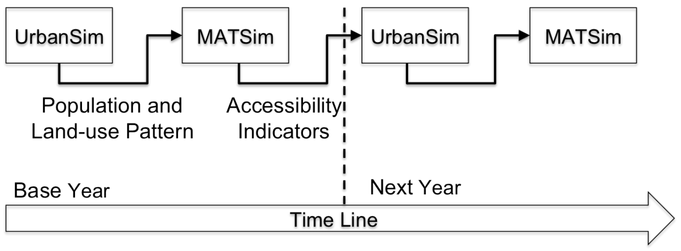


Figure 4: This shows the interaction sequence between UrbanSim and MATSim. UrbanSim calls MATSim in regular intervals and passes the current population and land-use pattern. MATSim computes the traffic based on the provided information and the resulting access and accessibility indicators. UrbanSim uses these indicators for the next year (iteration) as input for various models.

MATSim4UrbanSim simulates home-work-home commuting trips based on home and work locations of each individual person provided by the UrbanSim input tables. More complex, e.g. activity-based, demand patterns are possible with MATSim, but are not implemented in the present version of MATSim4UrbanSim.

4.2 Different options to couple UrbanSim with MATSim

Given that one of the objectives is to exploit the fact that both UrbanSim and MATSim are person-centric, it would be desirable to communicate between both packages directly at the object level. This would mean to invoke and execute MATSim by UrbanSim like other UrbanSim routines, and to exchange data, such as current land use information in UrbanSim or computed access or accessibility indicators by MATSim, directly via the main memory.

As stated earlier, UrbanSim is implemented in Python, while MATSim uses Java. Both languages use different and incompatible byte code representations. Java programs are translated into Java byte code that is executed by the Java Virtual Machine (JVM). The reference implementation for Python is written in C, called

CPython. In the same way as the JVM, CPython compiles Python source code into Python byte code. There is also a Java implementation for Python which would simplify the integration task. However, UrbanSim relies heavily on numerical and other libraries written in C or C++, so that was not an option.

Given the incompatible byte code representation, objects would need to be converted when passed from one package to the other. The arguably first package that comes to mind is the Java Native Interface (JNI; Liang 1999). JNI allows the calling of Java routines from C. Unfortunately, this does not yet resolve how to get the information from Python to C; such a layer would need to be separately implemented.

Other projects have taken up that challenge. The JPype project (JPype www pages 2013) allows Python full access to Java. This is achieved through interfacing the Python interpreter (CPython) and the JVM at the native level using JNI and PNI (Python Native Interface). The main problem is that JPype does not provide support for advanced data types, let alone objects. For example, to create an array, the number of dimensions and the number of elements in the array need to be declared at compile time. This is completely at odds with important concepts used both in UrbanSim and in MATSim, which decide on the size of data classes from reading the input files. Also, there is no support for making sure that the dimensions and sizes declared on the Python side are consistent with the dimensions and sizes used on the MATSim side. It is our belief that projects such as UrbanSim and MATSim have become possible for universities over the past years exactly because modern programming languages provide support for objects. Working without such support leads to code that neither the UrbanSim nor the MATSim team have the resources to maintain. In addition, the JPype project does not seem overly active, with the last blog entry from 2007, and the last two (minor) releases from 2007 and 2011, making it problematic to base an important part of the SustainCity project on JPype.

The only other project that we could find in that area was JEPP (Jepp www pages 2013). That project, however, looks at calling Python from Java, which is exactly the wrong direction for what was needed here.

Thus, in the end it was decided to resort to the conventional technique of writing and reading files. UrbanSim already had methods to write and read comma-separated files, and for MATSim they were easy to write.

A challenge was how to install and configure MATSim in a convenient and robust way. This meant that preferably, at least to get started, one file containing all the MATSim material should be unzipped at the right place, and relevant entries should be made in the OPUS GUI (Graphical User Interface), presumably already known to the UrbanSim user. This, however, implied that a communcation channel from UrbanSim to MATSim would need to be established which could be automatically tested, so that eventual errors during future refactorings of UrbanSim or MATSim that would destroy that communication channel could be detected.

This was resolved passing an XML (eXtensible Markup Language; W3C 2008) document from UrbanSim to MATSim, and using a so-called XML Schema Document (XSD; van der Vlist 2002) to standardize and certify the grammar of that document. For details, see Sec. 4.3.

Another challenge stems from the data type flexibility of UrbanSim, where virtually arbitrary attributes can be added to the tables, and related to each other through the configurable models. One consequence of this flexibility is that the attribute names are not standardized; for example, not even the keys (= attribute names) for coordinate entries are standardized in UrbanSim. In consequence, the passing of data between UrbanSim and MATSim has to rely on often-used names on the UrbanSim side (Nicolai 2012); projects which deviate from these naming conventions would need to implement their own data packing method on the UrbanSim side. This holds also for the direction from MATSim to UrbanSim: MATSim assumes the existence of tables under certain keys on the UrbanSim size; if they exist under a different key, the MATSim output will be added instead of replacing the data in the UrbanSim tables.

In short, the data exchange will work as long as the naming conventions from Nicolai (2012) are strictly followed. Any deviations from these naming conventions mean that changes in the computer codes need to be made.

Overall, not being able to couple the two packages at the object level is a bit disappointing, since this means that much of the agent-oriented expressiveness cannot overcome the barrier between the two packages. For example, it is not possible for a locating-seeking agent on the UrbanSim side to query MATSim about detailed transportation options, and it is also not possible for a synthetic MATSim traveler stuck in congestion to, say, go shopping at a different location if landuse information was not passed to MATSim pre-emptively. It is to be hoped that future technological developments will overcome this barrier.

4.3 MATSim4Urbansim configuration

Both UrbanSim and MATSim have, in principle, their own configuration file. However, to run the frameworks together for different scenarios would have the consequence to set up both configuration files separately. This easily leads to inconsistent, error-prone and fragile configurations that are inconvenient to maintain by users. To address this issue, it was decided to provide one joint configuration for both frameworks, at least for simple situations. At this point a brief overview is provided; a comprehensive description is given by Nicolai & Nagel (2010):

The UrbanSim side was selected as the place to manage the joint configuration, since it embeds MATSim as a travel model plug-in. It is achieved by embedding necessary MATSim parameters into the travel model configuration section of the UrbanSim configuration, as shown in Fig. 5. In this way, both simulation frameworks are conveniently configurable via the OPUS GUI, i.e. the graphical user interface that is available to configure the OPUS system and start UrbanSim runs. Necessary parameters, for instance, include input file locations, e.g. for the traffic network, the population sampling rate to accelerate computation times, the run description defining for which years to run MATSim, and more; in-depth information about this is provided in the user guide (MATSim extensions www page 2013, see MATSim4UrbanSim).

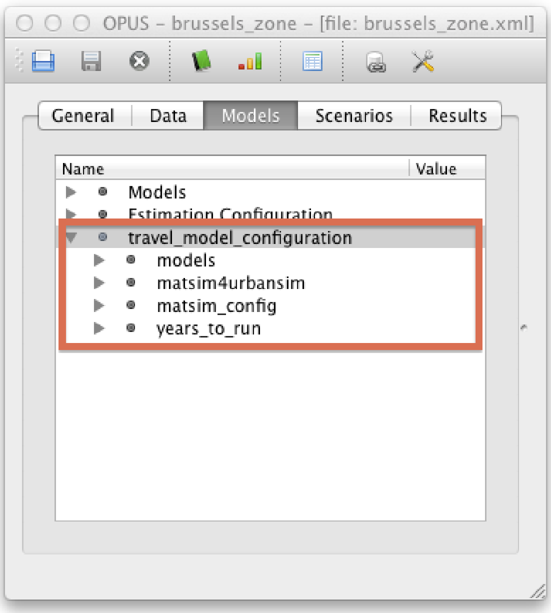


Figure 5: Relevant MATSim parameter are configurable via the OPUS GUI. They are embedded in the travel_model_configuration section (marked in red) as part of the UrbanSim configuration.

UrbanSim takes the parameter settings from the travel model configuration section and generates a separate configuration file in XML format to initialize MATSim. This is done each time the travel model is called by UrbanSim. To achieve a robust and reliable communication, a mutual consent between UrbanSim and MATSim regarding the structure and the parameter data types, e.g. string, int or float, of the generated configuration file is crucial. Therefore, the generated MATSim XML configuration is specified by a so-called XML Schema Document (XSD), which is an abstract collection of meta data about an XML document (van der Vlist 2002). The XSD is used to generate customized XML parsers in MATSim and UrbanSim that read, write and validate XML documents; this technique is called XML data binding.

In addition, this approach allows a convenient adaption of the MATSim configuration to new requirements, e.g. when additional configuration parameters are required. This can be done by adjusting the XSD and regenerating the XML parsers on both sides.

4.4 Cold, warm and hot start

Here, three notations are introduced: cold, warm and hot start. Cold start was in

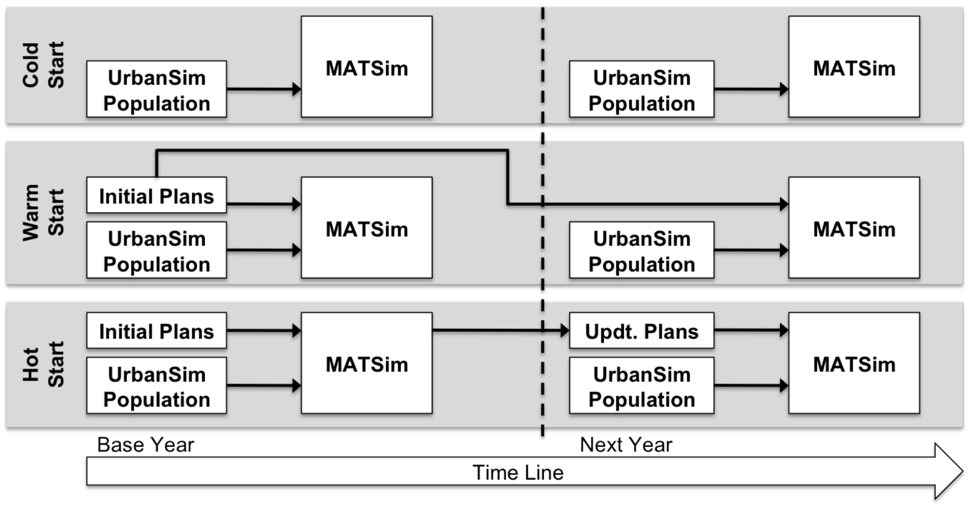


Figure 6: This illustrates the working of the (i) cold, (ii) warm and (iii) hot start in MATSim. With (i) cold start MATSim generates the initial demand from the current UrbanSim population. With warm and hot start MATSim recycles information from previous runs; as a result less iterations are required to reach a relaxed state of the system. (ii) Warm start will always use the same plans file, which becomes less and less correct when running UrbanSim over a long time span. Opposed to warm start, (iii) in hot start after each run an updated plans file is stored, incorporates changes from the current UrbanSim population. In this case, MATSim starts from the updated plans file instead of the initial plans file.

principle already described in Sec. 4.1 above. It means that MATSim generates the initial travel demand based on the provided UrbanSim population, see Fig. 6.

Warm and hot start describe the capability of MATSim4UrbanSim to start a simulation from a pre-existing, relaxed plans file (e.g. see Sec. 2) and to recycle information, such as route decisions and departure time choices, from previous runs. In other words, MATSim "remembers" the travel schedule (daily plan) of each traveller. Consequently fewer iterations are and thus less computing time is required to reach a relaxed state of the system.

In warm start, MATSim will always use the same relaxed plans file as shown (as "Inital Plans") in Fig. 6. This plans file is generated from a preparatory run, which relaxes the travel model for the base year. In subsequent UrbanSim years, those synthetic travellers that neither change their residence nor their workplace will start with those relaxed plans; also see below. All other travellers will start from scratch.

When running UrbanSim over many years with a changing population, because of the relocations the initial plans file will become less and less correct, and the system will require more and more iterations to get MATSim back into a relaxed state. This issue is addressed by hot start. Here, MATSim stores an updated plans file after each run, including all changes of the current UrbanSim population. As a result the differences between the updated plans file and the UrbanSim population are kept small. As opposed to warm start, hot start uses the updated instead of the initial plans file after the first run, as shown in Fig. 6.

Technically, when MATSim performs a warm or hot start it reads a plans file together with the current UrbanSim population. It keeps all persons from that plans file that have not changed; this means:

1. A person from the current UrbanSim population also exists in the plans file.
2. Has the same employment status.
3. Has the same home location.
4. And if applicable, has the same work location.

For all other persons, new initial plans are generated. Some computational results concerning warm and hot start are available from Nicolai (2013).

5 Access and accessibility indicators

This section looks at the access (= impedance) and accessibility indicators computed by MATSim as feedback for UrbanSim. UrbanSim uses the feedback to update its data base for the next year as stated in Sec. 4 above. The term "access" refers to a two-point-value such as travel time impedances between an origin and a destination; in contrast, "accessibility" is attached to one location and thus refers to an aggregated single-point-value. The MATSim feedback currently includes (i) so called zone-to-zone "skims", meaning a summary of key model predictions like travel times by mode or time of day (OPUS User Guide 2011), (ii) individual agent-based performances as well as (iii) accessibilities to work places. The following provides a summary, see Nicolai (2012) for details.

5.1 Zone-to-zone impedance matrix

The zone-to-zone impedance matrices contain travel times for several transport modes, travel distances, generalized travel costs and the number of vehicle trips for each pair of zones. Zones are assigned to the road network by determining the network node which is closest to the zone centroid. The centroid coordinates can be obtained from a variety of definitions, either (i) they are directly available in the UrbanSim model, or by (ii) averaging all parcel coordinates that belong to a zone. Typically, (i) applies to UrbanSim zone models and (ii) to UrbanSim parcel models.

Travel times are provided for congested and free speed car, public transport, bicycle and walk. Congested and free speed car travel times are accordingly based on congested or free speed link travel times of the MATSim road network. By default, congested car travel times are measured for the morning peak period, i.e. trips are assumed to start at 8am. Travel times on foot or by bicycle are based on the shortest path on the road network with a constant speed of $5km/h$ (walk) and $15km/h$ (bicycle). For public transport (pt) travel times are obtained by Eq. 7, which is composed of traveling on foot to the nearest pt stop from both the (i) origin and (ii) destination and (iii) the pt travel times queried from the pt matrix.

Travel distances are given using the shortest path on the road network.[1] Generalized travel costs are given by Eq. 3 – including congested car travel times, distances and toll. The number of vehicle trips per OD pair are obtained from counting the actually simulated trips of each synthetic traveler. The numbers are then scaled to the full population sample.

5.2 Agent-based performance

This feedback contains the individual travel performance for each MATSim agent based on the selected mode, congested car or public transport (pt), including travel times and travel distances, for the selected plan, e.g. see agent plans in Sec. 2. These are given for both directions, commuting from home to work and back. For the time being, only information on the congested car mode and on the pt mode is returned, corresponding to the fact that the default version of MATSim4UrbanSim only uses these two modes. Additional modes could be integrated with relatively little effort.

5.3 Accessibility computation

A comprehensive description about high resolution accessibility computations in MATSim is given by Nicolai & Nagel (in press, 2012). At this point a brief overview built on these references is provided.

For the present implementation a utility-based measure is selected (e.g. Ben-Akiva & Lerman 1985, Train 2003). It reflects the (economic) benefits, computed as the expected maximum utility, that someone gains from access to spatially distributed opportunities (Geurs & Ritsema van Eck 2001, de Jong et al. 2007). This is also known as the logsum term; it is defined as

$$A_i := \ln \sum_k e^{V_{ik}} \, , \tag{9}$$

where k goes over all possible destinations. V_{ik} is the disutility of travel in order to get from location i to location k; in discrete choice theory interpretation, these are the so-called systematic parts of the utility.[2]

[1]Note that a car will typically not take the shortest but rather the fastest/best path, and the bicycle/walk modes are not restricted to the road network. Thus, these distances should be seen as an approximation.

[2]There is sometimes an additional scale parameter in that definition. Here it is assumed that this

Origin and opportunity locations do not necessarily lie on the network. Thus, for the *network-oriented modes car, bicycle, and walk,* the calculation of V_{ik} includes the disutility of travel to overcome the gap between locations and the network; in the current implementation it is assumed that in these modes, opportunities can only be reached via the transport network. Overall, V_{ik} consists of the following components:

1. The walk disutility for reaching the transport network from a given origin location i. This gap is determined as the shortest distance to the network either given by (i) the distance to the nearest node or (ii) the orthogonal distance to the nearest link on the network.

   For the latter case, V_{ik} additionally includes the travel disutility to overcome the distance to the nearest node according to the given transport mode.

2. The travel disutility on the network towards k is given by

   $$V_{tr,i} = (\beta_{tr,mode} - \beta_{perf}) \cdot t_i + \beta_{td} \cdot d_i + \beta_m \cdot m_i \; ; \qquad (10)$$

   this is Eq. (3) *plus* the effect of using time as a resource. Using Eq. (5), the first term of the sum can also be expressed as $-\beta_m \cdot VTTS \cdot t_i$. The parameters are taken from the travel model.

3. The walk travel disutility for reaching opportunity k from the transport network. Here, the distance to the nearest node is used to determine the shortest distance to the network.

This results in

$$V_{ik,mode} \;:=\; V_{wlk,gap,i} \;+\; V_{mode,tr,k} \;+\; V_{wlk,gap,k} \; , \qquad (11)$$

where $V_{wlk,gap,i}$ is the walk disutility to overcome the gap between origin location i and the road network, $V_{mode,tr,k}$ is the travel disutility on the network traveling towards k (Eq. 10), and $V_{wlk,gap,k}$ is the walk disutility to overcome the gap between the road network and opportunity location k. The term $V_{wlk,gap,i}$ consists of

$$V_{wlk,gap,i} = \beta_{tt,wlk,gap,i} \cdot t_{wlk,gap,i} + \beta_{d,wlk,gap,i} \cdot d_{wlk,gap,i} \; , \qquad (12)$$

where $t_{wlk,gap,i}$ is the walk time and $d_{wlk,gap,i}$ is the distance to overcome the gap between location i and the network. $\beta_{tt,wlk}$ and $\beta_{d,wlk}$ are marginal utilities that convert travel times and distances respectively into utils. $V_{wlk,gap,k}$ is defined in the same way.

For the *pt mode*, the accessibility computation is only implemented when the "matrix based pt" approach described in Sec. 3.1 is used. Then, the walk times and

is absorbed into V_{ik}. In consequence, A_i is in the same units as V_{ik}; a conversion to, say, monetary terms needs to be done separately.

distances to the network are replaced by the walk times and distances to the nearest pt stop, and the travel times and distances on the network are replaced by the times and distances given by the pt matrix. In contrast to Sec. 3.1, the computation here uses different marginal disutilities for access/egress walk, and for the time spent "in the pt system".

Accessibilities can be calculated for two spatial systems, (i) zones and (ii) parcels.

- For zones, zone centroids are used as measuring points. The centroid coordinates can be obtained from a variety of definitions, as explained in Sec. 5.1.
- For parcels, the study area is subdivided into square cells, where the resulting cell centroids serve as origins or measuring points for the accessibility calculation. The spatial resolution of this is configurable. Once the calculation for each cell is completed, MATSim interpolates the accessibility value from the grid-cell to each UrbanSim parcel. For more information see Thunig & Nicolai (2013).

  The intermediate result based on the grid cells can be saved as well.

The current version of MATSim4UrbanSim returns workplace accessibilities, i.e. the opportunities mentioned above are workplaces. Accessibilities to other types of opportunities could be added with relatively little effort.

6 Some accessibility results

As an illustration, intermediate results of the Brussels case study are shown. Fig. 7 shows the morning workplace accessibility by car in the base case. One very clearly sees the influence of the network; since the code assumes that access to the network is by the walk mode, the distance to the transport network is the most decisive factor because of the low walk speed. This points to the necessity to make such studies with higher resolution networks – this was not an option here because of computational performance restrictions for the UrbanSim integration (see Sec. 7).

One of the scenarios of the Brussels UrbanSim case study was looking at the influence of a possible cordon toll of 5Eu, levied between 6am and 10am. The cordon was defined just outside the outer freeway ring; tolled links are shown in blue in Fig. 8. Car users from outside thus need to pay toll when using the ring. MATSim was then run again with that toll enabled, and the resulting accessibility differences are shown in the same figure. One finds two major effects:

- The accessibility inside the cordon area is visibly improved. This is due to reduced congestion.
- The accessibility outside the cordon area is visibly reduced. This is because the toll is part of the travel disutility. In consequence, persons living outside the toll area now have a higher travel disutility to reach some of their possible workplaces, and thus their (car) accessibility to workplaces is reduced.

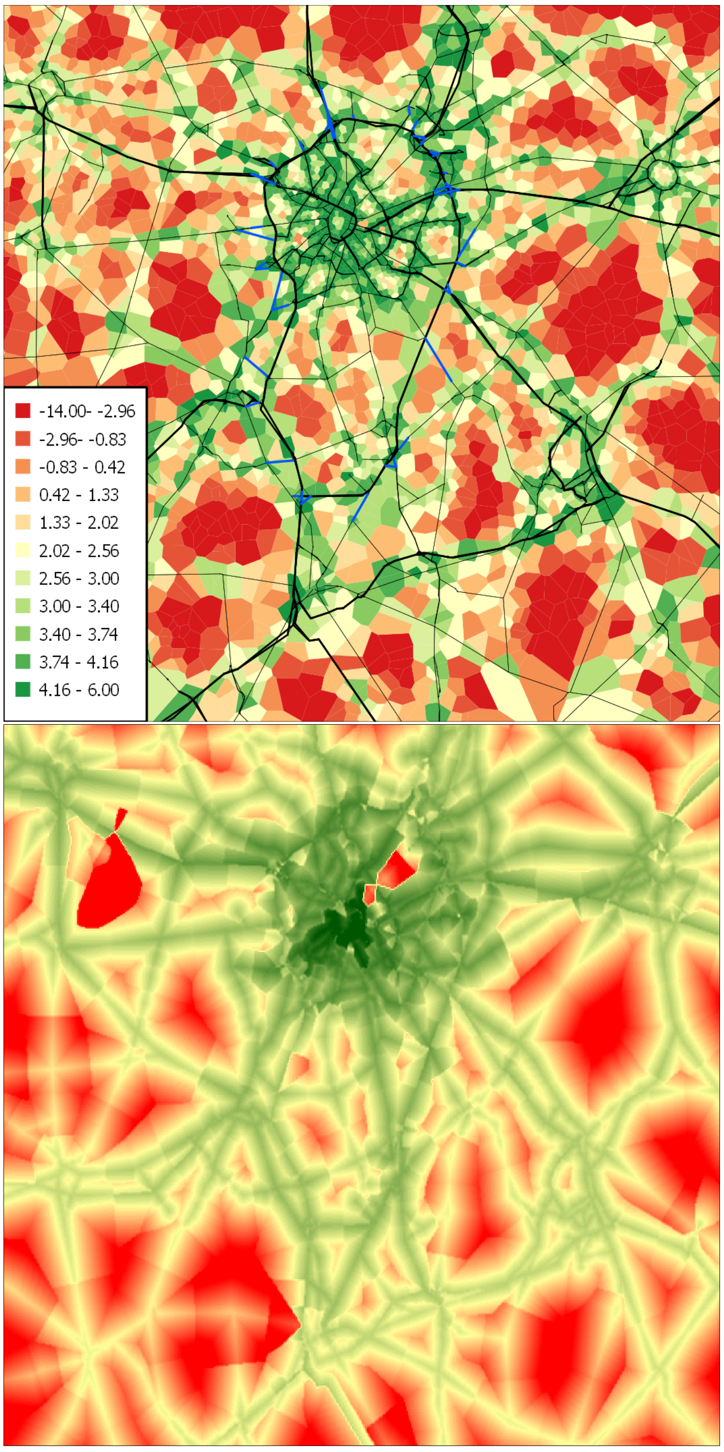


Figure 7: Brussels case study – intermediate results. Morning workplace accessibility by car in the base case. *Top:* Using zones. *Bottom:* Using high resolution (grid-based) accessibility.

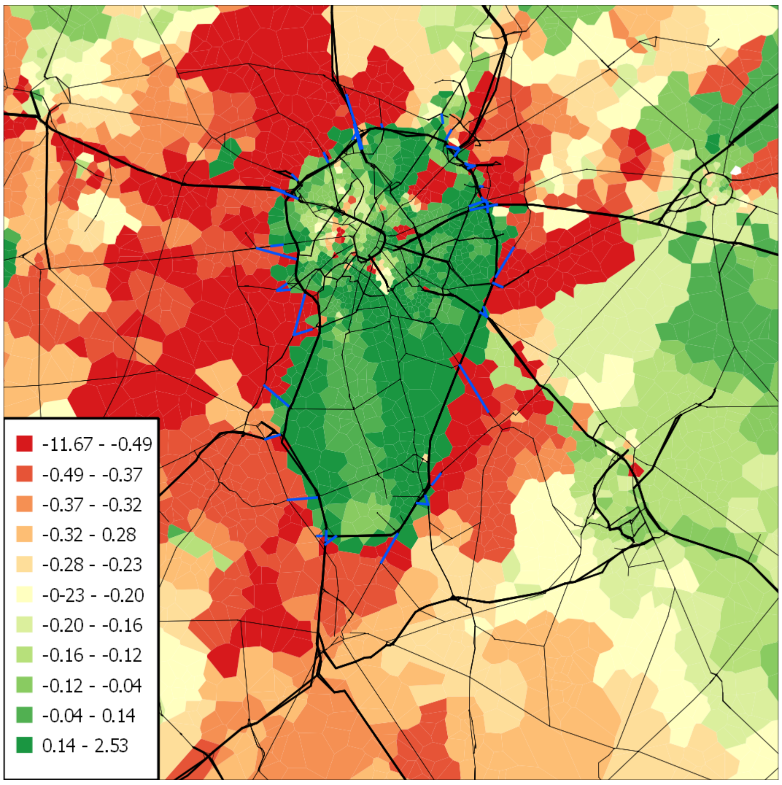


Figure 8: Changes in the morning workplace accessibility by car because of a morning cordon toll of 5 Eu in the Brussels area. The tolled links are marked in blue; the freeway ring is fully inside the cordon area.

7 Computational issues

With the case study set-up for Brussels, one MATSim run with 100 iterations took about 40 min on a regular desktop computer. Fig. 9 gives the breakdown between different modules, over the iterations. The accessibility computation itself is not contained in that breakdown, but takes about 2 min of computing time for a problem of this size (Nicolai & Nagel in press).

Compared to UrbanSim, which only needs a couple of minutes per simulated year, these 40 min per run were already a significant burden, and in consequence MATSim was only run for 3 different UrbanSim years. Keeping all other things constant, the computation time of a MATSim run scales roughly linearly in the number of links. Doubling or quadrupling the number of links, leading to MATSim computation times of 80 or 160 minutes, was not an option for the present study. Warm or hot start acceleration was not used since it had some isssues in conjunction with the matrix-based pt, and there was no time to resolve them.

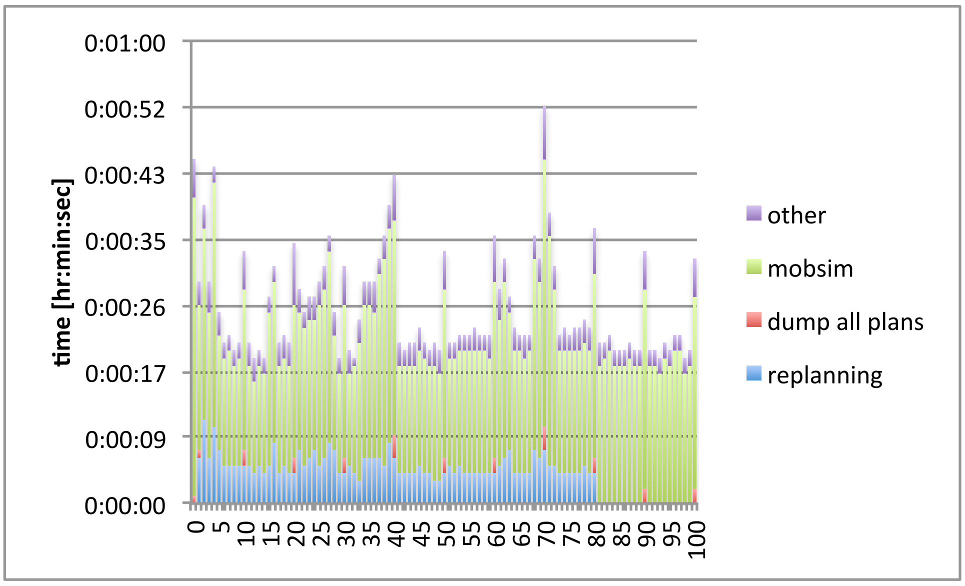


Figure 9: Computational performance of a typical MATSim run within the Brussels case study over the iterations. The machine had additional loads, which explains the fluctuations.

8 Discussion and outlook for the SustainCity project

This chapter discusses a so-called agent-based model as travel model plugin, as opposed to a trip-based model which will be discussed in Ch. **??** of the book. The most important difference is that the agent-based approach keeps the simulated person intact throughout the day. This allows, for example, to consider the effect of a morning toll on the afternoon mode choice, or the effect of a morning delay on the afternoon schedule.

In principle, agent-oriented coupling should be very natural between MATSim and UrbanSim, because both are person-centric models. And indeed, one set of quantities that is fed back from MATSim to UrbanSim is person-centric.

Notwithstanding these statements, coupling Java-based MATSim with Python/C/C++-based UrbanSim, as was done here, remains problematic. The coupling between MATSim and UrbanSim was achieved using the traditional method of writing and reading regular files. This does, however, not allow to directly call one software from the other. In particular, it is impossible for “persons” in UrbanSim to query the MATSim model directly concerning, say, how easy or difficult it is to reach the workplace from a considered housing location. This reinforces the assessment that there are currently separate streams of software that will remain difficult to integrate fully until technology improves.

In contrast, the computation of high resolution accessibilities proved to be rather fruitful. As anticipated, it is quite feasible to compute accessibilities on a $100m \times 100m$ grid, while taking into account congestion. This provides better insights and thus also better modelling input to UrbanSim than conventional accessibility computations that aggregate over zones. It is intended to use this in the future for various scenarios and also for other activity types besides work. This should, for example, help with the identification of regions with poor access to certain services.

References

Babin, A., Florian, M., James-Lefebvre, L. & Spiess, H. (1982), ‘EMME/2: Interactive graphic method for road and transit planning’, *Transportation Research Record* **866**, 1–9.

Balmer, M. (2007), Travel demand modeling for multi-agent transport simulations: Algorithms and systems, PhD thesis, Swiss Federal Institute of Technology (ETH) Zürich, Switzerland.

Balmer, M., Raney, B. & Nagel, K. (2005), Adjustment of activity timing and duration in an agent-based traffic flow simulation, *in* H. Timmermans, ed., ‘Progress in activity-based analysis’, Elsevier, Oxford, UK, pp. 91–114.

Balmer, M., Rieser, M., Meister, K., Charypar, D., Lefebvre, N., Nagel, K. & Axhausen, K. (2009), MATSim-T: Architecture and simulation times, *in* A. Bazzan & F. Klügl, eds, ‘Multi-Agent Systems for Traffic and Transportation’, IGI Global, pp. 57–78.

Ben-Akiva, M. & Lerman, S. R. (1985), *Discrete choice analysis*, The MIT Press, Cambridge, MA.

Carey, M. & Watling, D. (2003), ‘Introduction to a special issue “Dynamic Traffic Assignment II”’, *Networks and Spatial Economics* **3**, 403–406.

Cetin, N., Burri, A. & Nagel, K. (2003), A large-scale agent-based traffic microsimulation based on queue model, *in* ‘Proceedings of the Swiss Transport Research Conference (STRC)’, Monte Verita, Switzerland. Earlier version, with inferior performance values: Transportation Research Board Annual Meeting 2003 paper number 03-4272.
URL: *http://www.strc.ch*

Charypar, D. & Nagel, K. (2005), ‘Generating complete all-day activity plans with genetic algorithms’, *Transportation* **32**(4), 369–397.

de Jong, G., Daly, A., Pieters, M. & van der Hoorn, T. (2007), ‘The logsum as an evaluation measure: Review of the literature and new results’, *Transportation Research Part A: Policy and Practice* **41**(9), 874–889.

Gawron, C. (1998), ‘An iterative algorithm to determine the dynamic user equilibrium in a traffic simulation model’, *International Journal of Modern Physics C* **9**(3), 393–407.

Geurs, K. & Ritsema van Eck, J. (2001), Accessibility measures: review and applications, Technical report, National Institut of Public Health and the Environment, RIVM, P.O. Box 1, 3720 BA Bilthoven.

Grether, D., Chen, Y., Rieser, M. & Nagel, K. (2009), Effects of a simple mode choice model in a large-scale agent-based transport simulation, *in* A. Reggiani & P. Nijkamp, eds, ‘Complexity and Spatial Networks. In Search of Simplicity’, Advances in Spatial Science, Springer, chapter 13, pp. 167–186.

GTFS www pages (2012), ‘General transit feed specification’. Accessed April 2013.
URL: *https://developers.google.com/transit/gtfs/*

Jepp www pages (2013), ‘Jepp – Java Embedded Python’. Accessed December 2013.
URL: *http://jepp.sourceforge.net/*

JPype www pages (2013), ‘JPype – Java to Python integration’. Accessed June 2013.
URL: *http://jpype.sourceforge.net/*

Kickhöfer, B., Grether, D. & Nagel, K. (2011), ‘Income-contingent user preferences in policy evaluation: application and discussion based on multi-agent transport simulations’, *Transportation* **38**, 849–870.

Kickhöfer, B. & Nagel, K. (2013), ‘Towards High-Resolution First-Best Air Pollution Tolls’, *Networks and Spatial Economics* pp. 1–24.

Lefebvre, N. & Balmer, M. (2007), Fast shortest path computation in time-dependent traffic networks, *in* ‘Proceedings of the Swiss Transport Research Conference (STRC)’, Monte Verita, Switzerland.
URL: *http://www.strc.ch*

Liang, S. (1999), *The Java Native Interface: Programmer’s Guide and Specification*, Addison-Wesley Java series, Addison-Wesley.

MATSim extensions www page (2013). Accessed December 2013.
URL: *http://matsim.org/extensions*

Nagel, K. (2008), Towards simulation-based sketch planning: Some results concerning the Alaskan Way viaduct in Seattle WA, VSP Working Paper 08-22, TU Berlin, Transport Systems Planning and Transport Telematics. See www.vsp.tu-berlin.de/publications.

Nagel, K. (2011), Towards simulation-based sketch planning, part II: Some results concerning a freeway extension in Berlin, VSP Working Paper 11-18, TU Berlin, Transport Systems Planning and Transport Telematics. See www.vsp.tu-berlin.de/publications.

Nagel, K. & Flötteröd, G. (2012), Agent-based traffic assignment: Going from trips to behavioural travelers, *in* R. Pendyala & C. Bhat, eds, ‘Travel Behaviour Research in an Evolving World – Selected papers from the 12th international conference on travel behaviour research’, International Association for Travel Behaviour Research, chapter 12, pp. 261–294.

Nagel, K., Grether, D., Beuck, U., Chen, Y., Rieser, M. & Axhausen, K. (2008), *Multi-agent transport simulations and economic evaluation*, Vol. 228 of *Journal of Economics and Statistics (Jahrbücher für Nationalökonomie und Statistik)*, pp. 173–194.

Nicolai, T. W. (2012), Using MATSim as a travel model plug-in to UrbanSim, VSP Working Paper 12-29, TU Berlin, Transport Systems Planning and Transport Telematics. Also VSP WP 12-29, see www.vsp.tu-berlin.de/publications.

Nicolai, T. W. (2013), Investigating the MATSim warm and hot start capability, VSP Working Paper 13-06, TU Berlin, Transport Systems Planning and Transport Telematics. See www.vsp.tu-berlin.de/publications.

Nicolai, T. W. & Nagel, K. (2010), Coupling MATSim and UrbanSim: Software design issues, VSP Working Paper 10-13, TU Berlin, Transport Systems Planning and Transport Telematics. See www.vsp.tu-berlin.de/publications.

Nicolai, T. W. & Nagel, K. (2012), Sensitivity tests with high resolution accessibility computations, VSP Working Paper 12-22, TU Berlin, Transport Systems Planning and Transport Telematics. See www.vsp.tu-berlin.de/publications.

Nicolai, T. W. & Nagel, K. (in press), High resolution accessibility computations, *in* A. Conde co, A. Reggiani & J. Gutiérrez, eds, ‘Accessibility and spatial interaction’, Edward Elgar. Also VSP WP 13-02, see www.vsp.tu-berlin.de/publications.

Nicolai, T. W., Wang, L., Nagel, K. & Waddell, P. (2011), Coupling an urban simulation model with a travel model – A first sensitivity test, *in* ‘Computers in Urban Planning and Urban Management (CUPUM)’, Lake Louise, Canada. Also VSP WP 11-07, see www.vsp.tu-berlin.de/publications.

OPUS User Guide (2011), *The Open Platform for Urban Simulation and UrbanSim Version 4.3*, University of California Berkeley and University of Washington.
URL: *http://www.urbansim.org*

Ortúzar, J. d. D. & Willumsen, L. (2001), *Modelling transport*, 3. edn, John Wiley Sons Ltd, Chichester.

PTV AG (2009*a*), *VISUM 11.0 Benutzerhandbuch*, 76131 Karlsruhe.

PTV AG (2009*b*), *VISUM 11.0 Grundlagen*, 76131 Karlsruhe.

Raney, B. & Nagel, K. (2006), An improved framework for large-scale multi-agent simulations of travel behaviour, *in* P. Rietveld, B. Jourquin & K. Westin, eds, ‘Towards better performing European Transportation Systems’, Routledge, London, pp. 305–347.

Rieser, M. (2010), Adding transit to an agent-based transportation simulation concepts and implementation, PhD thesis, TU Berlin. Also VSP WP 10-05, see www.vsp.tu-berlin.de/publications.

Rieser, M., Dobler, C., Dubernet, T., Grether, D., Horni, A., Lämmel, G., Waraich, R., Zilske, M., Axhausen, K. W. & Nagel, K. (2013), ‘MATSim user guide’. Accessed 2013.
URL: *http://www.matsim.org/userguide*

Rieser, M. & Nagel, K. (2009), Combined agent-based simulation of private car traffic and transit. Also VSP WP 09-11, see www.vsp.tu-berlin.de/publications.

Röder, D., Cabrita, I. & Nagel, K. (2013), Simulation-based sketch planning, part III: Calibration of a MATSim-model for the greater Brussels area and investigation of a cordon pricing for the highway ring, VSP working paper 13-16, TU Berlin, Berlin, Germany. See www.vsp.tu-berlin.de/publications.

Thunig, T. & Nicolai, T. W. (2013), Spatial interpolation of accessibilities, VSP Working Paper 13-07, TU Berlin, Transport Systems Planning and Transport Telematics. Also VSP WP 13-07, see www.vsp.tu-berlin.de/publications.

Train, K. (2003), *Discrete choice methods with simulation*, Cambridge University Press.

van der Vlist, E. (2002), *XML Schema.*, 1st edn, O’Reilly.

W3C (2008), *eXtensible Markup Language (XML)*, World Wide Web Consortium (W3C). See www.w3.org/XML.

Wegener, M. (2004), Overview of land-use transport models, *in* K. Hensher, D.A.; Button, ed., ‘Transport Geography and Spatial Systems’, number 5 *in* ‘Handbook in Transport’, Pergamon/Elsevier Science, pp. 127–146.