Michael Balmer, Marcel Rieser, Konrad Meister, David Charypar, Nicolas Lefebvre, Kai Nagel

MATSim-T: Architecture and Simulation Times

Balmer, M.; Rieser, M.; Meister, K.; Charypar, D.; Lefebvre, N.; Nagel, K. (2009). MATSim-T: Architecture and Simulation Times. In A. Bazzan, & F. Klügl (Eds.), Multi-Agent Systems for Traffic and Transportation Engineering (pp. 57-78). Hershey, PA: IGI Global. https://doi.org/10.4018/978-1-60566-226-8.ch003

MATSim-T: Architecture and Simulation Times

Michael Balmer
IVT, ETH Zürich, CH-8093 Zürich, Tel: +41-44-633 27 80, Fax: +41-44-633 10 57, E-Mail: balmer (at) ivt.baug.ethz.ch
Marcel Rieser
VSP, TU Berlin, D-10587 Berlin, Tel: +49-30-314 25 258, Fax: +49-30-314 26 269, E-Mail: rieser (at) vsp.tu-berlin.de, and
IVT, ETH Zürich, CH-8093 Zürich, E-Mail: rieser (at) ivt.baug.ethz.ch
Konrad Meister
IVT, ETH Zürich, CH-8093 Zürich, Tel: +41-44-633 30 88, Fax: +41-44-633 10 57, E-Mail: meister (at) ivt.baug.ethz.ch
David Charypar
IVT, ETH Zürich, CH-8093 Zürich, Tel: +41-44-633 35 62, Fax: +41-44-633 10 57, E-Mail: charypar (at) ivt.baug.ethz.ch
Nicolas Lefebvre
IVT, ETH Zürich, CH-8093 Zürich, Tel: +41-44-633 49 97, Fax: +41-44-633 10 57, E-Mail: lefebvre (at) ivt.baug.ethz.ch
Kai Nagel
VSP, TU Berlin, D-10587 Berlin, Tel: +49-30-314 23 308, Fax: +49-30-314 26 269, E-Mail: nagel (at) vsp.tu-berlin.de

ABSTRACT

Micro-simulations for transport planning are becoming increasingly important in traffic simulation, traffic analysis, and traffic forecasting. In the last decades the shift from using typically aggregated data to more detailed, individual based, complex data (e.g. GPS tracking) and the continuously growing computer performance on fixed price level leads to the possibility of using microscopic models for large scale planning regions.

This chapter presents such a micro-simulation. The work is part of the research project MATSim (Multi Agent Transport Simulation, http://matsim.org). In the chapter here the focus lies on design and implementation issues as well as on computational performance of different parts of the system.

Based on a study of Swiss daily traffic – ca. 2.3 million individuals using motorized individual transport producing about 7.1 million trips, assigned to a Swiss network model with about 60'000 links, simulated and optimized completely time-dynamic for a complete workday – it is shown that the system is able to generate those traffic patterns in about 36 hours computation time.

KEYWORDS

traffic simulation; transport simulation; multi-agent simulation; large-scale simulation

1 INTRODUCTION

By tradition, transport planning simulation models tend to be macroscopic or mesoscopic (e.g. de Palma & Marchal, 2002; PTV, 2008). Reasons for this are access to aggregated data only (e.g. traffic counts, commuter matrices, etc.) and limitations in computational hardware to calculate and store detailed computations. These limitations have changed in the last few decades. Performance of computer hardware has been continually growing—and still grows—while the cost of machines stays fixed. For transport planning, the relevant developments in computer hardware are:

- The capacities of fast random access memory (RAM) have increased dramatically.
- Multi processor hardware allows one to perform parallel computation without using (maintenance intensive) computer clusters.
- Shared memory architectures allow fast on-demand access to the physical memory for an arbitrary amount of processes.

In the same manner, the available data used in transport planning are getting more detailed and complex. Good examples for that are person diary surveys (e.g. Hanson & Burnett, 1982; Axhausen et al, 2002; Schönfelder et al, 2002), and analysis of individual transport behavior based on GPS data (e.g. Wolf et al, 2004).

Therefore, the demands made to transport planning software are getting more complex, too. Micro-simulation is becoming increasingly important in traffic simulation, traffic analysis, and traffic forecasting. Some advantages over conventional models are:

- Computational savings when compared to the calculation and storage of large multidimensional probability arrays necessary in other methods.
- Larger range of output options, from overall statistics to information about each synthetic traveler in the simulation.
- Explicit modeling of the individuals' decision-making processes.

The last point is important since it is not a vehicle that produces traffic; it is the person that drives the vehicle. Persons do not just produce traffic; instead they try to manage their day (week, life) in a satisfying way. They go to work to earn money, they go hiking for their health and pleasure, they visit their relatives for pleasure or because they feel obliged to do so, they shop to cook a nice dinner at home, and so on. Since not all of this can be done at the same location, they travel, which produces traffic. To plan an efficient day, many decisions have to be made by each person. They decide where to perform activities, which mode to choose to get from one location to another, in which order and at which time activities should be performed, with whom to perform certain activities, and so on. Some decisions are made hours (days, months) in advance while others are made spontaneously as reactions to specific circumstances. Furthermore, many decisions induce other decisions. Therefore, it is important to model the complete time horizon of the decision makers.

Transport simulation models should be able to implement (at least part of) such an individual decision horizon and assign the outcome to a traffic model, since it is the complete daily schedules (and the decisions behind that) that produce traffic. This chapter presents such a micro-simulation, called MATSim-T (Multi Agent Transport Simulation Toolkit), implemented as a Java application, usable on any operating system. The work is part of the research project MATSim (http://matsim.org). In the chapter here the focus lies on design and implementation issues as well as on computational performance of different parts of the system. On the basis of integrated (daily) individual demand optimization in MATSim-T, the system is extended such that it provides flexible handling of a large variety of input data; extensibility of models and

algorithms; a simple interface for new models and algorithms; (dis)aggregation for different spatial resolutions; robust interfaces to third party models, programs, and frameworks; unlimited number of individuals; and an easily usable interface to handle new input data elements.

This chapter lays the focus on the modules. It analyzes how specific modules affect the functionality of the toolkit as well as how they affect the overall computational speed of the complete system. The chapter starts with an overview of MATSim-T and related work (Sec. 2). This is followed by sections about the modules of the iterative part of MATSim-T: the traffic flow simulation (Sec. 3), the scoring and plans selection modules (Sec. 4), the re-planning (Sec. 5), and finally a comprehensive view at the whole iterative process (Sec. 6). Sec. 7 sketches the computational demand of the initial demand generation. Although that process runs at the beginning of the study, some aspects of it are easier to explain after the iterative part of MATSim is laid out. The chapter closes with an overview of the current development processes which will enhance the system in size, speed and functionality.

2 OVERVIEW

2.1 MATSim-T

The term *multi agent micro-simulation* is used with different meanings in transport research. Often, the word "microscopic" is used to describe a "car following model" (e.g. Wiedemann, 1974) that is also used in some commercial products (e.g. VISSIM: PTV, 2008). In MATSim, the term is used to describe that each modeled person contains its completely individual settings. Each person is modeled as an *agent*, and the sum of all agents should reflect the statistically representative demographics of the region. The demand is modeled and optimized individually for each agent—not only for some parts of the demand like departure-time and route choice, but as a complete *temporal dynamic* description of the daily demand of each agent.

The demand of an agent is called a *plan* in MATSim. Figure 1 shows an example of one agent's daily plan, written in XML (W3C, 2008). This structure stays the same during all modeling and simulation of the demand. In particular, the assignment of the traffic demand does not only take single trips into account, but the complete daily plans—including the activities—are executed. Thus the term *micro-simulation* relates to the microscopic (individual) demand of each person in the scenario.

To produce individual plans for each agent with MATSim as shown in Figure 1 it is necessary to provide the user interfaces such that he/she is able to generalize and fuse the data available for the region of interest, so that a general dataset of the infrastructure, the population, and the demand can be created. To structure the process of demand creation/optimization, MATSim-T can be split up into four parts as shown in Figure 2:

- Scenario creation process
- Initial individual demand modeling process
- Iterative demand optimization process (including demand execution, scoring, and replanning)
- Post-process analysis

Since MATSim-T is a modular approach, all parts shown in Figure 1 (FUSION, IIDM, EXEC, SCORING and REPLANNING) are given as interfaces such that users are able to plug in their own modules.

```
<plans name="example plans file">
  ...
  <person id="393241" sex="f" age="27" license="yes" car_avail="always"
     employed="yes">
    <travelcard type="regional-abo" />
    <plan>
      <act type="home" link="58" start_time="00:00" dur="07:00" end_time="07:00" />
      <leg mode="car" dept_time="07:00" trav_time="00:25" arr_time="07:25">
        <route>1932 1933 1934 1947</route>
      </leg>
      <act type="work" link="844" start_time="07:25" dur="09:00" end_time="16:25"/>
      <leg mode="car" dept_time="16:25" trav_time="00:14" arr_time="16:39">
        <route>1934 1933</route>
      </leg>
      <act type="home" link="58" start_time="16:39" dur="07:21" end_time="24:00" />
    </plan>
  </person>
  ...
</plans>
```

Figure. 1: Description of the demand of a synthetic person (including demographic data) for a complete day. The agent with ID 393241 plans to leave home—located on link 58—to travel to his work place. He uses a route leading along 4 nodes (5 links) with an expected travel time of 25 minutes. The agent stays at work for 9 hours, then travels back home with an expected travel time of 14 minutes. The demand does not only describe single parts of the day, but the complete sequence for agent 393241 continually in time. – Source: Balmer, 2007.

The first two processes rely on the available data of the region of interest. Since the quality, quantity, and resolution of data can vary a lot from one scenario to another, the scenario creation and the initial demand modeling process steps can vary as well. MATSim-T therefore provides in its core only the resulting data representation of the infrastructure (network and facilities) and the population including each person's individual demand, plus parsers and writers for the XML data representation.

To clarify the functionality of a FUSION or IIDM module, here is an example: Let us assume that land-use information about the region of interest is given based on the resolution of municipalities, and that the number of work places is given for each municipality. The user implements a MATSim-FUSION module that parses this information and creates one facility (including the number of workplaces) per municipality. This gives a rough approximation of the existing work facilities and work places in the region. Let us now assume that at a later stage of the project the user has access to detailed buildings data including work facilities. The system allows one to add another module that replaces the already created work facilities with the new information and distributes the number of work places to the more realistic work facilities of that region.

While the resulting facilities of *both* situations are suitable for MATSim-T to start the third step of the overall process (demand optimization; Figure 1), the second version of the facilities delivers more detailed results. Even though the two described modules are implemented for a specific scenario, they can be part of the MATSim toolkit and therefore, another user with the same needs is able to reuse the modules for his/her own scenario.

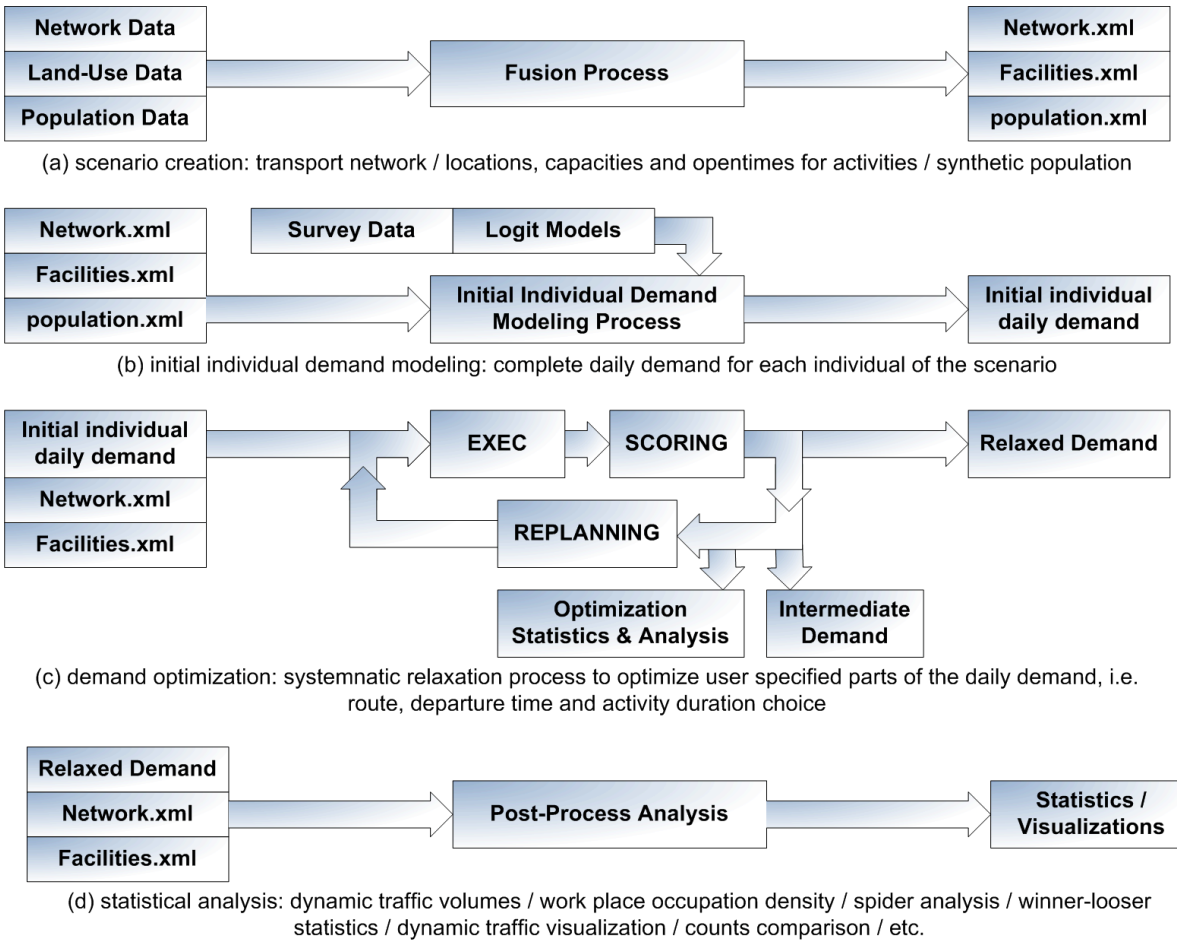

Figure 2: Process structure of MATSim-T

The post-process analysis part of MATSim-T (fourth part of Figure 2) works in the same way, with the difference that now the input data follows MATSim standards (MATSim XML formats of the network, facilities, population and demand) and therefore is useable for any given scenario.

The *iterative demand optimization* process (third part of Figure 1) is, in a way, the core of MATSim-T. While all other steps are run once in a sequential order defined by the user, part three optimizes the demand for each individual synthetic traveler in the scenario such that they respect the constraints (network, facilities) of the scenario and the interaction with all the other actors of that region.

Usually, a method of relaxation is used to find an equilibrium state. For route choice the Wardrop equilibrium (Wardrop, 1952) describes such a relaxed state. But importantly, not only the routes are optimized in MATSim-T. Instead, the complete daily plan—including routes, times, locations, sequence of activities, activity types, and so on—of each agent is optimized. Each agent tries to execute its day with highest possible utility. The utility of a daily plan depends on infrastructural constraints (capacity of streets, opening times of shops, etc.) and on the daily plans of the other agents in the system. This implies that the effective utility of a daily plan can only be determined by the interaction of all agents. This is the place where *co-evolutionary algorithms* (Holland, 1992; Palmer et al, 1994) come into play. An evolutionary algorithm basically consists of the following steps:

1) `Initialize P(t=0)` — Create the population of individuals at time t=0
2) `Score P(t)` — Calculate the "fitness"
3) `Select P'(t) out of P(t)` — "survival of the fittest"
4) `Recombine and mutate P'(t)` — "crossover" and "mutation"
5) `P(t+1) = P'(t); t = t+1` — the next generation of individuals
6) `GOTO item 2`

Applied to the demand optimization (optimization of daily plans) in MATSim, this means:

1) Initialize / generate the daily plans for each agent in the system
2) Calculate the utility of the execution of the individual daily plans for each agent
3) Delete "bad" daily plans (the ones with a low utility)
4) Duplicate and modify daily plans
5) Make those plans the relevant plans for the next iteration; increase the iteration counter by one
6) Goto 2.

It is important to note that the "individuals" of the evolutionary algorithms are the plans, while the synthetic travelers are the entities that *co*-evolve.

Figure 2(c) shows this optimization loop. For each of the steps listed above, specific modules are available. The execution of the daily plans (EXEC) is handled by a corresponding *traffic flow simulation* module, in which the individuals interact with each other, i.e. individuals may generate congestion on streets of high usage. The SCORING module calculates the utility of all the executed daily plans. Plans with a high utility (high "fitness") survive, while plans with a low utility (e.g. caused by long travel times because of traffic jams) are eventually deleted.

The creation and variation of daily plans (REPLANNING) is distributed among different modules that are specialized on varying specific aspects of daily plans. The modifications in the plan of a single agent are completely independent on the re-planning of all the other agents' plans.

2.2 Related Work

Many models have implemented the concept of activity based demand generation (e.g. "VISEM": PTV, 2008; Vovsha et al, 2002; Bowman et al, 1999, Bhat et al, 2004; Pendyala, 2004; Arentze et al, 2000). But the results are typically delivered as (time-dependent) origin-destination matrices, which are used as input for static or dynamic traffic assignment models. Completely agent-based micro simulations (e.g. "mobiTopp": Schnittger & Zumkeller, 2004)) are typically focused on telematics aspect or on effects of changes in infrastructure. Event driven simulations for transport planning (e.g. Axhausen, 1988; Balmer & Nagel, 2006; Axhausen & Herz, 1989) already presented the powers of micro-simulations, but they usually only work on small scenarios.

The work most related to the MATSim project is TRANSIMS (2008), which also generates individual activity schedules for large-scale scenarios. While the concepts are similar, there are some important differences. The most important differences are:

- MATSim is consistently constructed around the notion that travelers (and possibly other objects of the simulation, such as traffic lights) are "agents", which means that all information for the agent should always kept together in the simulation at one place. In this way, an agent in MATSim can access demographic characteristics or time pressure while he/she is moving around in the transport system. In TRANSIMS, such information is in principle available, but fragmented between many modules and many files.

- As a mirror of the coherent agent information, MATSim uses the hierarchical XML (W3C, 2006) format for the input or output of agent information. Because the file format is hierarchical, it can be filled out with different levels of detail. This means that in *all* places where agent information is exchanged between modules, the same file format is used. This has two important consequences: (i) *Arbitrary* modules can be combined to fill out the agent information. In TRANSIMS, the capabilities of the modules are given implicitly by the file formats. (ii) One DTD (Document Type Definition, see W3C, 2006) is sufficient to ensure correctness of all agent data files.
- As a consequence of the agent design, it is easy to maintain several plans per agent. This facilitates to interpret the iterative part of MATSim as a co-evolutionary algorithm, where every agent draws on a population of plans in order to find better solutions for him-/herself. Once more, this could be emulated in TRANSIMS, but it would be considerably more difficult to implement it, and in some sense the only option may be to add something similar to the MATSim agent database (Raney & Nagel, 2004) to TRANSIMS.
- The traffic flow simulations currently used in MATSim-T are simpler than that in TRANSIMS, and as a result run considerably faster, thus allowing meaningful runs in days instead of weeks. This is not really a conceptual difference, but it was an important design decision when starting MATSim: Iterations should essentially run over night.

Agent-based micro-simulation applications can also be found in related research fields to transport planning. Promising concepts in urban planning are land-use simulations, i.e. URBANSIM (Waddell et al, 2003), ILUTE (Salvini et al, 2005) or the models of Abraham/Hunt (Hunt et al, 2000).

2.3 Case Study ("all-of-Switzerland")

From a user point of view, it is of high interest how much time a simulation program needs to spend until results are produced. This chapter will present the performance measures of the toolkit on a typical large-scale transport planning study. Meister et al (2008) present the first results for the daily traffic for the whole of Switzerland created with MATSim-T. That case study will be used to present the computational performance of each part of the toolkit. The extents of the Swiss daily traffic demand study are:

- The national planning network ("Nationales Netzmodell": Vrtic et al, 2003) consists of ~24'000 nodes and ~60'000 links.
- Based on the enterprise census 2000 (SFSO, 2001) and the census 2000 (SFSO, 2000), ca. 1.7 million facilities are modeled. Up to five different activities ("home", "work", "education", "shop" and "leisure" activity) are assigned to each facility.
- With the census 2000 and the microcensus 2005 (SFSO, 2006), about 7 million synthetic persons (agents) are generated, incl. demographic attributes like age, gender, car license ownership, car availability, public transport ticket ownership and employed status.
- The generation of the initial, individual, time-dependent daily demand is described in detail in Ciari et al (2007) and Meister et al (2008). Overall, about 22 million trips are generated—about 7.1 million trips for motorized individual transport.
- The performance measures are produced on a machine with 8 dual-core processors with 2.2 GHz clock rate each.

- The case study needs about 22 GByte of RAM.

2.4 Computing times

This chapter concentrates on the MATSim architecture and the resulting computing times. The above scenario is close to the largest that is currently feasible. Since it is possible to obtain plausible results with runs with 10% of the population, this means that scenarios up to 70 million people can currently be addressed. If hardware keeps improving in similar ways as in the past, simulating even large mega-cities or "all-of-Europe" seems within reach.

Computing times are given with respect to that specific scenario. Unfortunately, it has turned out consistently that finding simple predictive rules for the computational performance of MATSim is quite difficult (Nagel & Rickert, 2001; Cetin et al, 2003; Cetin, 2005). This has to do with the fact that interwoven aspects of hardware, implementation, scenario details, and scenario size play a role. For example, hardware, implementation and scenario size together determine how much of a scenario fits into cache or memory, and if the computation is I/O- or CPU-bound. Scenario details decide, say, during how much of the simulated time there is activity in all parts of the system (as opposed to activity on a small number of links). It might be possible to give worst-case complexities. These, however, in our experience are completely unrelated to the actual computing times. This chapter rather gives computing times for a specific scenario, plus information on how these times change when important aspects, such as the number of travelers or the number of network elements, change.

3 TRAFFIC FLOW SIMULATION

Despite considerable work over more than the last decade (e.g. Nagel & Schleicher, 1994; Nagel & Rickert, 2001; Gawron, 1998; Cetin et al, 2003; Charypar et al, 2007), the traffic flow simulation remains the module with the largest computing requirements for the problem at hand. The traffic flow simulation is responsible for executing the daily plans in a physical environment. In principle, arbitrary models could be used, e.g. the model by Wiedemann (1974) or a cellular automata model (e.g. Nagel & Schreckenberg, 1992), but both require still too large amounts of computing power. Transport planning is not so much interested in the detailed driving behavior, but in the dynamic amount of traffic, traffic that reflects traffic jams, tailbacks, the dissolving of traffic jams, etc. The queue model (Gawron, 1998) fulfills all these requirements. Every street is modeled as a queue in which vehicles have to wait for at least the free speed travel time on that street. In addition, both the flow and the storage capacity of each link are limited. The former causes congestion, the latter causes spillback since links can become full and then upstream links also become jammed.

The traffic flow simulations produce information about where each agent is at a specific time of the day and what it is doing at that time. Each agent generates for each of its actions (begin/end of an activity, entering or leaving a link, etc) a temporally and spatially localized *event*.

3.1 Default traffic flow simulation

The current default traffic flow simulation of MATSim-T is a single CPU Java re-implementation of the micro-simulation described by Cetin (2005). As an integral part of the toolkit it has the advantage that it can directly access all the data in the MATSim object database, saving time-

consuming input and output of data. Because of the platform independence of Java, it runs on all major operation systems.

The default traffic flow simulation uses seconds as smallest entity of time. For each simulated second, all queues (all links of the network) synchronously get a new state assigned. As a result, the runtime is proportional to the number of links in the scenario:

$$T_{mobsim} \propto t_{sim} N_{links} / \Delta t$$

where t_{sim} is the real time window to be simulated (usually 1 day = 86,400 seconds), Δt the size of the time step (1 second), and N_{links} the number of links in the street network. There is, however, also some overhead to generate events, which depends on the number of agents in the system. Performing the Mobility Simulation on a 2.2 GHz processor, the computation time to simulate one day of the complete vehicular traffic of Switzerland (see above) takes about 70 minutes (ca. 20.5 times faster than real time).

The simulation performance in a naïve implementation of the queue model does not depend very much on the number of agents, respectively on their demand: Every link is processed once in every time step. This is acceptable for situations where all network elements are in use (e.g. morning rush hour), but the simulation will take just as long calculating low traffic during nightly hours. The current implementation in MATSim-T, however, switches off links that are completely empty, saving additional computing time but making it now even more dependent on the number of agents and their demand structure.

3.2 Deterministic, Event-based Queue-Simulation (DEQSim)

DEQSim, an alternative traffic flow simulation, extends the queue-model. In addition to the FIFO (First-In, First-Out) behavior of the queue model, this traffic flow simulation imitates backwards-traveling gaps produced by vehicles that leave congestion. This leads to more realistic dynamics of congested links. Also the implementation differs. Rather than updating all links every second, it only operates whenever a link actually changes its state. Despite the improved dynamics, such state changes are rare. In a pure queue model, the state of a link only changes when a vehicle enters or a vehicle leaves, and since the earliest possible leaving time is known for every vehicle, the link can be processed at exactly those times. It was possible to add the improved dynamics in a similar way, by adding "holes" that travel backwards, and that have, in consequence, also pre-computed times of when they arrive at the upstream end of the link. Therefore, computing time is only used when agents produce *events* on links. As a side effect, the simulation does not have to stick to discrete time steps anymore. A detailed description of the DEQSim can be found in Charypar et al (2007). The performance is

$$T_{deqsim} \propto e(a, N_{links})$$

where the number of events e is proportional to the number of agents (a), respectively the number of executed plans, and depends on the street network (number of links N_{links}). On a high-resolution network of the same region, an agent's route contains more links than on a low-resolution network, thus generating more events.

For the case study described above, 162 million events are generated. The total computing time for the single-CPU implementation of the DEQSim takes about 50 minutes (real time ratio =~ 28). Additionally, the DEQSim also runs in parallel using multiple CPUs with distributed

memory. The performance scales nearly linearly with the number of processors, implying that in 8 CPUs, the 50 minutes are reduced to less than 7 minutes.

In contrast to the default traffic flow simulation, DEQSim is written in the *C++* programming language. This prevents the direct access to the data from the traffic simulation. Instead, the data needed to run the DEQSim is first written to disk and later read by DEQSim. Similarly, DEQSim writes its events to a file on disk, from where the MATSim toolkit reads them after DEQSim has finished. This file input and output (including the processing of the read in events) requires an additional 20 minutes in the given case study. The input is proportional to the number of agents, the output once more proportional to the number of the events. Maybe somewhat surprisingly, the main overhead does not stem from the physical disk I/O, but from the handling of the events while they are processed inside MATSim.

4 SCORING AND PLANS SELECTION

The events produced by the traffic flow simulation make it possible to calculate the effective utility of each daily plan, including the influences and effects of the interaction of other agents. The "success" of a daily plan is specified by an individual utility function. This function describes the goals of each agent, and with that its behavior. In principle, any arbitrary utility function could be used, for example one coming from *prospect theory* (Avineri & Prashker, 2003). MATSim currently uses a simple but effective utility function described in Charypar and Nagel (2005). It is related to the Vickrey bottleneck model (Vickrey, 1969; Arnott et al, 1993), but is modified in order to be consistent with the approach based on complete daily plans (Charypar & Nagel, 2005; Raney & Nagel, 2006).

Without going into detail, the elements of the utility function are:

- A positive contribution for the (usually) positive utility earned by performing an activity.
- A negative contribution (penalty) for traveling.
- A negative contribution for being late.

Intuitively, being early should also be punished, but it turns out that this is not necessary since "doing nothing" is already indirectly punished by the fact that something with a positive utility could be done instead in a better plan.

The utility function induces the behavior of the agent, because the agent searches in the solution space of the utility function for the best possible score, which implies the best possible daily plan. The agent cannot optimize outside of the solution space. This aspect is documented in more detail later.

Scores are computed in two ways, depending on the type of the traffic flow simulation:

- In the case of the integrated (default) traffic flow simulation, scores are computed when events from the traffic flow simulation reach the scoring module. The computational effort to compute the scores is smaller than the overhead caused by the events handling mechanism. Any effort to accelerate the computation at this end would need to accelerate the events handling mechanism first.
- In the case of the external DEQSim traffic flow simulation, scores are computed when Java events that are generated from the events file reach the scoring module. This ends up being the same problem: The main computational effort is caused by the events handling mechanism.

There is, thus, a computational cost of the events handling mechanism, that is either hidden in the default traffic flow simulation, or in the file I/O when the events are read from file. This may be an element of future improvements.

A small, but important step in the whole process is the deletion of a "bad" plan. As there are new plans generated in each iteration for a subset of all agents, the population of plans per agent increases up to a user-defined maximum (typically between 3 to 6 plans per agent). Before a new plan can be created for an agent that already has as many plans as the maximum defines, the worst plan (the one with the lowest score) is deleted from the population. As a consequence, only "good" plans survive. This step takes about 10 seconds for the all-of-Switzerland study.

5 PLANS VARIATION (RE-PLANNING)

The re-planning is responsible for making sure that every agent explores its solution space. This happens by duplicating an existing plan of an agent, varying (mutating) the copy, and executing and scoring it in the next iteration. Each re-planning module takes charge for a specific part in the optimization process. As an example, the *Router* module calculates the routes of a plan based on the amount of traffic from the last traffic flow simulation. The Time Allocation Mutator module modifies departure times and activity durations of a daily plan. This module varies the corresponding times randomly. Additional modules could change activities' locations, or change the sequence of activities. An important fact is that all these modules work *independently* from each other. This allows one to add an arbitrary number of re-planning modules to the optimization process.

A characterization of modules is whether they modify a plan randomly (*Random Mutation*) or whether they search for the best solution based on the results of the last traffic flow simulation (*Best Response*). The former has the advantage not to use any significant amount of computing power. Additionally, it searches—sooner or later—over the complete search domain. The disadvantage is that such modules require (too) many iterations until the optimization relaxes. Best Response modules on the other hand help to relax the system much faster, but they are usually more complex and computationally intensive.

5.1 Time Allocation Mutator

The Time Allocation Mutator is a typical example of a *Random Mutation* module. It varies randomly the departure times and durations of activities in a daily plan. The Time Allocation Mutator needs about 2 seconds to process the 10% re-planning agents (approx. 220 000 agents) per iteration.

5.2 Router Module

The Router Module calculates the best routes in a daily plan, given the departure times for each leg and the dynamic travel times of *all* streets (based on the last traffic flow simulation). The best route is defined as the one with the least negative utility. This *Best Response* module uses the complete and dynamic traffic load of the system for finding routes.

Currently, MATSim has three different implementations of the Router module. They are all based on a time-dynamic variant of Dijkstra's algorithm for finding shortest paths in networks, and they return all the identical results. Our newest implementation, the Landmarks-A* module (Lefebvre & Balmer, 2007), gives the best performance in average: For the given case study it

needs in about 0.1 milliseconds to calculate one route in average. For the 7.1 million (motorized) routes of the "all-of-Switzerland" scenario and 10% route replanning rate this implies 71 seconds of computing time per iteration, which can, however, be shared between parallel CPUs.

Additional computational results for different routing algorithms and different networks sizes can be found in the paper by Lefebvre & Balmer (2007). Unfortunately, those results are not sufficient to make a prediction about the functional form of the average complexity of the Landmarks-A* implementation; the most probable fit may be $O(n^2)$ where n is the number of network nodes. It also plays a role that the Java implementation of the priority queue does not offer a fast "decrease-key" operation.

5.3 "planomat"

Another *Best Response* module available in MATSim is *planomat*, described in full detail in Meister (2004). This module not only optimizes one aspect of a daily plan, but all parts at the same time. It bases its assumptions heavily on the outcome (*events*) of the last execution of the traffic flow simulation (see above). Additionally, it is able to coordinate the daily plans of members of the same household (e.g. a common dinner at home). This module is written in C++, but can be called from the MATSim toolkit.

The C++-planomat is a genetic algorithm (GA) with a special encoding for activity sequences, activity locations, activity times, and activity participation. The encoding was constructed with the idea that a plan that is "good in the morning" and another plan that is "good in the afternoon" should be able to combine into a plan that is "good overall". This takes some input from the GA coding of the traveling salesman problem. One instance of the GA generates the plan(s) for one person or one household. As is common, the GA is not a particularly fast method to solve the problem, but it is extremely flexible with respect to the inclusion of additional constraints, for example facility opening times.

A simplified version of the planomat—written in JAVA—is an integral part of the MATSim toolkit and optimizes the time schedules. It is therefore a substitute of the *Time Allocation Mutator*. planomat uses an evolutionary algorithm for the optimization of departure times and activity durations. It is therefore far more computationally intensive than the Time Allocation Mutator module. In the above described case study, it uses about 5.7 milliseconds in average for the best response calculation of timing of a single daily plan. For the ca. 2.3 million (motorized) plans of the "all-of-Switzerland" scenario and 10% *planomat* replanning rate this implies 1331 seconds of computing time per iteration, which can, however, be shared between parallel CPUs.

5.4 Additional modules

It is important to recall at this point that MATSim-T is not limited to the modules described above. Any user can add his or her own modules; additional modules are also added by the developers. The computational performance of such modules will be assessed in due time when such modules have proven their value with respect to the transport simulation problem.

6 SYSTEMATIC RELAXATION OF THE EVOLUTIONARY ALGORITHM

According to the user's needs it is now possible to combine all the previously mentioned modules. The optimization process, i.e., the iterative processing of single tasks, is done by the toolkit.

However, with respect to the combination of modules one aspect has to be considered: Each additional re-planning module enlarges the solution space for the agent's day-plan. It is required that this solution space is completely covered by the utility function. Consider the following example:

If an agent is only allowed to optimize its route it would be feasible to reduce the above described utility function to

$$U_{total} = \sum_i U_{travel,i}$$

since this agent would not be capable to alter its time allocation.

However, if one adds a time allocation module, and therefore enlarges the solution space, this has to be considered by the utility function. On the other hand, it is legitimate to use the extended utility function for agents that consider only route choice, since it covers the complete solution space. On this account, the further functional development of the optimization process in MATSim-T (implementation of new re-planning modules) goes hand in hand with the extension of the agents' behavioral models.

In the following the relaxation behavior and the required computational time of the co-evolutionary algorithm will be analyzed.

6.1 Setup

For the suitable analysis of the relaxation process a typical and in the last years frequently used setup is used:

1. Each agent is capable of route-choice.
2. Each agent is capable of time allocation choice.
3. In each iteration, a randomly selected sample of 10% of all agents creates a new plan by altering the routes of an existing plan.
4. In each iteration, a randomly selected sample of 10% of all agents creates a new plan by altering the time allocation of an existing plan.
5. The remaining 80% of agents select an existing plan for repeated execution. The selection probability corresponds to the logit function $p_i = \exp(\beta S_i)/\sum_j \exp(\beta S_j)$, where S_i denotes the utility of plan i and β is an empirically estimated constant.
6. The utility function corresponds to the one given in the previous section.
7. The number of plans per agent is limited to a maximum of four.
8. The system will be considered relaxed once the trajectory of average utility per iteration represents a stationary process.

A detailed description of this setup with values for the parameters of the utility function can be found in Meister et al (2008).

6.2 Relaxation

The relaxed state of the co-evolutionary algorithm of MATSim-T is reached if the utility for each agent does not noticeably change through variation of the day plans. Since bad plans do not "survive", the utility of all remaining plans levels off eventually. Figure 3 depicts such a behavior. The light grey curve represents the utility of the plan that has been executed in the corresponding iteration averaged over all agents. The black and the medium grey curve, respectively, denote the average utility of the currently available best and worst plan, respectively. One can realize that in this example the utility converges to the "relaxed" state after iteration 70, and exhibits only a mean variance of approx. 2 units in iteration 100.

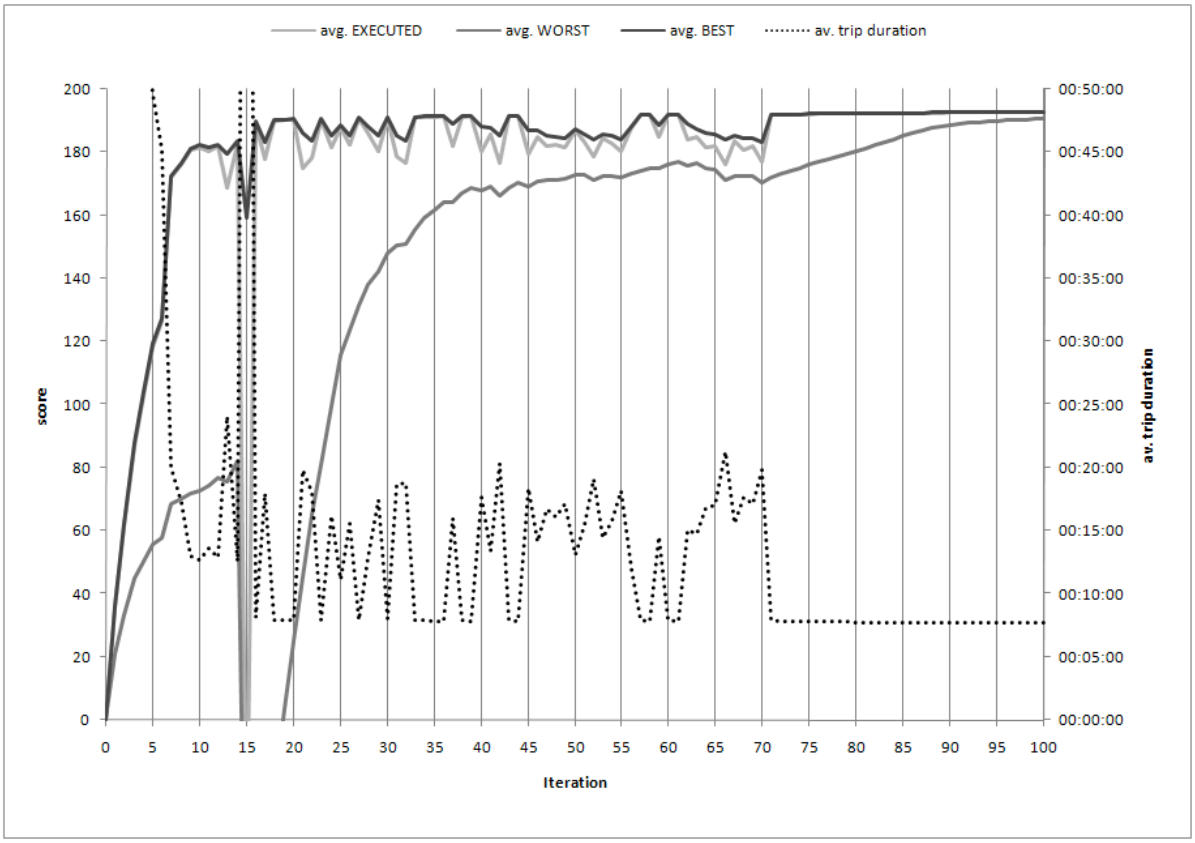

Figure 3: Average utility (score) and average trip travel time per iteration

More noticeable is the behavior before iteration 70, especially in iteration 15. Figure 3 shows that the average scores for the executed plans (light grey curve) as well as for the worst plans (medium grey curve) are remarkably low in iteration 15, which can be ascribed to a so-called "network breakdown". Due to the optimization process and the given constraints (such as the time window for the starting a work-activity) it is possible that a lot of agents simultaneously try out similar plans, which in turn leads to high traffic densities on preferred roads and therefore to highly congested situations. Due to this temporal overload, this congestion cannot be absorbed by the surrounding road network due to the overall high traffic density. Spillbacks build up and spread over a large part of the network. The model requires a long time to resolve such congestion, resulting in high travel times, and therefore in large disutility for traveling. Since the

last executed plan exhibits a low utility after such a network breakdown most of the agents switch their plans. Thus the last optimization step is discarded and the usage of more diverse plans will be reinforced. In the paper of Rieser & Nagel (2008) the "network breakdown" situations are analyzed in more detail.

Due to the diversification regarding departure times and route choice, average trips travel times decrease (black dotted curve in Figure 3), which in turn becomes noticeable in the resulting greater average utilities.

It appears that after iteration 70 a combination of plans arises which results in a stable traffic pattern that is robust towards variations of single agents. Good plans are duplicated during re-planning and the duplicates are kept if they also turn out to be good. Bad plans are discarded, so that finally only good plans will remain which can be observed in Figure 3 with the approximation of the medium grey curve to the other two curves.

6.3 Computational time for optimization process

The total computational time of a single relaxation process consists of the sub-processes as described in Figure 2(c). Additional time is required for storing temporal results and analysis (intermediate demand, statistics and analysis shown in Figure 2(c)). This latter feature can be switched off by the user, so that only the final result will be saved. However, this feature helps to analyze the optimization process and allows one to abort the process if needed. For that reason this part will not be excluded in the following discussion. In detail the process can be divided into the following chronological steps:

1. Initialization: Loading and managing of infrastructural data (network and facilities) and initial demand
2. Iteration 0: primary execution of the traffic flow simulation and calculating of utilities
3. Iteration 1 to *n*: re-planning, traffic flow simulation and scoring
4. Iteration 0, 1, 2 and every 10^{th} iteration: saving of temporal results and analyses
5. Finalization: saving of final state (relaxed day-plans)

Additionally to these steps, certain modules require extra computational time for initialization and finalization. For instance, the initialization of the "Landmarks-A*" router module as described earlier takes some seconds for calculating the landmarks (see Lefebvre & Balmer, 2007). The DEQSim requires several minutes for loading the network and individual demand and for storing them in optimized data structures. In case of the parallel DEQSim, additional initialization time is required. Several java internal processes such as the garbage collector and hardware constraints (file I/O) induce additional delays.

Figure 4 shows the contributions of time to the total calculation time for the first 40 iterations of a relaxation process. In this setup the routing is done by eight parallel running "Landmarks-A*" router modules. Time allocation is done by another eight "Time Allocation Mutator" modules. For the execution of the demand the parallel version of the DEQSim with eight threads is used. This setup results in a relaxation behavior as shown in Figure 2(c).

It turns out that the DEQSim requires in average 8-10 minutes per iteration as shown in Figure 4. There is, however, an additional overhead of 20 minutes for data exchange with the other modules of MATSim-T. Re-planning (time allocation and routing) requires about 90 seconds computational time, where the main fraction is consumed by the "Landmarks-A*" routing modules. The re-planning after a "breakdown" situation as shown in Figure 3 causes a significant increase in calculation time for the router (approx. nine minutes; iteration 16). The

cause for this is that the performance of the Landmarks-A* router decreases if link travel times differ significantly between the uncongested and congested traffic state.

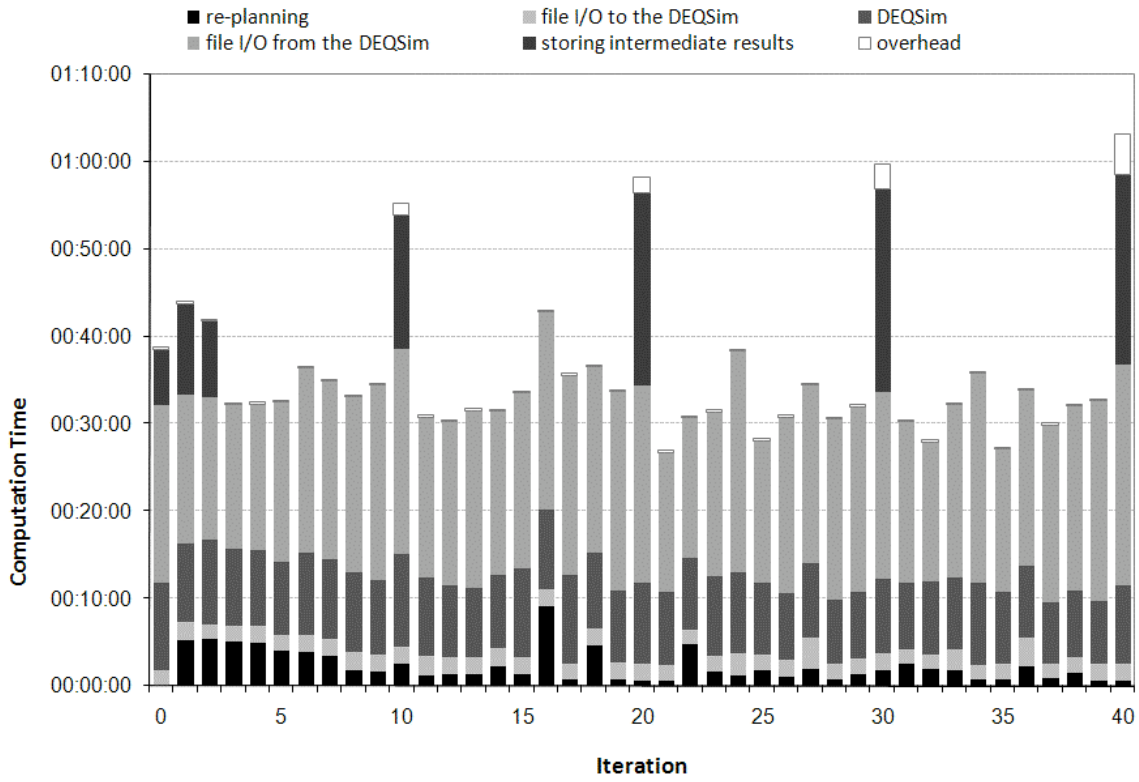

Figure 4: share of the overall computation time by process steps

One iteration of the co-evolutionary algorithm requires about 32 minutes for the calculation of the individual time-variant daily demand consisting of 7.1 million trips on a 60,000 links network of Switzerland. In addition, every 10 iterations 22 minutes are used for saving temporal results and analysis. Taking into account that the system reaches a relaxed state after about 100 iterations, the total time for calculating the resulting demand and the corresponding traffic takes about 3.2 days.

6.4 Combinations

It is possible to run the relaxations faster when using other modules. Table 1 lists a set of possible combinations of modules and their required average and total runtime to reach a stable state. With the replacement of the *random mutation* module (Time Allocation Mutator) with a *best response* module (planomat) a significant reduction of the number of iterations can be achieved. On the other hand, the planomat requires in average 30 minutes computational time per iteration. However, finally this trade-off pays off (the total runtime halves).

If one includes the additional overhead for data exchange between MATSim-T and the DEQSim, the performance of the default traffic flow simulation (in Java) and the DEQSim are

equivalent. With the parallel run of the DEQSim one can achieve a remarkable gain in performance, however, the overhead for file-I/O remains the same.

Finally it is worthwhile to mention that in terms of computational performance the results clearly show the applicability (large scenarios, time-dynamic and detailed) of micro simulations for transport planning.

Table 1: Computing times of different combinations of modules

traffic flow simulation	**routes**	**times**	**# of iterations**	**run time on computer**
DEQSim (1 CPU)	Landmarks-A*	Time Allocation Mutator	100	_5.2 Days
default traffic flow simulation (1 CPU)	Landmarks-A*	Time Allocation Mutator	100	_5.5 Days
DEQSim (1 CPU)	Landmarks-A*	planomat	30	_1.9 Days
default traffic flow simulation (1 CPU)	Landmarks-A*	planomat	30	_2.1 Days
DEQSim (8 CPU)	Landmarks-A*	Time Allocation Mutator	100	_3.2 Days
DEQSim (8 CPU)	Landmarks-A*	planomat	30	_1.5 Days

Module	Average runtime	Remarks
DEQSim (1 CPU)	ca. 70 minutes	ca. 50 minutes DEQSim and ca. 20 minutes I/O Overhead
DEQSim (8 CPU)	ca. 28 minutes	ca. 8 minutes DEQSim and ca. 20 minutes I/O Overhead
default traffic flow simulation	ca. 70 minutes	
Landmarks-A*	ca. 1.5 minutes	significantly longer after „break-downs“ (ca. 6–10 minutes)
Time Allocation Mutator	10 sec	
planomat	ca. 22 minutes	

7 INITIAL INDIVIDUAL DEMAND MODELING

In Fig. 2 the initial demand is stated to be a prerequisite for the optimization in MATSim-T. This section describes how the toolkit can be used to create the initial daily demand for each individual. The reason why this pre-process is introduced at the end of this article is that the solution space – as defined by the setup of the optimization – determines which aspects of the plan do not need to be modeled by the pre-process.

Thus the task of the *initial individual demand modeling* is to model aspects of the agents' plans that cannot be handled by the iterative optimization process. To get a best possible mapping of the real demand, this part is built upon knowledge, surveys and socio-demographic data of the investigation area. MATSim-T is built in such a way that it can operate on various types of input data. Depending on the scenario, existing input data can vary in quality, level of detail, and quantity. The modules for the initial demand modeling are adopted correspondingly, or replaced. For this reason the runtime for generating the initial demand varies. Basically, this pre-process is of sequential nature. All required modules need to be used only one time.

The modeling of the individual initial demand for the "all-of-Switzerland" application can be found in detail in Ciari et al (2007) and Meister et al (2008). The required runtime is about 14.4 hours.

MATSim-T operates on disaggregated information, i.e., the infrastructure is based on coordinates rather than aggregates, such as zones (districts, communes, etc.). Activities – and hence the facilities in which activities are preformed – are mapped to the links of the network. Since the network has a particular resolution, it defines the level of detail of the modeling. In other words, ultimately the investigation area has as many zones as the network has (directed) links, in the above case 60,000 zones. For high-resolution networks the number of zones can increase to more than one million. Since the raw data are typically of aggregated nature, they need to be disaggregated. MATSim-T provides several aggregation layers to store such data and to disaggregate them on to facilities, activities and persons if needed.

The modeling of the initial demand can be split up into several steps depending on the available raw data and the user's needs. Each of these processes is implemented in one module. These modules can be arbitrarily used, extended, replaced or skipped.

At each point of time during the modeling process it is possible to output intermediate results. This is important, since it is typically required to statistically validate the results of the model implemented in a module. The intermediate results can be used as input data for further modeling steps.

A further important aspect is the so-called streaming process for the generation of an initial demand. While infrastructural data (facilities, network and even aggregation layers) require relative little memory, the demand (= the initial plans) requires several gigabytes of memory. However, since the demand is generated individually for each synthetic person in the investigation area, it is possible to reduce the memory consumption: One loads the agent into memory, applies the demand-modeling module, writes the demand to file, and frees the memory. Then the next agent is loaded and so on. This allows one to model the individual demand for an unlimited amount of agents on standard consumer hardware. A detailed description of the features of the MATSim-T initial demand modeling can be found in the dissertation of Balmer (2007) and also in Balmer, Axhausen, & Nagel (2006).

8 DISCUSSION AND OUTLOOK

This work shows that the development of the last years considering hardware architecture, CPU performance, and optimization of programming implementations allows one to handle large-scale scenarios for transport planning with agent-based micro simulations in reasonable time. Furthermore it shows that the optimum of performance has not been reached yet. For instance, a re-implementation of the parallel DEQSim in JAVA as an integrated part of MATSim-T would avoid the overhead per iteration caused by the data exchange between DEQSim and MATSim-T (about 20 min for the discussed application), which in turn would decrease the total runtime by about 40%. A scenario of the magnitude of complete Switzerland could be handled in approximately one day.

The setup of the optimization process offers further possibilities of optimization. For instance, it is possible to reduce the number of iterations until the system becomes relaxed by introducing adaptive re-planning rates. Also the re-planning modules offer potential for optimization, in particular the routing module and the planomat. All these optimizations are to be aspired, since further functional extension, such as location and mode choice will certainly consume more computational time, be it because of the complexity of these modules or because more iterations will be required until the system reaches a relaxed state.

Finally it is worthwhile to mention that the results of MATSim-T are not only traffic patterns, but also rather a detailed description on the single agent level. In other words, it is possible to determine for each synthetic person at each point in time where she/he is and what she/he does. Still, the results should not be interpreted on the level of single agents, but rather at the level of aggregated sub-populations.

REFERENCES

Arentze, T., Hofman, F., van Mourik, H., & Timmermans, H. (2000). *ALBATROSS: A multi-agent rule-based model of activity pattern decisions*. Paper 00-0022, Transportation Research Board Annual Meeting, Washington, D.C.

Arnott, R., Palma, A. D., & Lindsey, R. (1993). A structural model of peak-period congestion: A traffic bottleneck with elastic demand. *The American Economic Review*, *83*(1), 161–179.

Avineri, E., & Prashker, J. (2003). Sensitivity to uncertainty: Need for paradigm shift. *Transportation Research Record*, *1854*, 90–98.

Axhausen, K. (1988). *Eine ereignisorientierte Simulation von Aktivitätenketten zur Parkstandswahl*. Ph.D. thesis, Universität Karlsruhe, Germany.

Axhausen, K., & Herz, R. (1989). Simulating activity chains: German approach. *Journal of Transportation Engineering*, *115*(3), 316–325.

Axhausen, K., Zimmermann, A., Schönfelder, S., Rindsfüser, G., & Haupt, T. (2002). Observing the rhythms of daily life: A six-week travel diary. *Transportation*, *29*(2), 95–124.

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

Balmer, M., Axhausen, K., & Nagel, K. (2006). Agent-based demand modeling framework for large scale micro-simulations. *Transportation Research Record*, *1985*, 125–134.

Balmer, M., & Nagel, K. (2006). Shape morphing of intersection layouts using curb side oriented driver simulation. In J. Van Leeuwen, & H. Timmermans (Eds.), *Innovations in Design & Decision Support Systems in Architecture and Urban Planning* (pp. 167–183).

Bhat, C., Guo, J., Srinivasan, S., & Sivakumar, A. (2004). A comprehensive econometric microsimulator for daily activity-travel patterns. *Transportation Research Record*, *1894*, 57–66.

Bowman, J., Bradley, M., Shiftan, Y., Lawton, T., & Ben-Akiva, M. (1999). Demonstration of an activity-based model for Portland. In *World Transport Research: Selected Proceedings of the 8th World Conference on Transport Research 1998*, vol. 3 (pp. 171–184). Elsevier, Oxford.

Cetin, N. (2005). *Large-Scale parallel graph-based simulations*. Ph.D. thesis, Swiss Federal Institute of Technology (ETH) Zürich, Switzerland.

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

Charypar, D., Axhausen, K., & Nagel, K. (2007). An event-driven parallel queue-based microsimulation for large scale traffic scenarios. In *Proceedings of the Word Conference on Transport Research*. Berkeley, CA.

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

Ciari, F., Balmer, M., & Axhausen, K. (2007). Mobility tool ownership and mode choice decision processes in multi-agent transportation simulation. In *Proceedings of Swiss Transport Research Conference (STRC)*. Monte Verita, CH. See www.strc.ch.

de Palma, A., & Marchal, F. (2002). Real case applications of the fully dynamic METROPOLIS tool-box: An advocacy for large-scale mesoscopic transportation systems. *Networks and Spatial Economics*, *2*(4), 347–369.

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.

Hanson, S., & Burnett, K. (1982). The analysis of travel as an example of complex human behaviour in spatially-constraint situation: Definition and measurement issues. *Transportation Research Part A: Policy and Practice*, *16*(2), 87–102.

Holland, J. (1992). *Adaptation in Natural and Artificial Systems*. Bradford Books. Reprint edition.

Hunt, J., Johnston, R., Abraham, J., Rodier, C., Garry, G., Putman, S., & de la Barra, T. (2000). Comparisons from Sacramento mode test bed. *Transportation Research Record*, *1780*, 53–63.

Lefebvre, N., & Balmer, M. (2007). Fast shortest path computation in time-dependent traffic networks. In *Proceedings of Swiss Transport Research Conference (STRC)*. Monte Verita, CH. See www.strc.ch.

Meister, K. (2004). *Erzeugung kompletter Aktivitätenpläne für Haushalte mit genetischen Algorithmen*. Master's thesis, IVT, ETH Zürich. See www.ivt.ethz.ch/docs/students/dip44.pdf.

Meister, K., Rieser, M., Ciari, F., Horni, A., Balmer, M., & Axhausen, K. (2008). Anwendung eines agentenbasierten Modells der Verkehrsnachfrage auf die Schweiz. In *Proceedings of Heureka '08*. Stuttgart, Germany.

Nagel, K., & Rickert, M. (2001). Parallel implementation of the TRANSIMS micro-simulation. *Parallel Computing*, *27*(12), 1611–1639.

Nagel, K., & Schleicher, A. (1994). Microscopic traffic modeling on parallel high performance computers. *Parallel Computing*, *20*, 125–146.

Nagel, K., & Schreckenberg, M. (1992). A cellular automaton model for freeway traffic. *Journal de Physique I France*, *2*, 2221–2229.

Palmer, R., Arthur, W. B., Holland, J. H., LeBaron, B., & Tayler, P. (1994). Artificial economic life: a simple model of a stockmarket. *Physica D*, *75*, 264–274.

Pendyala, R. (2004). *Phased Implementation of a Multimodal Activity-Based Travel Demand Modeling System in Florida. Volume II: FAMOS Users Guide*. Research report, Florida Department of Transportation, Tallahassee. See www.eng.usf.edu/~pendyala/publications.

PTV (accessed 2008). Traffic Mobility Logistics. See www.ptv.de.

Raney, B., & Nagel, K. (2004). Iterative route planning for large-scale modular transportation simulations. *Future Generation Computer Systems*, *20*(7), 1101–1118.

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* (p. 42). London: Routledge.

Rieser, M., & Nagel, K. (2008). Network breakdown "at the edge of chaos" in multi-agent traffic simulations. *European Journal of Physics*. Doi 10.1140/epjb/e2008-00153-6.

Salvini, P., & Miller, E. (2005). ILUTE: An operational prototype of a comprehensive microsimulation model of urban systems. *Network and Spatial Economics*, *5*(2), 217–234.

Schnittger, S., & Zumkeller, D. (2004). Longitudinal microsimulation as a tool to merge transport planning and traffic engineering models - the MobiTopp model. In *Proceedings of the European Transport Conference*. Strasbourg.

Schönfelder, S., Axhausen, K., & Antille, M., N.and Bierlaire (2002). Exploring the potentials of automatically collected GPS data for travel behaviour analysis – a Swedish data source. In J. Mölthen, & A. Wytzisk (Eds.), *GI-Technologien für Verkehr und Logistik – IfGI*, vol. 13 (pp. 155–179). Münster, Germany: Institut für Geoinformatik.

SFSO (2000). Eidgenössische Volkszählung. Swiss Federal Statistical Office, Neuchatel.

SFSO (2001). Eidgenössische Betriebszählung 2001 - Sektoren 2 und 3. Swiss Federal Statistical Office, Neuchatel.

SFSO (2006). Ergebnisse des Mikrozensus 2005 zum Verkehrs. Swiss Federal Statistical Office, Neuchatel.

TRANSIMS (accessed 2008). TRansportation ANalysis and SIMulation System. See transims.tsasa.lanl.gov.

Vickrey, W. S. (1969). Congestion theory and transport investment. *The American Economic Review*, *59*(2), 251–260.

Vovsha, P., Petersen, E., & Donnelly, R. (2002). Microsimulation in travel demand modeling: lessons learned from the New York best practice model. *Transportation Research Record*, *1805*, 68–77.

Vrtic, M., Froehlich, P., & Axhausen, K. (2003). Schweizerische Netzmodelle für Strassen- und Schienenverkehr. In T. Bieger, C. Lässer, & R. Maggi (Eds.), *Jahrbuch 2002/2003 Schweizerische Verkehrswirtschaft* (pp. 119–140). St. Gallen: SVWG Schweizerische Verkehrswissenschaftliche Gesellschaft.

W3C (2006). *eXtensible Markup Language (XML)*. World Wide Web Consortium (W3C). See www.w3.org/XML.

Waddell, P., Borning, A., Noth, M., Freier, N., Becke, M., & Ulfarsson, G. (2003). Microsimulation of urban development and location choices: Design and implementation of UrbanSim. *Networks and Spatial Economics*, *3*(1), 43–67.

Wardrop, J. (1952). Some theoretical aspects of road traffic research. *Proceedings of the Institute of Civil Engineers*, *1*, 325–378.

Wiedemann, R. (1974). *Simulation des Straßenverkehrsflusses*. Schriftenreihe Heft 8, Institute for Transportation Science, University of Karlsruhe, Germany.

Wolf, J., Schönfelder, S., Samaga, U., Oliveira, M., & Axhausen, K. (2004). Eighty weeks of Global Positioning System traces. *Transportation Research Record*, *1870*, 46–54.