Adding Transit to an Agent-Based Transportation Simulation
Concepts and Implementation

vorgelegt von

Dipl. Inf.-Ing. ETH Marcel Rieser

aus Affeltrangen (TG), Schweiz

von der Fakultät V, Verkehrs- und Maschinensysteme
der Technischen Universität Berlin
zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigte Dissertation

Promotionsausschuss:

Vorsitzende: Prof. Dr.-Ing. Chr. Ahrend
Gutachter: Prof. Dr. rer. nat. Kai Nagel
Gutachter: Prof. Dr.-Ing. Kay W. Axhausen

Tag der wissenschaftlichen Aussprache: 10. Juni 2010

Berlin 2010

D 83

Contents

List of Figures

List of Tables

Abstract

Transportation simulations are an important part of today's decision making process for transport infrastructure and management. While proposed changes are getting more and more complex, tools supporting the decision making process are struggling to keep up. Traditional, flow-based traffic assignment tools are limited in the number of different person groups that can be distinguished and do in most cases not offer fully time-dynamic results. Newer technologies like agent-based simulations overcome these problems, but usually demand high computational resources, even for simpler or mid-sized scenarios. In addition, agent-based simulations often take much longer to run than traditional assignment models, slowing down the work of transportation planners, but delivering more detailed results in exchange.

MATSim is an open-source, agent-based transportation simulation that is able to simulate large-scale scenarios within a useful amount of time. But MATSim is limited to simulate private car traffic only, trimming its capabilities to answer sophisticated questions posed to advanced models. An example of such an advanced question is given in this dissertation by the research of time-dependent road pricing schemes. The research showed that the introduction of a toll during the evening rush hours has a distinct influence on the traffic patterns during the morning rush hour. But as only car traffic was included in the model, it is not clear if the observed reaction in the model is indeed realistic. In reality, people could decide to switch to transit instead of continuing to commute with their private car; an outcome that could not be modeled with a simulation limited to private car traffic only.

To overcome this problem, this dissertation describes in detail how the existing simulation, MATSim, was extended to support additional modes of transportation. A mode choice model is integrated into the iterative structure of the simulation framework, effectively combining mode choice, departure-time choice and route choice together with the traffic assignment in the simulation. This is in

strong contrast to the way mode choice is handled in the four-step model traditionally used in transportation planning. The new mode choice model is mathematically validated on the basis of a simple test network, and then applied to a large-scale scenario.

The support for additional modes of transportation is then refined, by implementing a detailed simulation of schedule-based transit into the existing car simulation. Was the first extension more on the level of the complete framework to support additional modes, is this extension on the level of the actual traffic simulation. The required changes and applied extensions are explained in detail, highlighting computational issues in order to increase the computational requirements only as much as needed. Illustrative examples are given, demonstrating the features of the integrated simulation, as well as a large-scale application showing the applicability of the integrated simulation in a large, urban context.

Kurzfassung

Die Simulation von Verkehr ist ein wichtiger Bestand im Entscheidungsfindungsprozess für verkehrliche Massnahmen. Während die zur Diskussion stehenden Änderungsvorschläge immer komplexer werden, haben die zur Verfügung stehenden, unterstützenden Hilfsmittel immer mehr Mühe, mit der zunehmenden Komplexität mitzuhalten. Programme für die traditionelle, flussbasierte Umlegung von Verkehr unterstützen nur eine limitierte Anzahl an unterscheidbaren Personengruppen. Zudem bieten sie in den allermeisten Fällen keine wirklich zeit-dynamischen Resultate. Neuere Technologien wie agentenbasierte Simulationen haben keine Schwierigkeiten damit, erfordern aber bedeutend mehr Rechenressourcen, selbst für einfachere oder kleinräumige Szenarien. Zusätzlich benötigen agentenbasierte Simulationen oftmals markant längere Rechenzeiten als herkömmliche Umlegungsmodelle. Dies mag zwar die Arbeit von Verkehrsplanern erschweren, dafür stehen anschliessend bedeutend genauere Daten zur Verfügung.

MATSim ist eine agentenbasierte Verkehrssimulation, deren Programmcode frei verfügbar ist (*open source*) und die in der Lage ist, grossräumige Szenarien innert nützlicher Frist zu simulieren. Leider ist MATSim aber zur Zeit auf die Simulation von privatem Autoverkehr beschränkt, was den Nutzen der Simulationsumgebung stark beschneidet. Gerade anspruchsvollere Fragestellungen von Verkehrsmodellen lassen sich so nicht mehr ohne Weiteres beantworten. Ein Beispiel einer solchen Fragestellung sind die Auswirkungen einer zeitabhängigen Maut, wie sie in dieser Dissertation vorgestellt werden. Im Laufe der Untersuchung hat sich gezeigt, dass die Einführung einer Strassenmaut im abendlichen Stossverkehr klar erkennbare Auswirkungen im morgendlichen Verkehr hat. Weil aber nur Autoverkehr im Simulationsmodell enthalten ist, lässt sich nur schwer beurteilen, wie realistisch die beobachtete Reaktion ist. Es wäre gut denkbar, dass Personen entscheiden, mit dem öffentlichen Verkehr zur Arbeit zu fahren anstatt weiterhin mit dem privaten Auto zur Arbeit zu pendeln. Ein solches Resultat ist aber mit einer auf Autoverkehr beschränkten Simulation gar nicht möglich.

Um diese Beschränkung aufzuheben beschreibt diese Dissertation detailliert, wie das vorhandene Simulationswerkzeug, MATSim, erweitert wurde um zusätzliche Verkehrsmittel zu unterstützen. Ein Modell zur Verkehrsmittelwahl wurde in den iterativen Ablauf der Simulationsumgebung integriert. Dadurch werden die Verkehrsmittelwahl, Abfahrtszeitwahl und Routenwahl zusammen mit der Verkehrsumlegung kombiniert – im starken Gegensatz wie die Verkehrsmittelwahl im Vier-Stufen-Verfahren bislang in der Verkehrsplanung eingesetzt wurde. Die integrierte Verkehrsmittelwahl wird mathematisch mit Hilfe eines einfachen Test-Netzwerks validiert, bevor es dann in einem grösseren Szenario eingesetzt wird.

Die Unterstützung zusätzlicher Verkehrsmittel in der Simulationsumgebung wird dann weiter verbessert, indem fahrplanbasierter öffentlicher Verkehr detailliert abgebildet und in die Verkehrssimulation integriert wird. War von der ersten Modellerweiterung hauptsächlich die integrative Schicht der gesamten Simulationsumgebung betroffen, ist von dieser zweiten Erweiterung im Wesentlichen nur noch die eigentliche Verkehrssimulation betroffen. Die notwendigen Anpassungen und eingesetzten Erweiterungen werden ausführlich beschrieben, insbesondere um sicherzustellen, dass der benötigte Rechenaufwand nur um soviel ansteigt wie unvermeidbar ist. Die neuen Möglichkeiten der Simulation werden zuerst in einfachen Beispielen vorgeführt. Danach erfolgt die Anwendung der kompletten Simulation auf ein grossräumiges Szenario, um ihre Anwendbarkeit auch in einem grossen, städtischen Kontext zu zeigen.

1 Introduction

Traffic is everything — Without traffic, civilization could not have developed in the way we know it today. Traffic enables us to move goods, information and ourselves between different locations. But as important as traffic is, it does not mean that traffic must not be questioned. Far from it! Today's civilization is very dependent on functioning transport systems. Without traffic, today's civilization would not be possible.

The advent of motorized cars, electrified trains, and air planes during the twentieth century had a big influence on the behavior of people. Fast transportation of persons is no longer restricted to a (rich) minority, but also available to middle and lower income people. This gave new possibilities to the complete population, especially in urban regions with a dense network of infrastructures or services. No longer was it necessary to live close—most likely within walking distance—to the workplace. Instead, people could live at one end of a city and work at the other end, or even live outside the city boundaries and commute to the city for work. Goods could be fabricated in one city, and more easily shipped to other places in a country or even world-wide. All in all, the new and improved means of transportation opened a multitude of great possibilities and showed great growth potential for the civilization.

But the increase in travel and transportation, and thus in traffic, soon started to cast some shadows over the initial success. Infrastructure had to be expanded quickly, taking up more and more space (see Fig. 1.1). Noise, filth pollute the environment, maybe leading people to give up ancestral living places. Instead, cities spread further away from their center, leading to the so-called urban sprawl. Traffic—or its required infrastructure—actually shaped the landscape and influenced the development of cities.

Figure 1.1: Road infrastructure in Portland, OR; view from South to city center

In recent years, the focus has shifted: *While cities were shaped by traffic before, now cities try to shape their traffic*. No longer are just new highways built as a mean of combating against traffic jams. No longer is the personal claim to empty roads self-evident. Instead, a plenitude of regulations try to guide future demand for transportation capacity into manageable and sustainable ways. Urban sprawl, for example, is tried to be reduced by encouraging the building of multistory, multifamily houses. The use of private cars for commuting to work is made less convenient by reducing the number of public parking lots in city centers, or by requesting high parking fees in these areas for long-term parking. In addition, the number of cars used to commute to work is tried to be lowered by giving incentives to vehicles with high occupancies, e.g. by the use of exclusive high-occupancy vehicle (HOV) lanes. In addition, the attractiveness of public mass transportation, transit, is increased by introducing modern, rapid transport systems with high convenience standards such as low-floor vehicles and big windows. Often, such

services are strongly subsidized, sometimes even offered for free in certain areas, to convince people to use it.

Changes to existing transportation infrastructure, or the building of new infrastructure, need to be based on well-informed decisions. Not only will tax-payers watch very closely how their money is spent, but bad decisions may have lasting negative effects on the economy; a trauma for politicians. On the other hand, good measures may decrease the amount of time lost in traffic congestion enormously, leading to high economic gains. But how can good and bad measures be recognized in advance? How can politicians and stakeholders profoundly decide between several variants of a project?

Transportation planning is getting more and more complex. As people tend to commute longer and longer distances, the area that must be included in a scenario is getting bigger. Many systems already operate at their capacity limit, so changes must be carefully implemented to avoid break downs of the system. In addition, changes are less and less restricted to one transportation mode only, but often try to influence the mode share as well, resulting in more complex models.

Agent-based simulation models have proved to be very useful in transportation modeling. In such models, every person is represented by an object in software, a so-called agent. Each agent can have its own values for a given set of attributes, leading to very detailed description of a model-population. For this reason, such simulations are sometimes also called to be *microscopic*. In agent-based simulations, each agent may decide on its own—influenced by its specific set of attributes—how it behaves (within a given rule set). For this decision making process, additional simple or complex models could be used, e.g. route choice or departure-time choice models.

Current agent-based, microscopic tools are often limited to a single transportation mode only, or are only applicable to small areas. Traditional tools for transportation planning—which are not agent-based—on the other hand may support multiple transportation modes, such as private car and transit, and be applicable to large-scale scenarios. But the models themselves are not that detailed, such that typical questions nowadays can no longer be always answered by these tools.

This dissertation researches a way how an agent-based transportation simulation can be extended to support not only private car traffic, but also other modes of transportation, especially transit, while still being able to simulate large-scale scenarios with millions of agents.

The dissertation is organized as follows: The next chapter gives a short overview of public transportation systems, highlighting similarities and differences of different transit systems found in different parts of the world. It analyzes what kind of

data is used to describe transit services, and what data is required or useful to simulate it. Chapter 3 looks at existing transport simulations and assignment models, with a strong focus on the ability of these tools to model transit. The chapter also takes a close look at other agent-based simulations. The chapter following it introduces MATSim (Multi-Agent Transport Simulation), an open-source, agent-based simulation framework. The design of the framework is presented in detail, as it builds the technical base for all further chapters. Chapter 5 presents a case study, where MATSim is used to simulate time-dependent road-pricing strategies. This chapter shows the strengths of agent-based transportation simulations, but also the weaknesses of the existing MATSim toolkit, supporting only car traffic. Hence, chapter 6 describes a way how the existing simulation framework can be extended to support additional modes of transportation besides car, while the chapter following it completely focuses on the detailed simulation of transit. Chapter 8 presents examples and applications of the transit simulation, before chapter 9 concludes the thesis, giving an outlook of possible follow-up work and summarizing the key findings of this work.

2 Public Transportation Systems

World-wide, there exist a variety of different public transportation systems, each with its own characteristics. This chapter takes a short look at some of the most common ones and analyzes what data is required to operate in those transportation systems, drawing conclusions at what data is required for successful simulation of such public transportation systems. The chapter will focus mostly on urban public transportation, as that one is of bigger interest for the planned simulations than the long-distance travels between cities.

2.1 Definitions

When speaking about public transportation systems, or even travel in general, many different terms and expressions are used to describe certain characteristics of the systems or of their usage. This short section defines the terms commonly used in this dissertation.

A *transit system*, or public transportation system, contains several *transit lines*, run by different *operators*. It is designed for the mass transportation of people. Sometimes, the term transit system also includes additional parts like federal regulations, the available vehicle fleet, detailed pricing structure. This dissertation concentrates mostly on transit systems from a user's view, thus neglecting these additional parts the most time.

A *transit line* is commonly understood as a regularly available connection between two end points. Between the two end points, transit vehicles circulate along a given *route*. Along that route, additional locations can be specified besides the two end points where people can enter or exit the vehicles. The locations where the vehicles usually stop are referred to as *stop locations*, sometimes also as stop facilities. Examples of such locations range from airports and huge central train stations

to simple bus stops, marked probably only by a pole with a sign on it. Transit lines often have some kind of identification. Bus and train lines often use numbers or letters for simple identifications, but also the name of colors is not unusual, for example for subway lines. In a few rare cases, transit lines may have more than two end points that are regularly served. This may happen for example when the demand decreases significantly after a certain stop (e.g. in a larger village). Then, only every second vehicle of a line might drive further, serving additional stops. This measure is often used to increase the cost-effectiveness of public transport in rural areas.

The transit vehicles that drive on a transit line often follow a specific *transit schedule*. A schedule defines at what time a transit vehicle should depart from or arrive at a certain stop location. Often, the schedules are more dense (i.e. more available connections per hour) during the rush hours and in the city center, and are less dense in the off-peak hours (e.g. during night) and in rural areas.

Passengers are the users of a transit system. They use the offered transit services on their trips from one location to another. A *trip*, sometimes also referred to as *leg*, is a journey undertaken by a person to get from one location (e.g. home) to another location (e.g. work place). One trip consists of one or more *stages*. A stage describes a continuous part of a trip where a person uses one specific mode of transportation. The first stage is often done by walking, while a second stage may be undertaken in a vehicle (e.g. bus, or private car), and the trip ends with a walking-stage again. If a passenger has to change transit lines to get from one place to another, each travel with a vehicle of one line relates to a separate stage. So, a transit trip is usually a combination of several walking and transit stages.

2.2 Public Transportation in Developed Countries

2.2.1 Overview

In practically all European countries as well as typically in most developed countries, each major city has one or a small number of accredited public transportation companies that offer their services [28]. Additionally, there is usually one national railroad company or a small number of privately held companies that receive subsidies from a government to operate a network for long-distance travels, e.g. between cities.

It is common amongst the transportation companies that they operate on the basis of schedules. Usually, these schedules are published so they can be used by passengers for trip planning purposes. Only in rare situations, e.g. if the headway

on a line is very short (usually the case in subways during peak hours), the detailed schedules are not published, as they would not be of any help to the users. But even in those cases, the operators internally need schedules for assigning rolling stock and drivers, so it is safe to assume that almost all public transportation in developed countries is based on schedules.

Pricing of public transportation offerings varies greatly. While in some places pay-per-trip schemes are in use with fixed amounts per trip (e.g. trams and buses in Toronto, buses in Kyoto), many other cities have distance-based fares (e.g. the subways in Washington and New York) or zone-based fares (e.g. Berlin, Zurich). But even within this categorization, a lot of different realizations can be observed. In Washington, for example, the price per distance also depends on the time of day. And while both Berlin and Zurich have zone-based fares, the first one only differentiates three zones while the latter one has 45 different zones. In addition to zone-based fares, it is not unusual that short-distance tickets are also available. These tickets complement the regular, zone-based fares and are mostly used for short trips in the neighboring area of two zones.

Especially with zone-based fares, it is not unusual to have alliances of several transit operators with a common fare structure. This should help people to connect easily between transit lines from different operators and simplify the transit usage. Examples of such alliances are the "Zürcher Verkehrsverbund – ZVV" [126] in Zurich that allows customers to use the same ticket for buses, streetcars, trains and even ships operated by nearly 50 different companies, or the "Verkehrsverbund Berlin-Brandenburg – VBB" [116] that makes it possible to use one ticket to travel on the networks of over 40 operators.

2.2.2 Data Requirements

The existing agent-based simulation MATSim is heavily based on so-called plans: Detailed descriptions, when, where and why agents plan to be at a certain location or how they want to transfer from one place to another. It is easy to extend the current agents' plans to be suitable for describing public transportation trips. Instead of specifying the order of links or nodes the agent should drive along in its car, one could record the stop where the agent wants to access a transit vehicle, the transit line the agent wants to travel along, and the exit stop.

The simulation needs additional information to make use of the data in the agents' route descriptions. Table 2.1 gives an overview of data typically required to operate public transportation systems that are of interest to transportation simulations. The data can be differentiated between infrastructure data and opera-

Table 2.1: Data for operating public transportation systems in developed countries of interest for transportation simulations

Data item	Required for simulation	Required for routing
Network (road, rail, …)	yes	no
Stop locations	yes	yes
Transit lines	yes	yes
Routes of transit lines	yes	yes
Travel time along a transit route	no	yes
Departure times of vehicles for the transit routes	yes	yes
Information about vehicles	no	no
Driver schedules	no	no
Pricing schemes, fares	no	no

tional data. Network, stop locations and vehicles describe available infrastructure that usually cannot be easily adapted, whereas transit lines, their routes, travel and departure times can be changed by the operators within the bounds set by the available infrastructure.

Not all data is *required* for the simulation, but all the data mentioned in Table 2.1 could be used by the simulation to make it more realistic. Absolutely necessary for the simulation are the network and the location of stops from the infrastructure side, and information about lines, routes and departures from the operations side. The travel time along transit routes could be used for validation and analysis purposes, whereas detailed vehicle information as well as driver schedules would help to make the simulation more realistic in terms of capacities, vehicle load, maximum speed and effects of traffic jams on drivers' plans. Pricing schemes are important for economic evaluations. While they are not strictly required for a single simulation, their inclusion could improve the optimization process of the agents (see Sec. 4.7).

2.3 Public Transportation in Developing Countries

2.3.1 Overview

Public transportation systems for long distance travel in developing countries have often a strong resemblance to systems in developed countries. They also operate on lines, and often also have some kind of schedule (how good they are able to follow their schedule should not be discussed here). One possible explanation for this similarity may be the reason that such systems were often established by western people during colonial supremacy in the corresponding countries.

For short-distance trips, especially within cities, the situation often looks quite different. In many cities, a plethora of buses, mini-vans or other (mostly motorized) vehicles offer some kind of public transportation, sometimes also called paratransit. For the untrained tourist, it is often very hard to find out if there actually are transit lines or if the buses just follow random routes through the streets. Looking for printed description of routes is usually as futile as looking for departure time tables, as it is not uncommon for drivers to just depart when a certain vehicle load has been reached.

But more detailed observations often show that there are patterns in the transit offers. For example in Gauteng (the smallest province in South Africa, containing the two cities of Johannesburg and Tshwane/Pretoria), different operators have agreed on each serving some routes only, but not to compete on the same routes [58]. There are still no schedules, but there are clearly defined routes. In Padang (Indonesia), own observations have shown that all the buses have somewhere a line-number hidden. For the locals, this is enough to know which bus they can enter in the city-center to travel outbound, but for tourists it is still not of big use without a directory to look up the lines.

Most of those paratransit systems operate on the regular street network without real stop locations. The vehicles stop at the next possible place when there is demand for. Sometimes, certain locations become a kind of de facto stop location, e.g. important places in a city center or terminal locations where the vehicles turn around. Fares are often directly given to the driver, a fixed amount per ride is the default in many cities, without a differentiated pricing structure.

2.3.2 Data Requirements

Although the traffic may look quite different in cities of developing countries compared to cities in developed countries, the data the operators must have—maybe only implicitly in their knowledge—is basically the same as listed in Table 2.1 for

the developed countries. All the traffic operates on a network infrastructure, within urban regions mostly on roads, while for long-distance transportation rails are a frequent alternative. Stop locations are of higher importance for traffic between cities, but not so much for the paratransit within urban regions. Transit lines and routes seem to be distinguished as well in many places, and vehicle-data and driver schedules may be in use as well, although likely on a much simpler level. It is just the time data (expected travel time, departure time) that may be completely missing in an organized form for paratransit, but again may be used for long-distance traffic.

3 Related Work

3.1 Transportation Simulation in General

Traditional traffic assignment models are getting more and more complex, to a point where a mathematical formulation of the problem is often no longer feasible. Especially the request for dynamic (time-dependent) results adds a very big additional layer of complexity to these models. For such cases, simulation of a problem is in many cases the easier way—or at least the better understandable way—to get results. Vovsha and Bradley [118] list many reasons why trip-based models—and with that most models of the traditional four-step process—are no longer suitable and explain how activity-based models together with agent-based simulations may lead into the future of transportation planning.

Agent-based simulations, where elements (like single travelers) of the simulation are represented by so-called agents that have attributes and sometimes also some logic, are a very intuitive way to simulate traffic. One has to note that the term "agent-based" is a rather general expression, as it does not specify which parts of the simulation are modeled as agents, and which are not. In addition, as also mentioned in [57], the term sometimes also just refers to the object-oriented design of the simulation software.

Java [56], as a comparatively simple object-oriented programming language, has quickly become an often used language for developing agent-based simulations. To some extent this may be attributed to the fact that Java runs without problems on multiple computing platforms and offers built-in visualization commands that also work on all major operating systems. Java-Applets that can be embedded in web sites made the language even more attractive in its early years. Miller et al. [70] show an early attempt to use Java to simulate objects moving around on graphs. Repast, the Recursive Porous Agent Simulation Toolkit [97], offers a complete

framework written in Java to simplify the implementation of agent-based models.

Benenson et al. describe in [12] a minimalistic framework for simulations in an urban context, that makes heavy use of object-oriented software modeling. They analyze the different entities in the real-world and show how these entities can be transfered to an agent-based computer model. Jakovljević and Basch [54] use agents not only for the travel demand, but also for roads and nodes to support micro-, meso- and macrosimulation within the same environment.

One problem of agent-based simulations is their scalability. While the simulations often consist of simple concepts and models, the complexity arises from the interaction of the single agents. For large-scale studies, the number of agents can easily surpass a hundred-thousand or even a million, assuming that every traveler in an urban region is modeled as a single agent. The sheer number of agents quickly becomes a problem concerning memory consumption and computing time. New simulations often not only add more features, but strive also for new code architectures that allow larger scenarios (e.g. [101]). To speed up the computation, parallelism is becoming a more and more important aspect of simulations, especially as computers do not get faster per se, but can just work on more work items at the same time ("The free lunch is over", see [110]). Yoginath and Perumalla [125] show how they speed up a traffic simulation using parallelism. Especially, they use reverse computation-based optimistic execution to overcome problems synchronizing the multiple, parallel executions. Charypar [18] makes use of special model characteristics to minimize the number of synchronizations required in order to gain a high speed-up from the parallelization.

Another way to solve scaling issues is proposed by Jayakrishnan et al. [57], who combine microscopic and macroscopic models to forego problems with computational limitations. They show how the proposed concept can be used for simulating different kinds of traffic, having emergency vehicles or commercial fleets in mind. More focused on transit is the micro-simulation presented by Cortés et al. [27], which is implemented in PARAMICS. Their simulation handles fixed route systems with predefined stops and frequencies as well as other "flexible transit systems" like paratransit. Also as a module to PARAMICS, MILATRAS (Microsimulation Learning-Based Approach for Transit Assignment) [122] models behavioral responses of passengers under information provision, but handles only transit traffic. The conceptual design of the framework resembles MATSim (see Sec. 4).

3.2 Multi-Modal and Transit Simulation

The simulation of multi-modal traffic and transit has to be considered as very similar, because transit traffic usually consists of more than one mode, especially in urban contexts. Still, there are many simulation and assignment models that deal with one mode only, especially railways. In addition, most models concentrate on frequency- or schedule-based transportation systems as can be found in cities of developed countries. Nuzzolo and Crisalli [83] give an overview of recent developments in the schedule-based modeling of transportation systems.

Current research in this area can be grouped into four parts, which are looked at in more detail in the following subsections. It must be noted though that not all projects belong to one group only. For example, most transit assignment models (see Sec. 3.2.1) need some kind of routing algorithm, which are looked at in more detail in Sec. 3.2.2. The grouping is thus based more on the research focus of the work presented, and less on the applications shown the research is embedded in.

3.2.1 Transit Assignment

The first group, transit assignment, adapt network assignment models to work with transit traffic. Traditionally, such assignments do not include real temporal dynamics, but calculate a network equilibrium. Based on the long history of network assignment models—and therefore based on the experience with such models—many commercial software products like EMME/2 [7], VISUM [91] and OMNITRANS [85] offer some kind of transit assignment.

Many of the transit assignments models work on frequencies only, but not on the actual schedules. De Cea and Fernández [31] present a static, frequency-based transit assignment that respects congestion of passengers at transit vehicles and stop facilities, but does not explicitly differentiate between multiple transport modes.

In frequency-based models, walk links that act as access or egress to the transit network are often modeled with a very high or infinite frequency. In addition, passenger congestion is often modeled using volume-delay functions, known from traditional traffic assignment methods. Papola et al. [88] use FIFO (first-in, first-out) queues in their transit assignment model to overcome the problems in calculating the generalized costs for passengers when volume-delay functions are used at waiting/boarding links.

Using frequencies imposes additional problems, like the common line problem: If two or more lines share a line segment, the frequency of the single lines is not that relevant for people traveling only on that common line segment. Nor can a common frequency be derived from two frequencies without the actual schedule.

Schedule-based assignment forgoes such problems, but adds more complexity to the assignment. Poon et al. [89] for example show a model that calculates the route choice based on schedules, and combine it with a user equilibrium assignment that uses the method of successive averages (MSA). A similar approach in [50] also uses MSA for the iterative assignment on a time-expanded network.

Fernandez et al. [34] attempt to solve the network equilibrium problem for models with combined modes, i.e. with trips that use more than just one mode, like "park'n ride". While the models are cleanly laid out and analyzed, the number and complexity of the formulas presented implicitly question how well even more complex models can be solved analytically. A combined departure and route choice model using transit schedules, is presented by Nguyen et al. [79]. The model also respects vehicles' capacities, resulting in possible boarding failures for passengers.

3.2.2 Multimodal Route Choice

Multimodal route choice is most often also solved using network assignment algorithms, but explicitly includes different modes. It has a strong background in freight transports, where combinations of road, rail and water must be considered to deliver goods to their destination. One has to note that in most of these models the different modes do not interact with each other except than in terminal stations. This differs largely from multi-modal road traffic, where cars, buses, bikes, and also streetcars and pedestrian may influence each other.

Dafermos [30] shows an early approach solving the multi-modal network equilibrium problem, considering multi-modal lanes where different traffic influences each other. Fernandez et al. [34] try to solve the network equilibrium model analytically for trips that include more than one mode. They cleanly differentiate between decisions that the demand model has to make, and decisions the network/route model is responsible for. It is noteable that they mention the problem that both models should use the same generalized cost function in order to maintain compatibility between the different models. Although both examples may be seen as an early attempt with limited, computational resources, they seem mathematically rather complex, showing the limitations of a non-simulation based approach.

Southworth and Peterson [108] describe in detail how a multimodal network can be constructed and what specialties have to be taken into account. Although the presented application is for freight network modeling, the same ideas can also be applied for generating multimodal networks for person transportation. For example, it describes in detail how access to the network can be handled, and

how transfer terminals—comparable to transit stop locations—can be modeled. Pallottino and Scutella [87] use space-time networks for dynamic shortest-path calculation, whereas Lo et al. [64] use state-augmented multi-modal networks to especially look at transfers between different modes. State-augmented networks are based on regular transit networks, but expanded in their geometry to reflect usually internal states in such a way that the networks can be used with common traffic assignment or network analysis procedures.

But a suitable network alone is not enough. Modesti and Sciomachen [71] describe in detail how a generalized cost function must be designed to be used for finding shortest paths in multimodal transportation networks. They give an algorithm how the designed utility function can be used together with Dijkstra's shortest path algorithm to find the desired paths in multimodal transportation networks, using as example a network of Genoa, Italy, consisting of 60 nodes and 402 links.

While for many assignment models only small-scale applications with a few lines are shown, Nielsen and Frederiksen [80] describe a transit assignment based on timetables, and specifically describe sophisticated optimizations of their algorithms in [81] for the use in large-scale scenarios. In [39], Friedrich et al. present an algorithm for timetable-based route search that is currently implemented in VISUM for its transit assignment. In a follow-up, Friedrich [38] shows improvements to the original algorithm to also handle multi-day trips.

At least two experimental approaches to the transit route choice problem were undertaken within the MATSim framework. Maier [65] implemented a transit router in C++ working with data collected for the region of Zurich, whereas Titze [111] used data from Berlin to implement a transit router in Java. Both works showed advantages and disadvantages of their respective approaches, each serving as helpful inspiration for the design and implementation of the implemented transit router (see Sec. 7.4).

3.2.3 Operational Simulation

Real simulations of transit are often used in operations research studies to research operational and performance aspects of transit systems. Often, such simulations are limited to specific modes (e.g. rail) and do not take possible interactions with other modes into account. OpenTrack [75] limits itself to the simulation of operational aspects of railroads, but does so in a very detailed way. For example, the acceleration of trains depends on the traction of the engine and the type and number of attached coaches. Nash et al. [76] show an application of OpenTrack

for the S-Bahn in Zurich, Switzerland. In [61], Koutsopoulos and Wang propose a new rail simulation model, SimMetro, specifically designed for service performance analysis. The presented case study is limited to a single subway line in Boston, though.

But not only passenger flows are modeled in operation research, also the fleets and drivers must be managed. Daduna and Paixão [29] use transit networks to model vehicle scheduling problems. Balbo and Pinson [8] developed a custom domain-specific language to model the communication between buses and a control center. The automated communication provides input into a decision support system, used by the operators at the control center. The study shows a simple simulation applied to data from the Brussels bus transportation network to verify the functioning of the model.

Brill and Withney present a very detailed simulation of the Frankford Transportation Center in eastern Philadelphia in [14]. Although it is rather small scale, it includes multiple bus lines, trolleys, trains and also models private cars and pedestrian movements. The model was implemented in SLX (Extensible Simulation Language [51]) and includes detailed intersection logic as well as driver behavior. The main outcome of the study relates to the operation of the lines as well as the usage of the available infrastructure.

Other topics like the multiple-depot or multiple-vehicle-type scheduling problem are also handled in the operation research literature. But none of the project reports found take concurrent road traffic into account in the simulations. If at all, influences by other road users in reality are added as external effects to the simulations.

3.2.4 Multi-Modal Simulation

Although a lot of different assignment and simulation models exist, none of the aforementioned ones handles private car traffic and transit traffic in an integrated way. Either the private car traffic is simulated, or the transit traffic. In reality, especially in urban regions, buses and streetcars often have to share lanes with private car traffic, resulting in dependencies between the different modes.

TRANSIMS [107] was one of the first, large-scale microsimulators. It also includes the ability to simulate transit traffic, as outlined in [73]. Sadly, no newer research projects could be found that use TRANSIMS, despite the fact that the software is meanwhile available as open-source project [112].

In some cases, transit simulations were done with only regular traffic simulations available. In these cases, often special traffic lights were added to the models

that only influence transit vehicles in order to simulate transit stops [72, 82, 105]. Such behavior usually implies that transit lines are on special lanes, defeating the goal of an integrated simulation where the different modes influence each other.

4 Agent-Based Transportation Simulation

Agent-based transportation simulations consist of several functional entities that only in their entirety have the desired effect of (hopefully) realistically simulate and predict traffic. This chapter first takes a look at what those building blocks are and how they relate to each other, before looking at the individual blocks in more detail.

The chapter tries to remain as general as possible in the beginning. But the more detailed the discussed topics are, the stronger is the influence from the experiences of developing the Multi-Agent Transportation Simulation (MATSim, [66]). Especially in later sections, concrete examples from MATSim will be presented. Note that the state presented in this chapter does not yet include any special references to the simulation of transit. Changes and extensions required to successfully simulate transit traffic are discussed in chapter 7. Some of the concepts presented in this chapter are based on concepts used in the older MATSim implementation written in C/C++ (see, e.g. [16, 95]).

4.1 Overview

When talking about transport simulations, at first many people only think about the simulation of the physical aspects, i.e. the movement of vehicles in space. Often, people start to philosophize about how to handle in detail overtaking maneuvers, turning moves at crossings, how to include delays at crosswalks. Only on second thought, often triggered by corresponding inquiries, people realize that there are also many behavioral aspects that need to be considered: which route should the agents drive along, at what time do they leave home, where do they drive to work and where do the agents go shopping? This list of questions could easily be extended. Fig. 4.1 depicts the two layers, the physical world and the

mental world, and their relation to each other.

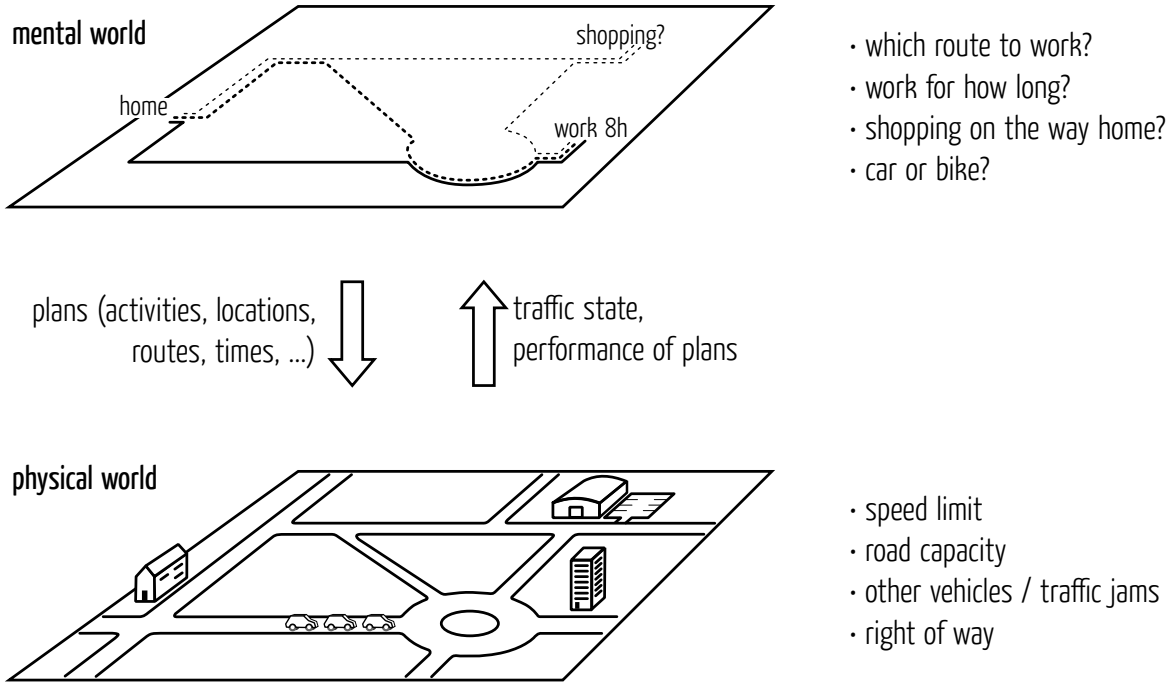


Figure 4.1: Mental and physical world must be considered in a transportation simulation

In the mental world, agents plan their day. They have to make decisions like when to leave home and go to work, if they go with their car or a bike, what route they want to drive, where they want to go for lunch, if they should do some shopping on the way home, etc. Balmer [9] differentiates between the following decisions agents have to make:

- **Mode choice:** Should I walk or take the bus?
- **Route choice:** Which route should I take to get to work?
- **Location choice:** Should I go shopping near my home or at the mall?
- **Activity type choice:** Should I go to visit my friend?
- **Activity chain choice:** Should I go swimming before or after work?
- **Activity starting time choice:** When should I do sports today?
- **Activity duration choice:** Should I drink another beer before going home?
- **Group composition choice:** Who should I take along?

This list of choice decisions is often also called the list of *choice dimensions* (e.g. [11, 13, 5]). Each plan of an agent describes exactly one point in a multi-dimensional

```
<person id="123" age="42">
  <plan>
    <act type="home" link="110" facility="82" x="60.0" y="110.0"
                                                        end_time="07:00:49" />
    <leg mode="car" trav_time="00:05:15">
      <route>23 86 14 57</route>
    </leg>
    <act type="work" link="498" facility="29" x="120.0" y="37.0" dur="08:00:00" />
  </plan>
</person>
```

Figure 4.2: An example of a person and its day plan, encoded in a MATSim-specific XML format

space defined by each choice dimension, where the values or parameters of the point are based on the actual decision.

The result of all these decisions of one agent is a detailed day's schedule (called "day plan" or just "plan" in the following). Fig. 4.2 shows an example of such a day plan, encoded in a custom XML-based file format. The day plans of all agents are then simultaneously executed by a traffic flow simulation, the so-called physical world. The traffic flow simulation tries to move the agents according to their plans, but respecting constraints and limits set by the physics of reality. Such limits can be, among others:

- **Speed limits:** An agent cannot drive faster than the road limit or the speed limit of its vehicle.
- **Flow capacity of road:** A vehicle cannot leave a road and enter a crossing together with 20 other vehicles at the same time (A well-known rule of thumb assumes that about 2000 vehicles can drive through a street per hour and per lane).
- **Storage capacity of road:** A vehicle cannot enter a road if the road is already jammed with other vehicles.
- **Position of other agents:** In most cases it is not desired to simulate traffic accidents, so vehicles should not crash into other vehicles.
- **Right of way:** No matter if a crossing has traffic lights or not, an agent has not always the right to just drive on.
- **Vehicle capacity:** If the subway or bus is already full, the agent has to wait for the next transit vehicle on the line to arrive before it can board.
- **Opening times:** An agent cannot do its shopping if the shop is closed.

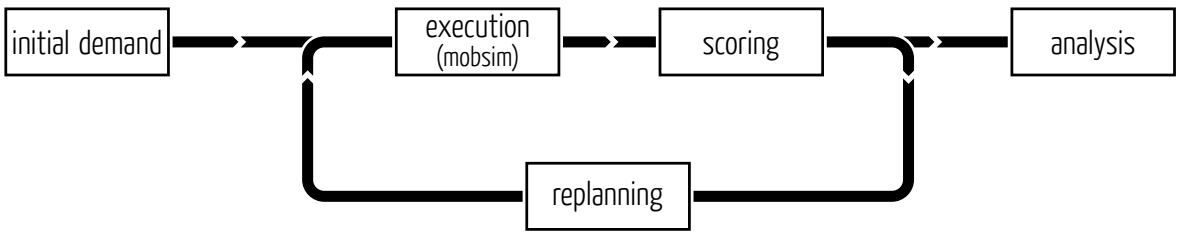


Figure 4.3: The building blocks of the Multi-Agent Transportation Simulation (MATSim)

The agents try to learn from their experience and to optimize their day plans. For this to happen, some feedback must be provided from the physical world to the mental world (see Fig. 4.1). This usually includes information about the state of traffic at certain times of day at certain locations, or some data about how much the actual day deviated from the day plan. With this information, agents can try to improve their day plan, maybe switching routes to avoid traffic jams, leaving work earlier so they arrive at the shop while it is still open, and so on. The new day plans will again be executed by the traffic flow simulation, and the agents will learn if their new plans were better or not accounting for the interaction with other agents, and may optimize again. This constitutes a loop between mental and physical world, in which the agents iteratively try to optimize their day plan. One round in the loop is usually called one *iteration* of the simulation.

Fig. 4.3 shows the schematic control flow of the Multi-Agent Transportation Simulation (MATSim, [66]). It starts with the *initial demand* that describes an initial set of day plans for the simulated population. This initial set of day plans is then *executed* by the traffic flow simulation or mobility simulation (mobsim). The performance of the day plans is evaluated during *scoring*, such that agents can try to optimize their day plans during *replanning*. The plans are then executed again. This iterative loop continues until some criterion is met. In the end, it is possible to *analyze* the data in regards to traffic state or behavior of the agents. The following sections will take an in-depth look at the mentioned building blocks, starting with the *Controler* which is responsible for the data flow and interaction between the different parts.

4.2 Controler

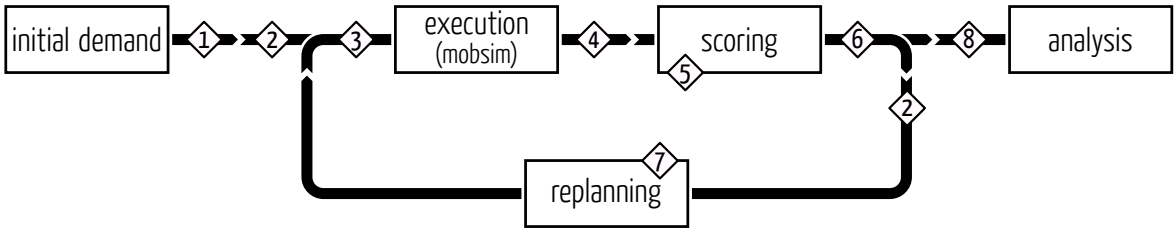
An agent-based transportation simulations consists of many different building blocks, as the previous section showed, especially Fig. 4.3. In MATSim, the job to

string all the blocks (in the following also called *modules*) together and ensure that the correct data is handled at the right time by the right module is done by the so-called "Controler"[1]. It builds the center piece of MATSim that holds all the single parts together and ensures that complete simulations with multiple iterations can be run. In earlier implementations of MATSim, Perl scripts or stand-alone tools were used to connect the various, at that time also stand-alone, tools together [95].

But to reduce the Controler to a simple "iteration manager" would be an understatement. The requirements for the Controler are very different and change from one simulated scenario to the next one. Especially as the power of agent-based simulation is researched within MATSim, the Controler needs to adapt to different needs without interfering with the work of other people that are—for example—doing production runs. Examples of applications and their influence on the Controler are:

- **Road-Pricing scenarios:** Compared to a "regular" simulation, the simulation of road-pricing scenarios requires additional input (the definition of the toll schemes) and produces additional output (e.g. toll-related analyses). But it also influences the scoring part (agents may have to pay money for a toll) and the replanning part (agents may prefer non-tolled routes).
- **Simulating retailer behavior:** Why should only "normal" agents try to optimize their plans, maybe by changing shopping locations? Why not let the shop owners themselves change places, looking for higher frequented locations such as to optimize sales [26]? So not only normal agents need to do scoring and replanning, but also other kinds of agents which may not even be known to the traffic flow simulation.
- **Environmental research:** This topic ranges from the simple calculation of emissions [44, 106] up to specialized replanning and analysis of effectiveness of electric cars [41]. The Controler must be able to deal with additional analyses, that may also influence the replanning parts.
- **Evacuation simulation:** Evacuations may have different movement characteristics, and may make use of additional input data (when and where is something bad going to happen? are there safe areas/shelters?) [62]. Not all of this information can be coded into the initial demand, so the Controler must be able to deal with additional input data, auxiliary analyses, modified replanning, and other extensions.

[1]Likely based on some misunderstanding, MATSim uses the wrong spelling "Controler" instead of the correct "Controller" since its beginning. In order to be consistent with the software, I use the wrong spelling knowingly in this thesis.



1 Simulation Starts ("Startup")
2 Iteration Starts
3 Before Plans Execution
4 After Plans Execution
5 Scoring
6 Iteration Ends
7 Replanning
8 Simulation Ends ("Shutdown")

Figure 4.4: Extension points offered by the MATSim-Controler

Just from the few examples above, plenty of non-standard requirements can be detected that a Controler should be able to fulfill.

First experiments with an object oriented software design allowed many methods to be overwritten—and the original functionality thus being replaced or extended—, but did not bring a lot of success as the inheritance constraints in Java did not allow to easily combine two extensions together at a later stage. In addition, it was rather error-prone by the fact that it was never clear if an overwriting method should call the original super-method, effectively extending a certain functionality, or not call the original super-method, replacing the functionality. This led to problems where critical functionality was no longer available because the essential method was overwritten but the original method no longer called, or that certain aspects were executed twice if the overwriting method did very similar things than the original one that was still called.

An analysis of the structure and data flow within the Controler showed eight different places where it made sense to offer extension points such that additional functionality could be added in a modular way. Fig. 4.4 shows these identified extension points in the MATSim-Controler. The extension points were implemented using the Observer pattern [42], sometimes also called Event listener or Publish-Subscribe pattern. For each of the extension points, the Controler offers events to which interested extensions can subscribe to be informed when that specific state in the Controler flow has been reached. Extensions can subscribe to one or more Controler events.

The observable events are:

- **Startup:** Describes that the simulation starts up and that extensions should load any additional data they may require to function properly.

- **Iteration Starts:** Informs extensions that a new iteration starts. This may be used to reset internal data structures.
- **Before Plans Execution:** There are some modules that analyze the exact plans that are fed into the traffic flow simulation.
- **After Plans Execution:** Some modules may pre-process the outcome of the plans execution, resulting in additional outcome that is relevant for the scoring.
- **Scoring:** Tells modules that the plans execution is over and no more events will be issued, so that the scoring can take place.
- **Iteration Ends:** Makes it possible for modules to write out analysis data that was collected during the iteration.
- **Replanning:** Informs the modules that now is the time to do replanning.
- **Shutdown:** Tells event listeners that the simulation is about to end, enabling extensions to write out final data or analysis results.

This event structure has proved to be so powerful that meanwhile even some of the core functionality is implemented that way: Plans Scoring and Plans Replanning are both realized as Controler events listeners, as is the writing out of the complete population with all plans to a file before the plans execution starts (this serves mostly practical reasons to be able to resume a simulation at a specific iteration).

The traffic flow simulation (plans execution) itself was specifically kept out of Controler events. As there should be anyway only exactly one "reality" in the simulation, this limitation never was an issue.

4.3 Initial Demand

Each simulation needs some initial data. In the case of an agent-based transportation simulation, this usually means a transportation network and the so-called "initial demand". The initial demand describes the initial day plans of all simulated agents.

As not necessarily all choice dimensions are optimized during the simulation, the initial demand should also provide useful values for the choice dimensions not-to-be optimized. For example, if the primary location choice is not optimized by the simulation, the initial demand should, for each agent, provide meaningful locations for work or school, respectively.

For choice dimensions that are optimized by the simulation (typically at least route and time choice) the corresponding data can be based on raw estimates or

even be missing in the initial demand. It is not wrong to assume a default duration for a work-activity of eight hours. If it is too long or too short, the optimization process will hopefully realize that and adapt its duration. If route data is completely missing, the simulation may just assign the fastest path through the network to the agent; it would do that anyway sooner or later in the optimization process.

As already mentioned, the initial demand should provide useful values for choice dimensions that are not optimized during the simulation. As currently the list of optimized dimensions is much smaller than the complete list of choice dimensions (see Sec. 4.1), the initial demand must be carefully prepared. This is usually done within special activity-based demand generation (ABDG) models. Balmer [9] describes the process and a framework used to generate an initial demand based on census data and other data sources. Other similar projects exist, e.g. the "Travel/Activity Scheduler for Household Agents – TASHA" [69] or "Prism Constrained Activity-Travel Simulator – PCATS" [60]. One has to note that most ABDG models have their own quirks and that taking data from an existing ABDG model is not always the best solution [100].

As can be seen, the generation of an initial demand is in itself a larger topic and must be done with great care. It will not be looked at in more detail here, as the exact creation of an initial demand is not of much relevance for the work presented.

4.4 Traffic Flow Simulation

4.4.1 Overview

The traffic flow simulation (sometimes also called *mobility simulation* or short "mobsim") is responsible for executing the day plans in a physical environment. It takes the agents' plans and executes the plans respecting physical constraints and limits. But still, it only operates on a model of the reality. This model can be more detailed or more generalized. As an example, traffic flow simulations could simulate in detail overtaking maneuvers, crossing oncoming traffic on intersections or the state of traffic lights. Depending on the resolution of the model, the diversity of the results will likely vary as well, as not each analysis is possible with every kind of simulation.

Usually, the more detailed a simulated model is, the more different kind of results can be produced. For example, the influence of intersection layouts or traffic lights coordination can only be researched if the underlying mobility simulation supports the detailed simulation of intersections. So, from the viewpoint of pos-

sible results, a simulation model with a high resolution and as many details as possible is clearly desirable.

The problem is that usually the execution time for the mobility simulation is highly depending on the level of detail simulated. The more features a traffic flow simulation has, the more processing power is required. But it may not always be required to simulate every possible detail, giving way for performance optimizations.

Thus, it is essential to decide from the beginning for what purpose a simulation will be used, and for what questions the simulation should provide results for. This allows the developer to carefully select those features for the simulation required for doing the proper analyses, but leaving unnecessary things to the side.

One of the most essential things for a mobility simulation is—besides the plans it has to execute—the network on which the traffic takes place. The network, basically an attributed graph, consists of nodes and links. In a road network, nodes often correspond to intersections, whereas links correspond to road segments. A rail network could be treated similar to a road network, with links representing the rails and nodes representing switches. Alternatively, nodes could also be used to model stops, and links would therefore stand for generalized connections between stops. In the first case, the network represents the real infrastructure. In the second case, the network behaves as a logical representation of possible connections between stops.

4.4.2 Flow Dynamics

In principle, arbitrary models could be used, e.g. the model by Wiedemann [123] or a cellular automata model (e.g. [74]). Both models offer a great level of detail, but also require large amounts of computing power. Transportation planning is often 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 [43] fulfills all these requirements while still being comparatively fast to compute.

The actual traffic flow simulation currently used in MATSim is for the bigger part a re-implementation of the algorithms described in [16] in Java. It is usually called *QueueSimulation*, reflecting the concept it is based on. The following paragraphs serve as a short summary for better understanding of later topics and to highlight differences to the original implementation. A further developed version of the QueueSimulation, the "Deterministic Event-driven Queue-based Simulation" (DEQSim, [18]), promising a faster traffic flow simulation without loss of

details, is currently evaluated for the use in MATSim.

Links (road segments) are modeled as first-in first-out (FIFO) queues. Vehicles in the queue cannot leave the queue before a certain amount of time has passed, the time corresponding to the free flow travel time of the link. In addition, only a limited number of vehicles can leave a link (or the link's queue, respectively) per time step, corresponding to the flow capacity of the link. Moreover, such a queue can also be full, effectively introducing a storage capacity of a link. This storage capacity corresponds to the limited amount of physical space on a road segment that limits the number of vehicles that can be located on that section. Besides the queue, a special buffer is located at the end of each link. That buffer stores those vehicles that are allowed to leave the link in the next time step, respecting the free flow travel time and the link's flow capacity. In addition, the splitting of the link into a queue and a buffer helps to implement a parallel update of the data structures, making the outcome independent of the order the links and nodes are processed [17]. This approach is borrowed from lattice gas automata, where particle movements are also separated into a "propagate" and a "scatter" step [40].

Contrary to the original implementation in C/C++ [16], the queue's size (and the link's storage capacity) is not measured as number of vehicles, but in vehicle-equivalents (VE). One VE is defined as the size of one average private car. Trucks, buses count for more than one VE, whereas bikes may count for less.

Nodes do not have a lot of internal logic. In each time step, the foremost cars of each in-coming link are moved over the node to the next link on their route, given that there is still (storage) space available on the next link and that the car can leave its current link (which may not be the case due to the aforementioned queue's constraints). The first car that cannot be moved from an in-coming link ends the handling of that link. The order in which the in-coming links are handled is based on Metropolis sampling [68] on the links' flow capacity. This means that links with a high flow capacity are more likely to be handled first, eventually taking up all the available free space on out-going links before lower-capacity links are handled.

The use of FIFO queues for modeling links has some important consequences regarding the details that can be modeled. Most obviously, effects like the overtaking of vehicles with different speeds or lane-changing behavior cannot be reproduced. As cars can only be removed from or inserted into the queue at its start or end, the beginning or the end of the link are the only place where vehicles can be removed from or inserted into the vehicle flow. In the current implementation, cars ending their trip (probably parking along the road or in a driveway) leave the traffic flow at the end of the link, just before being moved from the link's queue to its buffer. Vehicles departing will enter the traffic flow at the same place, first be-

ing added to the buffer where they have to wait until they can cross the upcoming intersection.

4.4.3 Interaction of Vehicles, Persons and Network

As others have noted before (e.g. [57]), many micro-simulations only look at private car traffic. In many cases, vehicles and drivers were combined into one agent in the simulation, often called *vehicle-person units* or *vehicle-driver units*. This was not different in the original C/C++ implementation of MATSim [16], and was even that way in the first version of the Java re-implementation.

In MATSim, the QueueSimulation moved such vehicle-driver units through the network. The combination of vehicles and drivers into one object made the implementation rather straight-forward: Although the simulation moved the vehicle forward and the driver needed to decide which link to take at an intersection, no additional interaction or communication between vehicle and driver was needed because vehicle and driver where essentially one object.

To successfully simulate transit, it is necessary to differentiate between vehicles and persons, as not every person is a driver itself. This leads to a higher complexity immediately, as drivers must now be somehow informed when their vehicles are moved by the QueueSimulation, so they can return the correct answers when asked for the next link. But not every person in a simulation supporting transit is a driver anymore. This means that the initialization of agents changes as well. It is no longer possible to just put every person together with a car on the links and wait until they depart. Instead, some agents may walk to a bus stop, wait for a transit vehicle and board it when it arrives, leaving it somewhere else. A clear distinction between trips where agents drive their own car and trips where agents are transported by other vehicles must be made.

Conceptually, the agent should be in control while performing an activity, while the simulation is in control while the agent is being moved through the network (as this is the real job of the traffic flow simulation). However, it is not very high-performance to give in every time step every agent performing an activity the possibility to decide about the ending of its activity (especially as the information is based on plans and thus predetermined). Thus, agents are required to specify when starting an activity at what time the activity is going to end. This information is managed by the simulation again, and the agents are informed just when their activity ends to start the next leg.

Fig. 4.5 gives a schematic overview in UML [93] over the messages passed between the QueueSimulation and an Agent to control when the agent is executing

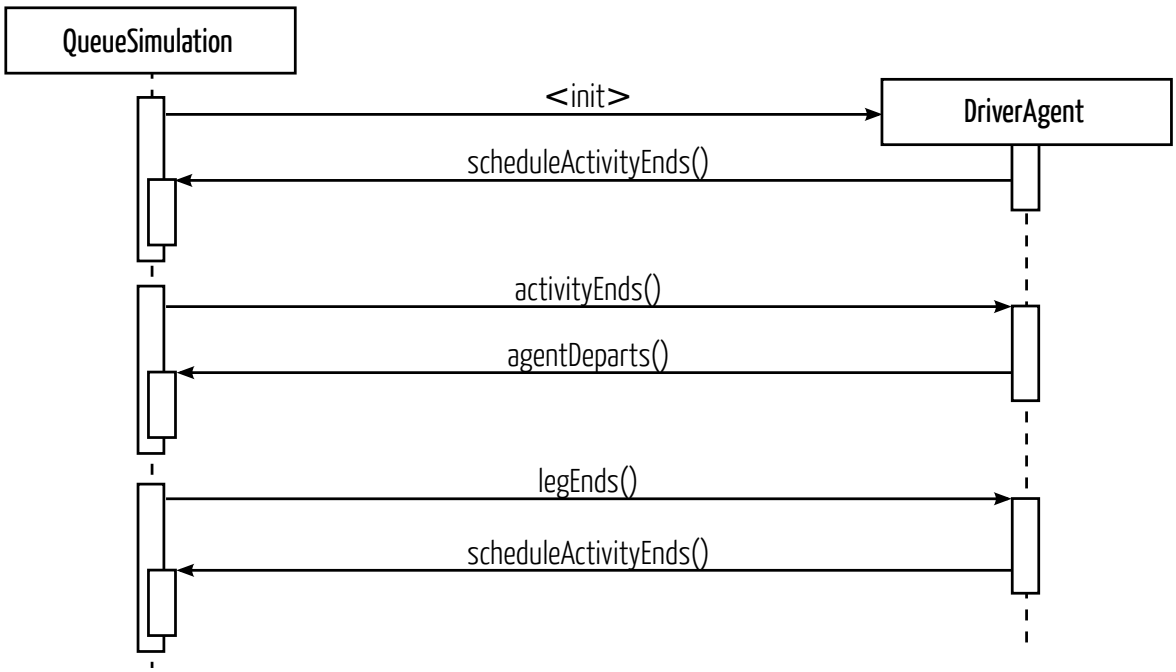


Figure 4.5: UML sequence diagram of the messages passed between QueueSimulation and DriverAgent

an activity and when it is starting and ending a trip. After being initialized, the agent registers its first activity's end with the simulation (`scheduleActivityEnds()`). Once that point in time is reached, the simulation "wakes" the agent, informing it about the time (`activityEnds()`). The agent can now look up in its plan what is next, and register itself with the simulation as departing (`agentDeparts()`), so it will be moved by the traffic flow simulation through the network. Once the agent has arrived the target of its trip, it is informed about that event (`legEnds()`), giving the agent the possibility to start a new activity and schedule its end (`scheduleActivityEnds()`) with the simulation. Should an agent arrive at the last activity of its plan, it will not schedule any activity end, essentially vanishing from the perspective of the simulation.

Even though the simulation is said to have the control over an agent while it is being moved through the network, there is still some communication going on between the simulation and the agent. Fig. 4.6 shows the messages passed between the involved entities for effectively simulating the traffic flow.

Whenever a driving agent departs (`agentDeparts()`), it is added to a special departure queue within a QueueLink. Then, in each time step the method `moveLink()`

is called for each QueueLink. The QueueLink first determines which cars driving on the link are able to leave the link in the current time step, moving those cars to the link's buffer. It does this by comparing each vehicles *earliest link exit time* with the current time, starting with the front most vehicle in the FIFO queue. The earliest link exit time is calculated by the time the vehicle was added to the link plus the free speed travel time on that link. If a vehicle is found whose earliest link exit time is not yet reached, no more vehicles are checked in the queue, as otherwise the first-in, first-out principle would be broken. Next, assuming that there were less cars to leave the link than the flow capacity allows, the queue of the departing cars is checked as well, moving cars from the departure queue into the buffer.

Also in every time step, every node is given the possibility to move cars from the in-coming links to the out-going links (`moveNode()`). The node only looks at vehicles in the aforementioned buffer of the link. This simplifies the logic as the node itself does not need to check whether an agent may leave the link. Instead, it can assume that every vehicle in the buffer is allowed to be moved away, as only such vehicles where placed in the buffer by `moveLink()`. For every in-coming link, the private method `moveVehicleOverNode()` is called with each vehicle from the buffer. This method first asks the vehicle's driver which way s/he wants to go (`chooseNextLink()`), then checks the corresponding link if there is still space available for that vehicle (`hasSpace()`). If this check returns positive, the driver is informed that it is being moved over the node (`moveOverNode()`), allowing it to update internal information (such as is required to return useful results to `chooseNextLink()`). Also, the vehicle is removed from the buffer in the previous link and added to the desired link.

It must be noted that the actual implementation differs slightly from the concept shown. The changes are only relevant for further performance optimization of the code (e.g. see Sec. 4.9.2) and to ensure that the actual order in which links and nodes are handled in one time step does not have an influence on the outcome (enforcing a so-called *parallel update*; see e.g. [17]).

4.4.4 Gridlock Prevention

As long as there is space available on a link, vehicles can be added to a QueueLink. This means that the in-flow of vehicles into a QueueLinks is unbounded. When more vehicles are added to a link in a time step than the link's (out-)flow capacity supports, traffic jams may start to build. In the simulation, congestion starts at the downstream end of a link when not all of the vehicles can leave the link at the expected time. If even more vehicles arrive, the queue of jammed vehicles grows, until it reaches the upstream end of a link. There, the traffic jam may spill over the

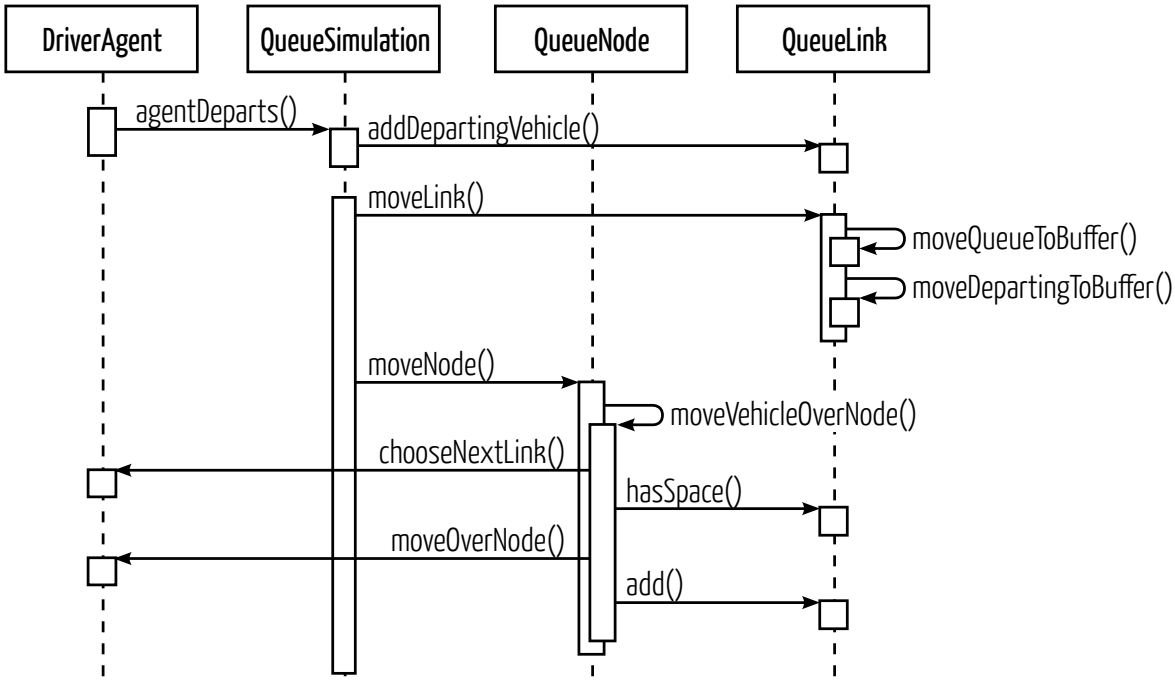


Figure 4.6: Messages passed for a single step of the simulation

intersection into subsequent link, as vehicles cannot be removed from other links and added to the jammed link. Under certain circular conditions, the tail of the traffic jam may hit its head again, effectively creating a gridlock. This can often be observed in traffic circles, but is not limited to such geometries. In reality, drivers may start to draw aside onto the sidewalk, effectively using more space capacity than foreseen to keep traffic moving.

The simulation monitors the front-most vehicle of every link to recognize if a gridlock may have happened. If a vehicle does not cross an intersection for a configurable amount of time, the vehicle is recognized as being stuck, suggesting the presence of a gridlock. The simulation has two possibilities to handle such situations:

- **Removing the stuck vehicle from the simulation.** The stuck vehicle is removed from the simulation, creating space on the link such that other agents can move forward, hopefully resolving the gridlock.
- **Moving the vehicle to the next link, ignoring capacity constraints.** This corresponds to what happens in reality, that additional space is used by the drivers. It opens up space on the link the vehicle was stuck before, allowing other drivers on that link to move again, hopefully resolving the gridlock.

For a long time, the first variant was used in MATSim. While it always solved the gridlocks, the removed agents created difficulties, especially when analyzing the performance of the executed plans (what is the performance of a plan that was aborted?). But also the ever-changing number of agents in each iteration that were able to complete their day-plan was cumbersome to deal with. The second variant, proposed by Charypar [18], solves the gridlocks just as well without imposing the problems when analyzing the simulation outcome. It is thus currently the default way of handling gridlocks in MATSim.

4.5 Events

The traffic flow simulation has to deliver some kind of output, otherwise it would not make much sense to simulate the traffic flow. But "output" is not clearly defined for a traffic flow simulation. Especially, different stakeholders are likely to have different interests regarding the output. A first person may be interested in the average velocities along certain routes, a second person wants to know the traffic volume in a specific region to calculate emissions, while a third person is interested in a visual representation of the traffic for PR reasons (also see Sec. 4.8 for more information about actual simulation output and analyses). In addition, the simulation framework itself also needs some characteristic values, like the actual travel time on links, to be able to calculate the fastest routes through the network. The variety of the requirements makes it clear that the traffic flow simulation must provide output as generically as possible, such that required analyses can be derived from the available data. To reach this goal, the traffic flow simulation generates so-called *events*. The concept of using events for feedback is also described in [95].

Events are small, atomic pieces of information. The traffic flow simulation generates events for every notable change of state within the simulation. Events include information about actions of agents, describing in detail what action happened at which time and—if available—where. Table 4.1 lists different events generated by the QueueSimulation together with their attributes. As the simulation supported driver-vehicle-units only for a very long time, all the events only include an agent-id, but no vehicle-id. The influence of having vehicles and passengers (not only drivers) will be detailed in Sec. 7, where also all the other changes related to the implementation of a transit simulation are discussed.

Fig. 4.7 gives a schematic overview in which situations events are created by the simulation when executing a plan of an agent. Whenever an agent starts or stops performing an activity, the events *ActivityStartEvent* or *ActivityEndEvent* are created. Likewise, when an agent departs or arrives from a leg, the events *Agent-*

Table 4.1: Different types of events generated by the traffic flow simulation

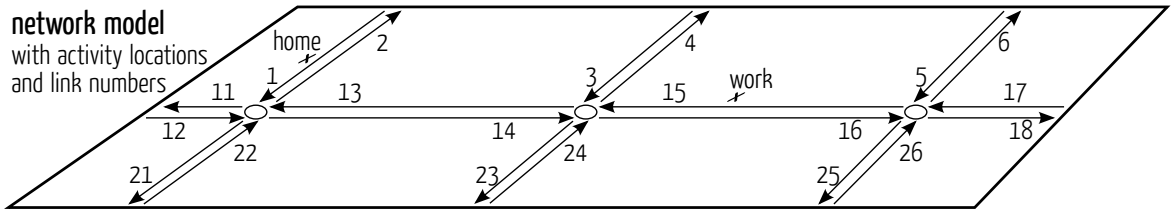
Type of Event	Attributes
ActivityStartEvent	time, agent-id, link-id, activity-type
ActivityEndEvent	time, agent-id, link-id, activity-type
AgentDepartureEvent	time, agent-id, link-id, leg
AgentArrivalEvent	time, agent-id, link-id, leg
AgentWait2LinkEvent	time, agent-id, link-id
LinkEnterEvent	time, agent-id, link-id
LinkLeaveEvent	time, agent-id, link-id
AgentStuckEvent	time, agent-id, link-id

DepartureEvent and *AgentArrivalEvent* are generated. As MATSim does not (yet) explicitly model the way from an activity location (like a flat or office) to a vehicle (or vice versa), the AgentDepartureEvent following an ActivityEndEvent has usually the same timestamp. The same holds true for the ActivityStartEvent after an AgentArrivalEvent. After an agent departs (i.e. starting the vehicles engine, maybe leaving a parking lot), it has to effectively enter the road. This is signaled by *AgentWait2LinkEvent* and may happen almost immediately in case there is no traffic jam or other vehicle blocking the road. After that, every time a vehicle crosses an intersection, two events are generated informing that the agent has left one road (*LinkLeaveEvent*) and entered another road (*LinkEnterEvent*).

Based on these types of events, a multitude of derived output can be generated. As an example, based on the number of LinkEnterEvents per hour, the hourly traffic volume can be calculated. By taking the difference in time between the LinkEnterEvent and LinkLeaveEvent of one agent on a link, the travel time for that agent on that link can be calculated, or the average speed on the link derived given the link's length. Sec. 4.8 gives more details and examples of possible analyses based on events.

The events are written to file for later post-processing. A simple, tabular text-based file format is available as well as a file format based on XML [121]. The tabular text-based file format was in use for several years in MATSim but has proven to be not flexible enough for additional types of events used in experimental studies. Thus, the XML-based format will be used for further studies.

To process events, code infrastructure was built according to the Observer pattern [42], which is tellingly also called *Event listener* pattern. For each type of



plan of agent 2:
- stay at home, located on link 1, until 07:05:00
- drive along links 14, 16, and 15
- start work, located on link 15

event type	event time	agent id	link id	additional data
ActivityEndEvent	07:05:00	2	1	home
AgentDepartureEvent	07:05:00	2	1	
AgentWait2LinkEvent	07:05:01	2	1	
LinkLeaveEvent	07:05:02	2	1	
LinkEnterEvent	07:05:02	2	14	
LinkLeaveEvent	07:06:24	2	14	
LinkEnterEvent	07:06:24	2	16	
LinkLeaveEvent	07:08:51	2	16	
LinkEnterEvent	07:08:51	2	15	
AgentArrivalEvent	07:10:43	2	15	
ActivityStartEvent	07:10:43	2	15	work

Figure 4.7: Events created by the traffic flow simulation when executing a single agent's day plan

event, an *EventHandler* interface is provided. By implementing the corresponding interfaces and registering itself at a central event handler manager provided by the Controler (see Sec. 4.2), objects are enabled to process exactly those type of events they need to generate the output the developer had in mind.

As outlined in the first paragraph of this section, the output of the traffic flow simulation is not only interesting at the end for analyses, but already while running multiple iterations for the optimization of plans (see Sec. 4.1 and 4.7). For this reason, the aforementioned event processing infrastructure is not only available for post-processing of events, but already at runtime during the simulation. This enables to feed back information efficiently from the physical world to the mental world of the simulation (see Fig. 4.1).

4.6 Scoring

This section describes the default scoring function used in MATSim. The description is heavily based on existing publications (e.g. [99]) and is reprinted here for completeness and for better understanding of the complete simulation setup.

In order to compare plans, it is necessary to assign a quantitative score to the performance of each plan. In principle, arbitrary scoring schemes can be used (e.g. prospect theory [6]). In this work, in order to be consistent with economic appraisal, a simple utility-based approach is used. The approach is related to the Vickrey bottleneck model [117, 4], but is modified in order to be consistent with our approach based on complete day plans [20, 94]. The elements of the approach are as follows:

- The total score of a plan is computed as the sum of individual contributions:

$$U_{total} = \sum_{i=1}^{n} U_{perf,i} + \sum_{i=1}^{n} U_{late,i} + \sum_{i=1}^{n} U_{tr,i} \ , \tag{4.1}$$

  where U_{total} is the total utility for a given plan; n is the number of activities, which equals the number of trips (assuming the first and the last activity—usually both "home"—are counted as one); $U_{perf,i}$ is the (positive) utility earned for performing activity i; $U_{late,i}$ is the (negative) utility earned for arriving late to activity i; and $U_{tr,i}$ is the (negative) utility earned for traveling during trip i. In order to work in plausible real-world units, utilities are often measured in Euro.
- A logarithmic form is used for the positive utility earned by performing an activity:

$$U_{perf,i}(t_{perf,i}) = \beta_{perf} \cdot t_{*,i} \cdot \ln\left(\frac{t_{perf,i}}{t_{0,i}}\right) \tag{4.2}$$

  where t_{perf} is the actual performed duration of the activity, t_* is the "typical" duration of an activity, and β_{perf} is the marginal utility of an activity at its typical duration. β_{perf} is the same for all activities, since in equilibrium all activities at their typical duration need to have the same marginal utility.

  $t_{0,i}$ is a scaling parameter that is related both to the minimum duration and to the importance of an activity. If the actual duration falls below $t_{0,i}$, then the utility contribution of the activity becomes negative, implying that the agent should rather completely drop that activity. As long as dropping activities from the plan is not allowed, $t_{0,i}$ has essentially no effect.
- The (dis)utility of being late is uniformly assumed as:

$$U_{late,i} = \beta_{late} \cdot t_{late,i} \ , \tag{4.3}$$

where β_{late} is the marginal utility (in Euro/h) for being late, and $t_{late,i}$ is the number of hours late to activity i. β_{late} is usually negative.

- The (dis)utility of traveling is uniformly assumed as:

$$U_{tr,i} = \beta_{tr} \cdot t_{tr,i} \ , \tag{4.4}$$

where β_{tr} is the marginal utility (in Euro/h) for travel, and $t_{tr,i}$ is the number of hours spent traveling during trip i. β_{tr} is usually negative. As the simulation only supported private car traffic for a long time, the disutility of traveling does not yet contain any transit specific terms (like number of line changes, waiting time, ...).

In principle, arriving early or leaving early could also be punished. There is, however, no immediate need to punish early arrival, since waiting times are already indirectly punished by foregoing the reward that could be accumulated by doing an activity instead (opportunity cost). In consequence, the effective (dis)utility of waiting is already $-\beta_{perf} t_{*,i}/t_{perf,i} \approx -\beta_{perf}$. Similarly, that opportunity cost has to be added to the time spent traveling, arriving at an effective (dis)utility of traveling of $-|\beta_{tr}| - \beta_{perf} t_{*,i}/t_{perf,i} \approx -|\beta_{tr}| - \beta_{perf}$.

No opportunity cost needs to be added to late arrivals, because the late arrival time is spent somewhere else. In consequence, the effective (dis)utility of arriving late remains at β_{late}. – These approximate values (β_{perf}, $\beta_{perf} + |\beta_{tr}|$, and $|\beta_{late}|$) are the values that would correspond to the consensus values of the parameters of the Vickrey model [4].

The utility function could be extended with additional terms. For example, one could think of tolls or parking costs that could be included into the score.

4.7 Replanning

4.7.1 Overview

As the initial day plans usually were generated independently—not knowing the plans of other agents—those plans usually do not reflect the best choices agents could do once they learn about the behavior of other agents. With the simultaneous plans execution by the traffic flow simulation and the scoring of the executed plans, the agents get a mean to compare different plans. This opens the possibility to try out different variants of a plan, keeping well performing plans and eventually forgetting about bad plans. Integrated into the iterative process steered by the Controler, this plans modification and optimization process builds the *replanning*

part of MATSim. It follows the concepts of co-evolutionary algorithms for optimizing the agents' day plans and thus, the travel demand. The concepts described in this section have been used in MATSim already for a long time (e.g. see [95]).

During the replanning, agents can explore the parameter space along the choice dimensions as outlined in Sec. 4.1. This is done by selecting one of the agent's existing plans, and optionally making a copy of it and modifying the copy. As agents can now have multiple plans, exactly one of them is always marked as being the "selected" one, which will be executed by the traffic flow simulation. All the plans of an agent build its memory. If a copy was made and modified in the replanning, the copy will be selected for execution by the traffic flow simulation. If no modifications were made to a plan (and thus no copy was made), one of the agent's existing plans will be selected for the next traffic flow simulation.

A number of replanning *modules* are available that each explores one or more dimension. The combination of zero, one or more of such modules builds a replanning *strategy*. In addition, each strategy has a plan *selector*, responsible for selecting one of the existing plans of an agent. Every agent randomly chooses one strategy in each iteration for the replanning. Each strategy has a weight, influencing the probability of being chosen, resulting in a Metropolis sampling [68] of the available strategies for each agent.

Each plan uses some amount of available memory in the computer. As the available memory is limited, the number of plan copies agents can have must be limited as well. The effective number of plans an agent can remember is configurable. Typical values are around 4 or 5 plans per agent. In every iteration, before the replanning takes place, supernumerary plans are removed from the agents. This way, agents can temporarily have one plan more than allowed in case the following replanning creates a new copy of a plan. This is consistent with the idea that agents should be able to try out new plans (or variations of existing plans) without consequences. If the new plan is bad, it may be removed before the next replanning, leaving the agent with the configured number of plans. If the new plan is good, another plan will be removed. Which of the agent's plan is removed is also decided by a plan selector, so not necessarily the worst plan must be removed.

4.7.2 Plan Selectors

Plan selectors are responsible for selecting one of the existing plans of an agent. This happens when an agent has too many plans—so the selected plan will be removed—or when an agent needs a plan for modification during replanning. Different policies can be thought of to choose a plan. Note that if a plan has not yet any

score assigned, it is always chosen before any other plan for replanning. This leads to an *optimistic behavior* (see e.g. [102]) in the replanning, where unknown plans (plans without score) are preferred and evaluated first. The following paragraphs describe the currently available and regularly used plan selectors in MATSim.

BestPlanSelector selects the plan with the highest score, useful to aggressively optimize already good plans. Note however that the chance of getting stuck in a local optimum rises when always the best plan is selected for further optimization.

WorstPlanSelector selects the plan with the lowest score. Mostly used to select plans for removal, if an agent has too many plans. Removing the plan with the lowest score corresponds to the evolutionary aspect of the replanning where the weakest elements are eliminated first.

KeepSelected just returns the plan that is already marked as selected for the person.

RandomPlanSelector just selects a random plan based on a uniform distribution over all plans of an agent. It is often used in conjunction with replanning modules for exploring the choice dimensions. To reduce the risk of ending up in a local optimum, bad and good plans should be equally chosen for further optimization.

ExpBetaPlanSelector selects a random plan according to a logit model: [11]

$$p_j = \frac{e^{\beta \cdot s_j}}{\sum_i e^{\beta \cdot s_i}} \tag{4.5}$$

where p_j is the probability for plan j to be selected and s_j its current score. β is a sensitivity parameter, set to 2.

ExpBetaPlanChanger is very similar to the aforementioned ExpBetaPlanSelector. It first draws a random plan, and then decides whether it should change to the drawn plan or stay with the currently selected plan. The probability to change to the randomly drawn plan is:

$$p_{change} = min(1, \alpha \cdot e^{\beta \cdot (s_{random} - s_{current})/2}) \tag{4.6}$$

with:

- α: The probability to change if both plans have the same score, set to 1%
- β: A sensitivity parameter, set to 2
- $s_{\{random,current\}}$: The score of the randomly drawn plan and the currently selected plan

In a steady state, the model shown in Eq. 4.6 converges on the logit model shown in Eq. 4.5. The advantage of the model described by Eq. 4.6 is that only small

numbers of agents change from one plan to another, which results in a smoother learning curve during the iteration cycle and lets the system better converge to a steady state.

Plan selectors may also take additional data than just the scores into account. Very recent experiments with integrating Cadyts [15, 36] into MATSim have resulted in a special plan selector that takes traffic counts into consideration when selecting plans, effectively resulting in a calibration of the simulation [37].

4.7.3 Replanning Modules

Replanning modules are responsible for exploring the choice dimensions in order to find a better plan for an agent. They form the most important part within the mental world of the agents to drive the optimization process of the whole simulation. Modules can be characterized in different ways.

First, modules can be distinguished if they modify one or multiple choice dimensions. The limitation to one choice dimension only usually helps to better understand what a module is really doing (smaller complexity, less side effects). In contrast, the combined optimization of multiple choice dimensions in one module allows for finding more elaborate plans than the combination of multiple modules that each only optimize one choice dimension is capable of (e.g. combined location and mode choice).

Second, modules can be distinguished by the kind and amount of data they use to modify a plan. Modules applying *random mutation* are usually very fast and do not require any data, but their modifications may degrade a plan's performance. This usually leads to a high number of iterations needed until (randomly) a good plan is generated and bad plans are eliminated. Modules that try to optimize their modifications, e.g. by using data from the last traffic flow simulation, are called *best response* modules. Often, they experience a longer computational runtime than random mutation modules for a single plan's modification, but usually require fewer iterations in the simulation to find optimal plans.

Other ways to classify modules are certainly possible. The following paragraphs will apply the two characteristics mentioned above when describing the currently available and regularly used replanning modules in MATSim.

TimeAllocationMutator is an example of a very simple random mutation module that only modifies one choice dimension, the time. For each activity in a plan, it adds (or subtracts, respectively) a random amount of time within a configurable bound. By default, the bounds are set to [–30, +30] minutes. The module ensures that no activity receives a negative duration, and that no activity ends later

than midnight (24 hours).

ReRoute only modifies the route dimension for car drivers by calculating the least expensive route from one link to another one in the network. The route cost is mostly influenced by the travel time needed for the route, so the least cost route is usually also the fastest route. The module uses the link travel times of the last traffic flow simulation, and is thus considered a best response module. Finding the least cost path in a network can be generalized to the shortest path problem, a known problem in graph theory. A specialty that must be considered when finding the least cost path are the time-dependent link costs, as they are based on the links' travel times which vary over the time of day. In MATSim, two different implementations for finding the least cost path are available. The first one is a simple implementation of Dijkstra's algorithm for shortest path calculation [32], adapted to work with time-dependent link costs. The second variant uses a version of Dijkstra's algorithm extended with some heuristics to reduce the computational costs, known as A* with landmarks [45]. While the A* implementation is faster than the simple implementation of Dijkstra's algorithm, the heuristics only work with networks in a Euclidean coordinate system and is thus not always applicable. A detailed comparison of the different implementations can be found in [63].

Planomat is an example of a module that modifies more than one choice dimension. The module combines an optimized time choice with route choice, resulting in a typical best response module. The module makes use of genetic algorithms [46] for finding an optimal time allocation of all activities in an agent's plan. Recently, the module also added mode choice to its modification process, being the first replanning module in MATSim modifying three different choice dimensions at the same time. More information about the original version of Planomat (without mode choice) can be found in [67], while [33] contains information about an updated version called PlanomatX that includes mode choice and improved computational time.

SecondaryLocationChoice is a rather new module that tries to optimize the locations of secondary activities like shopping and leisure in MATSim. It is also implemented as a best response module, trying to find the best solution within some constraints for each modified plan. Details of the implementation as well as first applications can be found in [52].

4.7.4 Typical Usage

Having all the plan selectors and replanning modules available, the question arises how to combine them to useful replanning strategies. Additionally, weights must

be defined for each strategy to help agents decide which strategy to apply.

As can be seen from the listing of available replanning modules before, time and route choice have been the dominant choice dimensions for quite some time. But to just define exactly two strategies—one for time choice and one for route choice—is not very useful. Best response modules usually take the results of the last traffic flow simulation into account when modifying plans. For this previous state being useful as a forecast for the next traffic flow simulation (and thus the optimized modification to be of any use), the traffic flow should not be completely changed by the replanning. In other words, it is likely not convenient to modify each and every plan.

A small example may demonstrate this problem: Given a very simple network with two roads leading from one place to another, with only a small difference in travel time and both with limited capacity. When the agents have first to decide for a route, they will chose the road with the smaller travel time. Because of the limited capacity, a traffic jam will occur on that road during the traffic flow simulation, increasing the travel time on that route. In the next iteration, the agents will decide to take the other route to avoid the traffic jam. If every agent changes to the—now faster—alternative route, the traffic jam will just move over to that one in the next traffic flow simulation. In the next iteration, the agents would switch back again to the first route—there it would have been possible to drive with the free flow speed in the last traffic flow simulation, so it seems to be faster. In each iterations, the agents would flip between the two alternatives.

An efficient way to solve such problems is to limit the amount of innovation in each iteration. By only allowing a certain percentage of the population to generate new plans, the probability of fluctuations can be hugely reduced and a more stable, albeit likely slower, optimization process can be reached. MATSim implements this solution by allowing replanning strategies to make no modifications. Such strategies consist of only a plan selector, but no replanning module. Together with accompanying weights for each strategy, the amount of new plans can be influenced. A typical setup in MATSim is the following:

- time choice strategy: uses the RandomPlanSelector and the TimeAllocationMutator module. Typical weight: 10%.
- route choice strategy: uses the RandomPlanSelector and the ReRoute module. Typical weight: 10%.
- select-only strategy: only uses the ExpBetaPlanSelector or ExpBetaPlanChanger, with a typical weight of 80%. This helps to change back from possible bad plans to good plans, and gives some stability if the agent already has a good plan by likely selecting the same plan again.

MATSim makes it possible to change the weights of the strategies during the iterations. This can be used to start with higher weights for time and route choice in the first few iterations to quickly generate some amount of diversity amongst the plans [19]. If one is interested in economic evaluations, it has proved to be helpful to switch off the generation of new plans completely after the optimization process stabilized and let the simulation run with the select-only strategy for some more iterations (e.g. [48, 99]). This avoids having new plans being created that turn out to be bad when evaluated by the traffic flow simulation.

4.8 Analyses

As already mentioned before, the number and diversity of desired analyses is quite high. The simulation framework has thus to provide a way to easily add custom analyses. MATSim makes this possible with a combination of extension points in the Controler (see Sec. 4.2) and event handlers (see Sec. 4.5). By listening to Controler events, a custom analysis can initialize specific event handlers at startup, letting the event handlers do the data collection and possibly aggregation, and outputting the results when an iteration ends.

MATSim includes some default analyses that are automatically executed when the Controler is run. In the following paragraphs, the different analyses will be highlighted, explaining what data they generate, how the data is used and, if of interest, how the data is generated from the simulation output.

The **LinkTravelTimeCalculator** calculates the actual travel times for each link. As the travel times vary over the time of day, depending on the current traffic volume, the travel time calculator aggregates the link travel times into time bins, i.e. the travel times of all agents traveling along a link within a certain time window are averaged. By default, a bin size of 15 minutes is used. The router (see "ReRoute" in the previous section about replanning) queries the LinkTravelTimeCalculator to receive the link travel times required for finding the fastest routes in the network. The LinkTravelTimeCalculator analyzes the time between a pair of LinkEnterEvent and LinkLeaveEvent for one agent on a link to calculate the travel time of that agent. In case the agent stopped on that link for an activity (indicated by the presence of an AgentArrivalEvent for that agent), the calculated travel time is ignored, as it does not represent the actual travel time.

The **VolumesAnalyzer** measures the traffic volume per link and per hour. It does so by counting the number of LinkLeaveEvents for each link and hour. This data is written to file for comparing to real-world traffic counts. In case the real-world traffic counts are available in a specific XML format, they can be loaded

by the Controler and a comparison of simulated traffic volumes versus real traffic volumes is automatically generated and output in different formats (e.g. as text-table for further analysis in statistical applications, or a special format for interactive visualization in Google Earth [47]).

CalcLegTimes measures the travel times for each leg and averages the values for each activity-pair connected by a leg. Each leg builds a connection between two activities. Legs between different types of activities usually have special characteristics, e.g. it is not uncommon that legs leading from home to work take longer than the legs from home to shopping. The calculated averages are written to a simple text file for further comparison with real-world values where available.

ScoreStats calculates some simple statistics at the end of each iteration based on the scores of the agents plans. The statistics are written to a file and a simple chart is generated, allowing one to easily track the progress of the simulation regarding the optimization of the scores during the iterations. ScoreStast is implemented as simple observer of Controler events, going through the population at the end of an iteration, analyzing the plans' scores.

The **Stopwatch** measures the execution times for specific tasks in the simulation, e.g. the time used for replanning, for writing files to disk, for running the traffic flow simulation. It is mostly used for the analysis of computational performance.

4.9 Optimizations

Computational speed is an important factor in simulations. It must be possible to calculate results in a reasonable time in order to be considered as a useful tool for engineers. Agent-based simulations offer several places at which performance optimizations can take place. But speed alone is not enough. As every simulated agent is represented as one—or more often as a group of multiple—software objects, memory consumption needs also to be respected. Even small changes to a single data structure may require a big amount of memory when several millions of agents use this data structure. To make matters worse, these two optimization directions—speed and memory—are often in opposition to each other: Higher speed could be achieved by caching more data in a suitable structure, while memory reduction can be achieved by re-calculating needed data instead of storing it. In the following subsections, some proven technologies for speed or memory optimizations are presented.

4.9.1 Simulation of Samples

One technique that has positive effects on both computational performance and memory consumption is the reduction of the actual number of simulated agents. For this, a random sample is drawn from the original demand, e.g. a 10% sample. Capacities in the network need also to be adapted accordingly. A link allowing 2000 vehicles per hour with the full demand should be reduced to pass only 200 vehicles per hour in a 10% sample. Also, the physical space must be adapted, such that similar spillback effects can be observed on jammed links.

The sampling reduces the memory consumption of the agents linearly to the sample size. Memory used by the network and other data structures (like collected data from analyses, or complementary data for routers and other mental modules) is usually not influenced by the sample size. The computational speed is slightly influenced, although not as strongly as the memory consumption. The simulation still needs to iterate over all links and nodes, which uses a big part of the computing time. But as the number of agents is smaller, fewer events are generated and need to be processed, reducing the time needed for computing.

Using a sample requires the thoughtful interpretation of the simulation outcome. Most obviously, the counted number of vehicles in the simulation must be scaled up again to be compared to real-world traffic volumes. But due to the discretization of the space on links, it is possible that some artifacts are created in the simulation that would not happen with a full demand, especially on very short links or links with a small flow capacity. While one has to be aware of these problems, they usually do not make it impossible to usefully interpret the results of such simulations.

4.9.2 Simulating Active Regions Only

Often, a large amount of traffic happens in certain regions only (like city centers), while other (rural) regions only have very light traffic or at certain times of day no traffic at all. The simulation still iterates over all links and nodes, checking in every time step every single link and node if there are some vehicles to be moved.

The amount of computing time can be reduced if the links that have cars on them are known. Then, the simulation only needs to iterate over those links. When an agent leaves a link, the link could be removed from the "simulated links" list if no other vehicle is on that link. When an agent departs, it must be ensured that the link is activated and simulated in the next time step.

A similar logic could be applied to nodes. As the time a node is active is usually rather small (an agent needs one time step to cross a node if its target link is not

jammed) and multiple links feed a node, this results in a high number of activations and deactivations of nodes. In our tests, the actual updating of the additional data structure containing the nodes to be simulated required about as much time as the simple checking of all nodes in every time step, which is why it was not implemented in the end.

The described technique is especially helpful when samples are simulated. In those cases, the probability of a link having no vehicles on it rises, allowing a higher number of links to be deactivated than with a full demand.

4.9.3 Multi-Threading

With the advent of multi-core CPUs or shared-memory machines (computers with more than one processor, accessing the same memory), concurrent algorithms become more and more popular to reduce computing times. Before, parallel programming was often limited to high-end cluster systems. But not every problem can easily be solved in parallel, limiting the applicability of parallel execution. One part in MATSim that can easily be solved in multiple threads is the replanning. Instead of sequentially processing one agent after the other, the workload can be split onto multiple threads, each processing a subset of all agents. This leads to a nearly linear reduction of computing time during the replanning phase, depending on the number of threads.

The idea arises that the QueueSimulation could make use of threads as well to process all links and all nodes in parallel per time step. A first, very simple approach that created new threads for each simulated time step did not perform well. This was to be expected, as the ramp-up and ramp-down of threads is involved with some overhead. A second approach of re-using the threads between time steps was not feasible due to the Java memory model [92]. As every thread has its own data cache, extensive data synchronization would have been necessary to ensure a correct working of the simulation. When one thread moves a vehicle from a link to its buffer, then another thread might move that vehicle from the buffer over a node to another link. For this to work, the second thread needs to see the vehicle in the buffer. But as each thread has its own memory cache, this visibility could only be reached with extensive flushing and reloading of the thread's caches, yielding in a worse performance than the single threaded variant. To successfully parallelize the simulation, one would have to reduce the amount of data synchronization that is required. This could be done by a complex domain decomposition, as it was done in earlier versions of MATSim [16] or by Charypar [18]. For the time being it was decided to keep the simulation single-threaded, accepting the worse performance

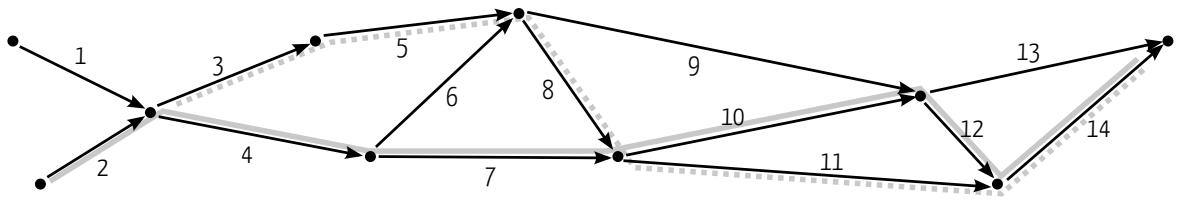
in favor of a cleaner design, that should also be less error-prone and easier extensible with new features.

4.9.4 Compressing Route Information

As one of the goals of MATSim is the simulation of large-scale applications, using samples is not always appreciated. In addition to the number of agents, also the number of links and nodes in the network has a major influence on the memory consumption. More links, usually in networks with a high level of detail, mean that the routes agents drive along consist of more links. Thus, the average number of links in a route influences heavily the memory consumption of a simulation. Especially in scenarios with very detailed networks and and a large number of agents, the amount of memory to store routes may make up a huge part of the total memory consumption (e.g. on a network with ∼880 000 links and 7 million agents, it was calculated that the routes use around 75 % of the total memory used).

By analyzing the network and calculating a designated *follow-up link* for each link of the network, the amount of data needed to store routes can be massively reduced. The follow-up link of a link should be the one that is most likely to be taken by agents. Some heuristics for finding such links are capacity, allowed speed, number of lanes or geometry. It seems plausible that agents often continue on links with the same or a higher capacity, or on links that continue in a similar direction as the current link the agents are on. Once such a follow-up link is identified for every link of a network, the routes only need to store the first link of a route and then each subsequent link of the route that is *not* the follow-up link of the previous link in the route. Fig. 4.8 shows a simple network and the lookup table containing the follow-up links for each link of the network together with two exemplary routes.

Compressing the data that needs to be stored for routes in this way yields in massive memory improvements in very large-scale scenarios. In the example mentioned above, memory consumption of routes could be reduced from about 75 % to 25 % of the total memory used, effectively saving more than half the memory. It must be noted though that this compression and decompression of the route information adds a small overhead, observable in a slightly higher runtime of the simulation.



Link:	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Follow-up Link:	4	4	5	7	9	9	10	10	13	13	14	14	.	.

	Start Link	End Link	Uncompressed Route	Compressed Route
Route 1	2	14	4 7 10 12 14	12
Route 2	3	14	5 8 11	8 11

Figure 4.8: Only links deviating from the follow-up link are stored in compressed routes

5 Road Pricing Case Study

The work reported in this chapter was presented as Modelling and simulation of a morning reaction to an evening toll *[98] at the conference* Innovations in Travel Modeling 2008 *in Portland, OR. Most of the following sections are excerpts from the presented paper, modified to integrate them into this dissertation.*

This chapter presents a case study performed with MATSim. In the case study, the effects of a time-dependent toll are researched. The hypothetical toll is applied to the area of Zurich, Switzerland.

5.1 Introduction

The modeling of traffic tolls becomes more and more important in transportation planning as there is usually no possibility to test toll schemes in reality. Time-dependent tolls are often preferred, as they allow for more fine-grained control of traffic flows. But the modeling of time-dependent tolls requires special attention to ensure that the model stays consistent with human behavior.

Traditional transportation planning tools work macroscopically, distributing static traffic flows onto a network. While this is a well-established technology, it is not able to fully model all aspects that are of interest when modeling tolls. In particular, they usually lack any meaning of time-of-day.

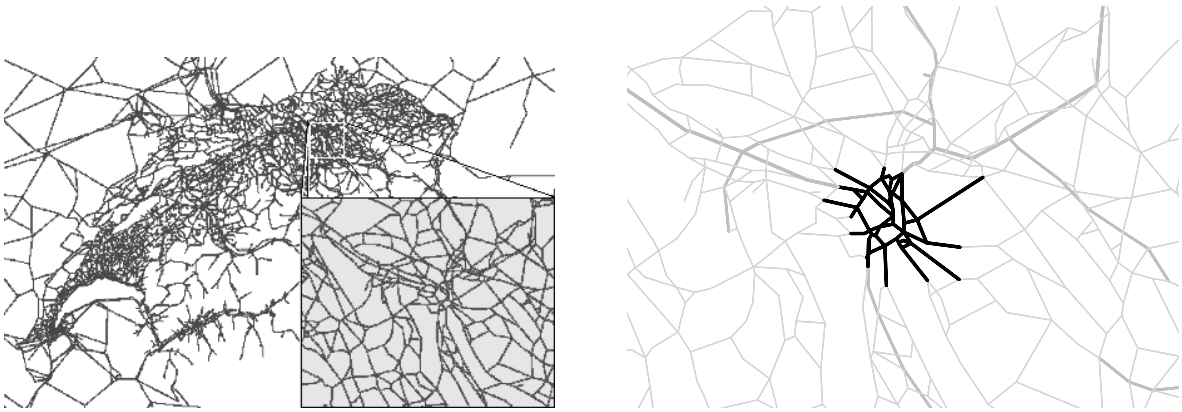
Dynamic traffic assignment (DTA) explicitly models the temporal development of the traffic. Demand, however, is typically given as fixed-period (e.g. hourly) origin-destination (OD) matrices, and does, in consequence, not adapt to the toll. Adaptation would need to happen in the demand generation modules that generate the OD matrices, but that implies rather intricate coupling between demand generation and DTA.

Every model that uses single trips only will have problems predicting useful reactions of travelers that span the whole day. This is because trips in real life are embedded in a complete day plan. This means that travelers cannot escape a toll at their will, but have to trade off between different utilities (working eight hours, being at a shop when it has opened, …) and disutilities (paying a toll, being late for work, …). Thus a toll may influence the whole day schedule of a person, and not only the duration the toll is active.

Our approach uses multi-agent simulations to model and simulate full daily plans. This allows us to research the influence of time-dependent tolls more thoroughly than traditional tools are able to.

5.2 Scenario

The scenario is the same as in [10]. It covers the area of Zurich, Switzerland, and has about one million inhabitants. The network is a Swiss regional planning network, extended with the major European transit corridors (Fig. 5.1a). It has the fairly typical size of 10k nodes and 28k links.



(a) Switzerland network with area of Zurich enlarged (b) hypothetical toll links in Zurich area

Figure 5.1: Switzerland network and toll links for Zurich

The simulated demand consists of commuters only that travel by car in the aforementioned region, resulting in 260k agents, all with an activity pattern home-work-home. The initial time structure has the agents leaving home in the morning at a randomly chosen time between 6am and 9am, work for 8 hours, and then returning to home. For the work activity a starting time window is defined between

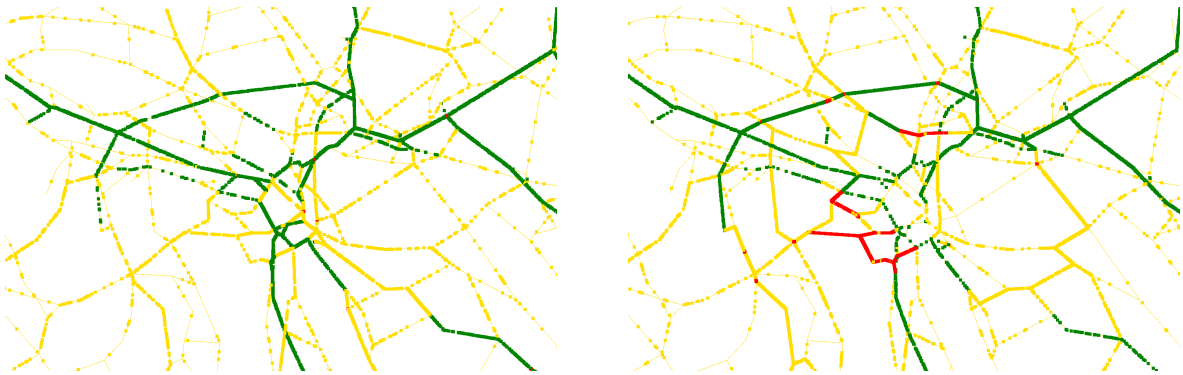
7am and 9am.

We defined a hypothetical toll area that covers the inner city of Zurich (Fig. 5.1b). The diameter of the toll area is about 6km. The toll is restricted to the evening (3pm to 7pm) only, and the toll amount of 2 Euro/km is subtracted from the plan's score. Restriction of the toll to the evening is done to illustrate that the agent-based approach is able to consider ramifications throughout the whole day. In particular, it will be shown that the morning traffic is significantly affected by the evening toll. As was discussed earlier, this is an effect that a trip-based model cannot represent. The covered area has a high density of offices and other work places, so the in-bound traffic is larger in the morning than the out-bound traffic, and vice versa in the evening.

A base case without the toll was first iterated until a relaxed state was reached. Based on this state, a new run was started with the toll switched on, again until a (new) relaxed state was reached. This allows researching the specific influence the toll has on the behavior of the travelers.

5.3 Results

A first visual validation is done by looking at the traffic volumes and velocities. Fig. 5.2 shows the velocity of agents at 5:30pm, during the toll hours. One can clearly see that there are more agents traveling around the toll area in the toll case by the traffic jams they produce.

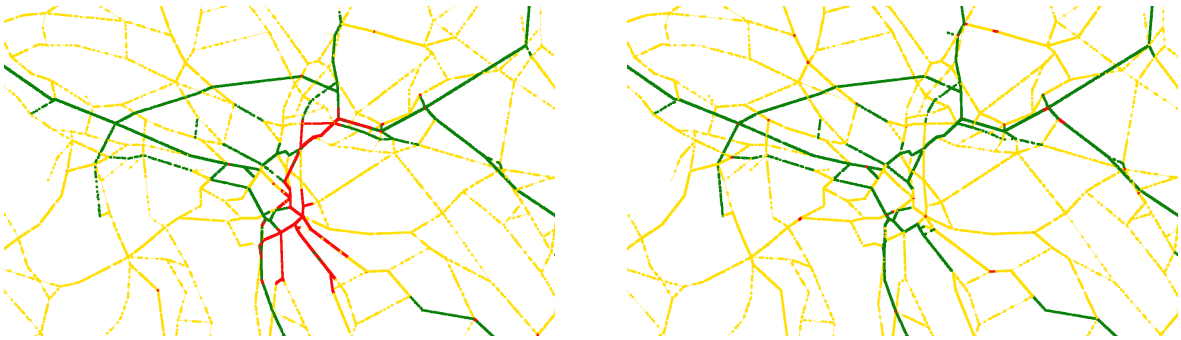


(a) base case (b) toll case

Figure 5.2: Travel speeds at 5:30pm during the toll time on the network: green are high speeds, red marks traffic jams.

One can also compare the two runs in the morning hours at 8am. Note that at this time of the day, there is no active toll either case. As can be seen in Fig. 5.3, there are traffic jams in the base case, but none of them in the toll case. This clearly shows that the toll in the evening rush hour has an influence on the morning rush hour.

Fig. 5.4a shows the departure time distribution for the base case and the toll case. Comparing the toll case with the base case in the evening peak, one can see how the number of travelers departing from work is higher in the toll case than in the base case in the time before the toll starts. It is also slightly higher after the toll ends. However, during the time the toll is active, the number of travelers departing is lower in the toll case than in the base case.

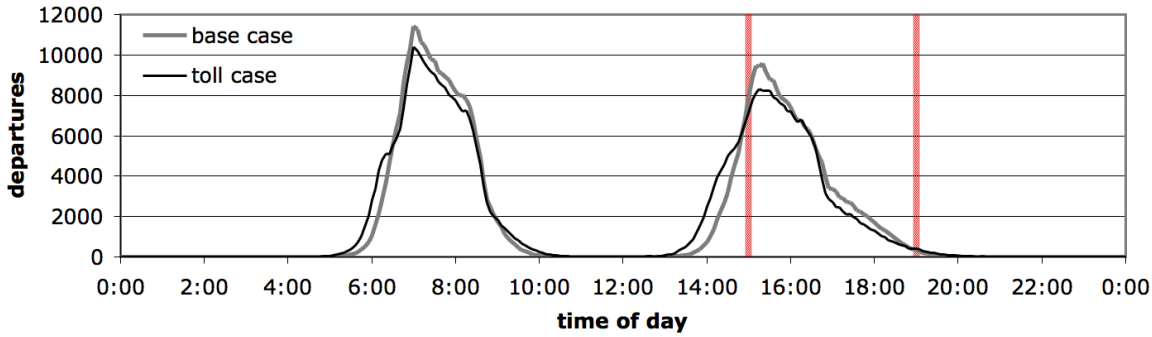


(a) base case (b) toll case

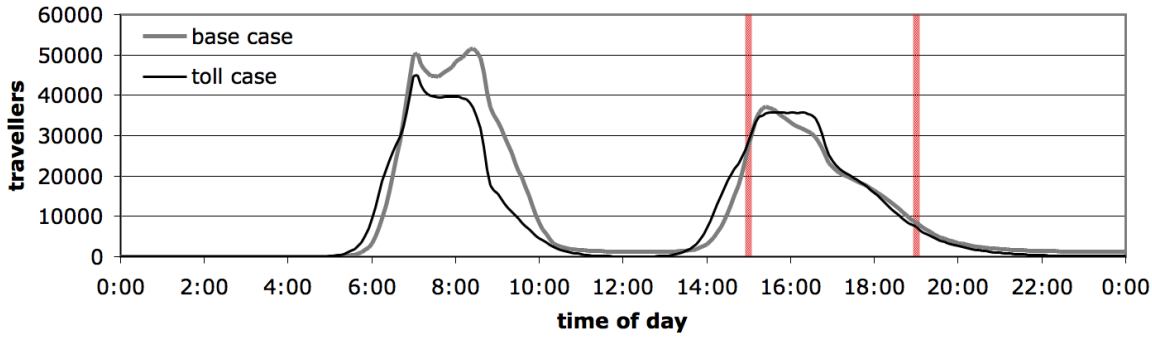
Figure 5.3: Travel speeds at 8am when no toll has to be paid: green are high speeds, red marks traffic jams.

As each traveler tries to work eight hours a day, the same characteristics can also be seen in the morning rush hour, as agents planning to leave before 3pm will also have to arrive at work earlier than the others. This leads to a general broadening also of the morning peak. Without full daily plans, there would be no difference in the morning peak.

If the peaks of departing travelers are broader but less high, this means also that there are likely fewer people traveling at the same time. Fig. 5.4b shows the number of travelers simultaneously on the road. Especially in the morning rush hour it is apparent that the area below the curve is significantly smaller than in the base case. The area below the curve can be interpreted as the total time agents spend on the road. A smaller area means that people spend less time in total traveling—and all this without a toll in the morning rush hour!



(a) Number of departures



(b) Number of travelers

Figure 5.4: (a) Number of departures and (b) number of travelers on the road over the time of day. The red lines mark the start and the end of the time a toll has to be paid.

The case is a bit different in the evening rush hour. Around 4pm we actually have more travelers on the road than in the toll case. This can be explained if one remembers that the toll area is only a small part of the whole simulated area: The travelers only have to get out of the toll area before the toll starts (as can be seen in the higher number of departures and travelers between 2pm and 3pm). However, this has the consequence that there may now be more travelers outside the toll area—and that's what can be observed in Fig. 5.4b at 4pm.

5.4 Conclusions

We have shown that the use of full daily plans in multi-agent simulations can be used to model travelers' reactions to a time-dependent toll in a way most existing transportation planning tools are not able to. In particular, the interdependence of different trips for a single agent throughout the day is taken into account by the simulation.

6 Simple Transit Integration

The work reported in this chapter was published at the 88th Annual Meeting of the Transportation Research Board in Washington, D.C. as Paper 09-2758, Adding Mode Choice to Multi-Agent Transport Simulation *[99]. Most of the following sections are excerpts from the published version, slightly modified to better integrate them into this dissertation.*

This chapter presents the extensions implemented into the MATSim simulation framework to support not only car legs, but also other modes of transport. The description is based on the concepts presented in chapter 4.

6.1 Mode Choice Model

The basic idea behind the mode choice model is that each agent always has at least one "car" plan and one "non-car" plan. Apart from that, plans are treated as described earlier. Since this always keeps both modes in the choice set, a decision between plans according to Eq. 4.5 is also a choice between modes.

This requires changes in many parts of the simulation framework, namely the transport simulation, the scoring of plans as well as the replanning. These changes are described in the following.

6.1.1 Generating Non-Car Plans

To generate non-car plans, an initial demand with car plans must exist already. Starting with that initial demand, the leg modes of all legs in each plan are set to "car", and the fastest routes are calculated. Then, each plan is duplicated, changing all leg modes in the duplicated plans to "non-car".

The duration of every non-car trip is assumed to take twice as long as the car mode at free speed, but no exact route is provided. This is based on the (informally stated) goal of the Berlin public transit company to generally achieve door-to-door travel times that are no longer than twice as long as car travel times. This, in turn, is based on the observation that non-captive travelers can be recruited into public transit when it is faster than this benchmark [96]. For the purposes of this paper, it is assumed that all non-car modes very roughly have the shared characteristics that they are slower than the (non-congested) car mode—this will be further disaggregated in future work. In the same vein, both for car and for non-car trips there are no separate considerations of access and egress.

6.1.2 Handling Non-Car Plans in the Transportation Simulation

Currently, the simulation only supports a road-network, but no walk- or rail-network. Thus, only car legs can be truly simulated. Agents on non-car legs are teleported from one location to the next. But the teleportation is not instantaneously, but takes some amount of time, which can be stored in the legs as planned travel duration. While this does not impose any transit vehicles' capacity constraints, it would still allow us to have individual travel times, depending on agents' demographics or chosen non-car mode (e.g. bike, walk, transit, …). The simulation still generates departure and arrival events for non-car legs, which can be used for analyses.

6.1.3 Scoring Non-Car Plans

The scoring of non-car plans is very similar to the scoring of car plans as described in Sec. 4.6, only the marginal disutility of traveling changes. This is expressed by using $\beta_{tr,nc}$ for the marginal utility of traveling, instead of $\beta_{tr,car}$. Note once more that $\beta_{tr,car}$ and $\beta_{tr,nc}$ are *not* values of time by themselves, but they are *additional* marginal disutilities caused by traveling, in addition to the opportunity cost of time. This is consistent with econometric approaches [55].

6.1.4 Replanning With Non-Car Plans

During replanning, plans are duplicated and modified (see Sec. 4.7). This also holds for non-car plans. The only difference is that the plans deletion module makes sure that at least one plan of every mode is kept for every agent. This is to make sure that all agents keep their ability to change mode until the end of the iterations.

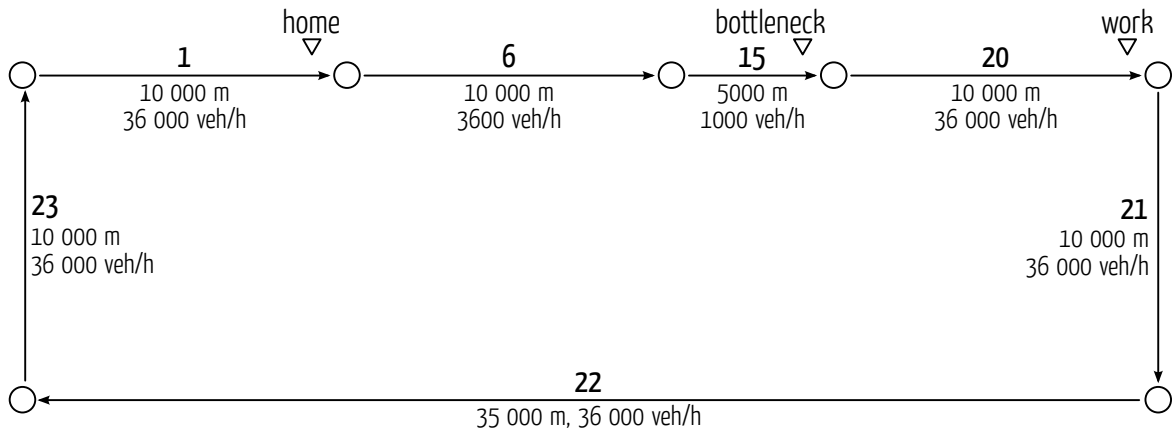


Figure 6.1: The links of the test network with their corresponding ids and attributes. Link 15 has reduced capacity. Traffic runs clockwise, i.e. agents have their home location at link 1 and work on link 20, so link 15 provides a bottleneck.

The above steps integrate mode choice into the replanning process that takes place iteratively with the simulation. Instead of precalculating the mode choice before the traffic assignment, as it is done in the traditional four-step process, mode choice is now treated at the same level as route choice in the traffic assignment.

6.2 Test Scenario

6.2.1 Network

To test the mode choice model, a simple test network was used (see Fig. 6.1), consisting only of a cycle of one-way links.[1] The capacity of all links except links 6 and 15 are (unrealistically) high as to minimize the influence these links have on the traffic, essentially making it possible for most agents to drive with free speed. Links 6 and 15 have reduced capacity, building a bottleneck.

6.2.2 Initial Plans

The synthetic population consists of 2000 agents. All agents have their home activity at link 1, which they initially leave at 06:00. They drive to work (located on link 20) with a car via links 6 and 15, where they stay for 8 hours, after which

[1]It is a simplified version of another test network that is used internally, which explains the numbering of the links.

they drive back home to link 1 via links 21–23. The free speed travel time from link 1 to link 20 is 15 minutes. The free speed travel time from link 20 to link 1 is 39 minutes. Thus the total free speed travel time driving by car is 54 minutes or 0.9 hours.

As the agents are forced to remain on that route, the scenario converts into the well-known Vickrey bottleneck scenario [4, 117]; also see below for more details.

In addition, each agent possesses an initially non-active plan that uses the non-car mode for both trips. These trips take twice as long as by car in an empty network, i.e. 30 minutes from link 1 to link 20, and 78 minutes from link 20 to link 1. The total non-car travel time is 108 minutes or 1.8 hours. In contrast to the car travel times, these non-car travel times are not affected by congestion. The first trip starts at 06:30, so the agents will arrive exactly at 07:00 at their work place.

6.2.3 Behavioral Parameters

The behavioral parameters are set and can be interpreted as follows:

- utility of performing an activity at its typical duration: $\beta_{perf} = 6$ Euro/h
- marginal disutility of coming late: $\beta_{late} = -18$ Euro/h
- additional marginal disutility of traveling with a car: $\beta_{tr,car} = -6$ Euro/h
- additional marginal disutility of traveling with non-car mode: $\beta_{tr,nc} \in [-10, -9, .., +2]$ (see below)
- constant in binary logit model (Eq. 4.5): $\beta = 2$
- "typical" durations of $t_{*,w} = 8$ and $t_{*,h} = 12$ hours for work and home mean that work and home times have a tendency to arrange themselves with a ratio of 8:12 (i.e. 2:3): Assume a fixed travel time budget. In this situation, for optimality of the scoring function the marginal utilities of duration, $\partial U_{perf,i}/\partial t_{perf,i} = \beta_{perf} t_{*,i}/t_{perf,i}$, need to be equal for all activity types, resulting in

$$\frac{t_h}{t_{*,h}} = \frac{t_w}{t_{*,w}} \tag{6.1}$$

  The result is only approximately correct when the overall travel time varies.
- A work start exactly at 7:00am means that (a) no utility can be accumulated from an arrival earlier than 7:00am, and (b) any late arrival is immediately punished with $\beta_{late} = -18$ Euro/h. Because of the argument made earlier regarding the opportunity cost of foregone activity time in situation (a), the *effective* marginal disutility of early arrival is $-\beta_{perf} t_{*,i}/t_{perf,i} \approx -\beta_{perf} =$

-6 Euro/h. Since the effective marginal disutility of car traveling is, by the same argument, $-\beta_{perf} t_{*,i}/t_{perf,i} \approx -\beta_{perf} - |\beta_{tr,car}| \approx -12$ Euro/h, the *effective* values of time of our study are approximately the same as the consensus values of (–6, –12, –18) of the Vickrey scenario [4, 117]. The return trip has no influence since there is no congestion.

6.2.4 Simulation Results

The simulation in the test setup was run with different values for $\beta_{tr,nc}$, resulting in different mode shares. Each simulation was first run for 1000 iterations. In each iteration, 10% of the agents were modified by the time allocation module, while all other agents chose an existing plan. After that, the simulation was continued for 100 more iterations, but without time adaptation. This allowed agents to select their best plan, no longer being forced to execute (possibly bad) plans after replanning.

$\beta_{tr,nc}$ was varied from +2 to –10 in increments of –1. Fig. 6.2 shows the resulting car mode shares as dots. It can clearly be seen that an increase of the marginal disutility of traveling in the non-car mode leads to an increasing number of agents choosing car as transportation mode. In the following section, these results are validated by comparing them to the theoretical values one should expect based on the aforementioned mode choice model.

6.3 Theoretical Calculations

Because of the simulation set-up, the mode share of the car mode, f_{car}, follows a binary logit model:[2]

$$f_{car} = \frac{\exp(\beta \cdot U_{car}(f_{car}))}{\exp(\beta \cdot U_{car}(f_{car})) + \exp(\beta \cdot U_{nc})} \tag{6.2}$$

U_{car} and U_{nc} are the total utilities of agents traveling either with a car or using the non-car transport mode. It is really important to note that these are utilities for the full daily plan, and not partial utilities for the mode choice contribution only. These utilities are defined according to Eq. 4.1, with only the two activities "home" and "work":

[2]This statement is, in fact, only correct when the number of car plans is equal to the number of non-car plans for every agent. See the end of the section for a comment on this.

$$\begin{aligned} U_{mode} &= \beta_{perf} \cdot t_{*,h} \cdot \ln\left(\frac{t_{h,mode}}{t_{0,h}}\right) + \beta_{perf} \cdot t_{*,w} \cdot \ln\left(\frac{t_{w,mode}}{t_{0,w}}\right) \\ &+ \beta_{tr,mode} \cdot t_{tr,mode} + \beta_{late} \cdot t_{late} \end{aligned} \tag{6.3}$$

As mentioned before, travel times depend on the transport mode.

6.3.1 The Non-Car Mode

Taking the "activity duration ratio" Eq. 6.1 together with the time budget equation

$$t_{h,nc} + t_{w,nc} + t_{tr,nc} = 24\ h\ ,$$

one obtains for people using the non-car mode:

$$t_{h,nc} = (24\ h - t_{tr,nc}) \cdot \frac{t_{*,h}}{t_{*,h} + t_{*,w}} \tag{6.4}$$

$$t_{w,nc} = (24\ h - t_{tr,nc}) \cdot \frac{t_{*,w}}{t_{*,w} + t_{*,h}} \tag{6.5}$$

At this point, all variables for Eq. 6.3 for the non-car mode, assuming on-time arrival, are expressed in the parameters of the simulation.

6.3.2 The Car Mode

For car users, the calculation is more complex. Following [4, 117] we will assume that at the end of the day every agent will have experienced the same total utility: While some may spend more time traveling (by being stuck in a traffic jam) but arrive at the right time at the work place, other agents may decide to leave early, traveling the whole route with free speed but also arrive at work early, foregoing any utility by performing an activity because the work place is still closed. Other agents again may stay longer at home, traveling after the jam has disappeared, arriving late at work and receiving the schedule delay penalty for that. One can obtain results by just looking at the first and the last agent to arrive at work. When equating Eq. 6.3 for these two, the travel time drops out because it is the same for both, and one arrives at

$$\begin{aligned} &\beta_{perf} \cdot t_{*,h} \cdot \ln\left(\frac{t_{h,car} - \tau_h}{t_{0,h}}\right) + \beta_{perf} \cdot t_{*,w} \cdot \ln\left(\frac{t_{w,car} - \tau_w}{t_{0,w}}\right) \\ &= \beta_{perf} \cdot t_{*,h} \cdot \ln\left(\frac{t_{h,mode}}{t_{0,h}}\right) + \beta_{perf} \cdot t_{*,w} \cdot \ln\left(\frac{t_{w,car}}{t_{0,w}}\right) + \beta_{late} \cdot t_{late}\ , \end{aligned}$$

where the LHS refers to the person who arrives early, and who suffers τ_h, τ_w reductions of his/her activity durations. After linearization and dropping terms that cancel out, this becomes

$$-\tau_h \cdot \beta_{perf} \cdot t_{*,h} \cdot \frac{1}{t_{h,car}} - \tau_w \cdot \beta_{perf} \cdot t_{*,w} \cdot \frac{1}{t_{w,car}} \approx \beta_{late} \cdot t_{late} \ ,$$

From the optimal time allocation, Eq. 6.1, one infers that also for the time deductions τ_h, τ_w one needs $\tau_h/\tau_w = t_{*,h}/t_{*,w}$ and therefore $\tau_h = t_{early} \cdot t_{*,h}/(t_{*,h} + t_{*,w})$ and $\tau_w = t_{early} \cdot t_{*,w}/(t_{*,h} + t_{*,w})$. Taking this and once more Eq. 6.1 directly, one obtains, after some algebra

$$t_{early}\, \beta_{perf} \frac{t_{*,h}}{t_{h,car}} \approx |\beta_{late}|\, t_{late} \ , \tag{6.6}$$

where it was also invested that β_{late} is assumed to be negative. In addition, one has the equation for the bottleneck,

$$t_{early} + t_{late} = \frac{|A| \cdot f_{car}}{C_b} \tag{6.7}$$

where $|A|$ is the total number of agents, C_b is the flow-capacity of the bottleneck, and f_{car} the share of car users. The equation states that the capacity of the bottleneck is exactly enough to serve all agents between the first and the last. Inserting Eq. 6.6, one obtains

$$t_{early} \approx \frac{|\beta_{late}|\, t_{h,car}}{|\beta_{late}|\, t_{h,car} + \beta_{perf}\, t_{*,h}} \cdot \frac{|A| \cdot f_{car}}{C_b} \tag{6.8}$$

The optimal activity durations for the "early" agent are, similar to Eq. 6.4 and 6.5:

$$t_{h,car} + t_{w,car} \quad + \quad t_{tr,fs} + t_{early} = 24\ h$$

$$t_{h,car} \quad = \quad (24\ h - t_{tr,fs} - t_{early}) \cdot \frac{t_{*,h}}{t_{*,h} + t_{*,w}} \tag{6.9}$$

$$t_{w,car} \quad = \quad (24\ h - t_{tr,fs} - t_{early}) \cdot \frac{t_{*,w}}{t_{*,w} + t_{*,h}} \ , \tag{6.10}$$

where $t_{tr,fs}$ is the free speed travel time by car. Substituting $t_{h,car}$ from Eq. 6.9 into Eq. 6.8 leads to an equation that only contains t_{early} and f_{car} as unknowns. One can see that the resulting equation contains the square of t_{early}. Solving that resulting equation provides two solutions for t_{early}, of which only one is useful, as the other one leads to negative times for either t_{early} or t_{late} in Eq. 6.7. Thus at this point one knows t_{early} and in consequence $t_{h,car}$ and $t_{w,car}$ as functions of f_{car}. The expressions can be written down, but are rather long and not easy to interpret.

6.3.3 The Complete Mode Choice

Recall that we are interested in an expression that relates the mode share, f_{car}, and the additional marginal disutility of the non-car mode, $\beta_{tr,nc}$. What we have at this point is:

- We can compute the utility of the optimal non-car plan as a function of $\beta_{tr,nc}$.
- We can compute the utility of the optimal car plan as a function of f_{car}.

What remains is to insert these expressions into Eq. 6.2, which can also be written as

$$U_{car} = \frac{1}{\beta} \cdot \ln\left(\frac{f_{car}}{1 - f_{car}}\right) + U_{nc} \tag{6.11}$$

Substituting U_{car} and U_{nc} with Eq. 6.3, one gets:

$$\beta_{perf} \cdot t_{*,h} \cdot \ln\left(\frac{t_{h,car}}{t_{0,h}}\right) + \beta_{perf} \cdot t_{*,w} \cdot \ln\left(\frac{t_{w,car}}{t_{0,w}}\right) + \beta_{tr,car} \cdot t_{tr,fs} =$$
$$\frac{1}{\beta} \cdot \ln\left(\frac{f_{car}}{1-f_{car}}\right) + \beta_{perf} \cdot t_{*,h} \cdot \ln\left(\frac{t_{h,nc}}{t_{0,h}}\right) + \beta_{perf} \cdot t_{*,w} \cdot \ln\left(\frac{t_{w,nc}}{t_{0,w}}\right) + \beta_{tr,nc} \cdot t_{tr,nc} \tag{6.12}$$

Recall that for the car mode we are considering the "first" (= most early) agent; the term regarding late arrival is thus dropped.

More variables can be substituted in Eq. 6.12 by their corresponding calculations in the previous equations. While it could still be solved analytically, it once more gets quite complex and not easily readable.

Evidently, $\beta_{tr,nc}$ can be isolated in Eq. 6.12, but not so f_{car} if one remembers that f_{car} is also part of t_{early} which is used to substitute $t_{h,car}$ and $t_{w,car}$ (Eq. 6.8–6.10).

Extracting $\beta_{tr,nc}$ and plotting it as a function of f_{car} ranging from 0 to 1, one gets the line shown in Fig. 6.2. Comparing it with the simulation (dots), one can see the the results are very similar. Only small variations can be seen, likely due to the discrete size of agents in the simulation as well as the not completely predictable behavior of random numbers used in the simulation. Additionally, the calculations assume that every agent has an optimal plan, which cannot be guaranteed in the simulation.

The fact that in spite of the noise the mode choice curve is "steeper" in the simulations than in the analytical calculations is due to the learning algorithm: If for an agent one mode is clearly better than the other mode, than that mode will

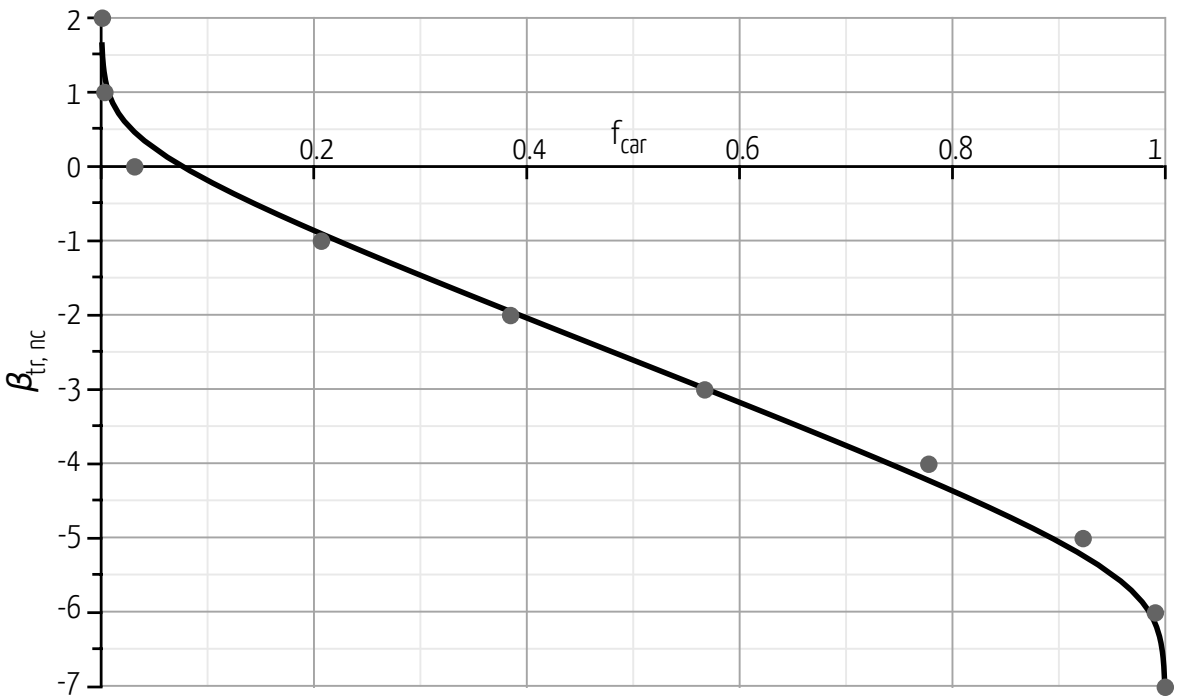


Figure 6.2: Non-car travel marginal disutilities ($\beta_{tr,nc}$) for different car shares (f_{car}). The line refers to the analytical result, the dots to the simulation results.

have more plans than the other mode. This gives an additional statistical advantage to the better mode, making the curve more steep.

Overall, one finds that the mode choice model is in excellent agreement with the theoretical calculations. This, on the one hand, verifies the implementation of the model. On the other hand, it means that, to an extent, it is possible to understand analytically what the simulation does, which will help to uncover and understand the economic and behavioral principles embedded in the implementation.

6.4 Large-Scale Application

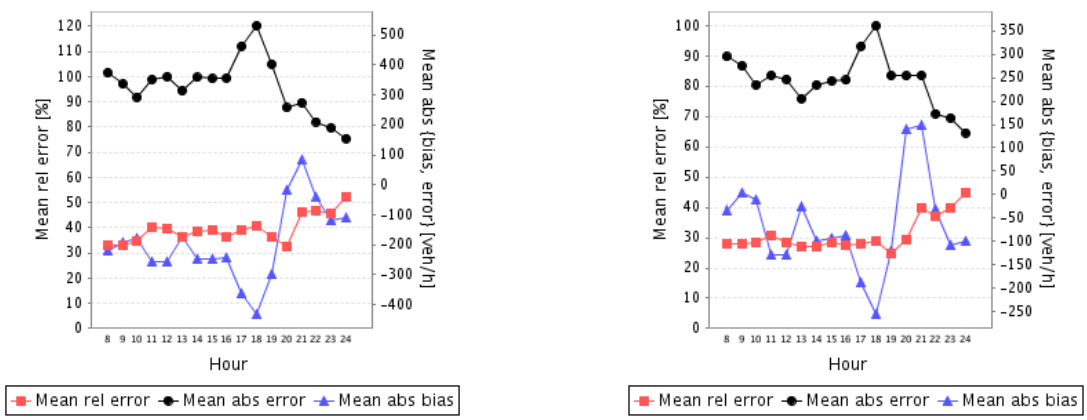
The mode choice model was also applied to a large-scale, real-world scenario. We used the area of Zurich, Switzerland, for this application, which has about 1 million inhabitants. The following paragraphs only give a simplified description of the scenario. A full description of the scenario can be found in [22].

The network used is a Swiss regional planning network that includes the major European transit corridors. It consists of 24 180 nodes and 60 492 links.

The simulated demand consists of all travelers within Switzerland that are inside an imaginary boundary around Zurich at least once during their day [22, 120]. All agents have complete day plans with activities like *home*, *work*, *education*, *shopping*, *leisure*, based on microcensus information [103, 104]. The time window during which activities could be performed was limited to certain hours of the day: *work* and *leisure* could be performed from 06:00 to 20:00, *shopping* from 08:00 to 20:00, while *home* and *leisure* had no restrictions. Unlike the sample scenario described in the sections above, there is no punishment for being late. This was not possible because agents could split their work activity into two or more parts, e.g. one in the morning and one in the afternoon. In such a case it would be complicated to specify when an agent starts an activity late or not.

To speed up computations, a random 10% sample was chosen from the synthetic population for simulation, consisting of 181 725 agents. In this large-scale application, the agents could not only perform time adaptation as described in a previous sections, but could also do route adaptation, which is essential for the car mode. For comparison, the same scenario was run with the pre-calculated mode choice (see [22]).

Simulated traffic volumes were compared with the hourly traffic volumes from 159 real-world counting stations. Fig. 6.3 shows, in red, the mean relative error between hourly flows in reality and hourly flows from the simulation. The left figure contains the result from the fixed, pre-determined mode choice, the right figure the result of the new adaptive mode choice which was explained in this paper. One notices a quite distinct reduction in the average error, from about 40% to about 30%. Also the absolute bias, in blue, is reduced.



(a) pre-calculated mode choice

(b) mode choice during simulation ($\beta_{tr,nc} = -3$)

Figure 6.3: Comparison of simulated traffic volumes with real-world counts. Note different scales on y-axis

For the large-scale tests, the marginal disutility for the car mode, $\beta_{tr,car}$, was set to –6 Euro per hour, while the marginal disutility for the non-car mode, $\beta_{tr,nc}$, was varied between 0 and –6. An interpretation of this might be that measures are discussed that change the attractiveness of the non-car modes, leaving everything else, including the travel times, constant. An obvious concrete example would be fare changes. And the importance of the results at this point is not so much the magnitude of the response itself, but the fact that the model displays the interaction between activity timing and mode choice. Fig. 6.4 shows the number of agents en route with cars over the time of day. It can be clearly seen that the number of car users decreases the lower the travel marginal disutility for the non-car mode gets. The peaks at 6am and 8pm are due to our opening time restrictions.

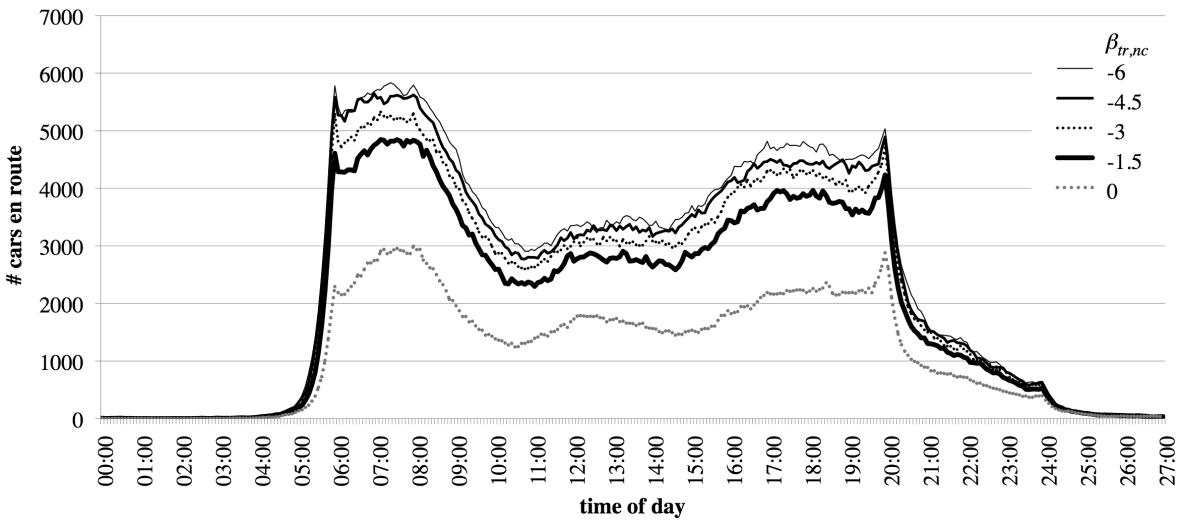
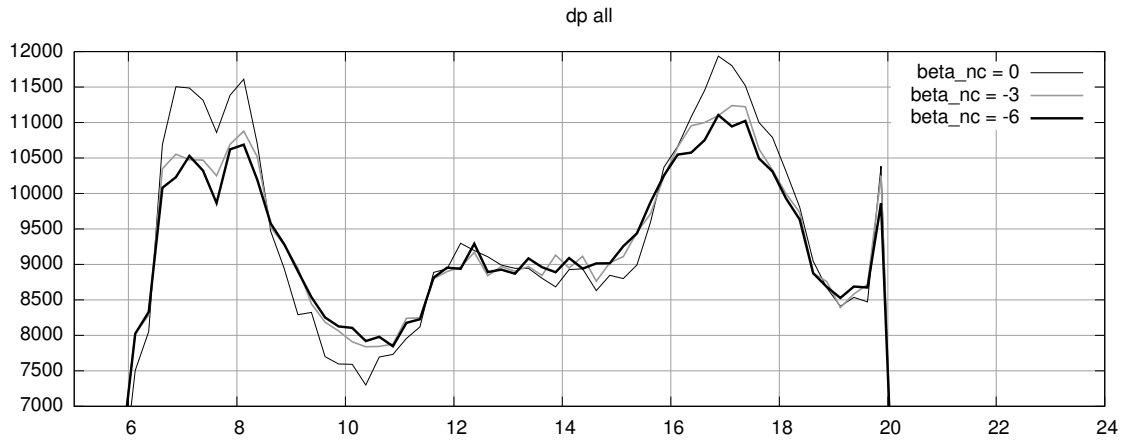


Figure 6.4: Car en route in large-scale scenario over time of day with different marginal disutilities for traveling with non-car modes.

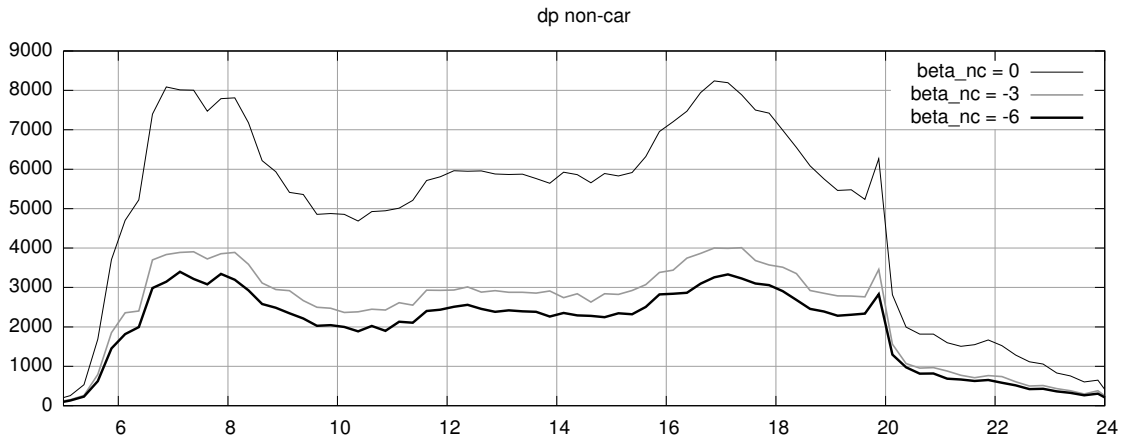
Figure 6.5a shows *all* departures as a function of the time-of-day, for different values of $\beta_{tr,nc}$. Since demand itself is inelastic, the area under all the curves is the same. One notices, however, a shift towards the peak periods when the marginal disutility of the non-car mode is reduced. Together with further results, discussed below, the reason is that, because of people moving away from the car, the peak period is less congested, meaning that other car-users re-adjust their schedules towards the peak hour.

Non-car departures (Fig. 6.5b) show the expected behavior: More non-car departures at all times as a function of a reduced marginal disutility.

Outside the time from 6am to 6pm, the non-car share is markedly lower than



(a) number of agents departing per 15 min over time of day



(b) number of agents departing with non-car mode per 15 min over time of day

Figure 6.5: Number of agents departing per 15 min over time of day.

during the day. This is due to the fact that during those times there is little car congestion, thus making the car more attractive.

6.5 Further Steps

At the moment, only one transport mode can be used for the complete plan. That is, all trips of a given day need to be done by the same mode. While the data structures, file formats and simulation could deal with different transport mode per leg, there are some conceptual points that we want to solve first before applying mode choice to a subtour level.

The simulation setup would allow to have different $\beta_{tr,nc}$ over the time of day, as every trip has a departure time. This could be used to model a changing attractivness to use the non-car modes during a day. One example might be to improve the quality of service in transit in the late evening or night hours, resulting in a lower absolute marginal disutility during that time of day.

An improved router for non-car modes would improve results. Possibilities are the usage of transit schedules instead of the "double free speed travel time" assumption currently used. Currently, that assumption makes the non-car mode highly unattractive for long distance trips. This will likely change by using more realistic travel times, especially for long distance trips that are served well by fast trains.

The simulation should not only teleport agents with non-car mode, but actually simulate them as well. Different aspects of this would be important to include, say, public transport vehicle overcrowding effects, or the effect of public transport being caught in car congestion. It would require to add transit vehicles, bikes and other means of transport, together with their characteristics, schedules and so on.

A car ownership model, or arguably a life style model, could be added in the demand modeling. This would reduce the choice between car and non-car mode to travelers that actually have access to a car. A preliminary attempt to do this for the Zurich scenario did *not* lead to improved results with respect to the real world traffic counts. This was presumably due to the fact that the car ownership model was based on zonal characteristics, while the mode choice model of our simulation at least on the car side picks up very detailed accessibility issues. It becomes quite clear that the behavioral basis of all relevant decision models needs to be consistent.

6.6 Conclusion

It was shown how to include a non-car mode into a multi-agent transport simulation with relatively few conceptual changes. The non-car mode was integrated by giving every agent *two* initial plans, one using the car for all trips, and one using the non-car mode for all trips. The non-car trips are assumed to use up twice as much travel time as the uncongested car mode. Travelers can then, in the simulation, adjust times and car routes; the performance of the resulting plans is scored after execution in the traffic flow simulation, based on a utility function that includes positive utility for performing an activity, different negative utilities for traveling by different modes, and opening times outside which no utility for performing an activity can be accumulated.

The model was first tested in a simplified scenario based on the famous Vickrey bottleneck example. The non-car mode was used as an alterative to the congested car mode. It was shown that the analytical calculation and the simulation model produce the same results when looking at the mode split as a function of the non-car mode marginal disutility.

The model was then applied to a realistic real world example for the Zurich metropolitan area. The reaction of users to changes in the non-car marginal disutility was analyzed in some detail, including temporal reactions. Adding the mode choice to the large-scale scenario improved the realism of the scenario when comparing the simulated traffic volumes to data from counting stations.

7 Design of the Agent-Based Transit Simulation

7.1 Goals and Features

The simulation of transit can have different goals, as it was already mentioned in chapter 3. In Sec. 4.4 it was discussed that one has to carefully decide—based on the desired goals—to what detail the traffic flow should effectively be simulated and what simplifications should be used, such that an adequate computational performance results. This also holds true when implementing (or integrating) a transit simulation.

In this work, the main reason to implement a transit simulation is to extend the realism of a (general) transportation simulation that includes behavioral decisions. Some motivation for this was given in chapter 5, where also the short comings of a simulation not taking transit into account were shown. This is in contrast to other tools that focus stronger on the detailed operational simulation of transit traffic, or even specializing on rail traffic only (e.g. OpenTrack [75]).

The following list gives an overview of desired features for the transit simulation, adding to the features of the existing traffic flow simulation:

- Movement of transit vehicles on the network
- Interaction of transit vehicles with private cars on roads
- Transit vehicles move according to a transit schedule
- Simulation agents can decide, which transportation system they want to use
- Simulation agents can board and alight transit vehicles
- Transit vehicles have capacity constraints
- Fares paid by agents must be tracked somehow

The limitation to this feature list forbids the implementation of some other aspects. E.g. a detailed track occupancy analysis will not be possible, nor will it be possible to model different type of controls of the transit vehicles (e.g. signal along the rails versus showing signals in the driver's cabin as it is done in high speed trains according to the European Train Control System – ETCS).

Still, the implementation should offer enough flexibility such that it can later also be extended to include additional features, like:

- Using detailed driver schedules for the simulation, where drivers can also change vehicles and lines during their work shift.
- Using vehicle schedules, such that delays on one course are carried forward to other courses as well.
- Implementing paratransit-like traffic, that does not follow a strict schedule.
- Implementing a wide variety of fare structures

The features mentioned above mostly relate to the traffic flow simulation only. To be able to fully use those possibilities in the traffic flow simulation, the surrounding simulation framework must also be extended. In the replanning part, if agents choose transit as transportation mode, a *Transit Router* must generate expedient routes using the transit transportation mode leading from one place to another. The *Scoring* function must be adapted to account differently for the time spent traveling in the transit, and should also take into account additional cost terms like fares, number of line changes, maybe even differentiating between time spent in vehicles and time spent waiting on a platform, or if the agent could sit or had to stand in the vehicle. In the end, some transit specific *analyses* may be implemented for better understand of the simulation outcome.

7.2 Data

In chapter 2, an overview was given of data available or used in real-world (see Table 2.1). This section will look in more details at the data required by the extended traffic flow simulation. For data structures already existing within MATSim it will be shown how they can be adapted to be suitable for the needs of a transit simulation. For data structures yet missing, the influences for the design will the discussed and the final design presented.

7.2.1 Network

The network describes the physical infrastructure available. Previously, it only represented the road network where cars could move around. Now, the network description must be extended to include multimodal aspects. To differentiate only between roads and rails is not sufficient. Not only have cars and trains to be distinguished, but e.g. also cars and buses, that can share a common road or not. In addition, links may exist that are both roads and rails, as it is the case for most streetcars. Given the existing structure of networks in MATSim, the primary step towards a multimodal network is the inclusion of an additional attribute for each link, listing the transportation modes that are permitted on each link.

If the list of allowed transportation modes varies between different lanes of the same street (e.g. a bus-lane), this must be modeled as multiple links in the network. Fig. 7.1 shows an example how a real-world situation could be modeled in the network.

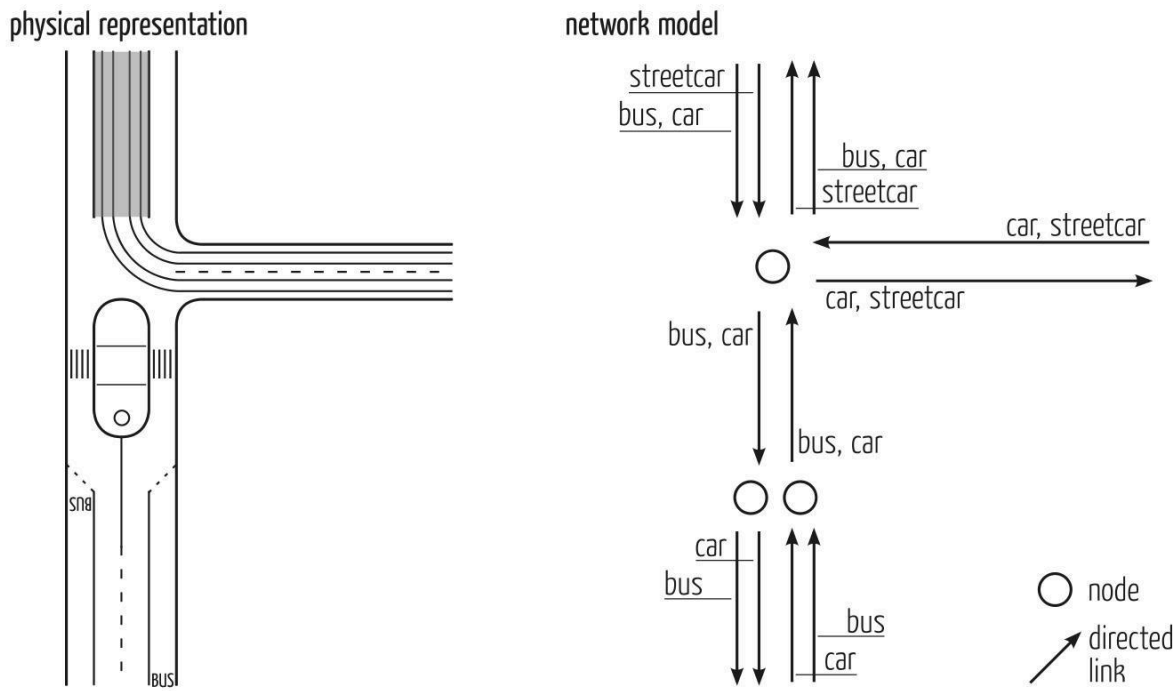


Figure 7.1: Example, how a physical intersection could be encoded in the network model. Each link is labeled with its allowed transportation modes.

Each link has attributes, describing physical aspects (like its length or the number of lanes) as well as traffic related ones (like the flow-capacity or free speed). Although these attributes were once added to describe car traffic, they are also suitable

for describing transit traffic in the queue model. Flow-capacity and physical space is measured in private car equivalents. Buses, streetcars and trains can also have a corresponding car equivalent assigned, causing them to allocate more space on a link than a regular vehicle.

7.2.2 Population and Routes

MATSim distinguishes already for a long time between legs driven with a car and legs not driven with a car, by assigning each leg a transportation mode. In the past, the traffic flow simulation just ignored all legs not carried out with a car. Each leg contains a route, describing how the agent should be moved around by the traffic flow simulation. To fully support the transit simulation, only this route information must be adapted, so the traffic flow simulation has all the data to move the agent around using transit offerings.

For car legs, the route currently describes the list of nodes an agent has to cross to reach its destination. At each node, the agent checks all outgoing links to find out which one leads to the next node. So, a route for car legs specifies a way through the road network, and is thus also called a *network route*.

For transit legs, such a route description is not practical, as transit users do not necessarily need to know what turns the vehicle they are in has to take to reach the next stop. For a passenger it is enough to known at which stop to board a transit vehicle of a specific transit line and at which stop to alight it. So, the data triple *entry stop*, *transit line*, *exit stop* is sufficient to describe the route of a transit user. Note the distinction between transit drivers and transit users: A transit driver needs a network route, but not the transit users.

7.2.3 Transit Schedule

The transit schedule is the central data structure regarding the simulation of transit. It contains information about one or more transit lines, their detailed routes, the stops approached on each route, and the departure time for each course along a route. The data can be split into two parts. The first part describes infrastructure, namely stop facilities, while the second part describes the services offered.

The **stop facilities** describe the locations where passengers can board and alight transit vehicles. Besides a unique identifier, stop facilities also have a coordinate used for estimating walk distances for agents or for finding the nearest stops at some place. An additional reference to a link from the network is used to specify on which link the stop can be reached by transit vehicles.

A **transit line** corresponds to a bus line, subway line or other "line" of a transit system. In the real world, a transit line often has a number or color assosciated (e.g. the "red" subway, bus 32, etc.). In the model, a transit line has an identifier and can have multiple transit routes.

A **transit route** describes one specific route of a transit line, including its time profile. The time profile describes how long it takes (or better: it is planned to take) to travel from one stop to the next. This information is used to project the departure time of one course onto arrival and departure times of the same course on later stops. Most transit lines have at least two transit routes, one in each direction. As the travel time during peak and off-peak hours may differ—either because of higher demand and thus higher stop times, or because of more dense traffic on roads, slowing buses and other vehicles with non-dedicated tracks down—the number of transit routes may increase further if the different travel times are anticipated in the schedule. In addition, some lines may not operate on the full extent of the line in off-peak hours, leading to additional transit routes for a transit line. The route is described twofold: once as a list of transit route stops, and once as network route. The latter is used by the driver of the transit vehicle, while the former is required such that passengers can do route planning.

Departures are stored for each transit route. A departure specifies at which time a vehicle starts the associated transit route. The vehicle can be specified as well, making it possible to use vehicles with different capacities according to the time of day (e.g. peak/non-peak hours).

Each transit routes contains an ordered list of **transit route stops**. A route stop references to a stop facility for its exact location, and contains arrival and departure time offsets. These offsets are relative to a departure time and build the time profile of a transit route.

The data is structured hierarchically, with many 1:n relationships. Example: One transit schedule can consist of multiple transit lines, one transit line contains several transit routes, and each route usually has several departures. Fig. 7.2 shows the structure of the transit schedule as entity-relationship model [21], drawn in UML syntax [93].

MATSim provides writers (and matching readers) to store the data to a file on disk. The file format is based on a custom XML grammar [121]. This approach is widely used throughout MATSim and has proven to be useful and powerful. Appendix B includes the full document type description (DTD) of the specified file format as well as an example of a transit schedule file.

The described structure relates closely how other tools store schedules. PTV VISUM [91] differentiates between routes (as a list of route stops) and time profiles

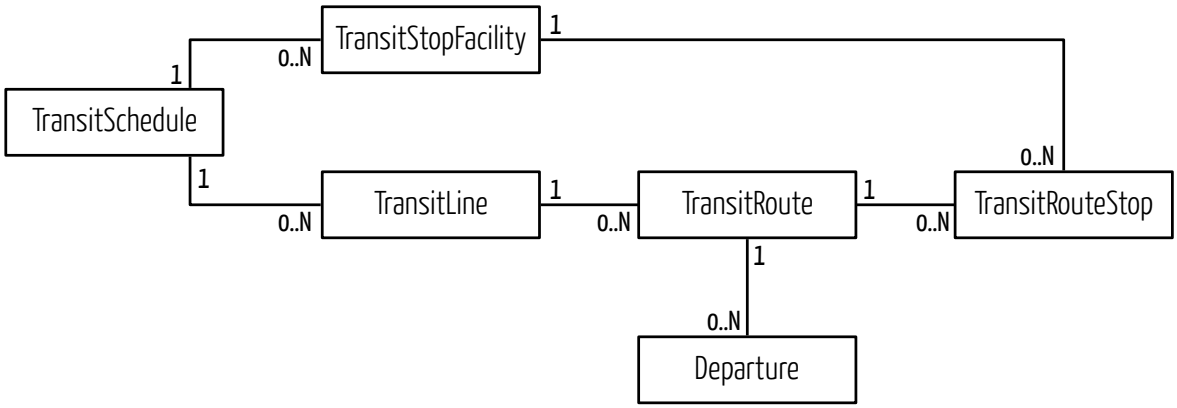


Figure 7.2: Entity-Relationship (ER) model of the transit schedule data structure, in UML syntax

(which reference the route stops), whereas the aforementioned data structure stores the time profiles directly in the route stops. On the one hand, separating route and time profile allows one to have multiple time profiles for the same route, giving a route a strong meaning. On the other hand, the additional hierarchy makes the data structure much more verbose (especially when written to file) and more complex due to the additional references from the time profile to the route stops. VISUM stores the data in flat ASCII tables, similar to a relational database schema.

The Association of German Transport Operators (Verband Deutscher Verkehrsunternehmen, VDV) publishes recommendations called *ÖPNV data model* [114, 115] that describes data structures and file formats for the exchange of transit related data. As the structures are mainly designed for data exchange, the number of tools that use the file formats natively is limited. Also, the recommendation dealt only with database structures in a first version for data exchange, before defining ASCII data formats in a later version. Still, it influenced the design of data structures in other tools such that a high level of compatibility could be reached, and "compatibility with the ÖPNV Data Model" is often claimed. The model, often also just called *VDV data model*, defines lines and route sequences (corresponding to list of ordered stop facilities), similar to MATSim or VISUM. The schedule is given as journeys with a departure time and assigned to routes and lines.

The Berlin Transit Company BVG used BERTA, as system developed by IVU [53], to manage its data and plan their services. BERTA allows to export data into hierarchical XML files, that resemble closely to the format specified by the aforementioned ÖPNV data model. A more detailed description of the format can be found in [111], where a transit router based on that data was developed.

Another format for transit schedules is HAFAS [49], which is widely used in many European countries. Despite its wide-spread usage, information about the "HAFAS Rohdatenformat" (HAFAS raw data format) is scarse. One of the few, good descriptions of the format can be found in the VISUM manual [91], where the import process of HAFAS data is explained. The data is given in multiple ASCII text files without fixed naming convention. The files contain information about operators, stops, vehicles, schedules and other, additional information. In contrast to many other data structures, schedule information is not hierarchically organized. Instead, single courses are listed in tables together with the stations they stop, time information and line information used for recognition of the course. This format may be explained by the fact that HAFAS data is mostly used for querying the schedules, e.g. in route-finding applications. For operational planning, the data will be most likely available in other tools that in the end output the HAFAS data format, or the HAFAS data may need to be analyzed and aggregated first before using it for operation planning purposes.

7.2.4 Additional Transit Data

In practice, a lot of additional data is used to manage and analyze transit operations. Such data compasses drivers' and vehicles' schedules, transit fares, passenger counts per line-segment, alternative schedules in the case of emergencies or blockages, and other data.

In this thesis, such data is not considered in more detail. The shown concepts and provided framework although are designed in such a way that the inclusion of additional data like passenger counts, fare structures and drivers' or vehicles' schedules should be doable without problems.

7.3 Traffic Flow Simulation

7.3.1 Overview

Up until now, the traffic flow simulation had to deal only with agents that either drove a car themselves (see Sec. 4.4), or that were teleported to their destination (see Sec. 6.1.2). Now, a third kind of agent movements needs to be added, namely to ride on a vehicle without steering it. While an agent riding along with a car does not need to know the exact route (as opposed to the driver), the travel time still depends on the current state of the network (unlike agents being teleported).

Sec. 4.4.3 detailed the messages passed between elements of the traffic flow

simulation (QueueSimulation, QueueNode) and the driving agent. The flow of messages show in Fig. 4.6 are only appropriate for agents driving a car on their own. Drivers of transit vehicles must take care of stops and interact with passengers, and passengers must decide which vehicle they want to board and where they want to alight. The new, transit specific interactions are as follows:

- **passenger – simulation:** A passenger, moved around by the simulation, must be able to reach a stop location and wait there for the arrival of a vehicle of the desired line. Also, a passenger must be able to start a new leg when exiting a transit vehicle, e.g. walking away to its next activity location, to a parked car, or to another nearby stop location for switching lines. These all are movements which are controlled by the traffic flow simulation.
- **transit driver – simulation:** The simulation has to recognize transit vehicles and give the drivers the possibility to handle transit stops along their route. As the drivers may stop their vehicles for some amount of time to let passengers enter and exit, the simulation also needs to take care whether the transit vehicle blocks the street, accumulating other vehicles behind it, or not.
- **transit driver – passenger:** Once a transit vehicle halted at a stop, passengers can alight or board the vehicle. This interaction is considered as an interaction with the passenger and the transit driver, and not the transit vehicle, as the driver has the "intelligence" and steers the car, thus defining the line the vehicle serves. Also, the driver is usually responsible for the safety of the passengers, maybe denying additional boarding when a vehicle has already reached its passenger capacity.

7.3.2 Initialization

Transit drivers are initialized based on the transit schedule, along their vehicles. For every departure, a driver and a vehicle is created. The driver receives one plan with exactly one leg, leading from the start of the route to its end, departing at the specified time. In addition, the driver is also informed about the stops along its line. At the end of the initialization, the transit driver is added to the QueueSimulation with a call to `scheduleActivityEnds()` (see Sec. 4.4.3). If drivers' schedules were available, drivers could be initialized with more than one leg, representing the different courses and routes they have to serve.

Passengers, i.e. the "normal" agents, can be initialized the same way as before. They are initialized with a day plan and schedule their activities' ends with the QueueSimulation, as shown in Fig. 4.5. Changes will only be necessary once they

start a non-car leg.

The simulation initializes a data structure to keep track of agents waiting at stops. For each stop, a list is maintained where agents waiting for a specific transit line are added to. Only one list is used per stop, even if the stop is served by more than one transit line.

7.3.3 Handling Stops

As it was outlined in Sec. 7.3.1, the traffic flow simulation is responsible for giving the transit drivers the possibility to handle stops. As was seen in Sec. 4.4, the QueueSimulation uses the so-called queue model to represent links. While this offers a comparably high computational performance over other models, it limits the amount of details that can be simulated on a link. Essentially, interactions can only take place at the very start of the link, when a vehicle is added to the queue, or at the very end of the link, when a vehicle is about to leave the queue. This implicates that transit drivers can only handle stops at one of those two occasions.

Agents departing from an activity—or arriving at one—are added to or removed from a link at its end in the current implementation. For reasons of consistency, the simulation lets transit drivers handle stops also at the end of a link. Thus, the location where agents board and alight transit vehicles is, in the model, the same location as where their legs with a different transportation mode would start or end.

Fig. 7.3 shows a conceptual sequence diagram how the traffic flow simulation enables transit drivers to handle stops. When moving vehicles from the queue to the special buffer used to identify those vehicles that are allowed to pass the next node (see Sec. 4.4.3), the simulation first checks if the vehicle's driver is a transit driver. If this yields true, the driver is queried for its next planned stop. If that stop is located on the link the vehicle is currently located on, then the stop can be handled. For this, the method `handleStop()` of the transit driver is called. The driver has then the possibility to let people out and in of the vehicle, and returns the time how long the vehicle remains blocked until it can continue on its course (`delay`). This delay can depend on the number of passengers alighting and boarding, can include some fixed amounts of time (e.g. for opening and closing doors) and other aspects. If a driver is too early at a stop (e.g. due to fewer than anticipated passengers and thus, shorter stop times), it can also return a delay that holds the vehicle longer at the stop then essentially needed for the change of passengers, in order not to depart too early at the stop.

If a transit driver returns 0 (zero) as delay, the vehicle is moved to the link's

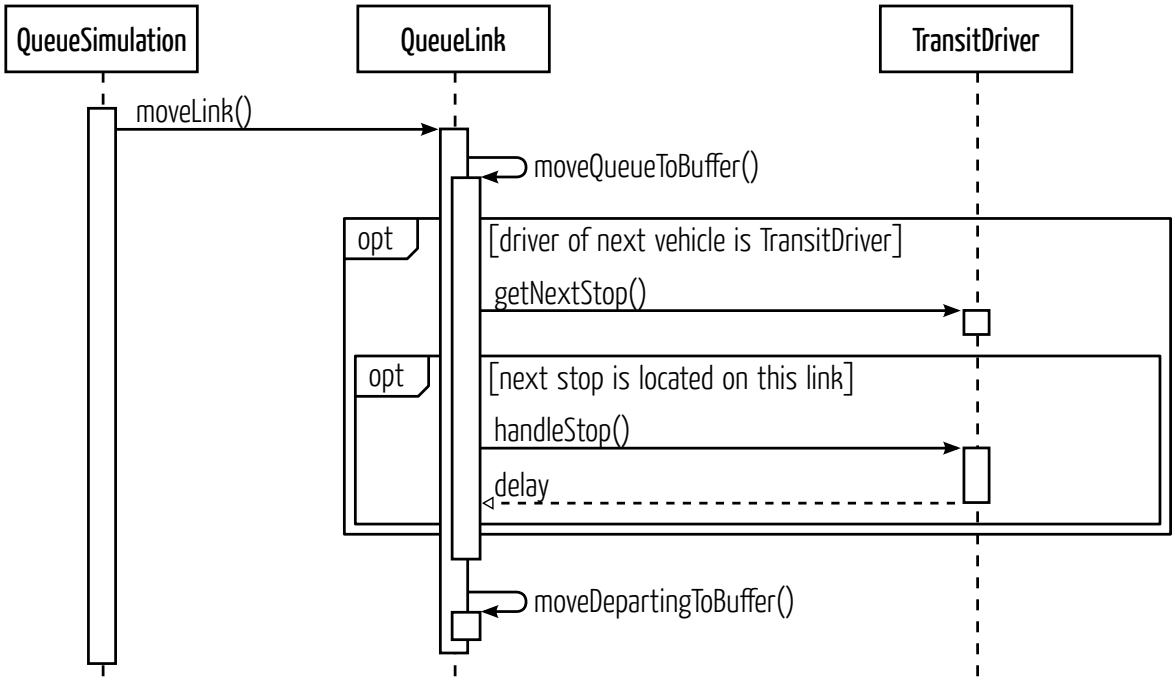


Figure 7.3: Sequence diagram of the relevant parts for handling transit stops

buffer, so it can be moved over the node to another link later. If the delay is larger than zero, it means that the vehicle remains at the stop for the specified time. The simulation has then two possibilities to proceed.

If the vehicle is blocking the link (effectively an attribute of the stop facility, as it depends on the physical geometry of the stop), the vehicle is left in the link's queue, but the vehicle's earliest link exit time is adapted to include the specified delay. When the queue is checked again in the next time step, the transit vehicle will still be front most in the queue, but not handled as its link exit time is still in the future, effectively blocking also the cars behind it.

If the vehicle is not blocking the link (e.g. when a bus stop is located in a special pocket along the road), the transit vehicle is removed from the queue and inserted into a special transit vehicle list. In addition, the vehicle's earliest link exit time is also adapted to include the specified delay. This special transit vehicle list is checked in each time step as the very first action for transit vehicles whose link exit time has been reached. If such vehicles are found, they are removed from the list and added to the front of the link's regular queue. This allows them to be treated normally as other vehicles, moving the vehicle eventually to the buffer if all other conditions (like a sufficient, remaining flow capacity in this time step) are fulfilled.

It must be noted that in both cases (blocking other traffic or not), the vehicles either remain in or are added to the link's queue at some point of time. This means, that the vehicles will be treated once more the same way as before. This, maybe surprising, fact is quite essential. Imagine a number of people waiting at a bus stop. When the bus arrives in the model, the driver counts the people who want to board the bus and thus calculates the delay. Now, while the agents are boarding, additional people could reach the stop. In reality, they would just enqueue and board the bus as well. In the model, the driver is not informed about agents arriving at a stop, possibly driving away while these people would wait at the stop. By handling the transit vehicle a second time exactly the same way as before, the driver can detect the newly arrived passengers and return an additional delay to allow them to enter the vehicle as well. This also makes the semantic very clear, when a transit vehicle can depart from a stop: Only if the driver returns a delay of 0 (zero), meaning no more entries or exits take place, the vehicle is moved to the link's buffer, and can thus be moved over the next node. Naturally, the driver has to take more care calculating the stop delays when it can be called multiple times to handle one stop. Parts of the delay not depending on the number of passengers should likely only be included in the delay the first time the stop is handled and not every time.

The transit driver is also responsible to generate events whenever it handles a stop. Every time the driver arrives at a stop facility, it generates a *VehicleArrivesAtFacilityEvent* that contains the current time, the vehicle's id and the id of the stop facility. After passengers have left or entered the transit vehicle and the bus is ready to depart (returning 0 from a call to `handleStop()`), the driver must also generate a *VehicleDepartsAtFacilityEvent*, containing the same amount of data as the corresponding arrival event.

7.3.4 Handling Transit Rides

In the model, each transit leg starts and ends at a transit stop facility. This is comparable to the assumption that every car leg starts and ends at the parking location of the car. The time to access these locations (either the stop facility or the parking facility) is ignored, unless it is explicitly modeled with a walk leg.

Whenever an agent is handed over to the traffic flow simulation with a call to `agentDeparts()` (see Sec. 4.4.3), the simulation now first checks with which mode the leg needs to be handled. If the leg has the mode set to "car", it is added to its departure link as described in the aforementioned section. If the leg has one of the supported transit modes, the agent is added to the stop's list of waiting passengers.

In all other cases, the agent is added to the teleportation list of the simulation (see Sec. 6.1.2). Fig. 7.4a visualizes this process as an UML sequence diagram.

When a transit vehicle arrives at a stop and the driver's method `handleStop()` is called (see previous subsection), the driver checks the list of the waiting passengers at that stop, querying every agent if it wants to board the vehicle (`transitLineAvailable()`, see Fig. 7.4b). If this is the case, the driver ensure that the passenger is removed from the stop's passenger list and added to its vehicle's passenger list. The driver queries as many agents as the list contains, or until the passenger capacity of its vehicle is reached.

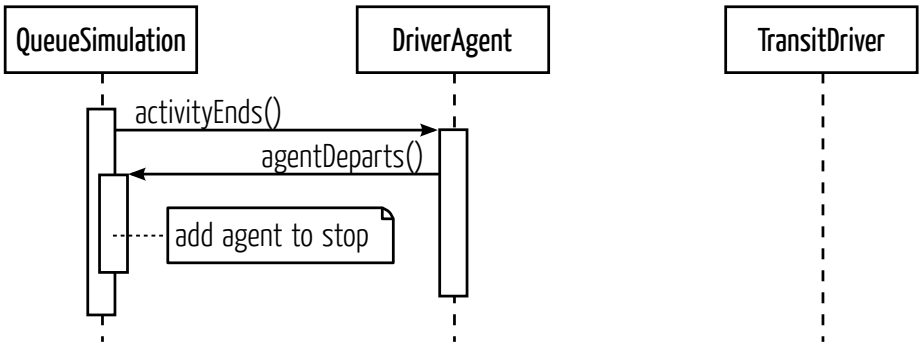
Also while at a stop, the transit driver asks each of the vehicle's passenger if s/he wants to get out (`arriveAtStop()`, see Fig. 7.4c). Agents about to leave the vehicle have their leg ended. They are removed from the vehicle as passenger, informed that they are now located at the stop's link (`teleportToLink()`, this is required as agent implementations may try to validate the locations they are at, but are naturally not aware of where the transit vehicle has driven along) and have finally officially their legs ended (`legEnds()`), allowing them to start the next activity at the location.

Note that in the implementation, passengers are first to leave the vehicle before new agents can board. This order is required to allow the maximum passenger capacity to be reached. Also, during their ride, agents do not know their current location. If one is interested in the exact location or the link the agents are currently on (e.g. for visualization), one needs to keep track which vehicle an agent entered and query the location of the vehicle instead. This can be done using events generated by the simulation.

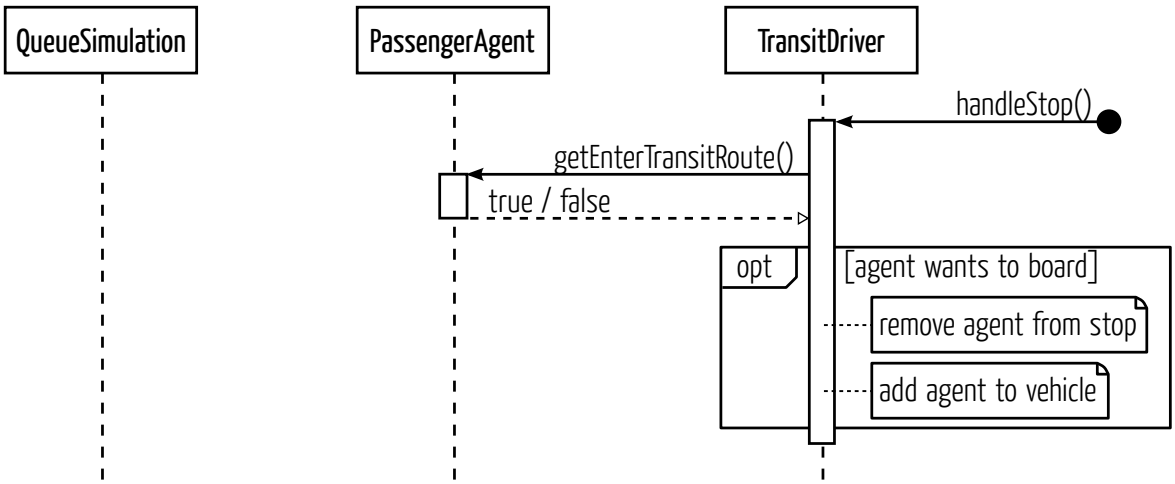
In addition to the steps outlined above, the simulation creates special events for each agent boarding or leaving a transit vehicle. The events contain the time, the vehicle's id and the agent's id. This information may be later used for analysis purposes.

7.4 Transit Router

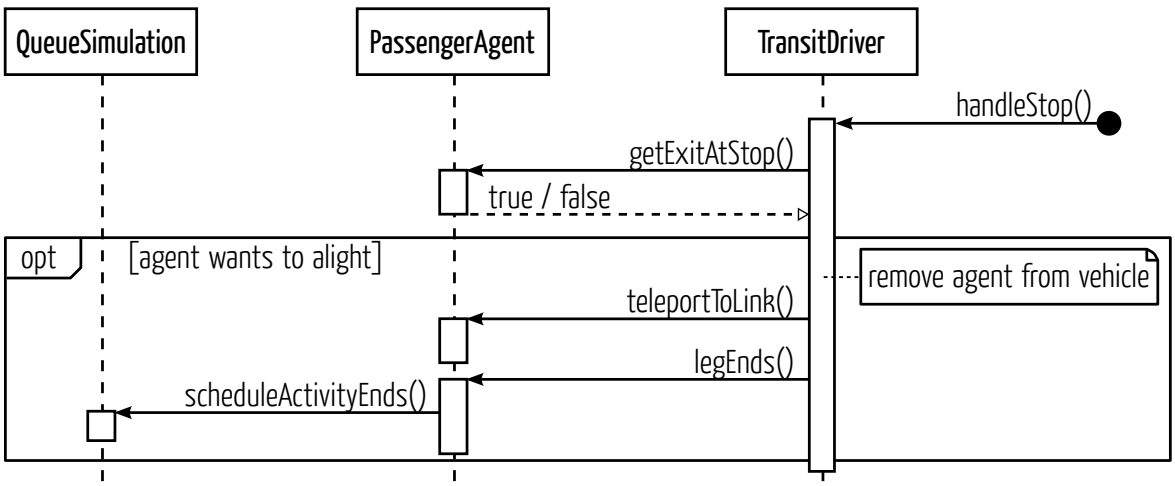
Simulating transit traffic is one thing. But as mentioned earlier, each transportation simulation needs a demand it can simulate. The demand for an agent-based transit simulation consists of agents that use the transit offerings to get from one location to another. The transit router is responsible for calculating and assigning agents useful routes using the given transit offerings, and is used during the initial demand preparation and the replanning phase of MATSim (see Sec. 4.7).



(a) Start of a transit leg, waiting at a stop facility



(b) Boarding a transit vehicle



(c) Alighting a transit vehicle, ending a transit leg

Figure 7.4: Sequence diagrams of the relevant parts for handling transit rides

7.4.1 Overview

Finding the best transit connection is a complex topic. Besides the obvious problem that every traveler may have different perceptions of waiting time, transfers, availability of free seats and other qualitative criteria, the task itself of finding the best connection given a fixed set of parameters is challenging. Real-world applications like the route finders available from many transit operators on the Internet even output multiple variants for one query, letting the customer decide which connection suites its needs best.

Within MATSim, several student projects have had a look at transit routers. Two notable works included also the implementation of such routers: Maier [65] implemented a transit router in C++ working with collected data for the region of Zurich, whereas Titze [111] used data from Berlin to implement a transit router in Java. Both projects were used to gain insight into the problems of transit routers, such as the representation of the network and the resulting complexity, but were rather "stand-alone" and not tightly integrated into the MATSim framework.

As this thesis concentrates more on the aspects of the traffic flow simulation, only a basic router was implemented that could be used during the replanning phase to run multiple iterations of the simulation. The implemented transit router uses a modified variant of Dijkstra's shortest path algorithm [32] to find the best connection on a graph representing the transit offerings. This graph, in the following also called *transit network*, is created from the data available in the transit schedule and represents a logical topology, in contrast to a road network that usually represents a physical topology.

As the router will be used as one mental module among others (see Sec. 4.7 about other mental replanning modules), it is possible to apply some simplifications usually not found in other transit routers. These simplifications are all based around the fixed departure time: In the simulation, the agents have a fixed point in time at which they end an activity. Starting from that point in time, the agents are interested in the next available connection that brings them to their target location at the earliest time possible. This is different from the way many people plan their trips in reality. Often people will decide to re-schedule their departure by some amount of time if they can take a faster (in the sense of total travel time) connection instead. In the simulation model, an agent may decide to take a big detour if it arrives a few minutes earlier than a direct connection for which the agent would have to wait a longer time.

While this may not seem realistic or useful at a first glance, it matches well with the modular and iterative concept of MATSim, where the combination of simple modules generate a useful outcome. Using a module that modifies the departure

times of agents (TimeAllocationMutator, see Sec. 4.7) may likely result in an agent finding the most useful connection between two locations including an improved departure time.

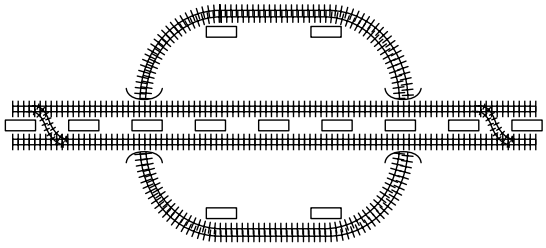
7.4.2 Transit Network Generation

The network the router operates on is generated from the data given in a transit schedule (see Sec. 7.2.3). Each transit route stop builds a node of the network, and for each transit route corresponding links are added, connecting the stops of the route. After this first step, the network consists of several linear strings of connected nodes, representing the transit routes from the schedule, but with no connection in-between them yet.

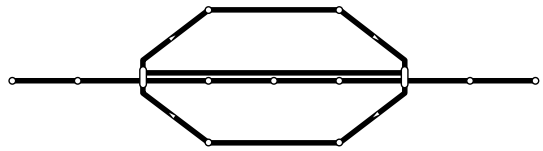
In a second step, each node is connected with additional links to other nodes within a configurable distance. These links, that can be seen as transfer or walking links, represent the interchange facilities; not only within one physical facility, but also between nearby stop locations of different lines. Fig. 7.5 shows an example of a simple physical network (Fig. 7.5a) and a matching transit line map (Fig. 7.5b), as well as the router network generated from it after each of the two involved steps (Fig. 7.5c and 7.5d).

The performance of the least-cost path calculation depends on the number of nodes and links in the network. In the second step of the router network generation process, some heuristics are thus applied to reduce the number of additional links added to the network. Nodes being the start locations of a transit route are seen as departure locations only, thus no transfer links being added to them starting at those nodes. Nodes representing the last stop of a transit route are arrival locations only, and will have no transfer links ending at those nodes. In addition, no transfer links are added between two nodes that belong to the same transit line and the same stop facility. This comes from the insight that U-turns on transit lines may never be part of a least-cost path and such transfer links are thus never needed. In addition, if two nodes represent the same stop facility they will have the same coordinate, resulting in that either both (or all) transit routes passing through that stop facility are found or none when searching for departure and arrival stop locations. This ensures that agents may not be forced to start in a certain direction on a transit line, when the opposite direction would be better.

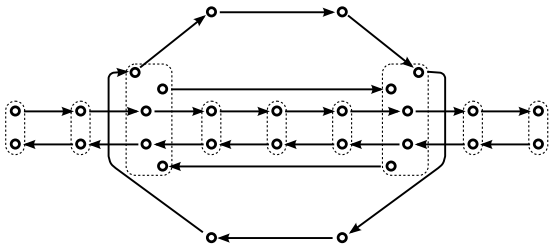
Re-using nodes for multiple transit routes, or using one link between two stops for multiple transit routes—in case multiple lines or routes share the same route section—could reduce the number of links in the generated network further. This option was not further researched, as it would make the cost calculation for



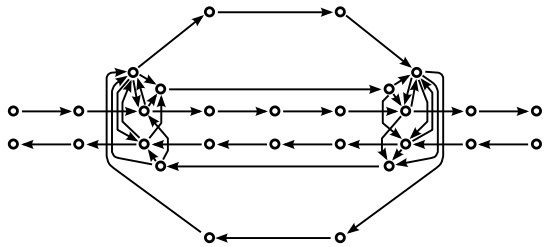
(a) Physical infrastructure. Squares represent platforms of stop facilities.



(b) Line map with 3 different lines: A circular line, an express line, and a regular line.



(c) Transit router network without transfer links. The dotted areas group the nodes belonging to the same stop facility.



(d) Complete transit router network

Figure 7.5: Generation of the Transit Router Network

links and nodes much more complex. With the current implementation, costs for changing lines can be clearly assigned to transfer links. The transfer links could even represent real transfer times, if such data is available. Without dedicated links for transfers and each transit route, the number of links would be reduced, but the complexity of calculating link travel times and link travel costs would be hugely increased. It is unclear if such a change would really bring a measurable performance improvement.

7.4.3 Least-Cost Path Calculation

Given the aforementioned transit network, traditional time-dependent least-cost path calculators could be used to find point-to-point connections. The link costs and travel times vary with the planned departure time, as they contain the actual travel time as well as the waiting time until the next departure on that link. A problem with traditional least-cost path calculators is, that they only return paths from exactly one location in the network to one other. An extension are shortest

path trees, generated when calculating the shortest path from one location to all other reachable locations in a network. But in reality, many transit trips could start or end at more than one stop. Neither is the nearest stop always the best one. It could well be that the nearest transit stop belongs to a bus with a headway of 30 minutes, but a few meters farther away is a stop of another bus line with a denser schedule, with a more direct route, or a stop of a faster express line.

The start or end location of a trip is usually not directly on the transit router network, but on some coordinates between links. Each transit trip starts and ends thus with walking stages to and from stop locations. Given a start or end coordinate, all stop locations located within a configurable distance are considered as start or end points.

A first, simple approach would be to route each combination of considered start and end points, always adding time and costs for accessing the stops by foot, and to finally decide for the combination that results in the lowest costs. The downside of this simple solution is the bad performance, as for each route request multiple route calculations are performed.

A second approach could be to add two additional nodes to the network, representing the start and end coordinate, and connecting those nodes with links to all considered stop locations. Then, many existing least-cost path algorithm could be used to search for the best route. While this approach uses only a fraction of the computational runtime than the previous approach, this solution has the drawback that it modifies the transit router network. This disables the possibility to re-use one transit router network in multiple threads for the parallel calculation of routes. In addition, some optimized algorithms like A* [45] require time-consuming pre-processing, which may have to be re-run every time additional nodes and links are added to or removed from the network. This would possibly result in a worse performance then a slower algorithm that does not need pre-processing.

The finally realized approach adapts current least-cost path calculators to work with multiple start and end locations. In the case of the implementation of Dijkstra's least-cost path algorithm [32], this is quite straight forward, as only two modifications to the original algorithm are needed:

- Instead of initializing set A—the set containing nodes for which the least-cost path is known—with only one node with cost 0 as it is proposed by Dijkstra, the set is initialized with all considered start locations, each with its own cost and time attribute, reflecting cost and time to reach those nodes from the originating coordinate. The current implementation of the transit router uses estimates for cost and time based on the beeline distance between the originating coordinate and the stop location.

- The algorithm ends, when all considered end nodes are contained in set A, or if the smallest cost of the nodes in set B—the nodes not yet in set A but connected with links to nodes in set A—is higher than the least cost of a path ending in an end node already in set A including the egress cost from that node to the destination. Then, to each (reached) end node costs and times to get from those locations to the destination coordinate are added. Based on those costs, the least-cost end node is determined and the least-cost path extracted.

This solution offers a very comparable, computational runtime as the one presented in the second approach (assuming the same least-cost path algorithm is used, in this case Dijkstra's algorithm), but works without any modification of the network.

This approach was only implemented for Dijkstra's algorithm, although it should not be hard to adapt it also to other algorithms. In the case of the—in MATSim already implemented—A* with landmarks algorithm [45, 63], the modifications would be very similar to the ones listed above for Dijkstra's algorithm, in addition to the adaptation of the heuristic to multiple end nodes instead of only one. Additional optimizations could be applied, like the reduction of heap operations described in [81].

The transit router returns a list of stages, starting and ending with a stage using the transportation mode *walk* to get to/from the transit stop. The stages between are either of mode *transit* or of mode *walk*. Transit stages describe at which stop to board a specific transit line and route, and at which stop to egress. If an agent must change lines to arrive at its target location, each line change ends one transit stage and starts another. If the connecting line departs at the same stop facility as the previous line arrived, no walk stage is contained between the two transit stages.

7.4.4 Integration

In MATSim, plans consist of an alternating list of activities and legs. A leg has a transportation mode assigned and includes one route object, specifying the detailed route the agent has to take to get from one activity to the next one. This has worked well in the past for car trips, but cannot easily satisfy the requirements of having stages in a trip. Several options have been looked at to reflect the more complex structure of transit trips within MATSim:

1. Extending the current legs to include a list of stages, effectively creating an additional hierarchy in the data structure.

2. Loosening the rule that activities and legs must occur alternatively, allowing multiple legs in succession. A single leg would represent a stage in this case, while a group of successive legs would build a trip.
3. Adding additional *transit interaction* activities between the stages, and storing each stage in the form of an existing leg.

The first option implies a large number of changes to the current data structures and file formats. Not only the simulation, but every existing mental module would need to be adapted to the new structure, as well as many analysis tools. The size of this task excluded it from realization.

The second option also requires quite some changes to the existing code base, as many modules currently assume legs and activities to occur strictly alternating. In contrast to the first variant, no changes to the file format is required. Although this variant is not (yet) realized, new code was written in a way to no longer assume legs and activities to be alternating, as this seems to be the most flexible solution for the future.

The third option, the one being realized, requires the least amount of changes to existing code. Each stage is translated into a leg with a route of corresponding mode. Between these legs, *transit interaction* activities with a duration of zero seconds are added. This list of legs with routes and activities is inserted in place of the one leg without a route into the plan. Fig. 7.6 visualizes this process of inserting the stages returned by the transit router as legs into an existing plan.

It must be noted that the transit router now modifies the list of activities within a plan. This has consequences for additional route searches and other replanning modules. If an already routed plan is routed again—e.g. because the departure times changed—the legs representing stages and the transit interaction activities must be ignored. Otherwise, agents would be forced to change lines at the exactly same stops as before (at the locations of the transit interaction activities), making new routing requests kind of useless. Instead, plans must be reduced again to their initial form, where one leg represents the total trip from one activity location to another.

7.5 Mode Choice During Replanning

In previous experiments with multiple modes in MATSim (see Sec. 6.1), the plans themselves were marked with a type to ensure agents have always the possibility to change the transportation mode. While this process worked, it required essential changes to the replanning mechanisms (see Sec. 6.1.4). In addition, that solution

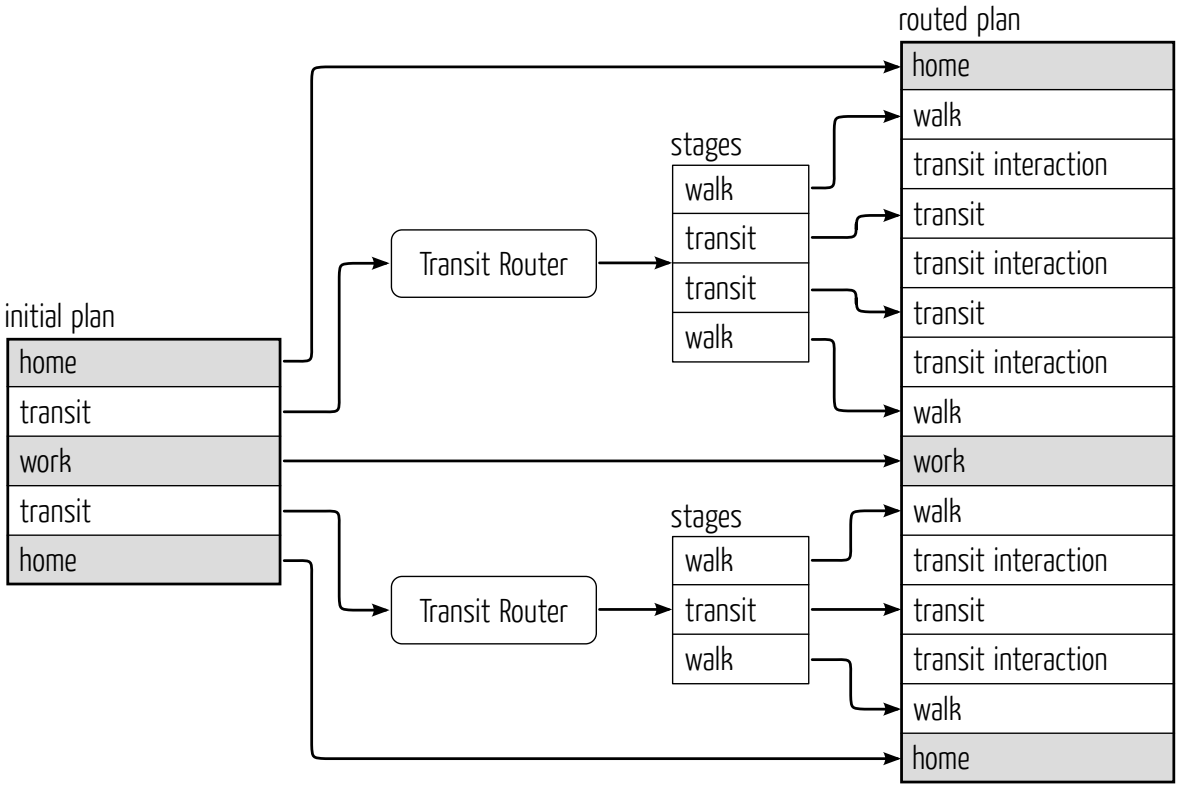


Figure 7.6: Modification of a plan by the transit router

would not scale well if more transport modes needed to be differentiated: The number of plans an agent can keep would need to be increased in order to allow agents to keep a reasonable number of possibly good plans besides one (likely) bad plan of each other allowed mode. Also, the demand had to be especially prepared to give each agent at least one plan with each possible mode of transportation.

These short-comings were solved by the implementation of a special mode choice replanning module, *ChangeLegMode*. For each plan handled by this module, a random transportation mode is chosen from a configurable list of modes (excluding the current transportation mode). This new mode is then set for all legs in the plan. This makes it possible that agents can try out new transportation modes at random, without having to keep a plan of each mode and without the need of special initialization.

The implemented module assigns the same mode to all legs. This may not be realistic, as people often mix different transportation modes during a day. Driving with the car to work implies in most cases that the people will also return with their car from work to home. But in-between, e.g. to go for lunch, they may

walk or choose to take the transit. The *ChangeLegMode* module could be extended in such a way to allow mixed mode plans. This would require to analyze a plan for possible sub-tours for which the transport mode could be changed (see [25]). Otherwise one could not ensure that non-replaceable transport modes, like using ones private car, are available at each location during the day. The latest version of the replanning module *Planomat* [67] already includes such an algorithm for sub-tour based mode choice, which could be extracted and used stand-alone as well.

7.6 Transit Controler

To successfully run multiple iterations of a transit simulation, a special Controler (see Sec. 4.2) is implemented, the *TransitControler*. It has the function to combine all the different parts correctly together. The TransitControler extends the normal Controler, but overwrites or adds certain functionality.

The most obvious change is the call of the TransitQueueSimulation instead of the regular QueueSimulation. In addition, the TransitControler ensures that replanning strategies work correctly together with the transit simulation, patching existing strategies if necessary. Such modifications could be integrated into the original code in case the transit simulation features would be integrated into the regular QueueSimulation. The actual patches the TransitControler does are:

- ensure that the router for the private car traffic uses only links that are open to cars
- modify the TimeAllocationMutator to not change the duration of transit interaction activities
- patch ChangeLegMode to first remove all transit interaction activities from a handled plan, reversing the effects of the transit router, as they are not needed when the leg modes are changed to non-transit modes

Especially the first item imposes some overhead for the route calculation, as the routing algorithm needs to verify for every link if it can actually be used.

As last, the TransitControler also makes sure that events are written to file in an XML format. By default—and for a long time—events are written to a simple text file in a tabular manner, with events' attributes being separated by tabs only. As the events get more complex, this tabular file structure is no longer suitable for all the different kinds of events. Thus, a newer file format based on XML is chosen to write out all types of events.

Table 7.1: Additional types of events generated by the transit simulation

Type of Event	Attributes
VehicleArrivesAtFacilityEvent	time, vehicle-id, stop-facility-id
VehicleDepartsAtFacilityEvent	time, vehicle-id, stop-facility-id
PersonEntersVehicleEvent	time, agent-id, vehicle-id
PersonLeavesVehicleEvent	time, agent-id, vehicle-id

7.7 Transit Analyses

Evaluation of simulation outcome depends heavily on the research focus of actual projects. The focus of this dissertation regarding analyses lays thus more on providing a useful infrastructure to create custom analyses. In the following, this infrastructure is presented and some examples are given to demonstrate the usage.

The base of the analysis infrastructure are events, as outlined in Sec. 4.8. The transit simulation introduces additional events (see Sec. 7.3.3 and 7.3.4), which are listed in Table 7.1. With these events, a very large number of analyses can be implemented.

The **VehicleTracker** keeps track which vehicle is currently at which stop facility. Every time a VehicleArrivesAtFacilityEvent occurs, an entry in a lookup table is stored with the vehicle id as the key and the facility id as the value. When a VehicleDepartsAtFacilityEvent takes place, the entry for that vehicle is removed from the lookup table. This allows to query the lookup table with a vehicle's id. If a facility id is returned, one knows that the vehicle is currently stopping at that location. If no entry is found it means that the vehicle is currently not at a stop and thus no information about its current location can be made.

The **TransitRouteAccessEgressAnalysis** listens to all PersonEnters- and PersonLeavesVehicleEvents. In combination with the VehicleTracker it calculates the number of passengers entering or leaving a transit vehicle at each stop. This information could then further be processed to calculate the occupancy of transit vehicles, seat availability, average number of stops passengers use a line and other passenger related metrics.

RouteTimeDiagram listens to VehicleArrivesAtFacility- and VehiclesDepartsAtFacilityEvents. In contrast to VehicleTracker, this class stores all the events to have a history of each vehicle. It offers methods to create route-time diagrams that show the position of all vehicles along one transit route over the lapse of time. An

example of such an analysis is given in Sec. 8.1.1.

Besides the aforementioned—and implemented—analyses, additional ones could be thought of. Given the information in the events, one could for example easily extract the average waiting time of passengers at stop facilities (which equals to the average of the time between an agent's ActivityStartEvent with activity type *transit interaction* and the next PersonEntersVehicleEvent concerning that agent), the number of passengers waiting at a stop, or the average number of transfers an agent has to make to get from one activity to the next one.

8 Applications

This chapter presents applications of the new transit simulation. In a first section, several smaller examples are given as proof of concepts. They help to demonstrate the correct functioning and to visualize certain features of the simulation. In a second section, a large-scale application with real-world data is presented. Data preparation is an important point for real-world applications that will be discussed as well. The third and last section serves as an outlook, what other applications could be realized using the infrastructure provided by the transit simulation. Several smaller examples are given and explained how they could be realized with the help of the features provided.

8.1 Illustrative Examples

Several small examples have been implemented to demonstrate single aspects of the transit simulation. While helpful to visually debug the simulation during the implementation, they also serve as simple proof of concepts.

8.1.1 AccessEgressDemo

This example shows how transit vehicles along a route are influenced by the number of passengers entering or leaving the vehicle.

The network for this example is very simple. It consists of only 15 links that are placed in a row, one after the other, building a road segment leading from left to right. At the end of each link, a bus stop is located. One bus line is defined with one route, serving each stop on its way from the far left to the far right side. All bus

vehicles have a capacity of 100 passengers. Buses depart every 5 minutes starting at 07:00. The bus departing at 07:45 is delayed by one minute, so it effectively departs at 07:46.

On the demand side, a new agent arrives at each bus stop (except at the last stop) every 60 seconds. The agents will board the next bus arriving at their stop, leaving it at the last stop location.

When the simulation is run, the buses stop at each stop facility and let people enter (or leave at the last stop). The duration how long the bus is at the stop depends on the number of passengers getting in and out:

$$stoptime = \begin{cases} m \cdot 4sec + n \cdot 2sec + 15sec & \text{if } m + n > 0 \\ 0sec & \text{else} \end{cases} \tag{8.1}$$

where m is the number of passengers entering the bus and n represents the number of passengers leaving the bus.

Fig. 8.1 shows a route-time diagram of the vehicles' positions. As can be nicely seen, all vehicles departing before 07:45 share a very similar progress along the route. Each bus collects at each stop 5 or 6 passengers, leading to a similar distribution of stop times and thus spacing between each bus. The bus starting one minute late has longer stop times at the stop facilities, as there will be at least one person more waiting for the bus at each stop. Due to the longer stop time, the delay will get even longer compared to the other buses on the route, resulting in even more passengers waiting at upcoming stops. As a contrast, the bus departing right after the delayed one, at 07:50, is faster due to fewer people waiting at the stops. This eventually leads to the situation where that bus catches up on the delayed one, as can be seen in Fig. 8.1. As the bus directly after the delayed one is faster, the next one following may again find more passengers waiting at each stop than the average would be, given each bus would have been on time. It takes several headways until the situation stabilizes.

This example shows nicely the effect that delayed departures can have on headways along one route, a phenomenon known as *bus bunching* or *pairing vehicles* [84, 90, 78]. These effects also happen in reality, mostly during rush hours, where headways are short and people do not exactly plan when to depart but just show up—more or less uniformly distributed—at stop locations.

In Fig. 8.1, a special effect can be observed between the last few stops of the bus departing at 07:50. Especially between stops 13 and 14, the travel time seems to increase massively. The same effect can already be observed between stops 11, 12 and 13, although at a lower scale. This seemingly increased travel time is an artifact of the simulation: As explained in previous sections, vehicles are processed

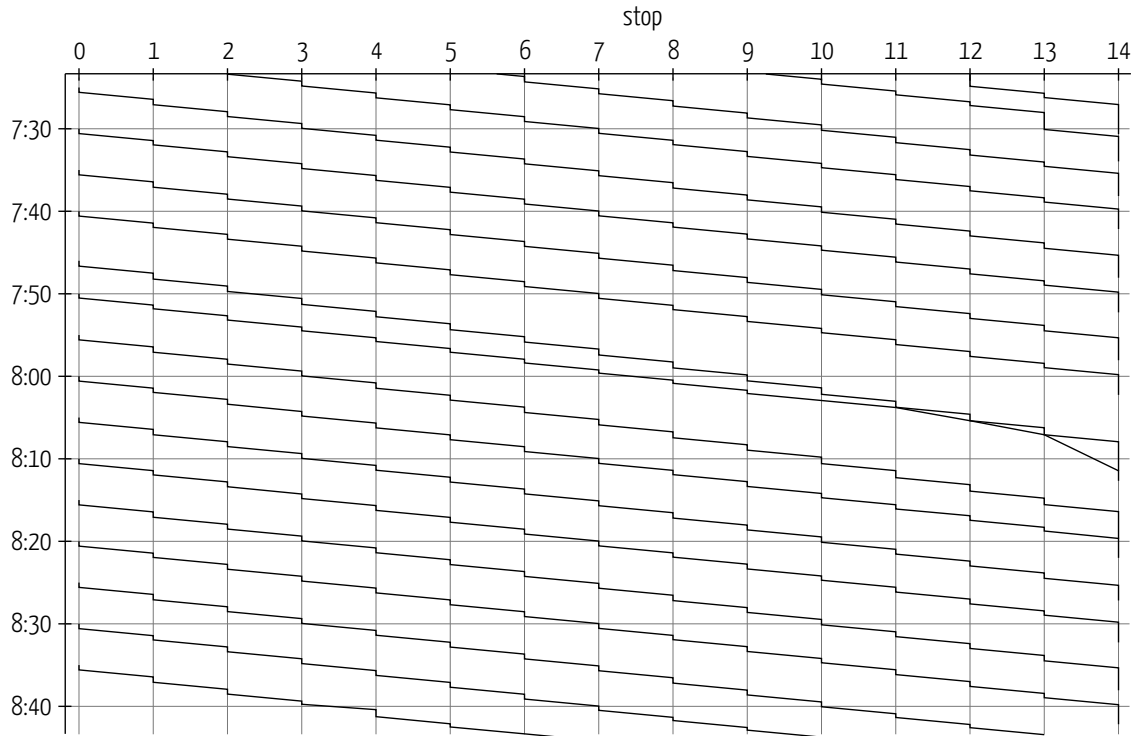


Figure 8.1: Route-Time diagram of the transit vehicles in the access-egress demo

in a strict first-in first-out order (see Sec. 4.4.2 and 7.3.3). This includes vehicles at a stop. The bus departing at 07:45 has to wait for a rather long time at stop 14 to let all the passengers exit. During that time, the bus departing at 07:50 is stuck behind the previous bus, and can only get to the stop once the earlier bus is empty and has left. As the analysis is based on events only (see Sec. 7.7), the actual being stuck cannot be observed. Only the extended time during departing at stop 13 and arriving at stop 14 can be detected, resulting in the longer travel time perceived.

8.1.2 BlockingStopDemo

This example shows how regular car traffic is influenced by buses stopping at stop locations that either block regular traffic or not.

The network consists of two parallel roads, each made up of 13 links. Along each road, 10 transit stop facilities are located, always at the end of a link. Fig. 8.2 shows the network with the stop locations.

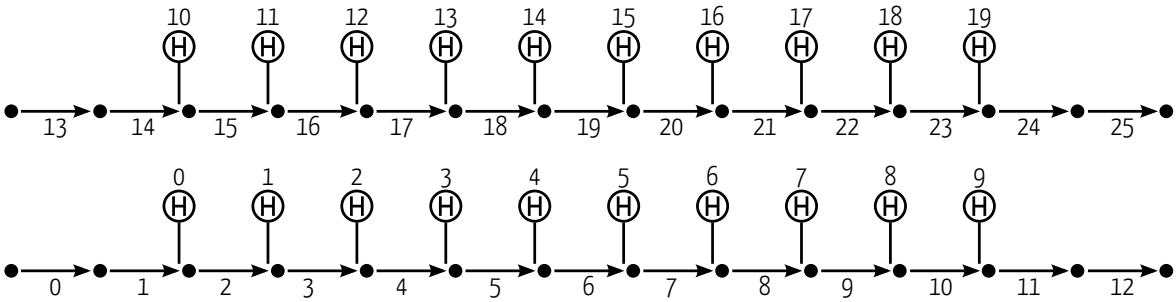


Figure 8.2: Network and stop locations for the blocking-stop example

Two bus lines are defined, one operating on the lower road (links 0 – 12) and one on the upper road (links 13 – 25). On each line, exactly one bus departs shortly after 7 a.m. At each stop, one passenger is waiting, requiring the buses to stop. If a bus halts at a stop on the upper road, it blocks the road such that other cars cannot pass the bus but must wait until the bus continues. If a bus halts on the lower road, regular traffic can pass the bus and is not blocked. In reality, this is the case if the stop has a bus bay or another, similar measure in the road geometry to allow regular traffic pass.

In addition to the buses, private cars are also on the roads. On each road, 30 cars travel from the left side to the right side. The first two cars (one on each road) start at exactly 7 a.m., after that, every 30 seconds starts another one.

Fig. 8.3 shows a screenshot from the visualizer when the simulation is run. It can be clearly seen, how on the upper road private cars are jammed behind the bus. On the lower road, cars can overtake the bus at stops, such that the bus is just one more vehicle in the flow of all the cars on the road.

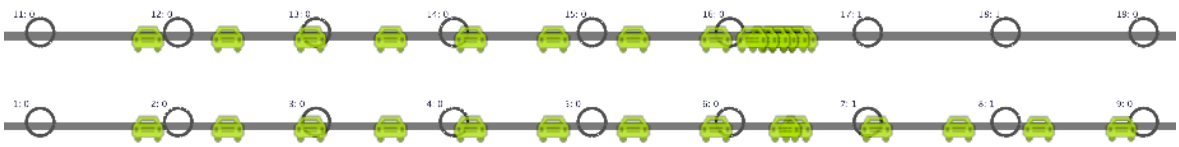


Figure 8.3: Screenshot from the visualization of the blocking-stop example. On the upper road, cars are jammed behind a bus. On the lower road, cars can overtake the bus at stops. Circles represent the location of bus stops.

Fig. 8.4 shows the measured travel time for each link in the simulation over the lapse of time. In the lower part, representing the lower road with the non-blocking stops, only minimal deviations from the free speed travel time can be observed. These small differences come from the additional time the transit vehicles spend on a link while serving the stops. On the upper road, much larger differences can be observed due to the fact that on that road the private cars must wait behind the bus at stops. This results in every car having a longer travel time on that link behind the bus, leading to a much higher average link travel time than on the lower road.

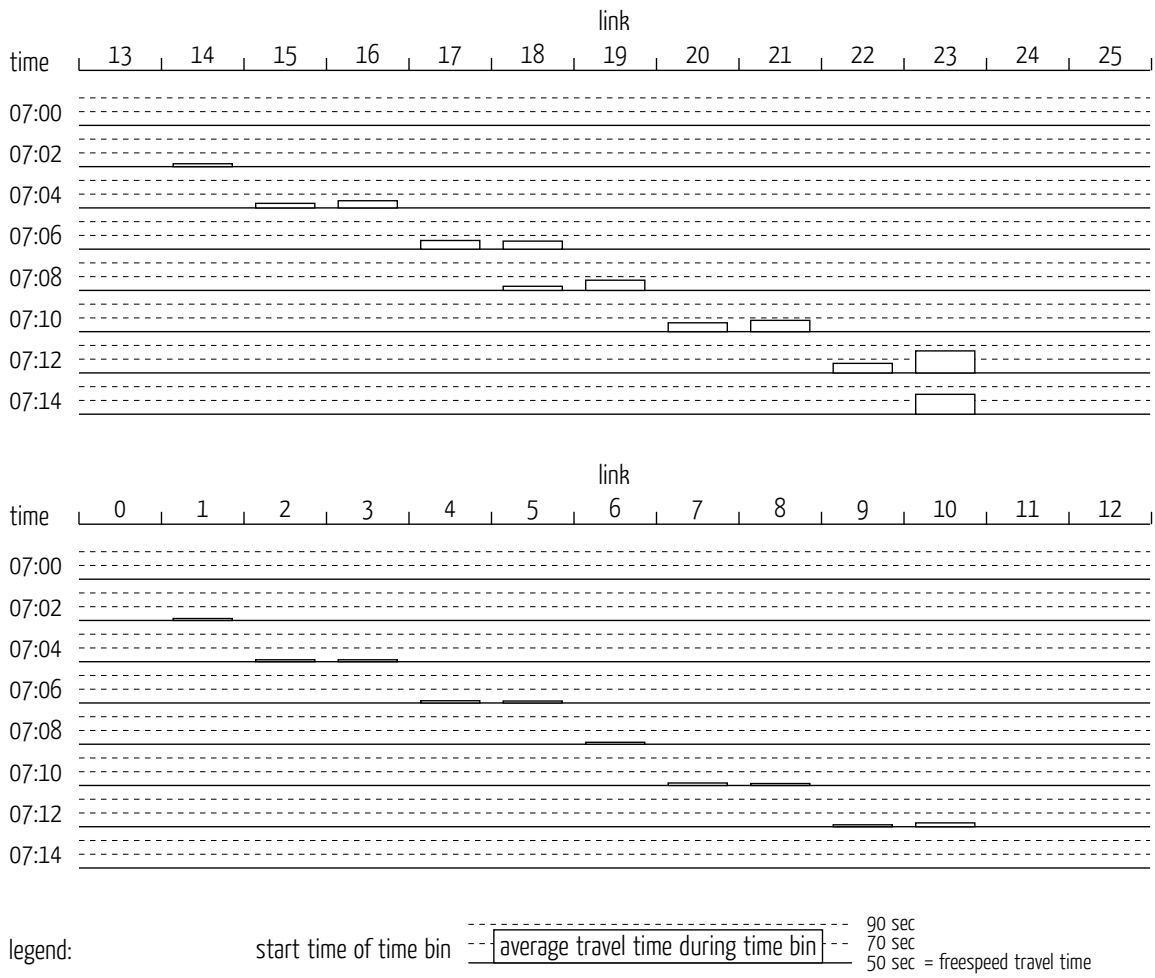


Figure 8.4: Travel times for each link over the lapse of time

8.1.3 TwoLinesDemo

This example shows how agents change from one transit line to another line to get from one location to the next one.

The network, depicted in Fig. 8.5, consists of a road corridor that splits into two roads at each of its two ends. Two transit lines serve each four stops along their route, two of them in common. Three agents want to travel from link 3 to link 12 at different times. To reach their target location, they have to change from transit line 1 to transit line 2, either at stop 3 or stop 4. Two of the agents change at stop 3, the third agent changes lines at stop 4.

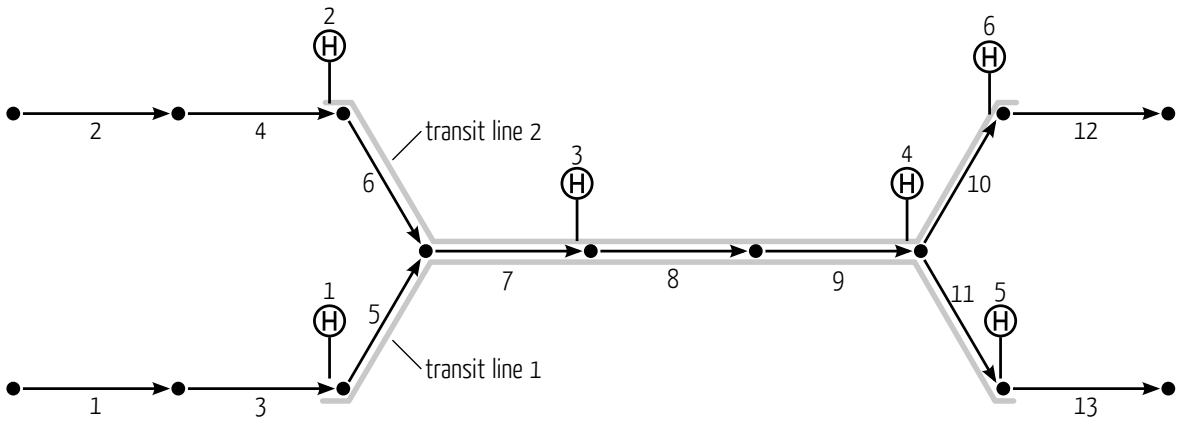


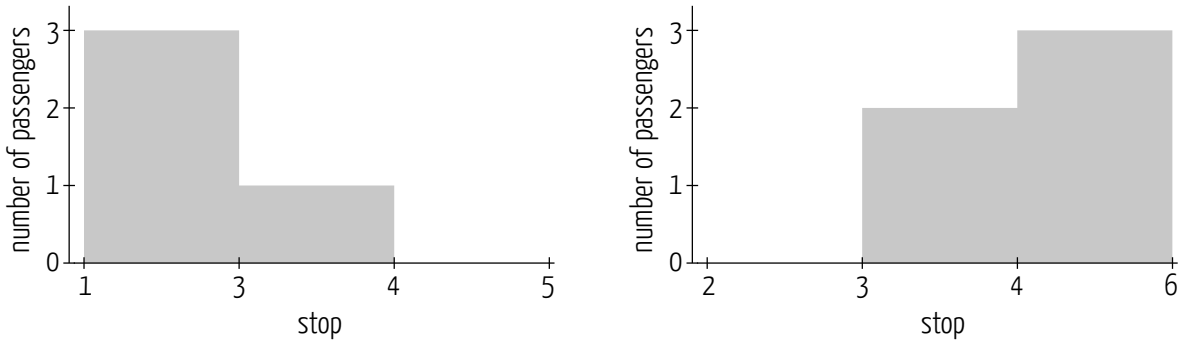
Figure 8.5: Network and stop locations for the two-lines example

Fig. 8.6 shows the number of passengers traveling on each line on the different sections between two stops. It can be nicely seen how the occupancy differs between the route segments depending on where agents change lines. The data for this figure is created with RouteOccupancy, presented in Sec. 7.7.

8.1.4 PseudoNetworkDemo

This example allows one to generate a network from a transit schedule on which the transit vehicles can move along. It is most useful to visually debug automatically generated transit schedule data.

The preparation of new scenarios for simulation usually includes the conversion of data. This is not different for transit schedule data. Especially in real world data, many special cases exist that may not be properly caught by automated conversion



(a) Number of passengers on transit line 1

(b) Number of passengers on transit line 2

Figure 8.6: Number of passengers traveling between the stops along the two transit lines

routines. For such cases, it is really helpful if the converted data can be visualized in some way or another. The visualization allows to quickly judge about the success of the conversion, showing visually apparent errors. A good visual representation does not mean the conversion is free of errors, but it facilitates to rule out some problems.

In the following, one simple method to generate a network from the given data is presented. The basic idea is to connect the stop facilities with links according to transit routes leading from one stop to another. In a first step, a node is generated in the network for every stop location. Next, for every transit route, links are added between two subsequent route stops. This already defines the basic geometry of the network and would be enough for the visualization of the network alone. More sophisticated methods for generating a transit network based on transit data could be thought of (e.g. taking an existing road network into account), but will not be discussed further in this example.

As it is helpful to see transit vehicles moving along the generated network, handling the stops along their route, the generated network must fulfill additional requirements. Each stop facility referenced from a route must thus be connected to exactly one link of the network (The limitation of at most one link per facility is a current limitation of the simulation framework). As the transit simulation handles transit stops at the end of a link (see Sec. 7.3.3), it was decided to connect transit stops (represented by nodes) to their incoming link. This approach leads to essentially two problems:

- The first stop along a route has no link it can be connected to.

- Multiple transit lines can have common route segments, where they serve the same stops. At the first jointly served stop, more than one incoming link exists at the node representing that stop.

The first issue is solved by adding a very short link at each stop where a transit route starts. The stop can then be assigned to this additional link. The second issue is solved by generating duplicates of a stop facility, such that each copy can be assigned to a different incoming link. This implies that also the transit routes must be adapted, replacing the original stop facility with the copy on the correct link. Fig. 8.7 shows the generated network based on the two transit lines used in the previous section (see Fig. 8.5). Note that stop facility 3 had to be duplicated, marked as 3.2, to accommodate the two lines joining for a common route segment.

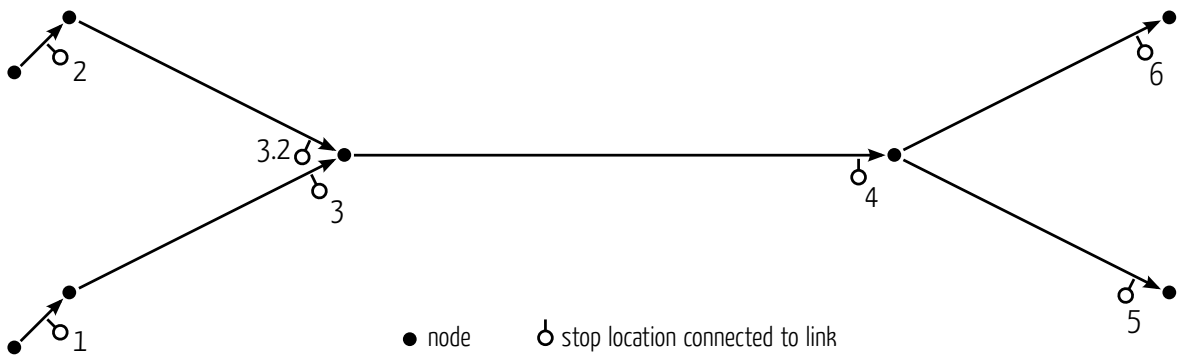


Figure 8.7: Generated Pseudo-Network from a transit schedule containing two uni-directional lines

8.2 Large-scale Application

For a simulation to be useful for transportation planning, it must not only support a wide variety of features, but it must also be applicable to large-scale, real-world applications. This section describes how the transit simulation was applied for the metropolitan area of Zurich in Switzerland, detailing the steps for data preparation as well as for configuring and running the simulation. It then analyses the computational performance of the large-scale application.

8.2.1 The Zurich Scenario

The Zurich region was already used several times throughout this dissertation for applications demonstrating the working of the simulation framework, e.g. in the

Road Pricing Case Study (see Sec. 5) and in the chapter about the simple transit integration (see Sec. 6.4). In contrast to these earlier usages, for the transit application the full population is used and not only a 10% sample.

The initial demand for this scenario consists of all travelers living in Switzerland that are inside an imaginary boundary around Zurich at least once during their day. This boundary is defined by a circle with a radius of 30 kilometers ($\approx$ 18.6 miles) with its center at "Bellevue", a central place in Zurich. This resulted in 1 817 944 agents. Every agent has a complete day plan with activities based on microcensus information [103, 104]. Fig. 8.8 shows the home locations of all 1.8 million agents on a map of Switzerland. The region of Zurich can be nicely recognized by the high density of agents. But it is also observable that a fair amount of agents travel from all over Switzerland into the Zurich region.

The road network used is a Swiss regional planning network that includes the major European freeway corridors (Fig. 8.9). It consists of 24 180 nodes and 60 492 links. The network was manually improved, especially in the region of Zurich, by adjusting links' flow capacity [22]. The road network is combined with the transit network in a following step (see next subsection), creating a multimodal network.

Transit schedule information is taken from the official model for transit in Zurich [119]. The transit lines and their schedules are given in a file for PTV VISUM [91]. The information in this file is converted into the transit schedule file format defined by MATSim, as it is described in the following subsection. Fig. 8.10 shows the resulting transit network after the data conversion.

8.2.2 Data Preparation

In real-world scenarios, preparation of data requires a substantial amount of resources. Besides the task of acquiring the data, the information usually must be converted to formats that MATSim can use. Quite frequently, data may contain encoding problems or wrong information due to automated data processing. If such problems are recognized—e.g. due to inexplicable observations in the simulation outcome—they must be manually fixed in an often time-consuming process.

The data used for the transit simulation of Zurich is given in a file for PTV VISUM. The network data can be written out to a text-based file format that also contains all data relevant for transit. A converter written within MATSim reads this file and builds a TransitSchedule. This file can then be read in by MATSim and converted into a MATSim TransitSchedule. The converter first creates stop facilities based on the stops given in the file. Next, it creates transit lines and routes

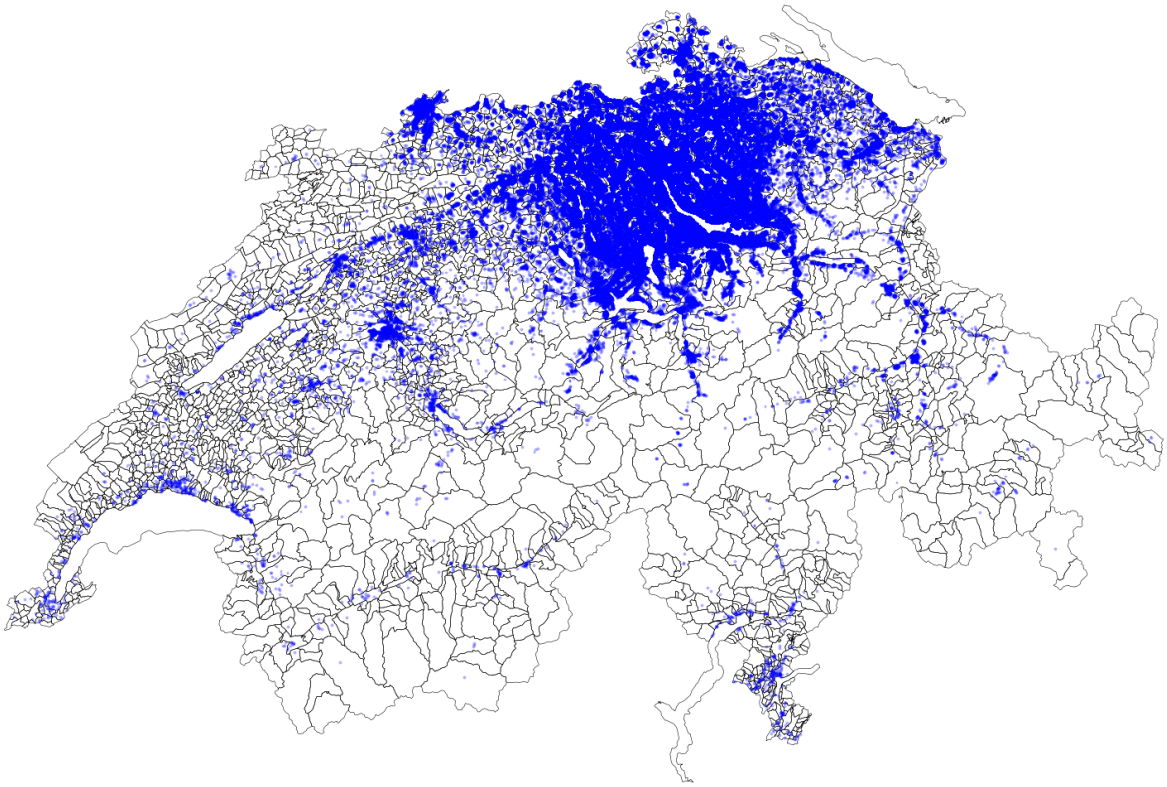
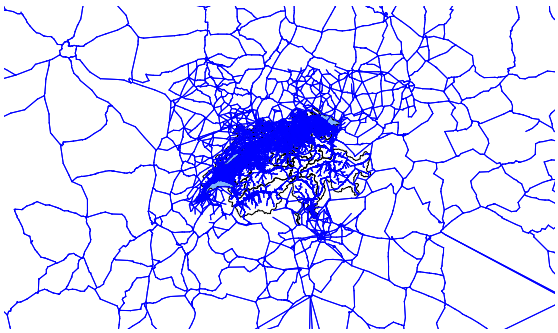
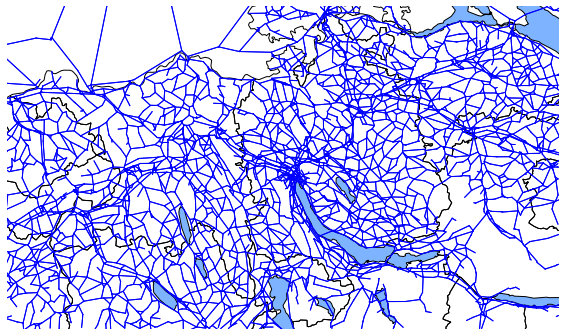


Figure 8.8: Home locations of the 1.8 million agents used in the simulation

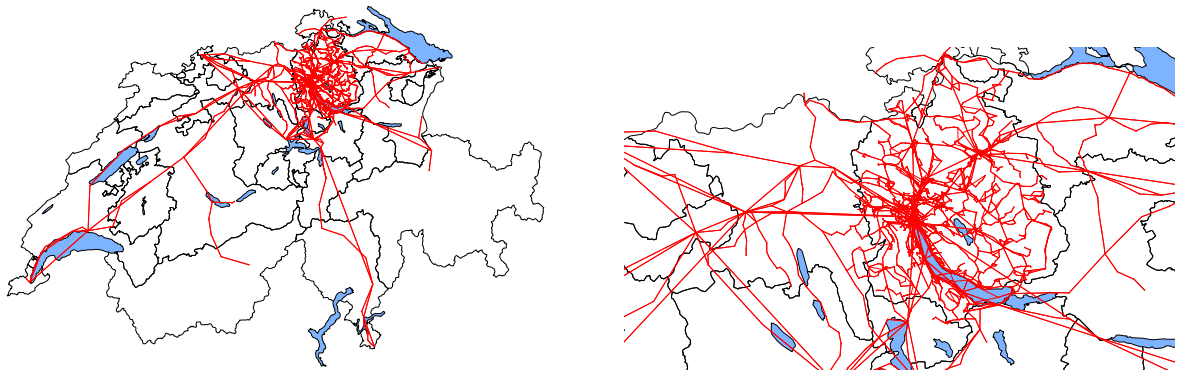


(a) Switzerland and neighboring countries



(b) Zurich area

Figure 8.9: Road network used in large-scale application



(a) Switzerland (b) Zurich area

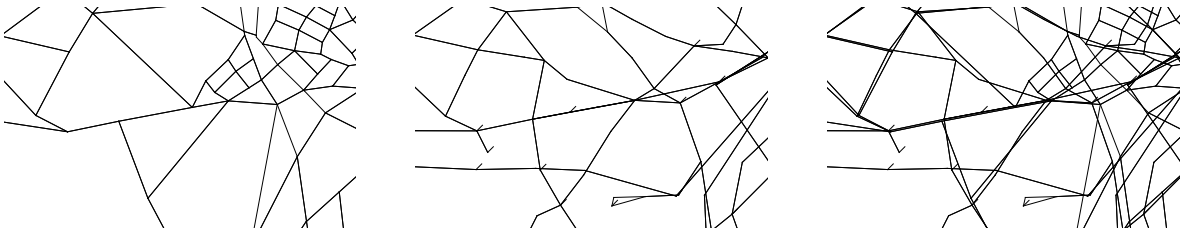
Figure 8.10: Transit network used in large-scale application

for each transit line in the VISUM file. In VISUM, a transit line is first separated into transit routes, and each transit route can have one or more time profiles. Each time profile has one or more departures assigned. In the data structure used in MATSim, each route has exactly one time profile assigned. This results in the converter creating a transit route in MATSim for each time profile in VISUM, duplicating the transit route information from VISUM if necessary. In addition to parsing the transit lines, the converter also creates the corresponding vehicle types based on the data given in the VISUM file. A vehicle of the corresponding type will be created for each departure. The transit schedule and vehicle definitions can than be written out to file in the native XML-based file formats that MATSim uses.

The converter handles only data related to the transit schedule, but not about the network. Thus, the network the transit lines operate on is generated automatically based on stop locations and transit routes. This is done by the pseudo network generator described in Sec. 8.1.4. The generated network encompasses 3245 nodes and 6533 links. As the simulation also includes private car traffic and only one network can be simulated, the generated (pseudo) transit network has to be merged with the existing road network. As a first step, a simple algorithm just lays the two networks on top of each other, so they build two separated layers or subnetworks. This results in transit traffic and private car traffic not influencing each other. Transit lines operated by road vehicles could be matched to use links from the existing road network instead of generated pseudo network. This process must likely be done manually, as the corresponding data is often not available because the route between two stops is usually "logical" to humans. Trépanier and Chapleau [113] propose a way to automatically match transit lines to road networks, as do Choi and Jang [23]. Both cases still require manual work, at least for

verification. But with freely available, high-quality data like from OpenStreetMap [86], it may be possible in the future to generate road and transit network completely automatically.

Fig. 8.11 shows cutouts from the different networks used to generate the final simulation network used. Fig. 8.11a depicts the road network with the limited resolution of the national road network model. Fig. 8.11b shows the transit network generated by the pseudo network algorithm described in Sec. 8.1.4. The short, diagonally running links that can be seen in the figure are the automatically generated links at the start of a transit route that the first stop of the route is connected to. In Fig. 8.11c, the merged network can be seen. As was mentioned before, this network could be simplified in some areas by merging road and transit links into links used by both modes—in case the two modes indeed share the same physical space. On the other hand, there also exist transit links for which no road links can be matched, either due to the missing resolution of the used road network or due to the fact that the transit does not use roads, but rails for example. Such differences make it hard to come up with a better, automated way to generate a merged network, resulting in manual work paired with knowledge of the location to clean up such a network.



(a) Cutout of the road network

(b) Cutout of the generated transit network

(c) Cutout of the merged network

Figure 8.11: Cutouts of the different networks used to generate the network used for simulation

8.2.3 Running the Simulation

Running a transit simulation is rather easy given the TransitControler presented in Sec. 7.6. Still, there are a few points that must be taken into account configuring a transit simulation.

- **Use XML events file format.** By default, the controler writes events in an older, tabular text file format. The TransitControler only ensures that events

are written in an XML file format, but does not disable the writing of text-based events. Usually, the XML format is enough. By setting the events file format to XML in the configuration (overwriting the default text-format), one can save some overhead during the simulation.

- **Add scoring parameters for transit interaction activities.** The transit router adds activities of type *transit interaction* to the plans. The scoring module will try to calculate a score for such activities, making it necessary to specify this activity type in the configuration for the scoring.
- **Set a simulation end time.** As the transit schedule is not repeated after midnight, it might happen that agents stand at a stop location, waiting for a vehicle to arrive, but no more vehicle will arrive there. As such agents are seen as still traveling in the simulation, the simulation will not end in such cases. Specifying an end time for the simulation (e.g. 30:00:00) will abort the simulation at that time if not yet all agents have arrived at their last activity.

The configuration of the plans scoring, especially the behavioral parameters such as the values of time for car and transit, are the same as in Sec. 6.2.3 and Sec. 6.4 respectively, with $\beta_{tr,transit}$ set to –3 Euro per hour and $\beta_{tr,car}$ remaining at –6 Euro per hour.

To be able to compare the detailed transit simulation with the previous, simple transit version (where transit trips take twice the time of the freeflow travel time by car for the same route; see Sec. 6) in full detail, the exact same setup is also run with the standard Controler.

8.2.4 Transit Simulation Outcome

In the following section, numerous geographical analyses are shown with text often referring to cities or regions. People not familiar with the geography of Switzerland and Zurich especially find maps of the discussed regions in Appendix C. All cities or regions mentioned in the following text are labeled in those maps. In addition, Appendix D contains an overview of the regional transit lines for the Zurich area as well as the city of Zurich.

Both simulations, the simple model and the detailed transit simulation, were run for 100 iterations. Fig. 8.12 shows again the locations of all agents, but differentiates between agents using a car and agents using transit to travel during their day. Already this rather simple analysis shows very promising results:

- **High share of transit users in large cities.** Large cities with good transit offerings have notably more transit users. Fig. 8.13 shows the larger Zurich area in more detail. The cities of Zurich (center), Winterthur (north-east of Zurich) and Schaffhausen (at the top border of the figure) as well as some small cities can be clearly recognized based on large amount of transit users.
- **High share of transit users for long-distance journeys.** The Swiss federal railway company maintains an attractive network of intercity connections between the larger cities of Switzerland. These connections usually offer a comparable travel time as if the route was undertaken with a private car. This fact can be seen in Fig. 8.12, where the especially cities (e.g. Basel, Berne, Geneva, Lausanne, but also many smaller ones) can be clearly recognized by the high number of red dots.
- **High share of car users in regions with bad transit accessibility.** In some regions, especially some valleys in the alps, there are significantly more agents using the car than transit to travel to the region of Zurich. Some such regions can be seen in Fig. 8.12, for example the valley of the Rhône in the Canton of Valais (the broadest of the parts in south of Switzerland), or around St. Moritz (in the south-eastern part of Switzerland). The few red dots spotted in these regions may be part of the 10% of agents trying out a new mode of transportation in this iteration, as part of the replanning process.

In comparison, the same analysis for the simple transit simulation (see Fig. 8.14) shows nothing comparable. Only in the city center of Zurich, transit seems to be preferred by the agents. This is likely due to the fact that there the number of transit stop locations is quite high, service quality is good, and the regular traffic (against which transit has to compete) is often jammed, leading to longer travel times, possibly even longer than the transit estimation.

The effects of the mode choice model are not only observable on a large scale. When zooming in into the region south of the city of Zurich, one can see nicely how the model is also able to reflect small differences in transit accessibility. Fig. 8.15 shows small cities and villages south of the city of Zurich, along the lake of Zurich. Green points show the location of transit stops; big ones represent stops of trains, while small ones represent stops of regional bus services. Some distance away from the eastern shore of the lake, a series of train stops can be seen building a line, starting at the center-top of the image, moving to the south-east. These stops belong to the S18, a typical commuter train connecting the city center with suburban areas. The train line leads through Egg to the end point in Esslingen (see Fig. C.3 in the appendix for information about named locations). Comparing the share of transit users in Egg and Esslingen with the share in the next village to the south-

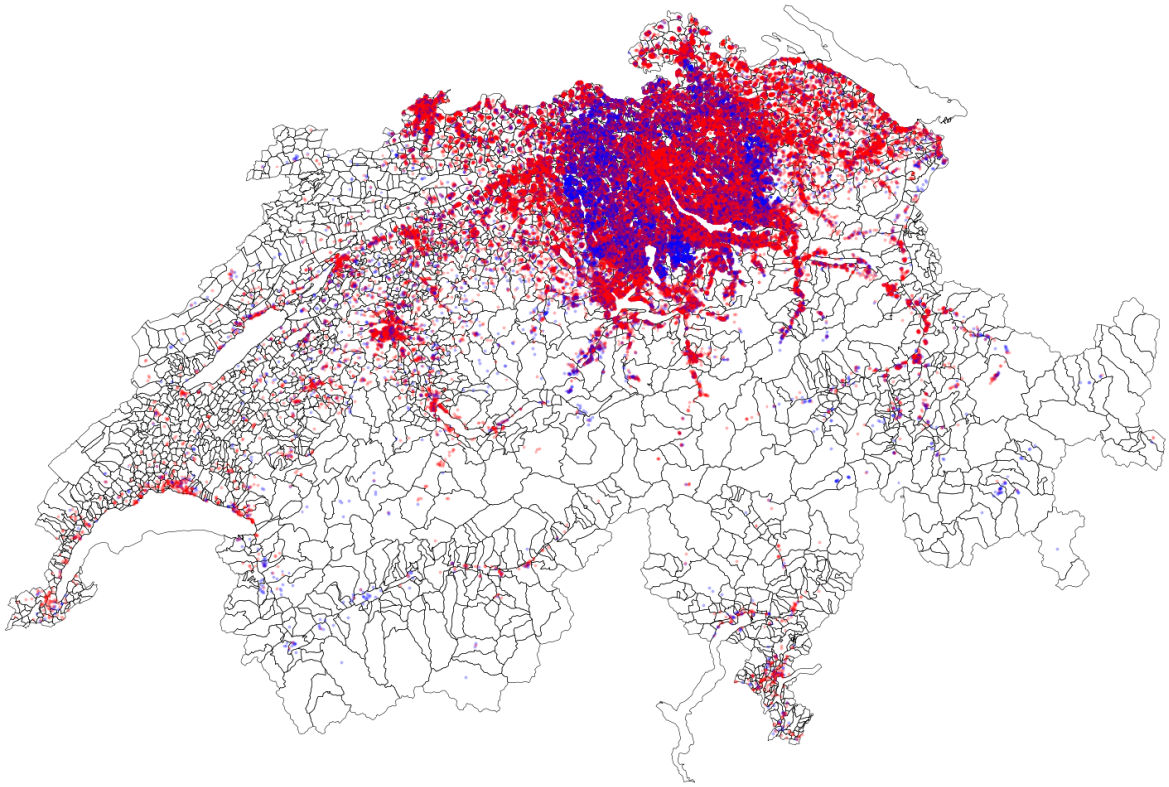


Figure 8.12: Chosen mode of transportation of all agents with the detailed transit simulation model. Iteration 100, red dots depict transit users, blue ones car users.

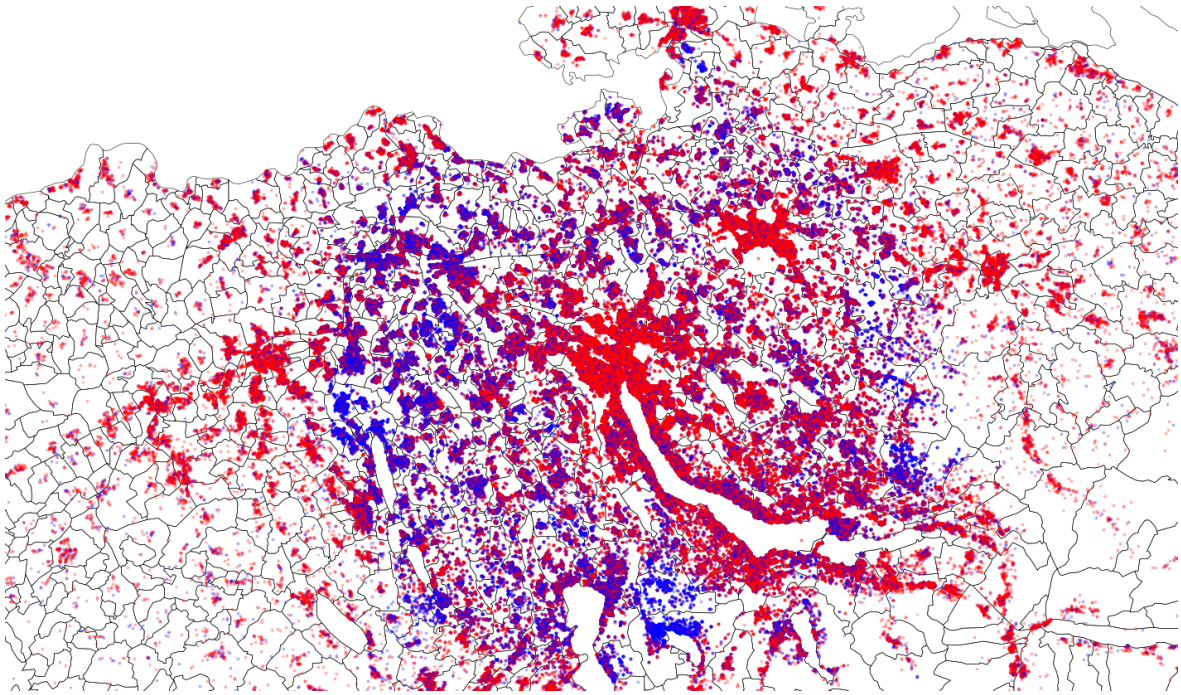


Figure 8.13: Chosen mode of transportation of agents living in the larger Zurich area. Iteration 100, red dots depict transit users, blue ones car users.

east, Oetwil am See, the difference is very nicely recognizable. Similar effects can be recognized along both shores of the lake, where the train stations are usually directly along the shore line, leading to a high number of transit users in these areas. Moving away from the shore, the number of car users (blue dots in Fig. 8.15) increases.

An interesting question is, how the detailed transit simulation impacts the realism of the large-scale transit simulation. Before answering that question, one has to recognize that the two shown simulations (the one with the detailed transit model, and the other with the simple transit model) were not especially calibrated. A good model, badly calibrated, may likely produce worse results than a mediocre model with reasonable calibration. But finding the best values for a simulation is a complex topic in itself (e.g. see [59, 37, 35]). Not only must values from the reality be retrieved, for example by surveys, but such values must also be related to the simulation model, as a model cannot reproduce every single aspect of the reality. As such, the following results are not represented to show the amount of realism an agent-based simulation can achieve—for that, the models would have needed to be calibrated in detail—but to show how the detailed transit simulation effects the results given the same (not really calibrated) parameters as the simple transit model.

Fig. 8.16 shows a comparison of the car traffic in the simulation models with 159 road traffic counting stations. Figures 8.16a and 8.16b show the mean relative error and the mean absolute bias of all counting stations over the time of day in hourly bins. As can be seen, the mean relative error is significantly higher in the detailed transit simulation than in the simple transit model. Looking at the mean absolute bias shows that in the simulation far fewer cars were counted along the counting stations than in reality. In addition, in the case of the detailed transit simulation the number of counted cars is even lower than in the case of the simple transit model. This decrease is reflected in the higher relative error. Figures 8.16c and 8.16d show a comparison of the actual counted number of cars for the hour from 7am to 8am. Again it can be clearly seen that in the case of the detailed transit simulation, in general far too few cars are counted in the simulation compared to the reality. In the case of the simple transit model (Fig. 8.16d), the number of counting stations that have too much traffic in the simulation is comparable to the number of counting stations with too few cars.

The inferior results are not surprising considering the previously mentioned notes about calibration. As described in Sec. 8.2.3, the behavioral parameters, especially the value of times required for the scoring, were taken from studies related to the simple transit model (see Sec. 6.2.3). Thus it is not surprising that this

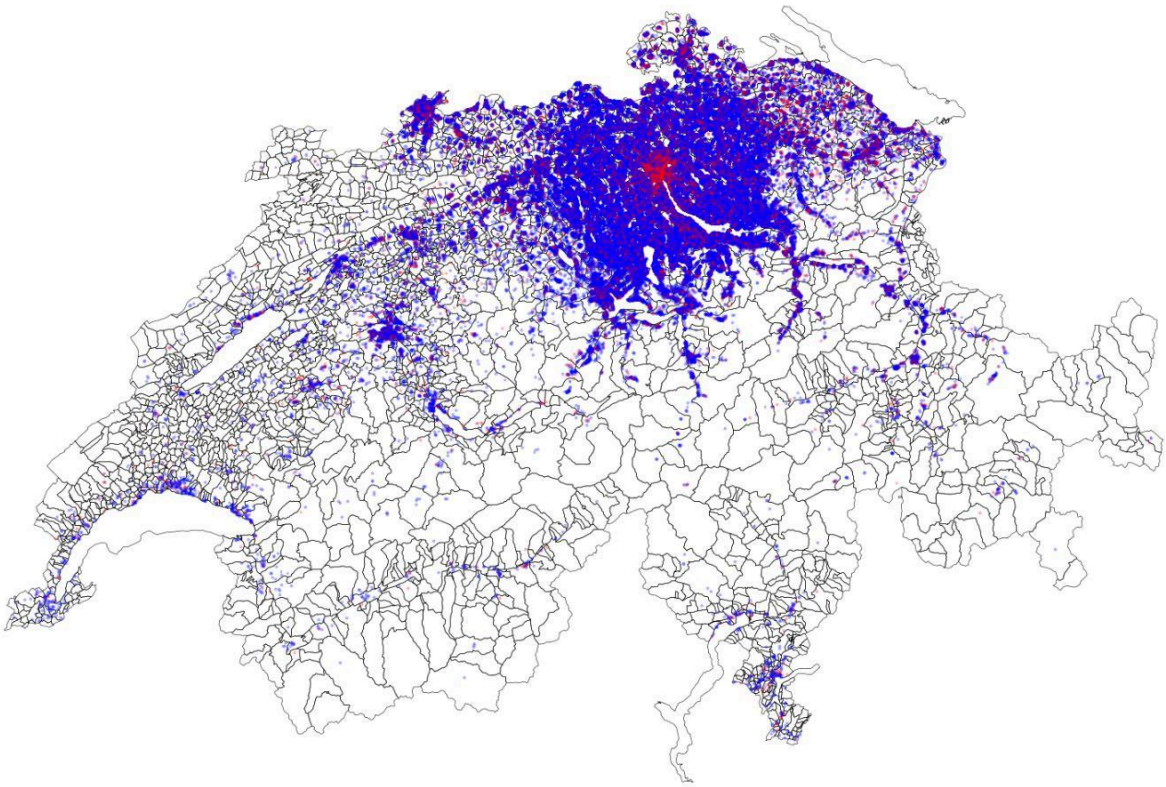


Figure 8.14: Chosen mode of transportation of all agents in the simple model. Iteration 100, red dots depict transit users, blue ones car users.

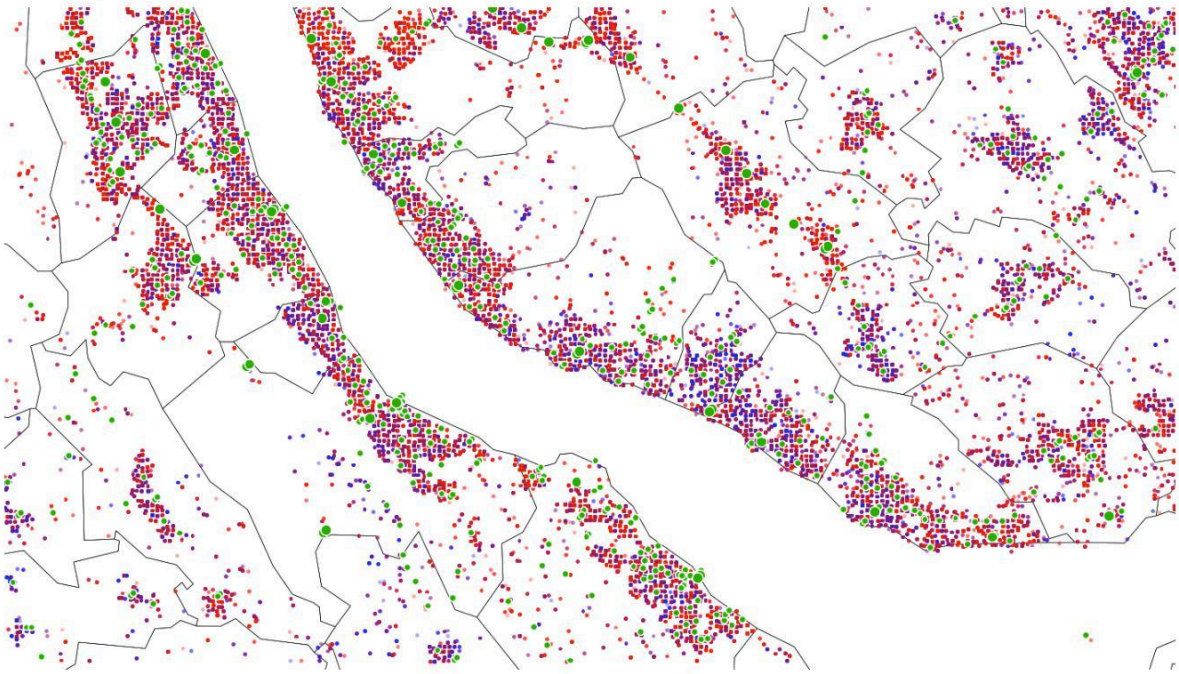
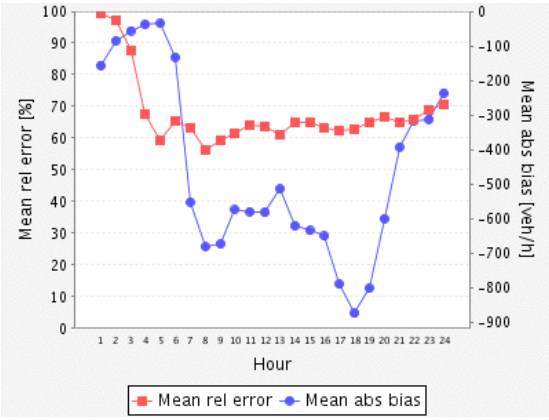
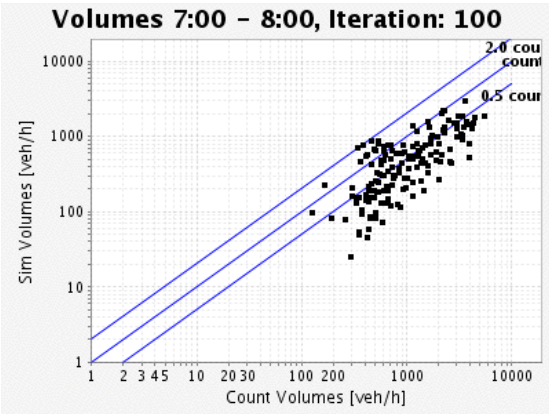


Figure 8.15: Chosen mode of transportation and location of transit stops in the detailed transit simulation model. Iteration 100, red dots depict transit users, blue ones car users, large green circles represent train stop locations, small green circles other transit stop locations.

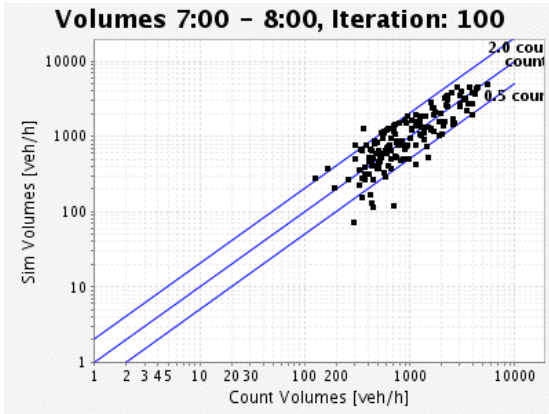


(a) Mean relative error with detailed transit simulation

(b) Mean relative error with simple transit model



(c) Comparison of traffic volumes from 7am to 8am with the detailed transit simulation



(d) Comparison of traffic volumes from 7am to 8am with the simple transit model

Figure 8.16: Comparison of simulated traffic volumes with real-world counts

model generates the better results, despite the change from a 10% sample to the full 100% of the population. On the contrary, it is what must have been expected. The simple transit model assumed a transit travel time equal to twice the time needed by car on a empty road network. While that assumption may hold true in the city center of Zurich, it is often not the case for long-distance travels. Not only decreases the influence of access and egress times on the total travel time when the actual travel time increases. Between larger cities, the transit travel time may also be equal to or even shorter than the travel time required when traveling by car. For example, the travel from Zurich to Berne takes just under one hour with the train. With the car, a distance of around 120 kilometers must be traveled, resulting in

a travel time of at least one hour. The 280 kilometers from Berlin to Hannover take usually around 100 minutes by train, much faster than what is possible in a private car.

As the new transit-model was especially designed to model the differences in the reachability by using detailed transit schedules, it is obvious that the parameters used for the simulation will have to be adjusted and newly calibrated. Especially, the time needed for access and egress will have to be considered in more detail (see [59]). In the parameters used, the value of times reflected the factor of 2 between the free flow travel times of car and transit in the simple transit model, such that the general costs for both alternatives would be around the same value. As the transit in the new model is often faster on long distance travels, the calculated costs in that model are now significantly lower than when using a car; likely even unrealistically low, resulting in the too high share of transit users. Estimating new values and calibrating the simulation which those will likely lead to better results in future runs.

8.2.5 Sensitivity Study

In order to test the sensitivity of the simulation and the mode choice model, a slight change to the available transit services was made: One suburban train line, labeled S7, was removed from the system. This line leads from Winterthur through regions north of Zurich city, passing through Zurich (main station), continuing further along the eastern shore of lake Zurich down to Rapperswil, a small city in the South of the area of interest. The S7 has a strong influence on the reachability especially for the municipalities along the south-eastern parts of the lake. While there are other train lines along the lake, the S7 does not stop in the municipalities being near the city of Zurich, providing an express connection for the municipalities farther away from the city.

If the S7 is removed, the villages along the most eastern parts of lake are losing the direct connection by train to the city of Zurich. Some villages nearer to the city still have other train lines available, albeit slower ones as they will stop in every municipality on their trip to Zurich. In addition, in the regions north of the city, the S7 provides train connectivity to a few places with a considerable amount of working places near the airport. These places can also be reached by different bus lines, so the accessibility is still guaranteed, even though with longer travel times and additional line changes.

For each municipality, the share of transit users living in that municipality can be calculated. Fig. 8.17 shows the difference between the share of transit users in

the simulation including the S7 and the simulation excluding the S7. A negative change means that less people are using transit in the model the line S7 is missing.

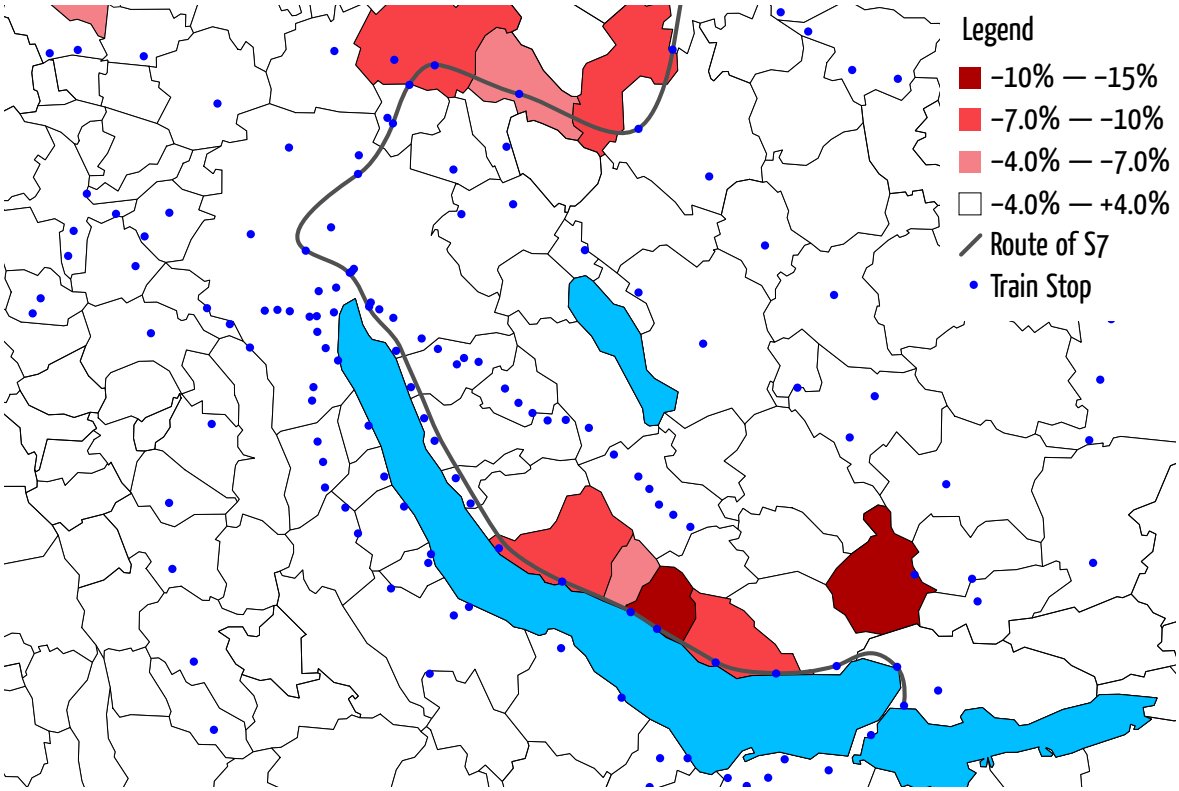


Figure 8.17: Absolute change in the share of transit users per municipality in percentage after train line S7 is removed from the model

It can be nicely seen how the municipalities along the lake and close to the city of Zurich are not affected by the simulated change of transit services. These are exactly those villages, where the S7 does not stop. In the municipalities farther away (namely Meilen, Uetikon am See, Männedorf and Stäfa), the share of transit users drops significantly. It is the largest in Männedorf, the first village along the shore only connected by the S7. Similar effects can also be observed in the municipalities north-east of the city, where the S7 is the only train stopping.

8.2.6 Performance analysis

Performance is an important aspect in simulating traffic, as it was already mentioned at several places in this dissertation. The following results are based on runs with the Zurich Scenario, consisting of 1 817 944 agents and a network with 67025

links and 27425 nodes. The simulations were run on a system having two Quad Core Intel Xeon Processors (X5550) running at 2.67 GHz. While each processor has 4 CPU cores, the cores support Intel's Hyper-Threading Technology, resulting in a total of 16 threads that can be executed simultaneously on the system. DDR3 Memory was connected through a front side bus clocked at 1066 MHz. Sun's Java Virtual Machine, Version JDK 6 Update 16, was used for execution. The virtual machine was given 31 GB of RAM. In addition, the option to use compressed object points [109] (JVM-Arguments: `-XX:+UseCompressedOops`, available since Sun's JDK 6 Update 14, by default enabled in the upcoming JDK 7 release) was enabled, effectively reducing the amount of memory required to run the simulations by partially using a 32-bit data model as well as reducing the runtime of the simulations due to the lowered amount of required 64-bit operations.

The simulations are configured to make use of all possible 16 threads for replanning. The traffic simulation itself does not support multiple threads, running single-threaded. Events generated by the traffic simulation are added to a queue, from which an additional single worker thread handles the events (so called *parallel events handling* in MATSim). With this setup, the simulation itself can just create events, but does not have to wait until each event is processed before continuing. Instead, the events are processed in a separate thread. In the case that the simulation generates events faster than they are processed, the simulation has to wait at the end of each iteration until all events are processed. This is necessary to ensure correct analysis results being written to disk at the end of the iteration, or to provide meaningful data to the next iteration's replanning. Events are written to a file only every tenth iteration to save disk space as well as processing time. In fact, events are only written to disk to allow later post-processing of the simulation results. Also every tenth iteration, the complete population including all agents and all their plans are dumped to a file on disk. This file is also useful for post-processing, but makes it also possible to resume a simulation at that specific iteration.

Fig. 8.18 shows the time needed for running the simulation, for each iteration and part of the simulation. The lowest part in the bars (colored blue) shows the time needed to run the replanning part. It averages at around 11 minutes. If the same workload should have been handled by a single thread instead of the 16 threads available, the replanning would have taken nearly 3 hours per iteration. The next part in the bars (colored green) shows the time the traffic simulation ran in each iteration. It takes about 7–8 minutes per iteration, except the iterations when events are written to disk. The third part (colored red) only occurs every tenth iteration and reflects the time needed to write the population data to disk. With

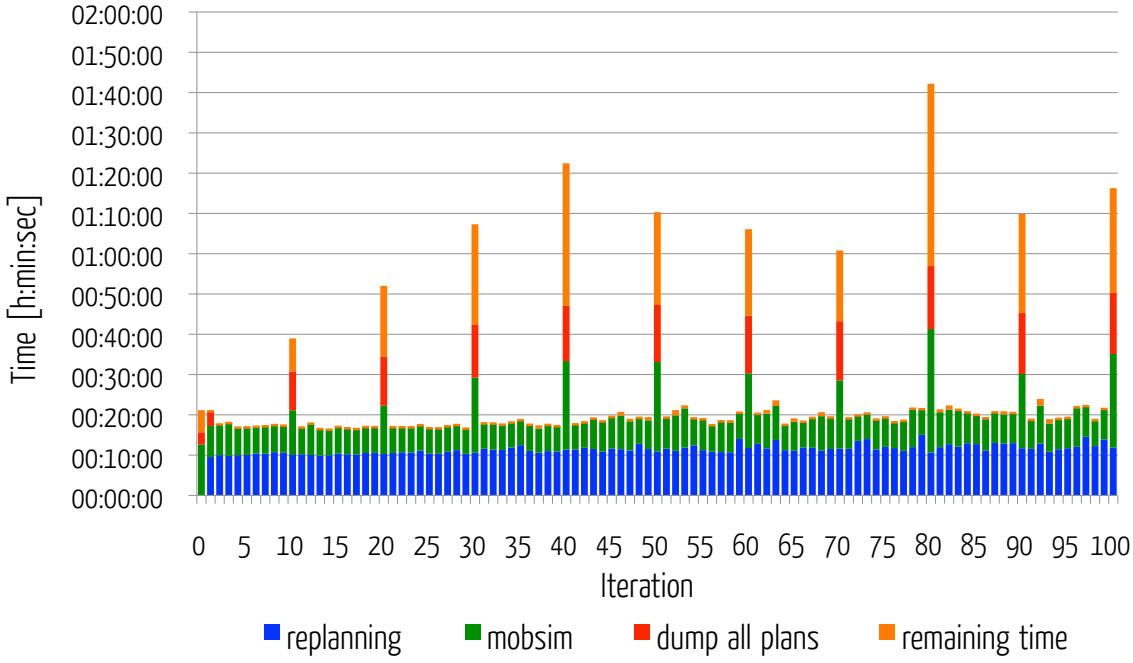


Figure 8.18: Time for computation per iteration and simulation part

about 15 minutes in the later iteration where the maximum number of plans per agent is reached, it actually takes longer than running the traffic flow simulation. The fourth and final part of the bars (colored orange) is best visible again every tenth iteration, but also exists in the other iterations. This parts contains all the remaining time per iteration that cannot be allocated to a specific task. It includes setting up each iteration, but also writing analysis data at the end of each iteration and is usually diminishable small. In each tenth iteration this remaining part also includes waiting on the events to be completely handled, which takes longer as in these iterations the events are also written to disk. By allocating more than one thread for the event-handling, this overhead might be reduced by some parts.

Table 8.1 shows a comparison of the simulation time between the detailed transit simulation and the simple transit model. In the detailed transit simulation, the average time per iteration is about three times the number of the simple transit model. On a closer look, the traffic flow simulation itself only takes about one third more time. But the replanning in the detailed transit simulation takes nearly 20 times as much time as in the simple transit model.

The increase in the traffic simulation's runtime can partly be explained with the additional complexity in the simulation. This additional work is reflected to some

Table 8.1: Performance comparison between the simple mode choice model and the detailed public transit simulation (times in [min:sec])

	simple model	detailed public transit simulation
avg. time per iteration	08:04	24:03
avg. time for replanning	00:37	11:27
avg. time for traffic flow simulation	06:02	08:24
number of events in iteration 100	94 138 552	107 399 914

parts also in the higher number of events generated by the simulation. In addition, the new events generated by the detailed transit simulation are not yet dispatched as efficient as the other, already existing types of events. Due to the high number of events in each iteration, quite some amount of work was invested in MATSim to optimize the way how incoming events are dispatched to the registered event handlers. These optimizations now need also to be applied for the new types of events.

The huge slow-down in the replanning has two causes: First, the transit router uses only a (comparatively) slow Dijkstra algorithm for the least cost path calculation, while in the simple transit module a highly optimized A* with landmarks algorithm is used (see Sec. 4.7.3 and 7.4.3). Second, the routing process for cars is also slow downed as that router now needs to make sure only links available to cars are used. Such a check was not needed as long as no multimodal network was used, but is now required with the multimodal network. Luckily, both causes can be resolved: As already mentioned in Sec. 7.4.3, it should be possible to apply the modifications made to Dijkstra's algorithm also to the A* with landmarks algorithm, leading to a notable speed-up. And instead of using a multimodal network for car-routing, the network could be once analyzed and a car-only subnetwork be generated that the car router could use, foregoing the need to check all the time if the corresponding links could be used. One has to note that this increases the memory usage, as the subnetwork must be stored in memory. Compared to all the other data required in the simulation although, this small memory overhead should be negligible.

8.3 Conceptual Applications

The modifications done to the QueueSimulation to simulate transit open up several possibilities to also simulate other forms of traffic besides private car traffic and urban transit traffic. Most of these possibilities come from the differentiation between vehicles and persons and the extension that vehicles can now transport more people than just the driver. In the following, a few of these additional forms of traffic are presented. It will be shown how the current simulation infrastructure could be used or extended to simulate those kinds of traffic.

8.3.1 Paratransit

Paratransit is usually referred to as a flexible mode of transport for passengers that does not follow fixed schedules or even routes. It is often operated by fleets of small buses or vans, although other vehicles can be used as well. While paratransit is often the only kind of public transportation available in cities of developing countries, it may exist as well in rural areas of developed countries where a fixed-route, fixed-schedule service is not cost-effective. In the latter case, the offered services are sometimes named as *Dial-a-ride*, *Ruf-Bus* (german) or similar.

One of the biggest differences between paratransit and the currently implemented infrastructure for simulating transit are the stop locations. Paratransit vehicles usually stop everywhere it is suitable when passengers want to get in or out the vehicle. The way how to simulate paratransit depends on the kind of paratransit.

For route-based paratransit, a simple approach would be to define a stop facility on each link that is served by paratransit. Passenger agents could then normally find the best connection using a router that includes the paratransit route. Instead of using a time table for routing, an average expected waiting time—maybe depending on time of day—could be used as time or cost for accessing a new paratransit vehicle in the routing process. In the simulation, a custom paratransit driver could be implemented that decides on itself if it should stop at a stop location or not. This decision could be—for example—depending on the number of passenger already in the vehicle. The same way, a paratransit driver could decide to wait a bit longer at a stop (by returning a value bigger than zero in `handleStop()`, and returning the same stop again in `getNextStop()`) in the hope to transport additional passengers arriving soon.

If paratransit offerings are not route-based, the simulation would likely need more adaptations. Although one could still define a stop facility on every link of a network, this may not be very usable. First, agents could not use those stops

for routing purposes. Second, agents could get stuck if they would wait at a stop location until a paratransit vehicle arrives, which could never happen. Instead, one might have to look into simulating also walking agents. This way, agents could search a route through the network and start walking along those links. The simulation could move the agents from one link to the next one according to their walking route. A custom implementation for paratransit drivers could announce their availability to agents walking along a link, giving them the option to hop in the paratransit vehicle. The same way, a paratransit driver would need to inform its passengers about every new link it drives along, so that the passengers could get out if the driver continues in a direction not appropriate for single agents.

As mentioned in the beginning, the main help by the implementation of the transit simulation is the differentiation of vehicles and passengers. Route-based paratransit should require only a small amount of changes, namely the construction of the paratransit routes network, a custom time-function for the transit router, and a custom implementation of the driver. Simulating paratransit not restricted to fixed routes will need a much higher effort.

8.3.2 Ride Sharing

Ride Sharing describes the state when two or more people travel together in a private car. This can be a family driving jointly to some leisure location, or co-workers that live close to each other and share a common trip to/from work. In some regions, agencies for arranged lifts exist, allowing one to easily find other people driving the same way in order to share rides.

Given the code infrastructure that differentiates persons and vehicles, the implementation of ride sharing should be rather simple. Similar to every transit departure specifying a vehicle to be used, each car-leg or car-route should also specify with which car the leg or route is undertaken. An additional attribute could define which of the persons is the driver. If such an attribute is missing, a simple algorithm could be used (e.g. a random person in the car that has a driving license; this would already clearly define all the cases where a parent brings its children to school, as children usually do not have a driving license).

One point that needs to be taken care of is that all persons that are scheduled to drive together indeed travel together. This means mostly that the driver should not depart before all passengers are also in the car. Assuming that such combined trips need a special replanning module that tries to put agents into the same car, this could be either solved with activity end times (passengers must be in the car before the driver) or better with additional route attributes. Such attributes, like

a list of passengers, could delay the departure of a vehicle until a condition based on the attributes is fulfilled. This would simplify to model actions like a parent picking up its child at a kindergarten, or one co-worker picking up another one somewhere along its route.

8.3.3 Car Sharing

Car Sharing can be used to describe several things. On the one hand, it could describe the offerings from car rental companies. On the other hand, it can simply relate to the fact that one car in a household can be used by more than just one person, although only one person can use the car at any point of time.

Both cases can be simulated rather straight forward once car-routes specify with which car a route should be undertaken. If a car is not available (i.e. it is not parked on the link where the agents wants to depart), the simulation could let an agent either wait until the car is available, or the agent could be marked as stuck, removed from the simulation and a penalty added to its score.

Special replanning modules could try to improve the allocation of cars to agents. This could be done either in household-aware modules, trying to find the optimal use of vehicles available in a household, or in modules managing the fleet of commercial car rental companies.

8.3.4 Automatic Transportation Systems

Automatic transportation systems that work without a driver steering a vehicle are coming up more and more. While such systems are currently mostly limited to fixed route systems (e.g. so called "people movers" at airports and theme parks, or modern subway systems as the line M2 in Lausanne, Switzerland) with accordingly large vehicles, people mover systems with much smaller vehicles are often part of visionary studies about future cities.

Similar to the implementation of Paratransit, the logic that drives a vehicle could be easily replaced to implement automatic transit systems. Instead of just following a given plan, containing a route and sequence of stops, the driver logic could query the agent for its target location and automatically find the best way through the network. This would require a passenger to actually accept every transit vehicle at a stop, as every vehicle would be able to bring the passenger to its destination. Stops could either be located on every link of the network or at special stop facilities, depending on the actual realization of a people mover system.

9 Conclusion

9.1 Outlook

The Multi-Agent Transportation Simulation, MATSim, has a lively history. It started less than ten years ago as a collection of multiple executables written in C/C++. Each executable had a well-defined job to do, like calculating routes, adapting departure times, or running the actual traffic flow simulation. As the speed of processors was still slow (compared to todays clock rates) and the amount of random access memory (RAM) that could be addressed was limited by 32-bit address pointers, the code supported the parallel execution of the simulation to overcome those limitations. Data was exchanged between the different executables by writing the data to disk, and reading it back from there later.

About four years ago, the project was started over, this time using Java for the implementation. Due to advances in computer technology, the structural composition was changed from multiple stand-alone executables to one integrated framework. Also, the possibility for parallel execution on a cluster was removed in favor of shared memory machines, which now offer big enough amounts of random access memory (RAM) to load large scenarios. These structural changes reduced the complexity enormously, allowing for quicker advances implementing new features. In addition, the number of contributors increased also substantively during the last few years, leading to an accelerated development speed.

The original C/C++ implementation of MATSim supported only private car traffic in the simulation, and used route and departure time choice as the only search dimensions. Now, in the newer Java implementation, additional features have been included: faster routing algorithms provided a big speed-up [63], the traffic flow simulation supports traffic lights [77], location choice is available as additional search dimension [52], and with Planomat exists an advanced, optimizing

replanning module that adapts route, departure time and mode choice [67, 33]. This dissertation adds the support for multiple modes of transportation to the framework as well as a detailed, integrated transit simulation. Ongoing projects look at the simulation of electric cars [41], usage of parking spaces, optimization of traffic lights and calculation of cars' emissions, to name a few.

Looking at this history naturally poses one question: What may the future bring? Respecting that this dissertation has evolved on the need to have more than just private cars in the simulation, the question could as well be asked as: What future requirements will MATSim have to fulfill? Obviously, this also poses the question: What will be feasible (by means of computational resources as well as understandable models) in the near future?

Based on the inclusion of transit, one may think about integrating additional kinds of traffic, especially freight and commercial traffic. But also paratransit, which was already mentioned in Sec. 8.3.1, should be looked at in more detail (which is to some parts already happening, see [24]). In the same vein, one could think about including bicycles and pedestrians in the model as well. On the other hand, the queue model used for the simulation imposes a kind of a barrier on the level of detail that can be simulated. This is already apparent in the existing simulation where interactions on links can only take place at the start or end of a link, e.g. limiting a close approximation to reality when handling bus stops. Researching improved ways to abstract traffic flows that will allow for more details on a link while still being efficiently computable may be a challenging task to be solved in the future.

The search dimensions (route, time, and now mode of transportation) used in the optimization process are still rather limiting. Additional modules that modify the activity patterns, either adding, removing or re-ordering activities in an agent's plan, can be thought of. Also a module to perform combined location and mode choice (as not all locations can be reached by transit, or by car, respectively) may help to improve the optimization process. One has to realize that additional or modified transportation infrastructure not only changes the actual traffic, but may also influence the land use development. Thus, a module updating land use information might be very interesting, especially in combination with the already existing location choice. This might be combined with the existing location choice module (for regular agents) or the research done for optimizing the locations of retailers [26].

This dissertation introduced the concept of vehicles and agents (either as drivers or passengers) to MATSim, which used vehicle-driver-units before. While vehicles are currently only really used for transit, their availability in the model opens up

additional possibilities. Currently, every person in MATSim has an attribute specifying if the person has a car available. In reality, mobility tools are often available to household members and have to be shared by them. Adding households to MATSim along with dedicated models to assign cars (and later maybe other mobility tools like bikes as well) to household members could lead to sophisticated modules regarding car-ownership, effectively implementing car-sharing as proposed in Sec. 8.3.3.

But even on a less general level, a lot of possibilities open up if one just looks at the new transit functionality. This dissertation has focused on the conceptual aspects of a transit simulation, only presenting a limited set of real applications of the simulation. Real-world case studies will certainly require additional extensions to be written, starting at a presumably large number of analysis modules up to modifications of the routing algorithms.

The current mode choice module could also be further improved. Instead of changing the mode of transportation of all legs in a plan, the module could differentiate more and change the modes based on sub tours. First experiments with sub tour based mode choice, although for the initial demand only, have been described in [25].

The transit router currently used offers lots of possibilities for improvements, not surprising as it was not the focus of this dissertation. The vast amount of literature concerning routing algorithms for transit give already many ideas. Without going into much detail, the following list highlights conceptual problems and challenges that apply to all implemented transit routers that should be integrated into MATSim:

- The router only uses the transit schedule currently. This is fine as long as it can be assumed that the transit vehicles indeed operate without significant delays. But as soon as buses or street cars get caught in congestion regularly during the rush hours, people may start to look at alternative ways of traveling in reality. An interesting alternative would be to use the actual transit travel times of the last iteration for the routing, as it is done currently by the car router. One would have to see if and what improvement such a change might bring.
- The router has no knowledge of capacity constraints. The current implementation will happily advise as many people as requested to use a specific transit line at a specific time of day, even if the transit vehicles used to serve that line may never offer enough capacity to transport all the people. In the same spirit is the router not able to respect quality demands of the passengers to be routed. If a handicapped passenger requires a seat at all cost, the

router would not be able to ensure this.

- The router must get much faster. The number of routing requests to be handled in each iteration is usually quite high in large scenarios. The process of finding routes for cars could be speed up immensely by the use of optimized algorithms (e.g. [63]). It would be interesting to see if similar optimizations could be added to the modified version of Dijkstra's algorithm used in the transit router, or if the transit router might take otherwise an advance of the sped-up car router.

Data availability is still a big challenge when initiating new scenarios. One promising approach is to take data from open sources like OpenStreetMap (OSM) [86], which has increased—and is still increasing—the amount of available data in huge steps in recent years. Experiments using data from OSM, converting it to a road network and using it for simulation have shown that this way is not only doable, but actually at least as useful and manageable than using private or commercial data sources. As OSM contains data about transit lines (at least subways, street cars and trains) in more and more regions, transit data could also be extracted from OSM. Data for the road network and the transit network originating from the same source could even make it possible to match transit lines more easily to roads automatically, as it is proposed in [23, 113].

9.2 Summary

This dissertation presented concepts and implementation details for building an agent-based transportation simulation, modeling in detail private cars as well as transit vehicles. Based on the existing C/C++ version of MATSim, the simulation tool was re-implemented in Java with an open, extensible structure (see chapter 4). The re-implementation was based on some data-structures created for the demand modeling framework presented in [9], especially on the network and population model.

The renewed simulation framework was then applied to a concrete research project, where the effects of time-dependent road-pricing measures were researched (chapter 5). The application showed that the simulation tool worked as expected by generating observable changes in the traffic pattern during the morning rush hour although the toll was only to be paid during the evening rush hours. Still, the validity of the simulated effects had to be questioned due to the fact that the model was restricted to private car traffic. Especially, no change in travel demand was accounted for, even if that is one of the possible reactions of people to road

pricing measures.

Consequently, it was researched how additional modes of transportation could be integrated into the simulation, giving the modeled agents the possibility to perform mode choice. Chapter 6 reports on the necessary modifications to the existing simulation framework to support other modes of transportation besides car. This goal was achieved by marking each plan with the mode of transportation to be used in its legs, resulting in the mode choice being done using the existing plan selection algorithm. The mode choice model was verified to work properly and then applied to a large-scale scenario.

Chapter 7 closes the conceptual part of this dissertation by giving a detailed look how the existing private car-only simulation was extended to model transit microscopically along the cars. It refines the aforementioned mode choice model implementing a "leg mode changer", replacing the need to mark plans with a mode of transportation.

Illustrative examples, showing the features of the new, integrated simulation supporting private cars and transit at the same time, are presented in chapter 8. A large-scale scenario, where more than 1 million agents are simulated in the greater region of Zurich, Switzerland, shows the applicability of the integrated simulation to real-world scenarios.

The presented simulation is currently already being used to research transit-related scenarios in Berlin, Germany, and will likely be part of an up-coming release of MATSim. Additional transit-related analyses will likely be implemented in the near future, driven by the actual demand of on-going and future research projects.

Acknowledgments

I would like to thank everyone who, in the last few years or for an even longer period of my life, supported me and thus enabled me to write this dissertation. My deepest gratefulness to all of you!

While the few lines above are certainly true, I do not think they really capture and mirror all my feelings after spending several years to work on this dissertation. I had the luck to work together with so many people, all of them unique in their own way, of all of them I have very special memories — I do not think that just a few generic lines would be adequate for thanking them. So let me bit a bit more extensive.

I would like to thank Kai Nagel very much for sharing his enthusiasm for agent-based transportation simulations, for giving me the possibility to work in his research group at TU Berlin, and being the examiner of my dissertation. I got to know Kai long before I started my work in Berlin. Actually, I first met Kai Nagel many years ago during my studies in computer science at ETH Zurich, Switzerland. In one of the exercises to a course he taught, we students had to implement a very simple simulation of an intersection. This was my first contact to the topic of traffic simulation. The fun implementing that simple model made me attend another course by Kai, which focused on agent-based traffic simulation. His enthusiasm for the topic caught me, and after that course, I started work on two semester projects, the first with a strong relation to transportation planning, the second to focus on traffic simulation. Despite his move from ETH Zurich to TU Berlin, we stayed in contact, and at the time I finished my studies, I got the opportunity to move to Berlin as well and continue to work in his research group

on agent-based transportation simulations. His unbroken passion for the work, his guidance on complex topics, the many interesting discussions we had, all that made my work stay in Berlin very enjoyable, which I would like to thank him for very much!

I am very thankful to Kay W. Axhausen for being the co-examiner of my dissertation. Similar to Kai Nagel, I met him for the first time during my studies at ETH Zurich, where he was the supervisor of my first semester project. During my dissertation, he made it possible for me to work part time at the Institute for Transport Planning and Systems (IVT) at the ETH, which I am very grateful for. The experience of working at the IVT was especially great for me as it eased the usage of the well-maintained Zurich scenario presented in Sec. 8.2.

Michael Balmer is the third person, next to Kai Nagel and Kay W. Axhausen, who played an important role in my academic career. Being a PhD student himself, I met Michael during my studies in computer science when he supervised one of my semester projects. His passion for agent-based traffic simulation was the same as Kai's, which made the work with him always a joy. He was the first one who mentioned to me that Kai was looking for PhD students in Berlin, asking me if I might not be interested. During all my work, I had lots of interesting discussions with him. Even though we did not always share the same opinion in our discussions, we always respected the position of the other, trying to find arguments for or against certain points, such that at the end all parties involved in the discussion gained knowledge. I have rarely had so intense, but also so satisfying, discussions with other people than Michael. I thank Michael Balmer very much for introducing me to MATSim, supporting my work as a student in Zurich, for the many interesting discussions we had, and for becoming such a good friend to me. I am especially happy to currently have the chance to continue to work with him together on MATSim.

My parents did a fantastic job raising me. They taught me to follow my plans and visions, supported my education, and showed me what it needs to be successful. They did so, leaving me enough space for realizing my own ideas and discovering the world at the same time, for searching for solutions to problems myself, while still being available in the background if needed. Thank you very much for all you have done! I also like to thank my sister Andrea. She supported my work and life, be it by proof-reading my early articles, or being there to discuss other matters of life.

I am very lucky to have found in Nadine Schüssler somebody who understands my passion for this work, who supports me, and who enjoys the life next to me. I

thank her very much to bear me and live with me.

MATSim, as a project, has reached a size, where it is impossible to achieve great results only by working alone for oneself. This is not different with this dissertation. The Zurich scenario, which was used in Chapter 5, Chapter 6 and Sec. 8.2, was contributed in large parts by Michael Balmer. Yu Chen improved the Zurich scenario further by manually updating the network based on data from OpenStreetMap. Mohit Shah implemented the first version of the converter to use transit schedules from VISUM in MATSim. David Strippgen was responsible for converting the original C++-based simulation to Java, before focusing on developing OTFVis, an interactive visualizer for MATSim. Andreas Neumann and Michael Zilske provided useful feedback when applying the developed transit features to a Berlin scenario. Andreas was furthermore a great office mate, with whom I also enjoyed debating topics not related to work. With Dominik Grether I had many great discussions about software engineering and software design principles on our quest to build a clean and stable code base for MATSim.

Finally, a big thanks to the MATSim-Community! I had great fun to work together with all the different people, no matter if they are living in Berlin, Zurich, Toronto, South-Africa, or in other places I'm not aware of. The interest they all have in MATSim, the different use cases MATSim is applied to, this all was—and still is—a big driving force for my work.

The work described in this dissertation is part of the open-source software project MATSim. The software can be freely downloaded from `http://matsim.org/` and used according to the terms of the GNU General Public License (GPL) version 2 or newer. To create this dissertation, several other free or open-source software was used. Text was set using the XƎTEX typesetting system, freely available from `http://scripts.sil.org/XeTeX` and distributed under the X11 free software license. The font family used for titles, headings and captions is the free font **Yanone Kaffeesatz** by Jan Gerner, licensed under the Creative Commons »By«-License, available from `http://www.yanone.de/typedesign/kaffeesatz/`. The font family used in listings, code examples or URLs is `Inconsolata`, an open-source font by Raph Levien, distributed under the Open Font License of SIL, available from `http://www.levien.com/type/myfonts/inconsolata.html`. The GIS plots shown in chapter 8.2 are created with Quantum GIS (`http://qgis.org`), an open-source Geographic Information System licensed under the GNU GPL. Many of the fig-

ures in this dissertations were created using the open-source software Inkscape (`http://inkscape.org/`), a powerful editor for vector graphics, also released under the GNU GPL.

The maps shown in chapter 8.2 and in appendix C are based on data from the Swiss Federal Statistical Office (Bundesamt für Statistik (BFS), GEOSTAT. Generalisierte Gemeindegrenzen der Schweiz 2008). The maps shown in appendix D are provided by and reprinted with the written permission of ZVV [126].

Part of the work presented in chapter 5 was founded by the Volvo Research and Educational Foundations within the research project "Environmentally-oriented Road Pricing for Livable Cities". Other parts of my work were founded by the Swiss innovation promotion agency CTI in the context of CTI project 8443.1 ESPP-ES, "Agenten-basierte Simulation für location based services". The remaining, and by far biggest, parts of my work were founded by the Technische Universität Berlin (TU Berlin).

Two different computer clusters were used to calculate the presented results. Early work was computed on the Beowulf cluster "simulant", at that time maintained by the Institute for Transport Research of the German Aerospace Centre (DLR) at Berlin-Adlershof. Later work, especially the large-scale simulations described in chapter 8, were run on a compute cluster maintained by the Faculty for Mathematics at TU Berlin. Miroslav Kolev administered and maintained the computer infrastructure used locally in our research group for several years. I am very grateful for his superb support, which made it possible to fully focus on the research.

Bibliography

[1] *Proceedings of Swiss Transport Research Conference (STRC)*, Monte Verita, CH. See www.strc.ch.

[2] *Proceedings of the meeting of the International Association for Travel Behavior Research (IATBR)*, Kyoto, Japan, 2006. See www.iatbr.org.

[3] *Proceedings of the 9th Swiss Transport Research Conference (STRC)*, Monte Verità, Switzerland, September 2009.

[4] Arnott, R., A. de Palma, and R. Lindsey. Economics of a bottleneck. *Journal of Urban Economics*, 27(1):111–130, 1990.

[5] Arunotayanun, K. and J. W. Polak. Accounting for supply chain structures in modelling freight mode choice behaviour. In *Proceedings of the European Transport Conference 2009*, 2009.

[6] Avineri, E. and J. Prashker. Sensitivity to uncertainty: Need for paradigm shift. *Transportation Research Record*, 1854:90–98, 2003.

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

[8] Balbo, F. and S. Pinson. *Engineering Societies in the Agents World II*, volume 2203 of *Lecture Notes in Computer Science*, chapter Toward a Multi-agent Modelling Approach for Urban Public Transporation systems, pages 160–174. Springer Berlin / Heidelberg, 2001.

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

[10] Balmer, M., B. Raney, and K. Nagel. Adjustment of activity timing and duration in an agent-based traffic flow simulation. In Timmermans, H., editor, *Progress in activity-based analysis*, pages 91–114. Elsevier, Oxford, UK, 2005.

[11] Ben-Akiva, M. and S. R. Lerman. *Discrete choice analysis*. The MIT Press, Cambridge, MA, 1985.

[12] Benenson, I., S. Aronovich, and S. Noam. Let's talk objects: generic methodology for urban high-resolution simulation. *Computers, Environment and Urban Systems*, 29(4):425–453, July 2005.

[13] Bhat, C. R. Analysis of travel mode and departure time choice for urban shopping trips. *Transportation Research Part B: Methodological*, 32(6):361 – 371, 1998.

[14] Brill, J. C. and D. E. Whitney. Development and application of an intermodal mass transit simulation with detailed traffic modeling. In Andradóttir, S., K. J. Healy, D. H. Withers, and B. L. Nelson, editors, *Proceedings of the 1997 Winter Simulation Conference*, pages 1230–1235, 1997.

[15] Cadyts web site. http://transp-or2.epfl.ch/cadyts, accessed 2009.

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

[17] Cetin, N., A. Burri, and K. Nagel. A large-scale agent-based traffic microsimulation based on queue model. In *Proceedings of Swiss Transport Research Conference (STRC)* [1]. See www.strc.ch. Earlier version, with inferior performance values: Transportation Research Board Annual Meeting 2003 paper number 03-4272.

[18] Charypar, D. *Efficient algorithms for the microsimulation of travel behavior in very large scenarios*. PhD thesis, Swiss Federal Institute of Technology (ETH) Zürich, Switzerland, 2008.

[19] Charypar, D., K. W. Axhausen, and K. Nagel. Implementing activity-based models: Accelerating the replanning process of agents using an evolution strategy. In *Proceedings of the meeting of the International Association for Travel Behavior Research (IATBR)* [2]. See www.iatbr.org.

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

[21] Chen, P. P.-S. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[22] Chen, Y., M. Rieser, D. Grether, and K. Nagel. Improving a large-scale agent-based simulation scenario. VSP Working Paper 08-15, VSP, TU Berlin, 2008. See www.vsp.tu-berlin.de/publications.

[23] Choi, K. and W. Jang. Development of a transit network from a street map database with spatial analysis and dynamic segmentation. *Transportation Research Part C*, 8:129–146, 2000.

[24] Ciari, F. Modeling collective taxis in a multi-agent traffic simulation framework. In *Proceedings of the 9th Swiss Transport Research Conference (STRC)* [3].

[25] Ciari, F., M. Balmer, and K. W. Axhausen. A new mode choice model for a multi-agent transport simulation. In *Proceedings of the 8th Swiss Transport Research Conference (STRC)*, Monte Verità, Switzerland, October 2008.

[26] Ciari, F., M. Löchl, and K. W. Axhausen. Location decisions of retailers: an agent-based approach. In *Paper presented at the 15th International Conference on Recent Advances in Retailing and Services Science*, Zagreb, July 2008.

[27] Cortés, C. E., L. Pagès, and R. Jayakrishnan. Microsimulation of flexible transit system designs in realistic urban networks. *Transportation Research Record*, 1923:153–163, 2005.

[28] Costa, A. The organisation of urban public transport systems in western european metropolitan areas. *Transportation Research Part A: Policy and Practice*, 30(5):349–359, 1996.

[29] Daduna, J. R. and J. M. P. Paixão. *Computer-Aided Transit Scheduling*, volume 430 of *Lecture Notes in Economics and Mathematical Systems*, chapter Vehicle Scheduling for Public Mass Transit – An Overview, pages 76–90. Springer Berlin / Heidelberg, 1995.

[30] Dafermos, S. The general multimodal network equilibrium problem with elastic demand. *Networks*, 12(1):57–72, 1982.

[31] De Cea, J. and E. Fernández. Transit assignment for congested public transport systems: An equilibrium model. *Transportation Science*, 27(2):133–147, May 1993.

[32] Dijkstra, E. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[33] Feil, M., M. Balmer, and K. W. Axhausen. New approaches to generating comprehensive all-day activity-travel schedules. Paper 10-1943, Transportation Research Board Annual Meeting, Washington, D.C., 2010.

[34] Fernández, E., J. De Cea, M. Florian, and E. Cabrera. Network equilibrium models with combined modes. *Transportation Science*, 28(3):182–192, August 1994.

[35] Flötteröd, G. *Traffic state estimation with multi-agent simulations*. PhD thesis, 2008.

[36] Flötteröd, G. Cadyts – a free calibration tool for dynamic traffic simulations. In *Proceedings of the 9th Swiss Transport Research Conference (STRC)* [3].

[37] Flötteröd, G., Y. Chen, M. Rieser, and K. Nagel. Behavioral calibration of a large-scale travel behavior microsimulation. In *Proceedings of the meeting of the International Association for Travel Behavior Research (IATBR)*, Jaipur, India, December 2009. See www.iatbr.org.

[38] Friedrich, M. *Schedule-based Modelling of Transportation Networks*, chapter Multi-day dynamic transit assignment, pages 51–61. Volume 46 of Wilson and Nuzzolo [124], 2009.

[39] Friedrich, M., I. Hofsäß, and S. Wekeck. Timetable-based transit assignment using branch & bound. *Transportation Research Record*, 1752:100–107, 2001.

[40] Frisch, U., B. Hasslacher, and Y. Pomeau. Lattice-gas automata for Navier-Stokes equation. *Phys. Rev. Letters*, 56:1505, 1986.

[41] Galus, M. D., R. A. Waraich, M. Balmer, G. Andersson, and K. W. Axhausen. A framework for investigating the impact of PHEVs. In *Paper presented at International Advanced Mobility Forum 2009*, Geneva, March 2009.

[42] Gamma, E., R. Helm, R. Johnson, and J. Vlissides. *Design Pattern: Elements of Reusable Object-Oriented Software*. Addison-Wesley professional computing series. Addison-Wesley, 2001.

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

[44] Gerike, R., T. Gehlert, F. Richter, and W. Schmidt. Think global, act local – the internalisation of external costs in transport. *European Transport / Trasporti Europei*, 38:61–84, 2008.

[45] Goldberg, A. V. and C. Harrelson. Computing the shortest path: A* search meets graph theory. In *SODA '05: Proceedings of the 16th annual ACM-SIAM symposium on Discrete algorithms*, pages 156–165, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[46] Goldberg, D. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[47] Google Earth. http://earth.google.com/. accessed 9/2009.

[48] Grether, D., Y. Chen, M. Rieser, and K. Nagel. *Complexity and Spatial Networks – In Search of Simplicity*, chapter Effects of a Simple Mode Choice Model in a Large-Scale Agent-Based Transport Simulation, pages 167–186. Advances in Spatial Science. Springer Berlin Heidelberg, 2009.

[49] HaCon. Hafas. http://www.hacon.de/hafas/. accessed 9/2009.

[50] Hamdouch, Y. and S. Lawphongpanich. Schedule-based transit assignment model with travel strategies and capacity constraints. *Transportation Research Part B*, 42(7–8):663–684, August 2007.

[51] Henriksen, J. O. An introduction to SLX. In *WSC '96: Proceedings of the 28th conference on Winter simulation*, pages 468–475, Washington, DC, USA, 1996. IEEE Computer Society.

[52] Horni, A., D. M. Scott, M. Balmer, and K. W. Axhausen. Location choice modeling for shopping and leisure activities with MATSim: Combining micro-simulation and time geography. *Transportation Research Record*, forthcoming.

[53] IVU Traffic Technologies AG. http://www.ivu.de/. accessed 9/2009.

[54] Jakovljević, G. and D. Basch. Implementing multiscale traffic simulators using agents. In *26th International Conference on Information Technology Interfaces*, pages 519–524, June 2004.

[55] Jara-Díaz, S. and R. Guerra. Modeling activity duration and travel choice from a common microeconomic framework. In *Proceedings of the meeting of the International Association for Travel Behavior Research (IATBR)*, Lucerne, Switzerland, 2003. See www.ivt.baug.ethz.ch.

[56] Java www page. http://java.sun.com, accessed 2009.

[57] Jayakrishnan, R., C. E. Cortés, R. Lavanya, and L. Pagès. Simulation of urban transportation networks with multiple vehicle classes and services: Classifications, functional requirements and general-purpose modeling schemes. Paper, Transportation Research Board Annual Meeting, Washington, D.C., 2003.

[58] Joubert, J. W. personal communication.

[59] Kickhöfer, B. Die Methodik der ökonomischen Bewertung von Verkehrsmaßnahmen in Multiagentensimulationen. Master's thesis, TU Berlin, 2009.

[60] Kitamura, R., A. Kikuchi, and S. Fujii. *Simulation Approaches in Transportation Analysis*, volume 31 of *Operations Research/Computer Science Interfaces Series*, chapter An Overview of PCATS/DEBNetS Micro-simulation System: Its Development, Extension, and Application to Demand Forecasting, pages 371–399. Springer US, 2005.

[61] Koutsopoulos, H. N. and Z. Wang. Simulation of urban rail operations: Application framework. *Transportation Research Record*, 2006:84–91, 2007.

[62] Lämmel, G., D. Grether, and K. Nagel. The representation and implementation of time-dependent inundation in large-scale microscopic evacuation simulations. *Transportation Research Part C: Emerging Technologies*, 18(1):84–98, February 2010.

[63] Lefebvre, N. and M. Balmer. Fast shortest path computation in time-dependent traffic networks. In *Proceedings of Swiss Transport Research Conference (STRC)* [1]. See www.strc.ch.

[64] Lo, H. K., C. Yip, and K. Wan. Modeling transfer and non-linear fare structure in multi-modal network. *Transportation Research Part B*, 37:149–170, 2003.

[65] Maier, D. Routing durch das VBZ und SBB Netzwerk in der Region Zürich. term project, ETH Zurich, 2003.

[66] MATSIM www page. MultiAgent Transport SIMulation. http://matsim.org/, accessed 2009.

[67] Meister, K., M. Balmer, K. Axhausen, and K. Nagel. planomat: A comprehensive scheduler for a large-scale multi-agent transportation simulation. In *Proceedings of the meeting of the International Association for Travel Behavior Research (IATBR)* [2]. See www.iatbr.org.

[68] Metropolis, N., A. Rosembluth, M. Rosembluth, and A. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[69] Miller, E. and M. Roorda. A prototype model of household activity/travel scheduling. *Transportation Research Record*, 1831:114–121, 2003.

[70] Miller, J. A., R. S. Nair, Z. Zhang, and H. Zhao. JSIM: A Java-based simulation and animation environment. In *Proceedings of the 30th Annual Simulation Symposium*, pages 31–42, 1997.

[71] Modesti, P. and A. Sciomachen. A utility measure for finding multiobjective shortest paths in urban multimodal transportation networks. *European Journal of Operational Research*, 111:495–508, 1998.

[72] Multisystems, Inc. Bus rapid transit simulation model: Research and development. Report, USDOT/SBIR Phase 1, Oct. 2000.

[73] Nagel, K. Multi-modal traffic in TRANSIMS. In Schreckenberg, M. and S. D. Sharma, editors, *Pedestrian and Evacation Dynamics*, Proceedings of the 1st international conference, Duisburg, 2001, pages 161–172. Springer, 2001.

[74] Nagel, K. and M. Schreckenberg. A cellular automaton model for freeway traffic. *Journal de Physique I France*, 2:2221–2229, 1992.

[75] Nash, A. and D. Huerlimann. Railroad simulation using OpenTrack. *Computers in Railways IX*, 2004.

[76] Nash, A., U. Weidmann, S. Bollinger, M. Lüthi, and S. Buchmüller. Increasing schedule reliability on the S-Bahn in Zurich, Switzerland: Computer analysis and simulation. *Transportation Research Record*, 1955:16–25, 2006.

[77] Neumann, A. Modellierung und evaluation von lichtsignalanlagen in queue-simulationen. diploma thesis, TU Berlin, Institute for Land and Sea Transport Systems, Berlin, Germany, 2008.

[78] Newell, G. F. Control of pairing of vehicles on a public transportation route, two vehicles, one control point. *Transportation Science*, 8(3):248, 1974.

[79] Nguyen, S., S. Pallottino, and F. Malucelli. A modeling framework for passenger assignment on a transport network with timetables. *Transportation Science*, 35(3):238–249, August 2001.

[80] Nielsen, O. and R. Frederiksen. Optimisation of timetable-based, stochastic transit assignment models based on MSA. *Annals of Operations Research*, 144:263–285, 2006.

[81] Nielsen, O. A. and R. D. Frederiksen. *Schedule-based Modelling of Transportation Networks*, chapter Large-scale schedule-based transit assignment – further optimization of the solution algorithms, pages 1–26. Volume 46 of Wilson and Nuzzolo [124], 2009.

[82] Note, I. M. Integration of a center-running guided busway into an arterial street. Presented at VISSIM User Group Meeting, Seattle, 2001.

[83] Nuzzolo, A. and U. Crisalli. *Schedule-based Modelling of Transportation Networks*, chapter The Schedule-Based Modelling of Transportation Systems: Recent Developments, pages 1–26. Volume 46 of Wilson and Nuzzolo [124], 2009.

[84] O'Loan, O., M. Evans, and M. Cates. Jamming transition in a homogeneous one-dimensional system: The bus route model. *Phys. Rev. E*, 58:1404–1418, 1998.

[85] Omnitrans International. OmniTRANS transport software. http://www.omnitrans.nl/. accessed 2/2010.

[86] OpenStreetMap. The free wiki world map. http://www.openstreetmap.org/. accessed 2009.

[87] Pallottino, S. and M. Scutella. Shortest path algorithms in transportation models: Classical and innovative aspects. In Marcotte, P. and S. Nguyen, editors, *Equillibrium and Adavanced Transportation Modelling*, pages 245–281. Kluwer, 1998.

[88] Papola, N., F. Filippi, G. Gentile, and L. Meschini. *Schedule-based Modelling of Transportation Networks*, chapter Schedule-Based Transit Assignment: New dynamic equilibrium model with cehicle capacity constraints, pages 145–170. Volume 46 of Wilson and Nuzzolo [124], 2009.

[89] Poon, M., S. Wong, and C. Tong. A dynamic schedule-based model for congested transit networks. *Transportation Research Part B*, 38(4):343–368, May 2004.

[90] Powell, W. B. and Y. Sheffi. A probabilistic model of bus route performance. *Transportation Science*, 17(4):376–404, 1983.

[91] PTV AG. *VISUM 10.0 Benutzerhandbuch*, 2007.

[92] Pugh, W. Java memory model. http://www.cs.umd.edu/ pugh/java/memoryModel/. accessed 9/2009.

[93] Quatrani, T. *Visual Modeling with Rational Rose and UML*. Addison-Wesley, 1998.

[94] Raney, B. and K. Nagel. An improved framework for large-scale multi-agent simulations of travel behaviour. In Rietveld, P., B. Jourquin, and K. Westin, editors, *Towards better performing European Transportation Systems*, page 42. Routledge, London, 2006.

[95] Raney, B. K. *Learning Framework for Large-Scale Multi-Agent Simulations*. PhD thesis, Swiss Federal Institute of Technology (ETH) Zürich, Switzerland, 2005.

[96] Reinhold, T. Konzept zur integrierten Optimierung des Berliner Nahverkehrs. In *Öffentlicher Personennahverkehr*. Springer Berlin Heidelberg, 2006.

[97] Repast 3. Recursive porous agent simulation toolkit. http://repast.sourceforge.net/, accessed 2009.

[98] Rieser, M., U. Beuck, M. Balmer, and K. Nagel. Modelling and simulation of a morning reaction to an evening toll. In *Innovations in Travel Modeling (ITM) '08*, Portland, Oregon, June 2008.

[99] Rieser, M., D. Grether, and K. Nagel. Adding mode choice to a multi-agent transport simulation. Paper 09-2758, Transportation Research Board Annual Meeting, Washington, D.C., 2009.

[100] Rieser, M., K. Nagel, U. Beuck, M. Balmer, and J. Rümenapp. Truly agent-oriented coupling of an activity-based demand generation with a multi-agent traffic simulation. *Transportation Research Record*, 2021:10–17, 2007.

[101] Riley, G. F. The Georgia Tech network simulator. In *MoMeTools '03: Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, pages 5–12, New York, NY, USA, 2003. ACM.

[102] Russel, S. and P. Norvig. *Artificial intelligence: A modern approach*. Series in Artificial Intelligence. Prentice Hall, 1995.

[103] SFSO. Eidgenössische Volkszählung. Swiss Federal Statistical Office, Neuchatel, 2000.

[104] SFSO. Ergebnisse des Mikrozensus 2005 zum Verkehr. Swiss Federal Statistical Office, Neuchatel, 2006.

[105] Shalaby, A., B. Abdulhai, and J. Lee. Assessment of streetcar transit priority options using microsimulation modelling. *Canadian Journal of Civil Engineering*, 30(6):1000–1009, 2003.

[106] Shiftan, Y. and J. Suhrbier. The analysis of travel and emission impacts of travel demand management strategies using activity-based models. *Transportation*, 29(2):145–168, May 2002.

[107] Smith, L., R. Beckman, D. Anson, K. Nagel, and M. Williams. TRANSIMS: TRansportation ANalysis and SIMulation System. In *Proc. 5th Nat. Transportation Planning Methods Applications Conference*, Seattle, WA, 1995.

[108] Southworth, F. and B. E. Peterson. Intermodal and international freight network modeling. *Transportation Research Part C*, 8:147–166, 2000.

[109] Sun Microsystems. Compressed oops in the Hotspot JVM. http://wikis.sun.com/display/HotSpotInternals/CompressedOops. accessed 10/2009.

[110] Sutter, H. The free lunch is over: A fundamental turn toward concurrency in software. *Dr. Dobb's Journal*, March 2005.

[111] Titze, T. Entwicklung eines ÖV-Routingmoduls für Multiagentensimulationen. Term project report, TU Berlin, 2007.

[112] TRANSIMS Open Source Project. http://www.transims-opensource.net/. accessed 9/2009.

[113] Trépanier, M. and R. Chapleau. Linking transit operational data to road network with a transportation object-oriented GIS. *Journal of the Urban and Regional Information Systems Association*, 13(2):23–30, 2001.

[114] Verband Deutscher Verkehrsunternehmen. ÖPNV data model 5.0, File format for data transfer between ÖPNV applications. VDV Schriften 451, Köln, 1999.

[115] Verband Deutscher Verkehrsunternehmen. ÖPNV data model 5.0, VDV standard interface network / timetable applications. VDV Schriften 452, Köln, 1999.

[116] Verkehrsverbund Berlin-Brandenburg. http://www.vbbonline.de/. accessed 2009.

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

[118] Vovsha, P. and M. Bradley. Advanced activity-based models in context of planning decisions. *Transportation Research Record*, 1981:34–41, 2006.

[119] Vrtic, M., P. Fröhlich, N. Schüssler, K. Axhausen, P. Kern, F. Perret, S. Pfisterer, C. Schulze, A. Zimmerman, and U. Heidl. Verkehrsmodell für den öffentlichen Verkehr des Kantons Zürich. Report to the Amt für Verkehr, Kanton Zürich, IVT, Zürich, Ernst Basler + Partner AG, Zürich and PTV, Karlsruhe, 2005.

[120] Vrtic, M., N. Schüssler, A. Erath, K. Meister, and K. Axhausen. Tageszeitliche Fahrtenmatrizen im Personenverkehr an Werktagen im Jahr

2000. Research report, Swiss Federal Department for Environment, Transport, Energy and Communication, Swiss Federal Office for Spatial Development, Swiss Federal Roads Authority and Swiss Federal Office for Transport, IVT, ETH Zürich, Zürich, 2007.

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

[122] Wahba, M. and A. Shalaby. *Schedule-based Modelling of Transportation Networks*, chapter MILATRAS: A new modelling framework for the Transit Assignment Problem, pages 171–194. Volume 46 of Wilson and Nuzzolo [124], 2009.

[123] Wiedemann, R. Simulation des Strassenverkehrsflusses. Schriftenreihe Heft 8, Institute for Transportation Science, University of Karlsruhe, Germany, 1974.

[124] Wilson, N. H. M. and A. Nuzzolo, editors. *Schedule-based Modelling of Transportation Networks*, volume 46 of *Operations Research / Computer Science Interfaces Series*. Springer US, 2009.

[125] Yoginath, S. B. and K. S. Perumalla. Parallel vehicular traffic simulation using reverse computation-based optimistic execution. In *Principles of Advanced and Distributed Simulation, 2008. PADS '08. 22nd Workshop on*, pages 33–42, June 2008.

[126] Zürcher Verkehrs-Verbund. http://www.zvv.ch/. accessed 9/2009.

Appendices

A

Files in MATSim

Table A.1 shows an overview of the currently standardized file formats that MATSim most commonly uses. The table includes the new files used for simulating transit.

Table A.1: Common standardized input data files for MATSim

Input File	Description
network.xml	description of road and rail network
population.xml	describes the population and one or more day-plans per agent
facilities.xml	description of facilities where activities can be performed e.g. homes, work places, shops, leisure facilities (e.g. cinemas, sports grounds), and so on
counts.xml	values from traffic count stations to validate road traffic
world.xml	definition of zones and regions, mostly for initial demand modeling
vehicles.xml	defines vehicle types and vehicles use by the agent for traveling around in the network. Currently only used for specifying transit vehicles.
transitSchedule.xml	list of the available transit lines, their routes and departures

B Transit Schedule

This appendix contains detailed information about and examples how transit schedules are stored in MATSim.

B.1 transitSchedule_v1.dtd

The following document type definition (DTD) specifies the XML grammar used to store transit schedules in MATSim.

```
<?xml version="1.0" encoding="utf-8"?>

<!ELEMENT transitSchedule  (transitStops?,transitLine*)>

<!ELEMENT transitStops     (stopFacility)*>

<!ELEMENT stopFacility     EMPTY>
<!ATTLIST stopFacility
          id               CDATA   #REQUIRED
          x                CDATA   #REQUIRED
          y                CDATA   #REQUIRED
          linkRefId        CDATA   #IMPLIED
          name             CDATA   #IMPLIED
          isBlocking       (true|false)  "false">

<!ELEMENT transitLine      (transitRoute)*>
<!ATTLIST transitLine
          id               CDATA   #REQUIRED>

<!ELEMENT transitRoute     (description?,transportMode,routeProfile,route?,departures)
      >
<!ATTLIST transitRoute
          id               CDATA   #REQUIRED>
<!-- transitRoute.id must be unique within a transitLine only. -->
```

```
<!ELEMENT description      (#PCDATA)>

<!ELEMENT transportMode    (#PCDATA)>
<!-- describes the mode of which vehicles are that serve that line/route -->

<!ELEMENT routeProfile     (stop)*>

<!ELEMENT stop             EMPTY>
<!ATTLIST stop
          refId            CDATA   #REQUIRED
          departureOffset  CDATA   #IMPLIED
          arrivalOffset    CDATA   #IMPLIED
          awaitDeparture   (true|false)  "false">
<!--
 * stop.departureOffset is required for all stops but the last.
   The offsets are to be added to the departure times of the single
   departures listed in the transitRoute.
 * stop.refId are id-references to facilities. -->

<!ELEMENT route            (link)*>

<!ELEMENT link             EMPTY>
<!ATTLIST link
          refId            CDATA   #REQUIRED>

<!ELEMENT departures       (departure)*>
<!-- the single departures along that transitRoute -->

<!ELEMENT departure        EMPTY>
<!ATTLIST departure
          id               CDATA   #REQUIRED
          departureTime    CDATA   #REQUIRED
          vehicleRefId     CDATA   #IMPLIED>
<!-- the departure.id must be unique within a transitLine only -->
```

B.2 Example

The listing below shows a very simple example, how a single bus line is specified. Fig. B.1 shows the network model the example refers to and also depicts the locations of the stop facilities.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE transitSchedule SYSTEM "http://matsim.org/files/dtd/transitSchedule_v1.dtd">

<transitSchedule>

  <transitStops>
```

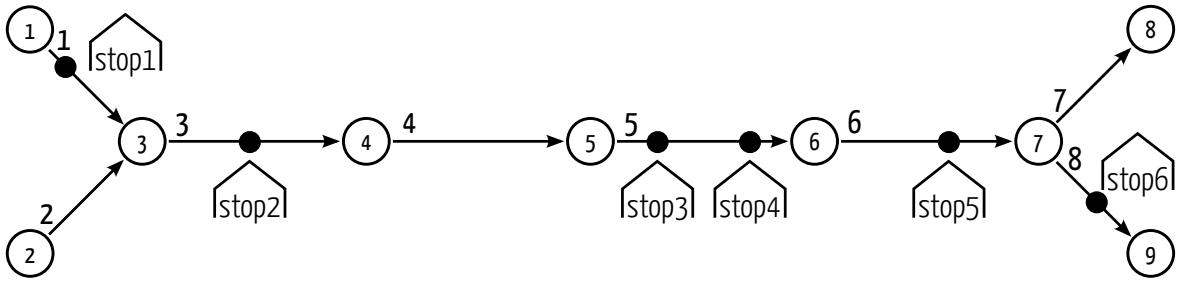


Figure B.1: Very simple example of a road network and location of bus stops.

```
    <stopFacility id="stop1" x="100.0" y="900.0" linkRefId="1" />
    <stopFacility id="stop2" x="1000.0" y="500.0" linkRefId="3" />
    <stopFacility id="stop3" x="2800.0" y="500.0" linkRefId="5" />
    <stopFacility id="stop4" x="3300.0" y="500.0" linkRefId="5" />
    <stopFacility id="stop5" x="4100.0" y="499.0" linkRefId="6" />
    <stopFacility id="stop6" x="4800.0" y="200.0" linkRefId="8" />
  </transitStops>

  <transitLine id="T1">
    <transitRoute id="1">
      <description></description>
      <transportMode>bus</transportMode>
      <routeProfile>
        <stop refId="stop1" departureOffset="00:00:00" />
        <stop refId="stop2" arrivalOffset="00:01:20" departureOffset="00:01:40" />
        <stop refId="stop3" departureOffset="00:03:00" />
        <stop refId="stop4" departureOffset="00:03:50" />
        <stop refId="stop5" departureOffset="00:04:50" />
        <stop refId="stop6" arrivalOffset="00:05:30" />
      </routeProfile>

      <route>
        <link refId="1" />
        <link refId="3" />
        <link refId="4" />
        <link refId="5" />
        <link refId="6" />
        <link refId="8" />
      </route>

      <departures>
        <departure id="T1.1" departureTime="07:00:00" />
        <departure id="T1.2" departureTime="07:10:00" />
        <departure id="T1.3" departureTime="07:20:00" />
      </departures>
    </transitRoute>

  </transitLine>

</transitSchedule>
```

C

Maps

C.1 Switzerland

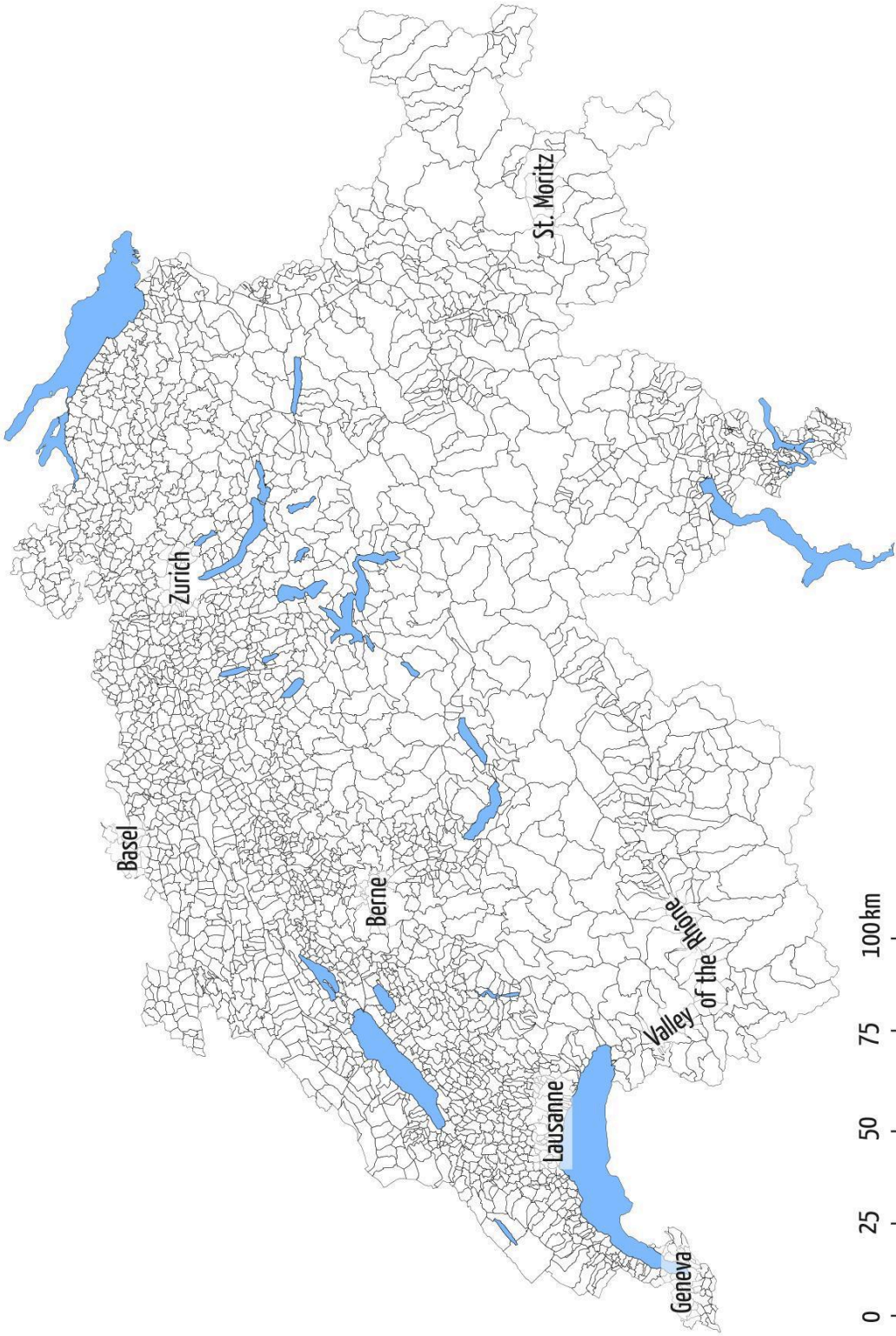


Figure C.1: Map of Switzerland

C.2 Area of Zurich

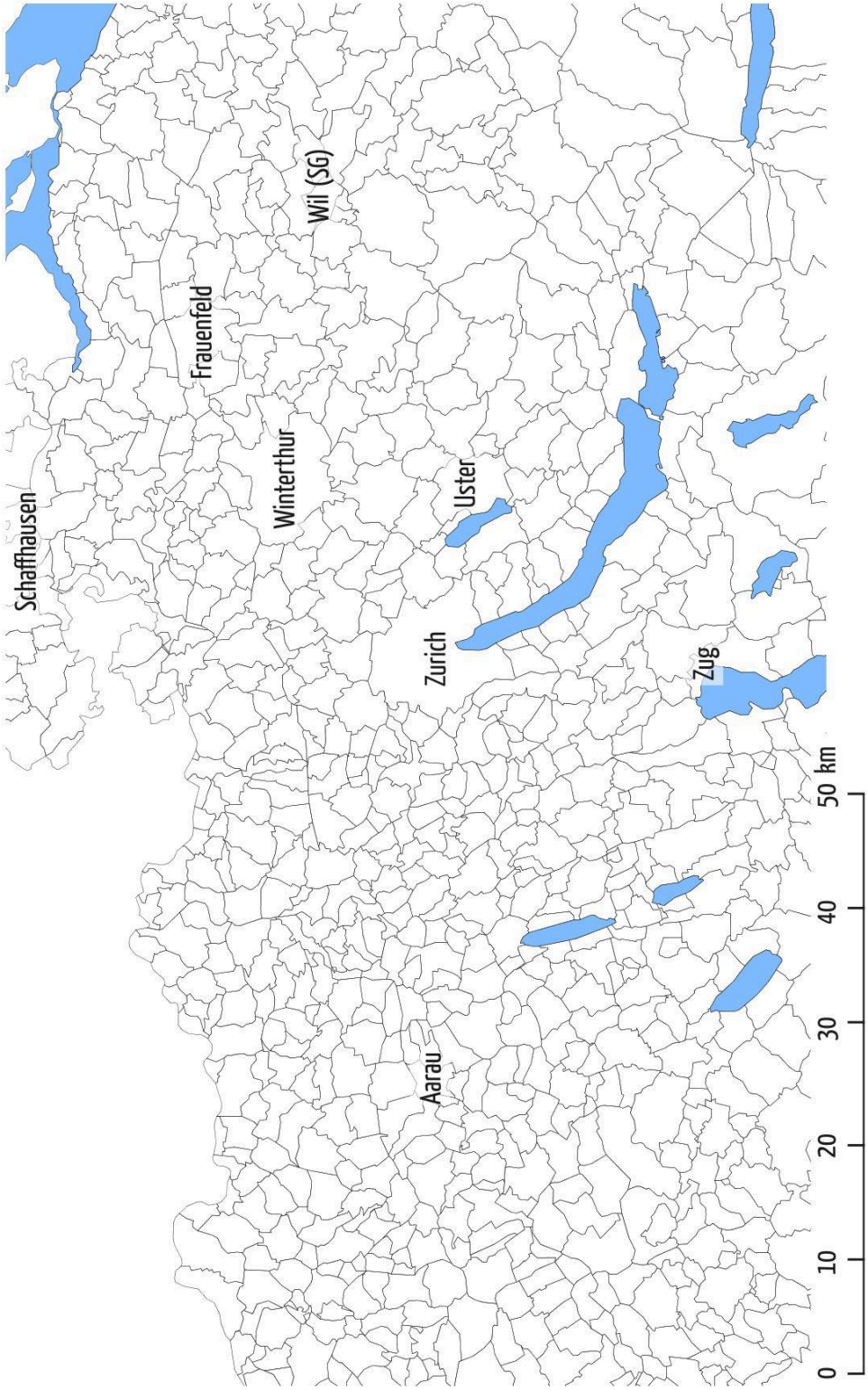


Figure C.2: Area of Zurich

C.3 South of City of Zurich

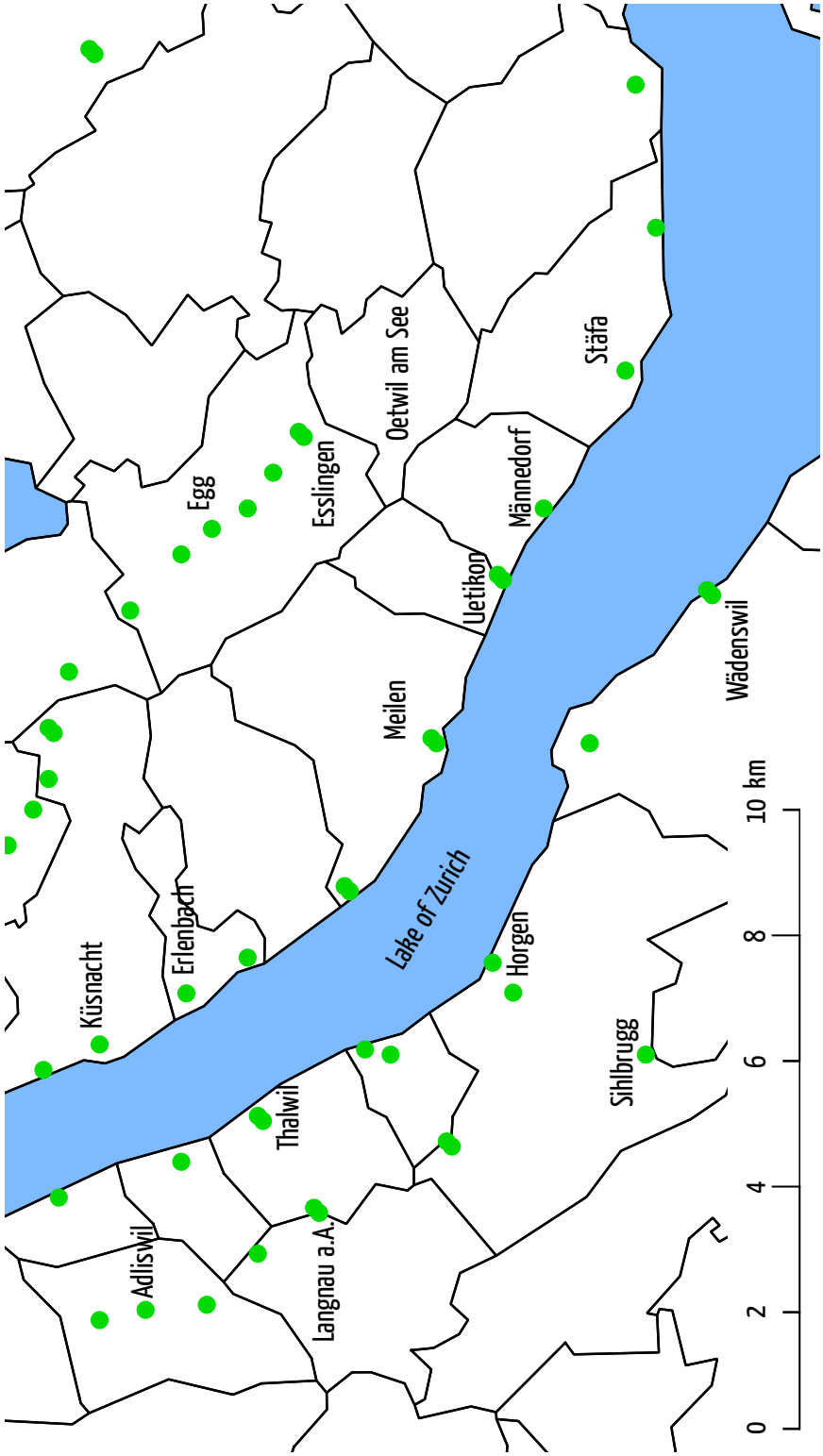


Figure C.3: South of Zurich, including stop locations of trains. Labels refer to the name of the stop locations.

D

ZVV Maps

The following two figures are provided by *Zürcher Verkehrsverbund – ZVV* [126] and are reprinted with written permission. Around 50 transport companies operating in the region of Zurich are united in ZVV, sharing the same fare system and trying to coordinate and optimize the services for the transit users. The first figure shows the train lines in the whole area covered by ZVV, while the second figure shows the tram and bus lines in the city of Zurich.

D.1 Complete Area of Operations

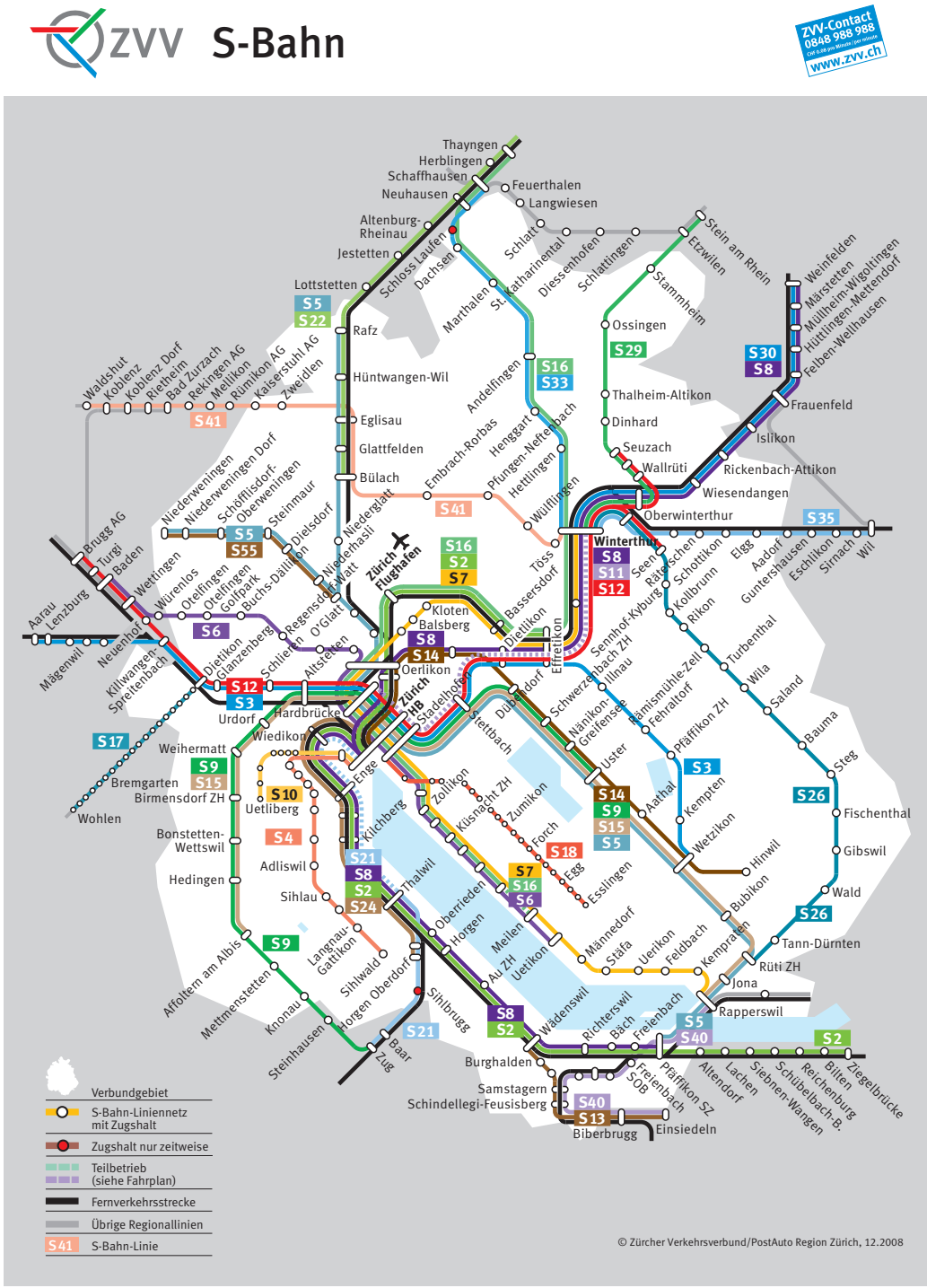


Figure D.1: Train lines in the area of Zurich

D.2 City of Zurich

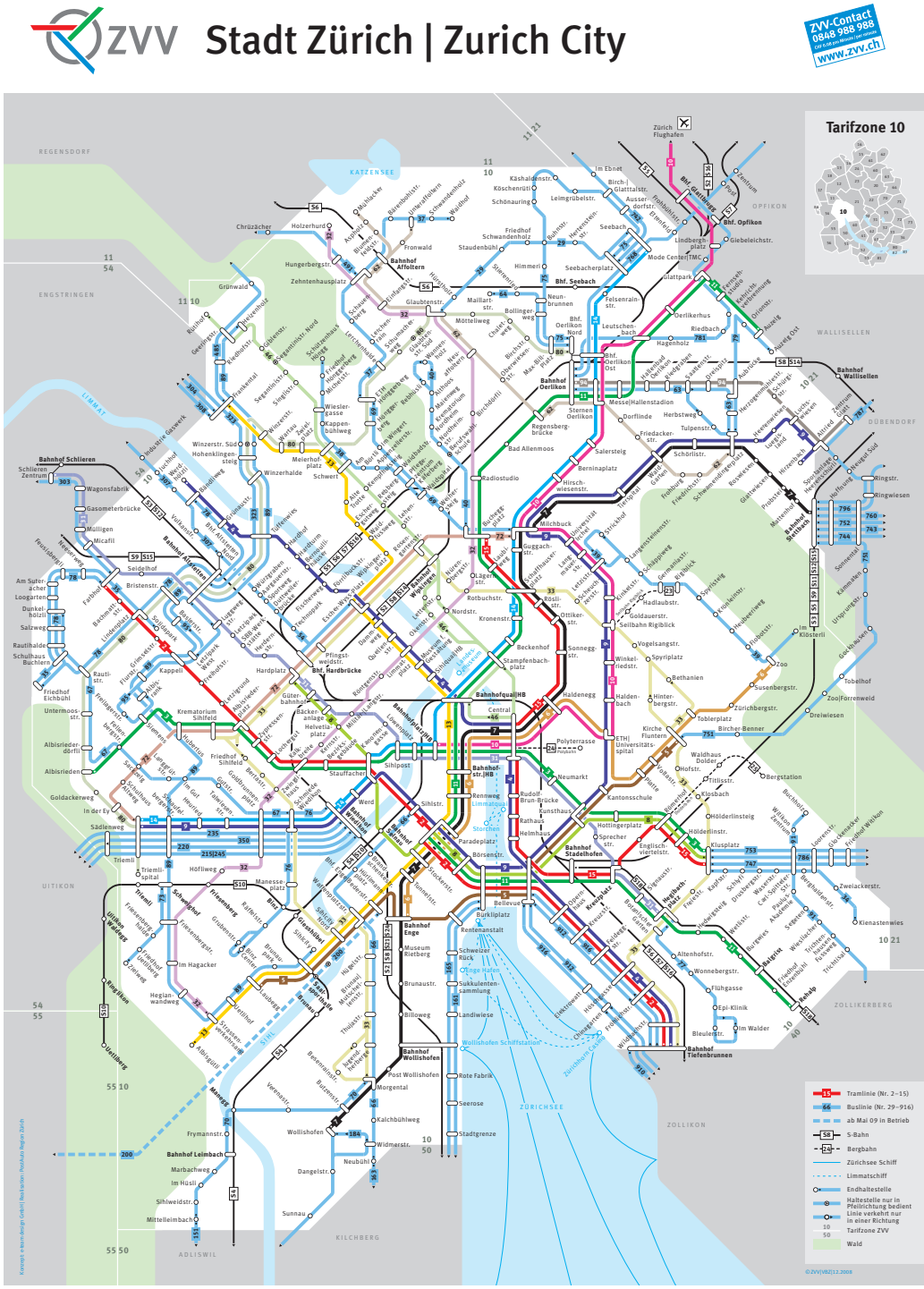


Figure D.2: Transit lines in the city of Zurich