Introducing MATSim

CHAPTER 1

Introducing MATSim

Andreas Horni, Kai Nagel and Kay W. Axhausen

1.1 The Beginnings

The MATSim project (MATSim, 2016) started with Kai Nagel, then at ETH Zürich, and his interest in improving his work with, and for, the TRANSIMS (TRansportation ANalysis and SIMulation System) project (Smith et al., 1995; FHWA, 2013); he also wanted to make the resulting code open-source.[1] After Kai Nagel's departure to Berlin in 2004, Kay W. Axhausen joined the team, bringing a different approach and experience. A collaboration, successful and productive for more than 10 years, was thus established, combining a physicist's and a civil engineer's perspective, as well as bringing together expertise in traffic flow, large-scale computation, choice modeling and CAS (Complex Adaptive Systems):

- **Microscopic modeling of traffic:** MATSim performs integral microscopic *simulation of resulting traffic flows* and the congestion they produce (see Section 1.3).
- **Microscopic behavioral modeling of demand/agent-based modeling:** MATSim uses a microscopic description of demand by *tracing the daily schedule* and the synthetic travelers' decisions. In retrospect, this can be called "agent-based".
- **Computational physics:** MATSim performs fast microscopic simulations with 10^7 or more "particles".
- **Complex adaptive systems/co-evolutionary algorithms:** MATSim *optimizes the experienced utilities* of the whole schedule through the co-evolutionary search for the resulting equilibrium or steady state (see Section 1.4).

[1] TRANSIMS has, since then, also become open-source (TRANSIMS Open Source, 2013); but in 2000, it was difficult to procure in Europe.

How to cite this book chapter:
Horni, A, Nagel, K and Axhausen, K W. 2016. Introducing MATSim. In: Horni, A, Nagel, K and Axhausen, K W. (eds.) *The Multi-Agent Transport Simulation MATSim*, Pp. 3–8. London: Ubiquity Press. DOI: http://dx.doi.org/10.5334/baw.1.

At the end of the 1990s, the scene was set for these research streams' mergence into a computationally efficient, modular, open-source software enabling further development on travel behavior, network response and efficient computation: MATSim.

1.2 In Brief

MATSim is an activity-based, extendable, multi-agent simulation framework implemented in Java. It is open-source and can be downloaded from the Internet (MATSim, 2016; GitHub, 2015). The framework is designed for large-scale scenarios, meaning that all models' features are stripped down to efficiently handle the targeted functionality; parallelization has also been very important (e.g., Dobler and Axhausen, 2011; Charypar, 2008). For the network loading simulation, for example, a queue-based model is implemented, omitting very complex and computationally expensive car-following behavior (see Section 1.3).

At this time, MATSim is designed to model a *single day*, the common unit of analysis for activity-based models (see, for example, the review by Bowman, 2009a). Nevertheless, in principle, a multi-day model could be implemented (Horni and Axhausen, 2012b).

As shown in Section 1.4, MATSim is based on the co-evolutionary principle. Every agent repeatedly optimizes its daily activity schedule while in competition for space-time slots with all other agents on the transportation infrastructure. This is somewhat similar to the route assignment iterative cycle, but goes beyond route assignment by incorporating other choice dimensions like time choice (Balmer et al., 2005b), mode choice (Grether et al., 2009), or destination choice (Horni et al., 2012b) into the iterative loop.

A MATSim run contains a configurable number of iterations, represented by the loop of Figure 1.1 and detailed below. It starts with an initial demand arising from the study area population's daily activity chains. The modeled persons are called agents in MATSim. Activity chains are usually derived from empirical data through sampling or discrete choice modeling. A variety of approaches is suitable, as evidenced in the scenarios' chapters (cf. Chapter 52). During iterations, this initial demand is optimized individually by each agent. Every agent possesses a memory containing a fixed number of day plans, where each plan is composed of a daily activity chain and an associated score. The score can be interpreted as an econometric utility (cf. Chapter 51).

In every iteration, prior to the simulation of the network loading with the MATSim *mobsim (mobility simulation)* (e.g., Cetin, 2005), each agent selects a plan from its memory. This selection is dependent on the plan *scores*, which are computed after each mobsim run, based on the executed plans' performances. A certain share of the agents (often 10 %) are allowed to clone the selected plan and modify this clone (*replanning*). For the network loading step, multiple mobsims are available and configurable (see Horni et al., 2011b, and Section 4.3 of this book).

Plan modification is performed by the *replanning* modules. Four dimensions are usually considered for MATSim at this time: departure time (and, implicitly, activity duration) (Balmer et al.,

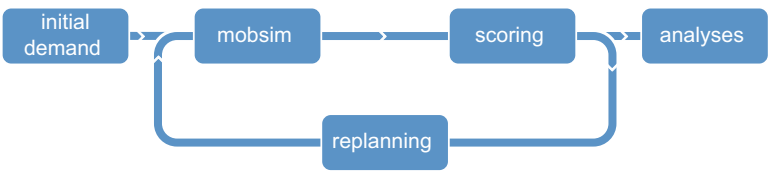

Figure 1.1: MATSim loop, sometimes called the MATSim cycle.

2005b), route (Lefebvre and Balmer, 2007), mode (Grether et al., 2009) and destination (Horni et al., 2009, 2012b). Further dimensions, such as activity adding or dropping, or parking and group choices are currently under development and only available experimentally. MATSim replanning offers different strategies to adapt plans, ranging from random mutation to approximate suggestions, to best-response answers where, in every iteration, the currently optimal choice is searched. For example, routing often is a best-response modification, while time and mode replanning are random mutations.

Initial day chains do not have to be very carefully defined for the replanning dimensions included in the optimization process. Plausible values just speed up the optimization process.

If an agent ends up with too many plans (configurable), the plan with the lowest score (configurable) is removed from the agent's memory. Agents that have not undergone replanning select between existing plans. The selection model is configurable; in many MATSim investigations, a model generating a logit distribution for plan selection is used.

An iteration is completed by evaluating the agents' experiences with the selected day plans (*scoring*). The applied scoring function is described in detail in Chapter 3.

The iterative process is repeated until the average population score stabilizes. The typical score development curve (Figure 1.2, taken from Horni et al., 2009) takes the form of an evolutionary optimization progress (Eiben and Smith, 2003, Figure 2.5). Since the simulations are stochastic, one cannot use convergence criteria appropriate for deterministic algorithms; for a discussion of possible approaches for the MATSim situation, see Sections 47.3.2.2 and 48.2 as well as Meister (2011).

MATSim offers considerable customizability through its modular design. Although implementing alternative core modules, such as an alternative network loading simulation, may entail substantial effort, in principle, every module of the framework can be exchanged. MATSim modules are described in Chapter 5 and following.

MATSim is strongly based on events stemming from the mobsim. Every action in the simulation generates an event, which is recorded for analysis. These event records can be aggregated to evaluate any measure at the desired resolution. The event architecture is detailed in Section 45.2.5.

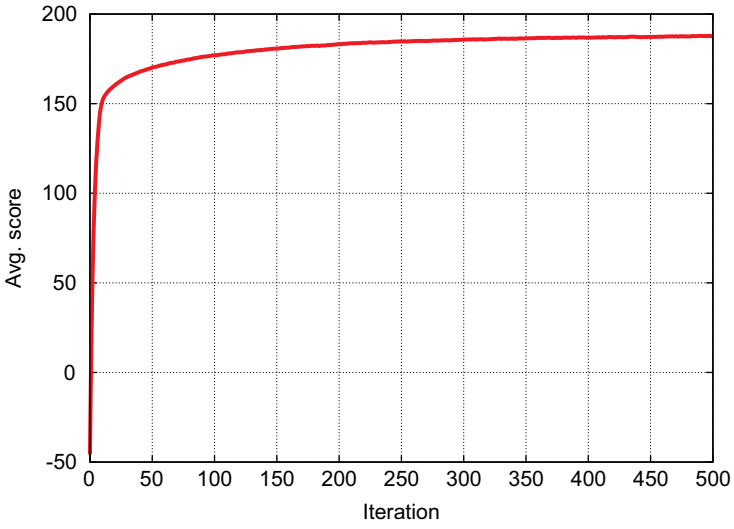

Figure 1.2: Typical score progress.

1.3 MATSim's Traffic Flow Model

MATSim provides two internal mobsims: QSim and JDEQSim (Java Discrete Event Queue Simulation); in addition, external mobility simulations can be plugged in. Some years ago, the DEQSim written in C++ and described by Charypar (2008); Charypar et al. (2007b,a, 2009) was plugged into MATSim and frequently used. The multi-threaded QSim is currently the default mobsim.

Charypar et al. (2009) distinguishes between

- physical simulations, featuring detailed car following models,
- cellular automata, in which roads are discretized into cells,
- queue-based simulations, where traffic dynamics are modeled with waiting queues,
- mesoscopic models, using aggregates to determine travel speeds, and
- macroscopic models, based on flows rather than single traveler units (e.g., cars).

As MATSim is designed for large-scale scenarios, it adopts the computationally efficient queue-based approach (see Figure 1.3). A car entering a network link (i.e., a road segment) from an intersection is added to the tail of the waiting queue. It remains there until the time for traveling the link with free flow has passed and until he or she is at the head of the waiting queue and until the next link allows entering. The approach is very efficient, but clearly it comes at the price of reduced resolution, i.e., car following effects are not captured. In JDEQSim, for computational reasons, the waiting-queue approach is combined with an event-based update step (Charypar et al., 2009). In other words, there is no time-step-based updating process of any agent in the scenario. Instead agents are only touched if they actually require an action. For example, links do not have to be processed while agents traverse them. Update events triggering is managed by a global scheduler. QSim, however, is time-step based. The MATSim traffic flow model is strongly based on the two link attributes: storage capacity and flow capacity. Storage capacity defines the number of cars fitting onto a network link.

Flow capacity specifies the outflow capacity of a link, i.e., how many travelers can leave the respective link per time step. It is an individual attribute of the link. The current implementation of QSim has no *maximum* inflow capacity specified. In contrast, in the earlier DEQSim and current JDEQSim, an inflow capacity can also be specified, which may move jams at merges from the end of the first common link, where the QSim generates them, upstream to where the links merge and where they plausibly should be (Charypar, 2008, p. 99). However, additional data is needed for this, which is often not available.

This basic traffic flow model has been extended with various modules: Signals and multiple lane modeling have been added (Chapter 12), backward-moving gaps, as investigated by Charypar (2008), are included in JDEQSim, but only available on an *experimental basis* for QSim (Section 97.5). Interactions between different modes are described in Section 4.6 and Chapter 21.

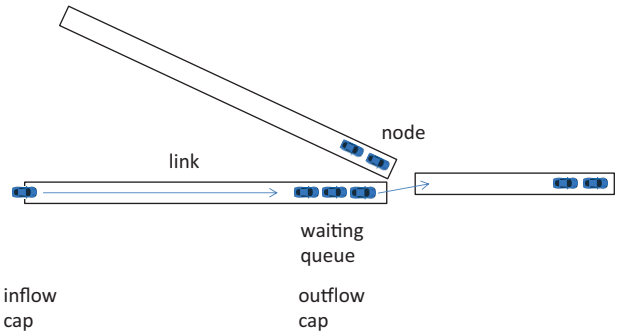

Figure 1.3: Traffic flow model.

1.4 MATSim's Co-Evolutionary Algorithm

As illustrated in Figure 1.4, the MATSim equilibrium is searched for by a *co-evolutionary algorithm* (see, e.g., Popovici et al., 2012). These algorithms co-evolve different species subject to interaction (e.g., competition). In MATSim, individuals are represented by their plans, where a person represents a species. With the co-evolutionary algorithm, optimization is performed in terms of agents' plans, i.e., across the whole daily plan of activities and travel. It achieves more than the standard traffic flow equilibria, which ignores activities. Eventually, an equilibrium is reached, subject to constraints, where the agents cannot further improve their plans unilaterally.

Note that there is a difference between the application of an evolutionary algorithm and a *co*-evolutionary algorithm. An evolutionary algorithm would lead to a system optimum, as optimization is applied with a global (or population) fitness function. Instead, the co-evolutionary algorithm leads to a (stochastic) user equilibrium, as optimization is performed in terms of *individual* scoring functions and within an agent's set of plans.

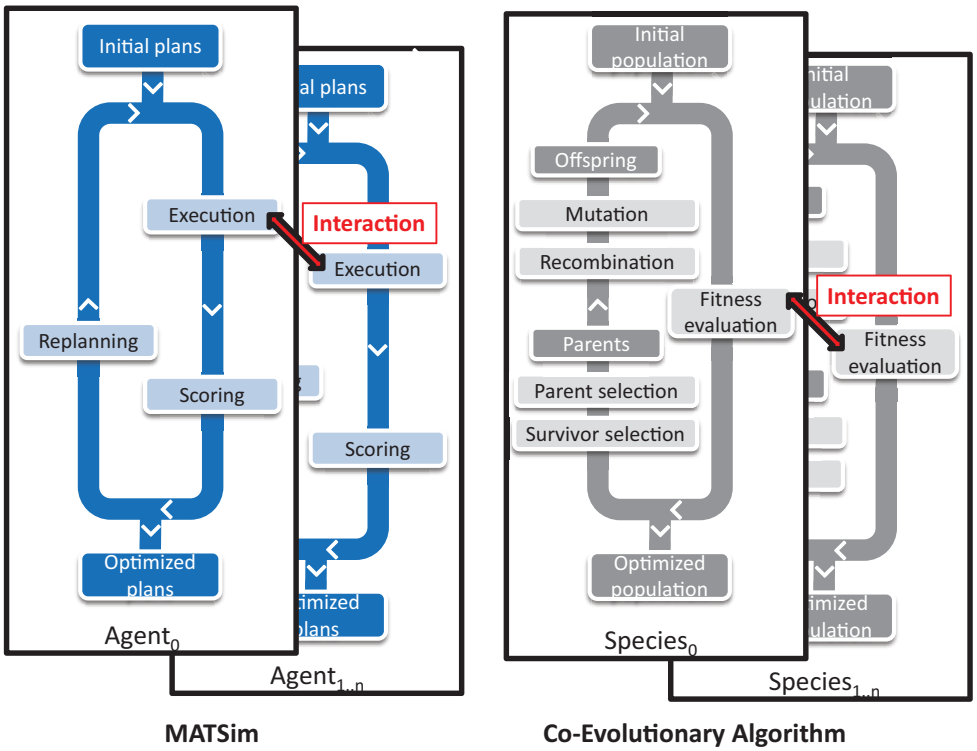

Figure 1.4: The co-evolutionary algorithm in MATSim.