Simulation-Based Traffic Assignment

Computing User Equilibria in Large Street Networks

Inaugural-Dissertation
zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultät
der Universität zu Köln

vorgelegt von
Christian Gawron
aus Köln

Köln 1998

Berichterstatter: Prof. Dr. R. Schrader
PD Dr. E. Dahlhaus

Tag der mündlichenPrüfung: 8. Februar 1999

Contents

1 Introduction 1

2 Traffic Assignment 5
- 2.1 Introduction 5
  - 2.1.1 Route Choice Models and Equilibria 6
  - 2.1.2 Static and Dynamic Assignment 6
- 2.2 Static Assignment 7
  - 2.2.1 Network Representation and Notation 7
  - 2.2.2 Problem Formulation: User Equilibrium 8
  - 2.2.3 Problem Formulation: System Optimum 10
  - 2.2.4 Uniqueness of the Solution 12
  - 2.2.5 Link Cost Functions 13
  - 2.2.6 User Equilibrium, System Optimum and Braess's Paradox 16
  - 2.2.7 Solution Techniques 19
- 2.3 Dynamic Assignment 22
  - 2.3.1 Network Model 22
  - 2.3.2 Dynamic User Equilibrium Conditions and Variational Inequality Formulation 25
  - 2.3.3 Formulation as Variational Inequality Problem 27
  - 2.3.4 Relaxation Procedure 27
  - 2.3.5 Solution Method 29
  - 2.3.6 Complexity 29

3 Traffic Simulation Models 31
- 3.1 Introduction 31
- 3.2 Macroscopic Models 31
- 3.3 Microscopic Models 32
  - 3.3.1 Car-Following Models 32
  - 3.3.2 Mesoscopic Models 35
- 3.4 FASTLANE: A Queuing Model 36
  - 3.4.1 Model Description 36
  - 3.4.2 Implementation 38
  - 3.4.3 Calibration and Validation of the link model 38
  - 3.4.4 Computational Performance 41

4 Simulation-based Traffic Assignment 45
- 4.1 Probabilistic Modeling of Route Choice 46
  - 4.1.1 Introduction 46

4.1.2 Traveler Model 46
4.1.3 Update Rules 47
4.2 Stability Analysis 49
4.2.1 Convergence 53
4.2.2 Numerical example 53

5 Applications 57
5.1 A Dynamic Version of Braess's Paradox 57
5.2 The Highway Network of Nordrhein-Westfalen 61
5.3 Wuppertal 66

6 Summary and Outlook 71

A Mathematical Prerequisites 73
A.1 First-order Optimality Conditions with Inequality Constraints 73
A.1.1 Kuhn-Tucker Conditions 73
A.1.2 Non-negativity and Affine Constraints 76
A.2 The Frank-Wolfe Algorithm 77

B Details on PLANSIM-T and FASTLANE 81
B.1 The PLANSIM-Traffic Simulator 81
B.2 FASTLANE 82
B.2.1 Input Formats 83
B.2.2 The Driver File 84
B.2.3 The Route File 85
B.2.4 The Programs and their Output Formats 86

C Networks 91
C.1 The Network of 'Seestrauch' 91
C.2 The Highway Network of the Landschaftsverband Rheinland 91
C.3 The Network of Wuppertal 91

D Glossary of Notation 95

Bibliography 97

Deutsche Zusammenfassung 105

Danksagung 109

List of Figures

1.1 ‘Flowchart’ of the FVU 2

2.1 Non-uniqueness of path flows 13
2.2 Fundamental diagram (ρ–q) 14
2.3 Fundamental diagram (ρ–v) 15
2.4 Fundamental diagram (q–v) 15
2.5 Link cost functions 17
2.6 Braess’s paradox 18
2.7 User equilibrium travel times in Braess’s paradox 18
2.8 Convergence of the Frank-Wolfe algorithm 20
2.9 Auxiliary network for the linear subproblem 30

3.1 Update algorithm of FASTLANE 37
3.2 Comparison of queuing model and CA 40
3.3 Comparison of queuing model and the Krauß model 40
3.4 Comparison of the flows in the queuing model and the Krauß model . . . 41
3.5 Comparison of travel time in PLANSIM-T and FASTLANE 42
3.6 Comparison of travel time in PLANSIM-T and FASTLANE 42
3.7 Influence of the link length on the computing time. 43
3.8 Influence of the time step size on the computing time. 44

4.1 ‘Learning function’ f (4.5) 48
4.2 Consequences of a perturbation of the travel times 50
4.3 Oscillating route choice probabilities 52
4.4 Test of Stability 55
4.5 Test of Convergence 55

5.1 Dynamic version of Braess’s paradox 58
5.2 Flows in the different versions of Braess’s network 59
5.3 Travel times in the different versions of Braess’s network 60
5.4 Travel time differences between network 1 and the other networks. 60
5.5 Comparison between static and dynamic assignment for the highway network of NRW 62
5.6 Average daily traffic flows for the NRW highway network 63
5.7 Comparison between static and dynamic assignment 64

5.8 Average number of routes per OD-relation generated by the static assignment for the NRW highway network 64
5.9 Distribution of the trip lengths for the NRW highway network 65
5.10 Number of alternative routes generated by the DTA algorithm for trips in the NRW highway network 65
5.11 Convergence rate of the Frank-Wolfe algorithm for Wuppertal 67
5.12 Convergence of the simulation-based DTA algorithm for the Wuppertal network 67
5.13 Average daily traffic in Wuppertal 68
5.14 NO_x emission in Wuppertal computed with PLANSIM-T 69

A.1 Kuhn-Tucker conditions 75
A.2 Frank-Wolfe algorithm 78

B.1 An example of a network in PLANSIM-T 82
B.2 Flow chart of the DTA algorithm 83
B.3 Compression of routes 85
B.4 Example graph files 85
B.5 An example `.odm` file for the Braess network 86
B.6 An example `.rinfo` file for the Braess network 86

C.1 The ‘Seestrauch’ network 92
C.2 The Wuppertal network 93

Chapter One

Introduction

Of all federal states of Germany, Nordrhein-Westfalen (NRW) has the highest population and harbors one of Germany's major conurbations, the Ruhrgebiet. Both the high population density and the high concentration of heavy industry result in a high demand for transportation of goods and people. Further, with Europe getting a single market, the amount of transit traffic through NRW has increased a lot due to NRW's situation at the western border of Germany.

For these reasons, it is not surprising that NRW has a lot of problems related to traffic. From 1980 to 1992, the amount of goods carried on the roads of NRW has increased by 50% (c.f. [63]). During the same time, the passenger kilometers traveled on NRW's roads increased by about 15%. Already today the growing demand for transportation often exceeds the capacity of the transportation networks, and the prognoses ([63], [13]) indicate that the demand for transportation of both goods and people will rise further in the next years.

Due to a growing environmental awareness and the large density of population in NRW, the increased demand for transportation cannot be handled by simply building new infrastructure. Further, people have become less willing to accept the negative impacts of traffic, like noise, pollution, and accidents. Therefore, the political decision makers are looking for new strategies to manage the increased demand for transportation.

Traffic networks are complex, and Braess's paradox (see sections 2.2.6 and 5.1) is only one example of how the results of a simple measure — like building a new road — might be counter-intuitive and, still worse, counter-productive. In the last years, faster computers and improved traffic simulation models have made real-time simulations of large networks, like the whole German freeway network, feasible. Such simulations might help decision makers to assess the effects of different available options. However, until now such fast traffic simulation models running on high performance computers are not used in 'real world' applications.

To support the development of new traffic simulation models and to promote the application of these models, in 1995 the Ministerium für Wissenschaft und Forschung[1] of NRW founded the 'Forschungsverbund Verkehrssimulation und Umweltwirkungen' (FVU)[2], consisting of twelve institutes working on different aspects of the simulation of

---

[1] Ministry of Science and Research

[2] Research Cooperative for the Simulation of Traffic and Environmental Impacts

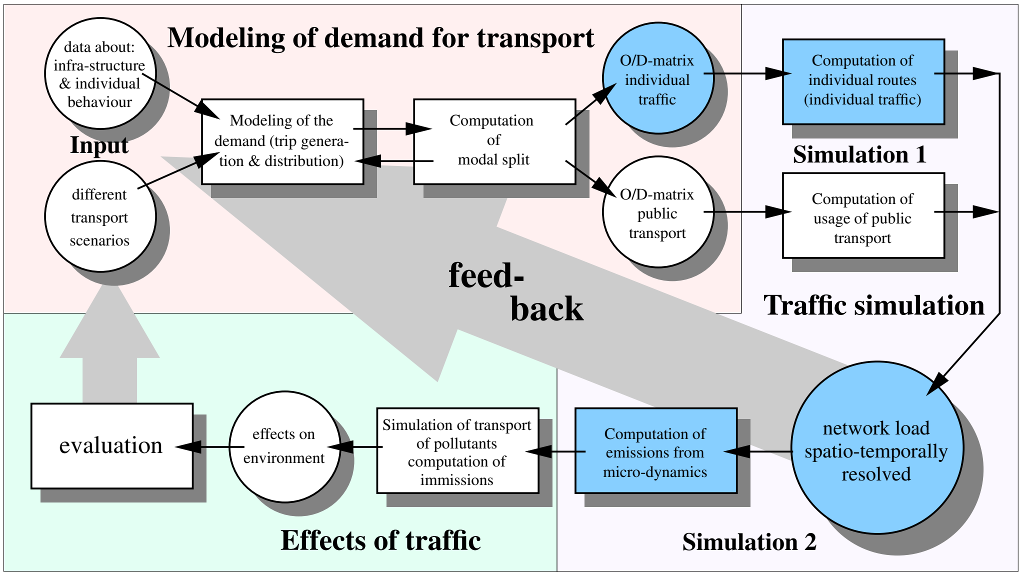

Modeling of demand for transport
data about: infra-structure & individual behaviour
Input
different transport scenarios
Modeling of the demand (trip generation & distribution)
Computation of modal split
O/D-matrix individual traffic
O/D-matrix public transport
Computation of individual routes (individual traffic)
Simulation 1
Computation of usage of public transport
Traffic simulation
feed-back
network load spatio-temporally resolved
Computation of emissions from micro-dynamics
Simulation 2
Simulation of transport of pollutants computation of immissions
effects on environment
evaluation
Effects of traffic

Figure 1.1 ‘Flowchart’ of the FVU. Figure taken from [2] by courtesy of P. Wagner. Tasks to be performed by the traffic simulation group at the ZPR are shaded in grey.

transportation systems, ranging from macro-economical demand models to models of the environmental impacts of traffic. A schematic overview of the FVU is given in Figure 1.1.

Our group at the Center for Parallel Computing (ZPR) of the University of Cologne — and later also at the German Aerospace Center — participated in the FVU from the beginning, contributing the traffic simulation models which were the result of the work of K. Nagel [65, 69], M. Rickert [78], and S. Krauß [51]. Using these models, the task of the ZPR was to perform microscopic, i.e. vehicle oriented, simulations of traffic in large street networks and to compute traffic flows, travel times and emissions with a high time-resolution.

To do these microscopic simulations, detailed input data are needed: the trips through the network, i.e. a set of drivers together with their departure times and their routes. To provide these input data, a model for the demand for transportation and a model for the route choice behavior is needed. Modeling the demand for transportation with a high time-resolution is a complex task, and a major part of the FVU (see figure 1.1) was concerned with providing these data.

The modeling of route choice, however, cannot be done independently from the simulation since route choice depends on the travel times, which are a result of the simulation. Therefore, our group not only had to do the simulation, but also had to model route choice. The accepted model for route choice is Wardrop’s first principle [89]: Every driver chooses a route for which the cost (usually the travel time) is minimal. The state in which every route choice satisfies this condition is called the *user equilibrium*.

The demand for transportation is usually provided by means of an *origin-destination* *matrix* (ODM) which contains the demand, i.e. the number of trips per time unit, for each pair of origins and destinations. The problem of calculating the user equilibrium route choices is called the *traffic assignment problem*, which we will discuss in chapter 2. It is still common practice to simplify this problem by neglecting the time-dependence of the ODM and solve a static traffic assignment problem assuming some time-independent relationship between traffic flow and travel time. We will see in chapter 2 that this simplification drastically reduces the complexity of the traffic assignment problem.

In reality, however, the ODM is time dependent. During the rush hours the demand often exceeds the capacity of a road network leading to the buildup of traffic jams. Thus, the travel time depends on the history of the system, i.e. the current lengths of the traffic jams. Since the aim of the FVU is to provide a simulation model with a high time-resolution, this time-dependence cannot be neglected when calculating the route choices.

Several authors have proposed models for the dynamic user equilibrium problem based on mathematical programming methods (see section 2.3). The task of solving these models is referred to as *dynamic traffic assignment*. However, as we will see later, even using state-of-the-art computers these models cannot be solved for large networks with several thousand nodes and OD-pairs like the network of Wuppertal (see appendix C.3), which was the main object of study by the FVU.

A pragmatic way to find the time-dependent user equilibrium is by an iterative simulation: Choose some initial routes assuming zero traffic. Now calculate the network load and the travel times by simulation and update the route choices of the drivers. Iterate this process until the travel times are stationary, i.e. a fixed point of the iteration is found. As we will discuss in chapter 4, this approach has several problems. For example, it is easy to construct situations where this simple approach does not converge and the equilibrium is unstable. One major objective of this thesis is to develop a modification of this approach which can be shown to be stable empirically and — in simple cases — theoretically. Since the approach is still based on iterative simulations, we call it *simulation-based traffic assignment*.

Another problem of such an iterative algorithm is performance. Even with a fast traffic simulation model like the Nagel-Schreckenberg model, doing multiple simulations of the traffic of one day in Wuppertal is still computationally expensive. In chapter 3.4 we will describe a queuing model which neglects the interactions between individual vehicles. Instead, each link is only described by its capacity, its length, and the number of vehicles which fit into the link. The travel time of vehicles is described as a sum of the free-flow travel time and the time spent waiting in the queue which will build up if the number of cars entering the link exceeds the capacity of the link. Despite its simplicity, this queuing model is capable of giving good estimates of the travel times in both the Nagel-Schreckenberg model and the Krauß model while needing much less computing time. Therefore, it is ideally suited for the simulation-based traffic assignment algorithm.

Using this simulation model, we were able to calculate the dynamic user equilibrium route choices for the Wuppertal network, providing the input data for the microscopic simulation of Wuppertal and enabling us to provide the data needed by the groups evaluating the environmental effects of traffic. Using the queuing model mentioned above, this task — 40 iterations of the simulation-based assignment model — took 300 hours of

CPU time on a UltraSPARC-II processor with a clock rate of 366MHz. While this is a large amount of computing time, we will see that a single iteration of the Frank-Wolfe algorithm to solve a dynamic user equilibrium model for Wuppertal would have taken more than 5 *years* of CPU time on the same machine, i.e. we would simply not have been able to perform a dynamic user equilibrium assignment using such a model to provide the data needed for our task within the FVU.

Chapter

Two

Traffic Assignment

The problem of calculating route choices for a set of drivers is closely related to the *traffic assignment problem* which is described in this chapter. The focus of traffic assignment is more on the mathematical description of route choice than on a realistic description of traffic flow, and it provides fundamental concepts like Wardrop's principles of route choice.

Unfortunately, we will see that the more realistic *dynamic assignment models* are too complex to be solved for large urban networks, and thus not suitable for our purpose.

2.1 Introduction

To calculate the route choices in a traffic simulation model, we do not only need an origin-destination (O-D) matrix but also need information on the travel times in the network. On the other hand, these travel times will depend on the route choices. Therefore, we have to determine a set of route choices which is self-consistent, i.e. consistent with the resulting travel times. In fact, even the O-D matrix has to be chosen in a self-consistent way since in a certain sense it is a function of the travel times: If a commuter cannot get to his job in reasonable time, he might move closer to his job.

Let us formulate the problem of finding self-consistent route choices by viewing a traffic simulation model as a function

$$\begin{aligned} S : \mathcal{R} &\to \mathcal{T} \\ \mathbf{r} &\mapsto \boldsymbol{\tau} \end{aligned}$$

describing the dependence of the travel times $\boldsymbol{\tau} \in \mathcal{T}$ on a set $\mathbf{r}$ of route choices chosen from the set $\mathcal{R}$ of all possible route choices. For most types of simulation models, $\mathcal{R}$ would be the set of all maps from the set of drivers to the set of paths $\mathcal{P}$ in the network, and $\mathcal{T}$ would be the set of all maps from the set of all pairs consisting of a path in the network and a departure time to the corresponding travel time in the simulation, i.e. $\mathcal{T} = \mathbb{R}_{\geq 0}^{\mathcal{P}\times[0,T]}$, where $[0, T]$ is the time interval which is simulated.

On the other hand, a route choice model can be viewed as a function

$$\begin{aligned} C : \mathcal{T} &\to \mathcal{R} \\ \boldsymbol{\tau} &\mapsto \mathbf{r} \end{aligned}$$

describing the dependency of the route choices $\mathbf{r}$ on the the travel times $\boldsymbol{\tau}$.

A set of route choices $\mathbf{r} \in \mathrm{R}$ is self-consistent if

$$\mathbf{r} = C\left(S(\mathbf{r})\right). \tag{2.1}$$

If S is not given by a simulation model but by a link cost function, i.e. an analytical dependence of the link travel time on the link flow, the problem of determining self-consistent route choices is usually called *traffic assignment problem* since it can be viewed as assigning the OD matrix onto the network.

This section discusses basic definitions related to traffic assignment. An overview of the recent development in traffic assignment can be found in [91, 77, 71].

2.1.1 Route Choice Models and Equilibria

There are two fundamentally different classes of route choice models. In one class, some global authority can choose the route of every driver. In this case, the usual assumption is that this authority tries to optimize some global cost function, e.g. the sum of all travel times. Traffic assignment with such a route choice model is called *system optimal traffic assignment*.

In the other class, all drivers choose their route individually, and the assumption is that each individual tries to optimize his own travel time. The resulting route choices satisfy a condition which is known as Wardrop's first principle [89]:

> All used routes [for a fixed origin-destination pair] have equal costs and no unused route has a lower cost.

This case is referred to as *user equilibrium traffic assignment* because the resulting network state can be viewed as an equilibrium since nobody can improve his travel time by unilaterally changing his route[1].

Of course, in this case one silently assumes that each individual has a perfect knowledge of the network state. In the case of commuter traffic, where the traffic demand and hence the network state is more or less the same every day, this assumption is more or less satisfied. Further, a perfect rational and uniform behavior is assumed.

The latter assumptions can be relaxed by making a distinction between the actual travel time and the travel time *perceived* by the individual. The perceived travel times are described as random variables distributed across the travelers. The equilibrium where no traveler believes he can improve his travel time by unilaterally changing routes is called the *stochastic user equilibrium*.

2.1.2 Static and Dynamic Assignment

If the origin destination matrix and the link flows are assumed to be time independent, the problem is known as *static assignment*. If, on the other hand, the time dependence is taken

---

[1] Strictly speaking, this is only true for *separable* cost functions. See chapter 3 of [71] for a discussion of the mathematical subtleties.

into account, one speaks of *dynamic assignment*. We will see that dynamic assignment is much more complex – both computationally and conceptually – than static assignment.

Of course, the link flows are never time independent, but for the discussion of, for example, a peak period, static assignment may be a good approximation, describing a 'stationary limit' of the traffic flow pattern which would evolve if the duration of the peak period would be infinite.

It should be noted, however, that for congested networks this 'stationary limit' is not realistic. Since the outflow from a link cannot exceed the capacity of the link, an inflow greater than the capacity of a link results in a queue building up on the link. This queue will grow as long as the inflow on the link exceeds the capacity. If we assume that the flow is stationary, the length of the queue must be infinite, which is not only unrealistic, but also gives an infinite travel time. Therefore, in the stationary limit the route choice model would not exceed the capacity of a link.

However, in reality the inflow on a link might exceed the capacity of the link during the rush hour, because the travel time on the shortest but jammed route might be still less than the travel time on the second shortest route. This is one of the reasons why civil engineers usually use a link cost function which remains finite above the capacity for static assignment applications. Nevertheless, the situation in a congested network cannot be described in a satisfactory way by a static approach.

Another major shortcoming of static network models is the fact that a travel demand results in a flow which allocates capacity on the whole route. Therefore, flows on routes sharing a common arc of the network always interact, while in reality drivers on different routes might use a link at different times, especially in large networks like the German highway network.

2.2 Static Assignment

Despite its shortcomings, static assignment is still widely used by civil engineers for planning purposes. It is therefore worthwhile to discuss the basics of static assignment before we introduce the dynamic assignment models. It is also easier to understand some of the concepts of dynamic assignment if one is familiar with their static counterparts.

2.2.1 Network Representation and Notation

We represent a traffic network by a directed graph[2] $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ with nodes $\mathcal{N}$ and arcs $\mathcal{A}$. Let $\mathcal{O} \subset \mathcal{N}$ and $\mathcal{D} \subset \mathcal{N}$ denote origin and destination nodes, respectively. Let x_a denote the flow of vehicles on link a. We assume that the costs of traveling on link a can be expressed by a function $\tau_a(x_a)$. As mentioned before, this assumption of static traffic assignment may not hold in real traffic networks due to effects like queuing and spill-back.

---

[2] In some cases it is more convenient to use a line-graph representation of the traffic network, since in this representation turning restrictions can be included easily.

For each pair $(r, s) \in \mathcal{O} \times \mathcal{D}$ the set of paths from r to s is denoted as $\mathcal{P}_{rs}$ and the travel demand from r to s as d_{rs}. Further, the flow on path $p \in \mathcal{P}_{rs}$ is denoted as f_p^{rs}, the costs of traveling on p as τ_p^{rs}. For each link $a \in \mathcal{A}$ and each path $p \in \mathcal{P}_{rs}$ let

$$\delta_{a,p}^{rs} = \begin{cases} 1 \text{ if } a \in p \\ 0 \text{ if } a \notin p \end{cases} .$$

Using $\delta_{a,p}^{rs}$, we can express the link based variables by the path based and vice versa:

$$\tau_p^{rs} = \sum_{a \in \mathcal{A}} \tau_a(x_a)\delta_{a,p}^{rs} \qquad \forall p \in \mathcal{P}_{rs} \quad \forall (r, s) \in \mathcal{O} \times \mathcal{D} \tag{2.2}$$

$$x_a = \sum_{\substack{(r,s) \in \mathcal{O} \times \mathcal{D} \\ p \in \mathcal{P}_{rs}}} f_p^{rs} \delta_{a,p}^{rs} \qquad \forall a \in \mathcal{A}. \tag{2.3}$$

In what follows, we will often use a simplified notation, for example we will refrain from stating the sets in sums and quantors explicitly whenever these sets can be inferred from the context.

2.2.2 Problem Formulation: User Equilibrium

With the notation introduced in the preceding section, the user equilibrium traffic assignment problem can be stated as follows:
For all $(r, s) \in \mathcal{O} \times \mathcal{D}$, find f_p^{rs} satisfying

$$\sum_{p \in \mathcal{P}_{rs}} f_p^{rs} = d_{rs} \qquad \forall r, s \tag{2.4a}$$

$$f_p^{rs} \geq 0 \qquad \forall p, r, s \tag{2.4b}$$

$$f_p^{rs} \cdot \left(\tau_p^{rs} - \min_{q \in \mathcal{P}_{rs}} \tau_q^{rs} \right) = 0 \qquad \forall p, r, s. \tag{2.4c}$$

Note that equation (2.4c) is just a concise formulation of Wardrop's first principle stating that only paths with minimal costs have a nonzero flow assigned to them.

To get a solution algorithm, a formulation of (2.4) as an optimization problem would be more convenient. Since the set of feasible assignments f_p^{rs} is a convex polyhedron and the link cost functions τ_a are usually assumed to be convex (see section 2.2.5), a formulation as an optimization problem should be helpful as convex optimization is a well-studied problem.

In fact, Beckmann[4] has given an equivalent formulation of (2.4) as a minimization program:

$$\min z(\mathbf{x}) = \sum_{a \in \mathcal{A}} \int_0^{x_a} \tau_a(x) dx \tag{2.5a}$$

subject to

$$\sum_{r,s}\sum_{p\in\mathcal{P}_{rs}} f_p^{rs}\delta_{a,p}^{rs} = x_a \qquad \forall a \tag{2.5b}$$

$$\sum_{p\in\mathcal{P}_{rs}} f_p^{rs} = d_{rs} \qquad \forall r,s \tag{2.5c}$$

$$f_p^{rs} \geq 0 \qquad \forall p,r,s \tag{2.5d}$$

This program is called the user equilibrium (UE) program.

To demonstrate that the UE program is equivalent to (2.4), we consider the Lagrangian of (2.5) with respect to the equality constraints (2.5c)

$$L(\mathbf{f},\mathbf{u}) = z(\mathbf{x}(\mathbf{f})) + \sum_{r,s} u_{rs}\left(d_{rs} - \sum_{p\in\mathcal{P}_{rs}} f_p^{rs}\right), \tag{2.6a}$$

where we have used equation (2.5b) to express the link flows in terms of the path flows. Program (2.5) is equivalent to the minimization of $L(\mathbf{f},\mathbf{u})$ with respect to nonnegative path flows

$$f_p^{rs} \geq 0 \qquad \forall p,r,s. \tag{2.6b}$$

At the stationary point of the Lagrangian, the following first-order optimality conditions (see corollary A.1.4) hold:

$$f_p^{rs}\cdot\frac{\partial L(\mathbf{f},\mathbf{u})}{\partial f_p^{rs}} = 0 \qquad \forall p,r,s \tag{2.7a}$$

$$\frac{\partial L(\mathbf{f},\mathbf{u})}{\partial f_p^{rs}} \geq 0 \qquad \forall p,r,s \tag{2.7b}$$

$$\frac{\partial L(\mathbf{f},\mathbf{u})}{\partial u_{rs}} = 0 \qquad \forall r,s. \tag{2.7c}$$

Using

$$\frac{\partial z(\mathbf{x})}{\partial x_a} = \frac{\partial}{\partial x_a}\sum_{b\in\mathcal{A}}\int_0^{x_b}\tau_b(x)dx = \tau_a(x_a) \tag{2.8}$$

and (2.2) we have

$$\begin{aligned}\frac{\partial z(\mathbf{x}(\mathbf{f}))}{\partial f_p^{rs}} &= \sum_{a\in\mathcal{A}}\frac{\partial z(\mathbf{x})}{\partial x_a}\frac{\partial x_a}{\partial f_p^{rs}}\\ &= \sum_{a\in\mathcal{A}}\tau_a(x_a)\delta_{a,p}^{rs}\\ &= \tau_p^{rs}.\end{aligned} \tag{2.9}$$

Thus

$$\frac{\partial L(\mathbf{f},\mathbf{u})}{\partial f_p^{rs}} = \tau_p^{rs} - u_{rs}, \tag{2.10}$$

and the first-order optimality conditions (2.7) are

$$f_p^{rs} \cdot \left(\tau_p^{rs} - u_{rs}\right) = 0 \qquad \forall p, r, s \tag{2.11a}$$

$$\tau_p^{rs} - u_{rs} \geq 0 \qquad \forall p, r, s \tag{2.11b}$$

$$\sum_{p \in \mathcal{P}_{rs}} f_p^{rs} = d_{rs} \qquad \forall r, s \tag{2.11c}$$

$$f_p^{rs} \geq 0 \qquad \forall p, r, s. \tag{2.11d}$$

Condition (2.11b) states that the Lagrange multiplier u_{rs} is less or equal to the path cost on any path connecting r and s, and condition (2.11a) states that u_{rs} is equal to the path cost for paths with nonzero flow. This implies that

$$u_{rs} = \min_{p \in \mathcal{P}_{rs}} \tau_p^{rs} \qquad \forall r, s \tag{2.12}$$

so (2.11) – and hence (2.5) – is equivalent to (2.4), which is what we wanted to demonstrate.

2.2.3 Problem Formulation: System Optimum

Unlike the user equilibrium traffic assignment problem, the system optimal traffic assignment problem, i.e. the minimization of the total travel time of all travelers, can be formulated as an minimization program in a straightforward way:

$$\min z(\mathbf{x}) = \sum_{a \in \mathcal{A}} x_a \tau_a(x_a) \tag{2.13a}$$

subject to

$$\sum_{r,s} \sum_{p \in \mathcal{P}_{rs}} f_p^{rs} \delta_{a,p}^{rs} = x_a \qquad \forall a \tag{2.13b}$$

$$\sum_{p \in \mathcal{P}_{rs}} f_p^{rs} = d_{rs} \qquad \forall r, s \tag{2.13c}$$

$$f_p^{rs} \geq 0 \qquad \forall p, r, s. \tag{2.13d}$$

Program (2.13) is called the system optimization (SO) program.

On the other hand, just as the user equilibrium problem can be formulated as an optimization problem with respect to a special cost function, the system optimum problem can be formulated as a user equilibrium problem with respect to *marginal costs*. Since rational behavior by the individuals leads to the user equilibrium, these marginal costs are the costs which should be enforced by a network operator in order to ensure that the rational behavior of the individuals leads to the system optimal traffic flow pattern. This is especially important in traffic management applications where the objective is to apply some control measures to get the traffic flows to the system optimal state.

As for the UE program, the optimality conditions in terms of the Lagrangian

$$L(\mathbf{f},\mathbf{u}) = z(\mathbf{x}(\mathbf{f})) + \sum_{r,s} u_{rs}\left(d_{rs} - \sum_{p} f_p^{rs}\right) \tag{2.14}$$

are

$$f_p^{rs} \cdot \frac{\partial L(\mathbf{f},\mathbf{u})}{\partial f_p^{rs}} = 0 \qquad \forall p,r,s \tag{2.15a}$$

$$\frac{\partial L(\mathbf{f},\mathbf{u})}{\partial f_p^{rs}} \geq 0 \qquad \forall p,r,s \tag{2.15b}$$

$$\frac{\partial L(\mathbf{f},\mathbf{u})}{\partial u_{rs}} = 0 \qquad \forall r,s. \tag{2.15c}$$

As in the case of the UE program, (2.15c) simply restates the flow conservation restraints. Further, we have

$$\begin{aligned}
\frac{\partial z(\mathbf{x}(\mathbf{f}))}{\partial f_p^{rs}} &= \sum_{a\in\mathcal{A}} \frac{\partial z(\mathbf{x})}{\partial x_a}\frac{\partial x_a}{\partial f_p^{rs}} \\
&= \sum_{a\in\mathcal{A}} \delta_{a,p}^{rs} \frac{\partial}{\partial x_a} \sum_{x_b} x_b \tau_b(x_b) \\
&= \sum_{a\in\mathcal{A}} \delta_{a,p}^{rs} \left(\tau_a(x_a) + x_a \frac{d\tau_a(x_a)}{dx_a}\right).
\end{aligned} \tag{2.16}$$

If we define

$$\tilde{\tau}_a(x_a) = \tau_a(x_a) + x_a \frac{d\tau_a(x_a)}{dx_a}, \tag{2.17}$$

we get

$$\frac{\partial z(\mathbf{x}(\mathbf{f}))}{\partial f_p^{rs}} = \sum_{a\in\mathcal{A}} \delta_{a,p}^{rs} \tilde{\tau}_a(x_a) = \tilde{\tau}_p^{rs} \tag{2.18}$$

where $\tilde{\tau}_p^{rs}$ are the path costs resulting from the link costs $\tilde{\tau}_a(x_a)$.

The additional link cost term $x_a \frac{d\tau_a(x_a)}{dx_a}$ can be viewed as the cost which an additional traveler on link a would inflict on the other x_a 'travelers' which already use the link, since $\frac{d\tau_a(x_a)}{dx_a}$ is the amount by which the travel cost per flow unit increases if the flow is increased by an infinitesimal amount, i.e. one traveler.

We can now write the conditions (2.15) in the same form as (2.11)

$$f_p^{rs} \cdot \left(\tilde{\tau}_p^{rs} - u_{rs}\right) = 0 \qquad \forall p,r,s \tag{2.19a}$$

$$\tilde{\tau}_p^{rs} - u_{rs} \geq 0 \qquad \forall p,r,s \tag{2.19b}$$

$$\sum_{p\in\mathcal{P}_{rs}} f_p^{rs} = d_{rs} \qquad \forall r,s \tag{2.19c}$$

$$f_p^{rs} \geq 0 \qquad \forall p,r,s, \tag{2.19d}$$

so the solution of the SO program is equivalent to the user equilibrium with respect to the link cost function $\tilde{\tau}_a(x_a)$.

Since under the cost function $\tilde{\tau}_a(x_a)$ the user equilibrium is equivalent to the system optimum for the cost function $\tau_a(x_a)$, a traffic manager can 'push' the traffic to the system optimum state by putting a toll of amount $\tilde{\tau}_a(x_a) - \tau_a(x_a) = x_a \frac{d\tau_a(x_a)}{dx_a}$ on each link $a \in \mathcal{A}$.

2.2.4 Uniqueness of the Solution

From the theoretical point of view it is interesting to ask under which conditions solutions of the UE and SO programs exist and whether they are unique. Since the conditions $\sum_{p\in\mathcal{P}_{rs}} f_p^{rs} = d_{rs}$, $\sum_{r,s}\sum_{p\in\mathcal{P}_{rs}} f_p^{rs}\delta_{a,p}^{rs} = x_a$ and $f_p^{rs} \geq 0$ describe a non-empty convex polyhedron, the strict convexity of $z(\mathbf{x})$ would suffice to ensure that z has exactly one local minimum[3]. The (strict) convexity of z is equivalent to the (strict) positive definiteness of the Hessian

$$\nabla^2 z(\mathbf{x}) = \left(\frac{\partial^2 z(\mathbf{x})}{\partial x_n \partial x_m}\right)_{(n,m)\in\mathcal{A}^2}. \tag{2.20}$$

For the UE program, we have

$$\begin{aligned}\frac{\partial^2 z(\mathbf{x})}{\partial x_n \partial x_m} &= \frac{\partial^2}{\partial x_n \partial x_m}\sum_a \int_0^{x_a} \tau_a(x)dx \\ &= \frac{\partial \tau_m(x_m)}{\partial x_n} = \frac{d\tau_n(x_n)}{dx_n}\delta_{n,m},\end{aligned} \tag{2.21}$$

since we always assume that the costs on link n do not depend on the flow on link m for $n \neq m$. Equation (2.21) implies that z is (strictly) convex if $\tau_a(x_a)$ is (strictly) monotonic increasing.

In the SO case, we get

$$\begin{aligned}\frac{\partial^2 z(\mathbf{x})}{\partial x_n \partial x_m} &= \frac{\partial^2}{\partial x_n \partial x_m}\sum_a x_a\tau_a(x_a) \\ &= 2\frac{\partial \tau_m(x_m)}{\partial x_n} + x_m \frac{\partial}{\partial x_n}\frac{d\tau_m(x_m)}{dx_m} = \left(2\frac{d\tau_n(x_n)}{dx_n} + x_n\frac{d^2\tau_n(x_n)}{dx_n^2}\right)\delta_{n,m},\end{aligned} \tag{2.22}$$

so a sufficient condition for z being (strictly) convex is $\tau_a(x_a)$ being (strictly) monotonic increasing and (strictly) convex.

In the next section we will see that both conditions are usually assumed to be true, which ensures that both programs have unique solutions with respect to the link flows. However, this does *not* imply that the solution is unique with respect to the path flows since there may be different path flows yielding the same link flows, as figure 2.1 demonstrates.

---

[3] See, for example, theorem 3.4.2 in [3].

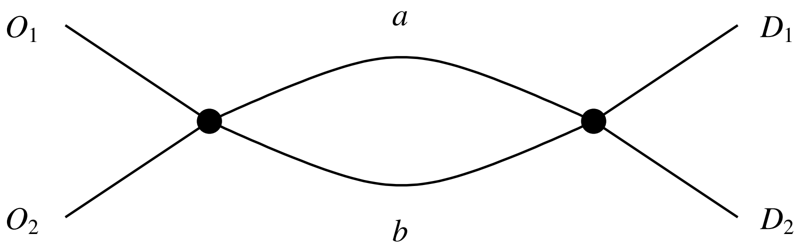


Figure 2.1 A simple example of non-unique path flows. Suppose we have $d_{O_1D_1} = d_{O_2D_2} = d \neq 0$ and all other demands are zero. For both OD-pairs there are exactly two routes, using either link a or link b. If we set $f_a^{O_1D_1} = f_b^{O_2D_2} = f$ and $f_b^{O_1D_1} = f_a^{O_2D_2} = d - f$, we have $x_a = x_b = d$ regardless of f.

2.2.5 Link Cost Functions

The cost of traveling on a link depends on the traffic volume on that link. Usually, we will assume these costs are proportional to the travel time[4], and if we neglect the fact that the conversion coefficient between time and money may vary among different travelers, we can express these costs in terms of the travel time.

From everyday's experience we can deduce some basic properties of the relation between travel time and traffic volume on a road:

- The travel time increases with increasing traffic volume.
- At small traffic volumes, a traveler can choose his velocity freely within the physical constraints set by his car.
- At high traffic volumes, the velocity of a traveler is determined by the surrounding cars.
- There is a maximum number of cars which can pass the road per time unit. This number depends mainly on the type of the road.

We used the term 'traffic volume' instead of 'traffic flow' because the actual flow is not a good indicator for the traffic volume. In a traffic jam, the flow in terms of vehicles per time is actually lower than the maximum attainable flow. A better indicator for the traffic volume is the *traffic density*, i.e. the number of vehicles per unit length, on a road. Figure 2.2 and 2.3 show how traffic flow and velocity depend on the traffic density on a Californian highway. The exact functional dependence varies from road to road and may even vary on the same road under different environmental conditions such as visibility or rain. Nevertheless, the basic features – linear dependence of the flow on the density for small densities, decreasing flow for high densities – are universal. Especially noteworthy is the fact that neither traffic density nor the mean velocity can be expressed as a function of traffic flow. An important dynamic characteristic of traffic flow is not contained in the fundamental diagram, namely the fact that the flow becomes unstable at a certain critical

---

[4] Recently, the case of nonlinear relations between time and money have been considered by Bernstein and Gabriel [7]. In this case, even the problem of finding shortest paths in a network becomes NP-hard, since sub-paths of shortest paths are not necessary shortest paths.

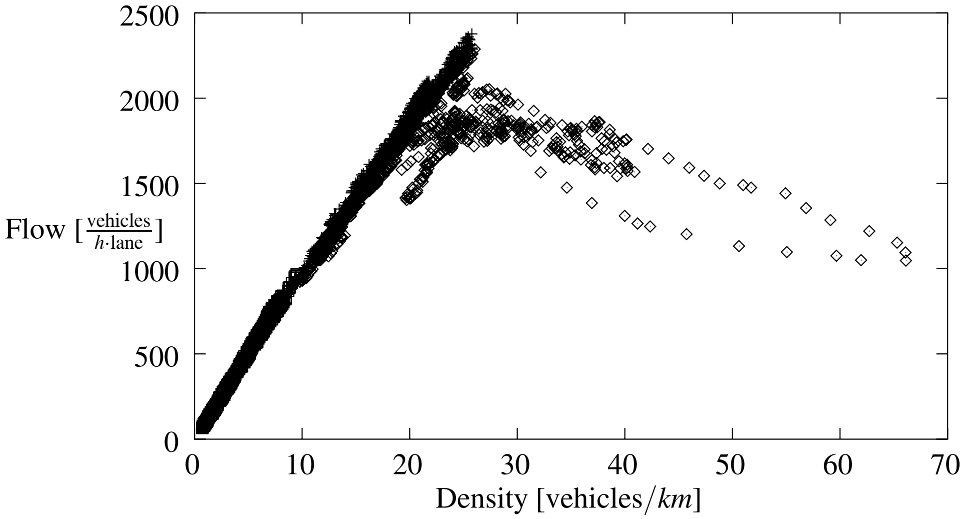


Figure 2.2 So-called fundamental diagram, showing the relation between traffic density and traffic flow which has the characteristic *reverse lambda* shape. For densities less then about 25cars/*km* the flow grows more or less linear with the density. This part of the fundamental diagram is called the *free-flow regime*. For larger densities, the flow decreases due to traffic jams. Note that this means that the density cannot be expressed as a function of the flow. At intermediate densities, both states – free-flow or jam – are possible, but the free-flow state is unstable. The data set was collected on a Californian highway.

density. In this state there is a certain non-zero probability – which increases with the density – that a traffic jam arises and the flow breaks down. This metastable state is the reason for the characteristic 'reverse lambda' shape of the fundamental diagram 2.2. The *capacity* of a road is the highest flow which is stable.

For static assignment, we have to express the travel time, or equivalently the mean velocity, as a function of the traffic flow. Figure 2.4 shows that this is problematic since there are two different velocities – corresponding to the free-flow and the jammed regime – for each flow. If we would insist on the fact that for static assignment the traffic state has been assumed to be static, and therefore to be in the free-flow regime, we would have to use the free-flow branch of the flow velocity relation to calculate the travel time. Further we had to ensure that the flow does not exceed the value where it gets unstable, i.e. the value where the probability of the flow breaking down gets positive.

However, this approach is unpracticable for two reasons. The first one is that the

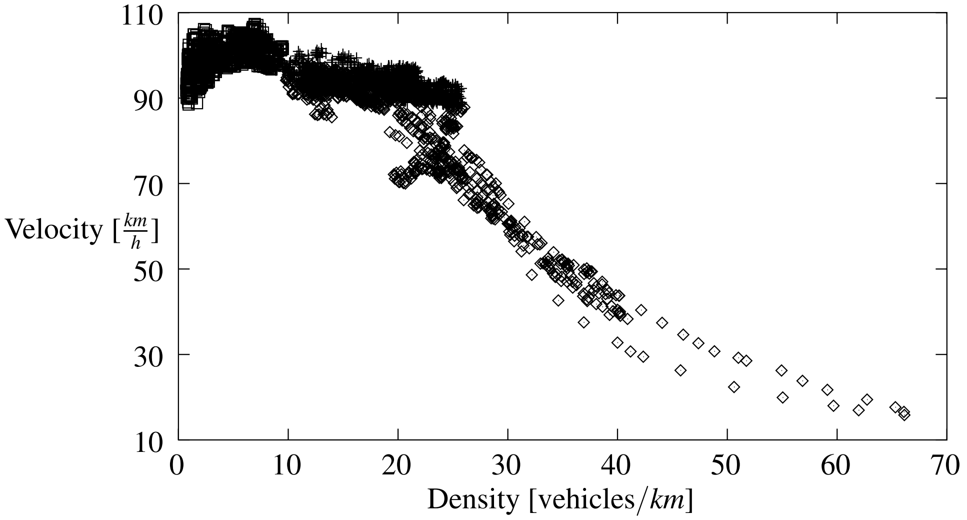


Figure 2.3 Relationship between traffic density and velocity for the same data set as Figure 2.2. In the free-flow regime the velocity depends only weakly on the density, whereas for densities greater 25cars/km the velocity drops drastically due to jamming.

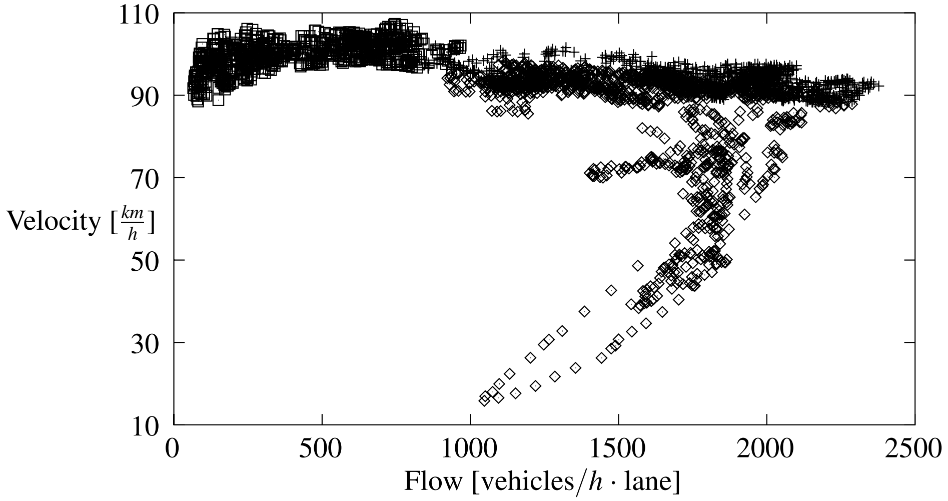


Figure 2.4 Relationship between traffic flow and velocity for the same data set as Figure 2.2. Although there are states where the traffic flow exceeds 2400 $\frac{\text{vehicles}}{h\cdot\text{lane}}$, the flow gets unstable at about 1800 $\frac{\text{vehicles}}{h\cdot\text{lane}}$.

resulting capacities of the links would be too small compared to reality, since in reality unstable flows occur, albeit for a short time. The second one is that the velocity does not change significantly in the free-flow regime, so the corresponding link cost function would be more or less constant. To fix these problems, link cost functions are used which allow flow values which are unstable, but add some cost for the probability of jamming. Although these cost functions are not compatible with the basic assumption of a static and stable network state, they are widely accepted by practitioners. In fact, authorities like the U.S. Bureau of Public Roads (BPR) [86] or the German Department of Transportation [38] proposed standard link cost functions. The BPR travel time function is

$$\tau_a = \tau_a^0 \cdot \left(1 + \alpha \left(\frac{x_a}{c'_a}\right)^\beta\right). \tag{2.23}$$

In (2.23), τ_a^0 is the free-flow travel time and c'_a is the ‘practical capacity’ of link a. The model parameter α and β are usually chosen as $\alpha = 0.15$ and $\beta = 4$. This implies that the practical capacity is the flow at which the travel time is 15% higher than the free-flow travel time and that the BPR function sets no limit to the actual flow on a link.

Davidson [21] proposed a link cost function of the form

$$\tau_a = \tau_a^0 \cdot \left(1 + J \frac{x_a}{c_a - x_a}\right), \tag{2.24}$$

where c_a is the capacity of link a and J is a parameter which has to be estimated from field measurements. Davidson’s cost function can be deduced if one assumes that a link can be viewed as a queuing system with Markovian[5] inter-arrival and service intervals and a server capacity of 1 (a so-called M/M/1 queue, see [48]). For such queues it can be shown that the average queue length is

$$\overline{Q} = \frac{\rho}{1-\rho}, \tag{2.25}$$

where ρ is the *utilization* of the queue, which corresponds to x_a/c_a, so (2.25) has the same type of divergence as (2.24).

The link cost functions proposed by the German Department of Transportation [38] are defined by a piecewise linear dependence of the mean velocity on the traffic flow.

All these link cost functions are both strictly monotonic increasing and strictly convex, implying that the solutions to the UE and SO programs are unique.

2.2.6 User Equilibrium, System Optimum and Braess’s Paradox

In economics it is usually assumed that the rational, i.e. cost optimizing, behavior of each individual tends to optimize the whole system, as long as certain boundary conditions are met. Therefore, one could expect the user equilibrium to be nearly optimal from the

---

[5] In this context, Markovian means uncorrelated, i.e. the time intervals between two arriving cars are Poisson distributed. This assumption holds for low traffic densities.

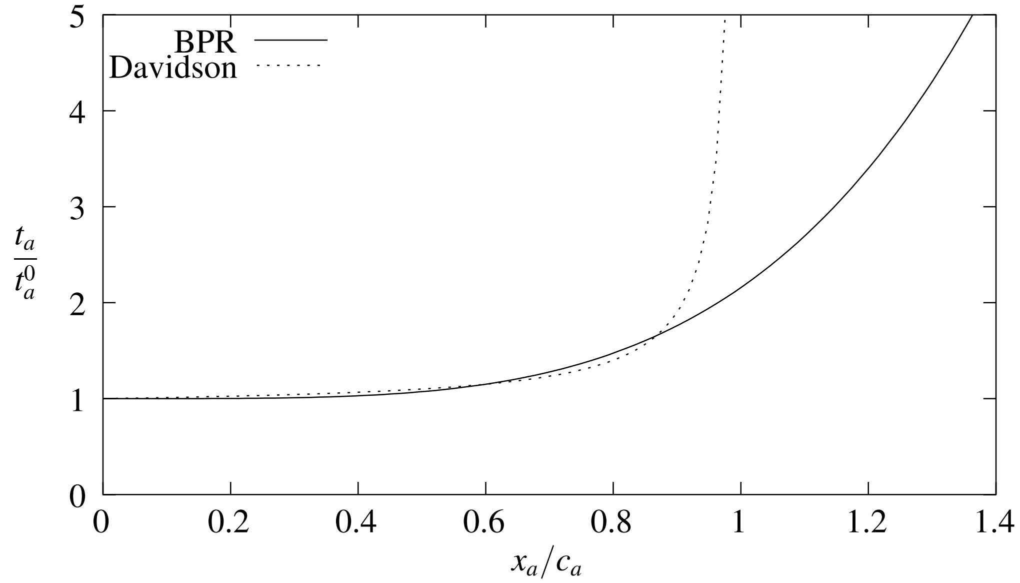


Figure 2.5 BPR and Davidson link cost functions. The parameter J of the Davidson function is set to 0.1, the parameters of the BPR function are $\alpha = 0.15$ and $\beta = 4$. The ‘practical capacity’ c'_a of the BPR, which corresponds to the flow at which the travel time is increased by a fraction of α, is chosen as 0.6 which is the correct value with respect to the Davidson function.

system point of view. Of course, due to the nonlinearity of the link cost function there might be cases in a congested network where many traveler might gain from the fact that a few others choose an alternative route which is suboptimal for them.

So it is very surprising that even an ‘improvement’ of a network, i.e. the addition of a link, might lead to a longer travel time *for every driver* under the user equilibrium condition. This paradox was first discovered by Braess [9, 64].

Figure 2.6 shows the network devised by Braess. The two narrow links can be viewed as bottlenecks since their costs increase strongly with the flow compared to the other links. Without the dashed link, the optimal solution of both the UE and SO program is $f_{O\to 1\to D} = f_{O\to 2\to D} = \frac{1}{2}d_{OD}$.

With the additional link a, the optimal solution still satisfies $f_{O\to 1\to D} = f_{O\to 2\to D}$. Therefore, all link flows can be expressed in terms of $f_{O\to 1\to 2\to D} = x_a$ by

$$x_1 = x_4 = \frac{d_{OD} - x_a}{2} \tag{2.26a}$$

$$x_2 = x_3 = \frac{d_{OD} + x_a}{2}, \tag{2.26b}$$

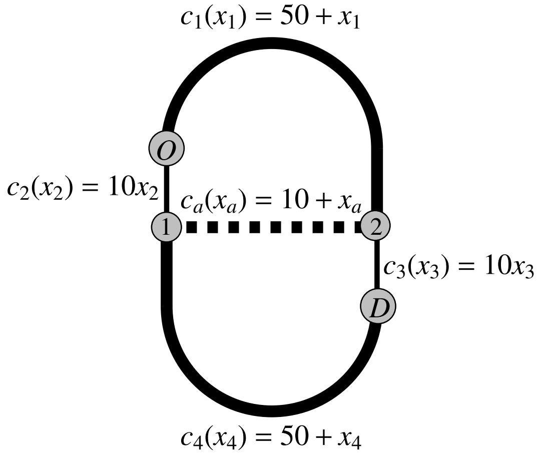


Figure 2.6 Braess's paradox. The picture shows the network and the link cost functions. The cost functions are chosen such that the narrow links can be viewed as short bottlenecks, whereas the thick links have a higher capacity but are longer. The dashed link a is the additional link which 'improves' the network. Due to symmetry, the flow on routes $O \to 1 \to D$ and $O \to 2 \to D$ are equal in both the SO and UE solution. So given the demand d_{OD}, there is only one independent link flow variable x_a, the flow on the additional link.

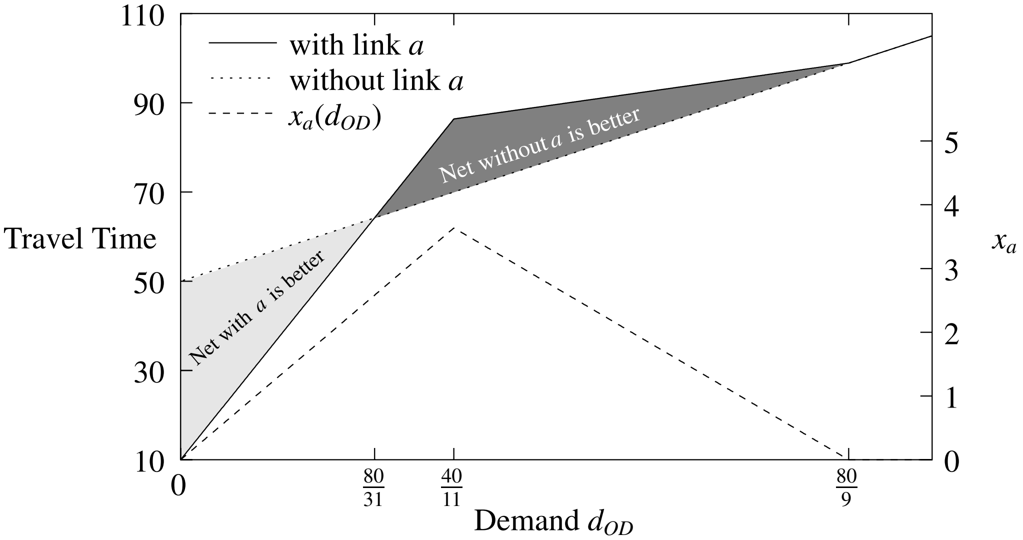


Figure 2.7 User equilibrium travel times in Braess's network. With the additional link a, for $80/31 < d_{OD} < 80/9$ the user equilibrium travel times are higher for all travelers, i.e. the 'rational' behavior of the individuals leads to a state where *everybody* is worse off.

and the travel times are

$$\tau_{O\to 1\to D} = 50 + \frac{d_{OD} - x_a}{2} + 10\frac{d_{OD} + x_a}{2} \tag{2.27a}$$

$$\tau_{O\to 1\to 2\to D} = 10 + 20\frac{d_{OD} + x_a}{2} + x_a. \tag{2.27b}$$

Some basic algebra yields the user equilibrium flow

$$x_a = \begin{cases} 0 & \text{if } \frac{80-9d_{OD}}{13} < 0 \\ \frac{80-9d_{OD}}{13} & \text{if } 0 < \frac{80-9d_{OD}}{13} < d_{OD} \\ d_{OD} & \text{if } d_{OD} < \frac{80-9d_{OD}}{13} \end{cases} \tag{2.28}$$

It turns out that for $d_{OD} < 80/31$ the travel time with the additional link is less than without the link – the 'shortcut' bypassing the 'long' links 1 and 4 improves the network. For $d_{OD} > 80/9$ the travel time via link a would be higher than the travel time via the long links, so the flow on link a is zero.

For $80/31 < d_{OD} < 80/9$ something interesting happens: For $x_a = 0$, the travel time via a is less than the travel time via the long links, so travelers will have an incentive to change to the route via a. In the resulting user equilibrium, however, the travel time is *higher* than the travel time without a. A fathomable explanation for this is the fact that if one traveler (or an infinitesimal flow unit) changes to the route via a, their travel time improvement is less than the additional travel time inflicted on the other travelers.

2.2.7 Solution Techniques

Both the UE and the SO programs are convex optimization problems which can be solved numerically by various methods. The still most widely used solution algorithm is the Frank-Wolfe algorithm described in A.2.

The linear subproblem A.14 of the nth iteration of the Frank-Wolfe algorithm is of the form

$$\min_{\mathbf{x}} \mathbf{x} \cdot \nabla z(\mathbf{x_n}) \tag{2.29a}$$

$$\text{subject to} \quad \sum_{r,s} \sum_{p\in\mathcal{P}_{rs}} f_p^{rs} \delta_{a,p}^{rs} = x_a \tag{2.29b}$$

$$\sum_{p\in\mathcal{P}_{rs}} f_p^{rs} = d_{rs} \tag{2.29c}$$

$$f_p^{rs} \geq 0, \tag{2.29d}$$

i.e. we have to find the minimum cost flow with respect to the link costs $\nabla z(\mathbf{x_n})$, where $\mathbf{x_n}$ is fixed. Since our problem formulations contain no explicit capacity limits, this problem is equivalent to solving a shortest path problem for each OD pair. These shortest path problems can be solved very efficiently [23, 1], and since these subproblems are independent, the Frank-Wolfe algorithm can be parallelized very easily.

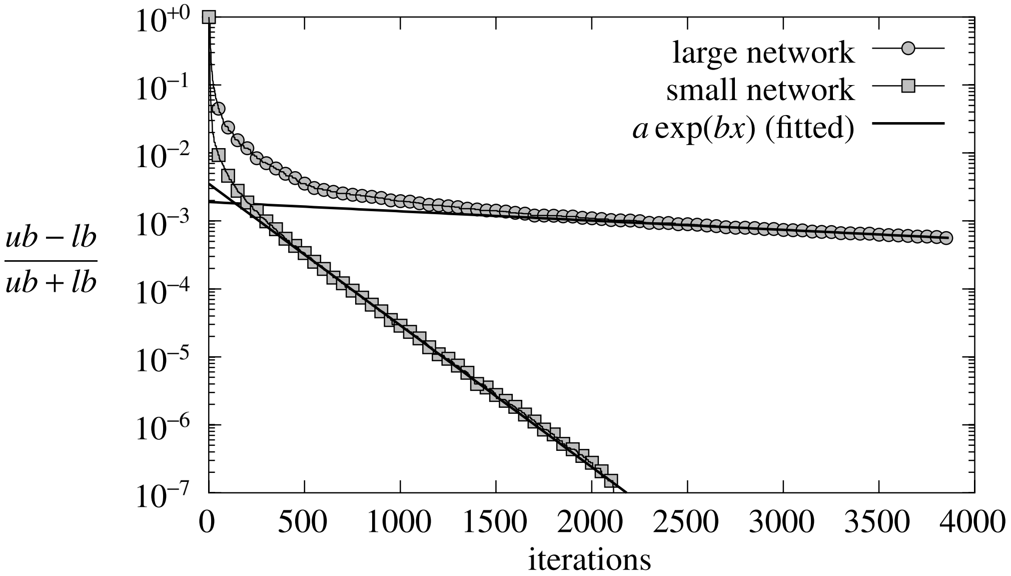


Figure 2.8 Typical convergence of the Frank-Wolfe algorithm. The relative difference between *ub* and *lb*, the upper and lower bounds on the objective function value, is used as the indicator for convergence. For the example network, the relative gap decreases exponentially with the number of iterations, although the theoretical convergence rate is only arithmetic.

The only problem with the Frank-Wolfe algorithm is its rather slow convergence (see figure 2.8). It can be shown [71] that the theoretical convergence rate is *arithmetic*, i.e. $z(\mathbf{x^n}) - \underline{z}(\mathbf{x^n}) = O(1/n)$, where $\underline{z}(\mathbf{x^n})$ is the lower bound provided by (A.17). Figure 2.8 shows that the practical convergence rate is often better than the theoretical one. The convergence rate can be improved by the simplicial decomposition method [54, 71], which replaces the line-search step by a minimization over the convex hull of more than two points. However, the discussion of these methods is beyond the scope of this thesis.

Since the set $\mathcal{P}_{rs}$ grows exponentially with the size of the network, it is very important that in an actual implementation $\mathcal{P}_{rs}$ has not to be enumerated explicitly. This is possible since routes p with $f_p^{rs} = 0$ can simply be neglected. In the language of mathematical programming this approach, which is also useful in other problems related to network flows (see chapter 17 in [1] for an example), is called 'column generation'. For stochastic assignment models like the model of Fisk [25], which has been studied by A. Vildósola Engelmayer in her diploma thesis [88], the set of routes $\mathcal{P}_{rs}$ has either to be enumerated or some subset $\mathcal{P}'_{rs} \subset \mathcal{P}_{rs}$ of 'reasonable' routes has to be given a priori, since all possible routes have a flow $f_p^{rs} > 0$ assigned to them.

The following 'pidgin code' algorithm for the static assignment problem can be easily implemented using the **LEDA**[6] class library [60], which contains all necessary graph-related data structures.

```
procedure UE_Assignment
begin
  $n := 0$
  $\mathbf{x}_0 := \mathbf{0}$
  foreach $a \in A$
    $\mathbf{c}[a] := c_a(0)$                                   cost vector for empty net

  foreach $(r, s) \in \mathcal{O} \times \mathcal{D}$           perform all-or-nothing
                                                                assignment on empty net
  begin
    $p :=$ ShortestPath$(r, s, \mathbf{c})$                     shortest path from $r$ to $s$ with
                                                                respect to $\mathbf{c}$
    $\mathbf{f}^{rs}[p] := d_{rs}$
    $\mathcal{P}^{rs}_0 := \{p\}$
    foreach $a \in A$
      $\mathbf{x}_0[a] := \mathbf{x}_0[a] + \delta^{rs}_{a,p} d_{rs}$
  end

  repeat
    $\mathcal{P}^{rs}_{n+1} := \emptyset$foreach $a \in A$
      $\mathbf{c}[a] := c_a(\mathbf{x}_n[a])$                   set cost vector for $\mathbf{x}_n$

    $\mathbf{y} := 0$
    foreach $(r, s) \in \mathcal{O} \times \mathcal{D}$         perform all-or-nothing
                                                                assignment with respect to $\mathbf{c}$
    begin
      $p :=$ ShortestPath$(r, s, \mathbf{c})$
      $\mathcal{P}^{rs}_{n+1} := \mathcal{P}_{n+1} \cup \{p\}$
      foreach $a \in A$
        $\mathbf{y}[a] := \mathbf{y}[a] + \delta^{rs}_{a,p} d_{rs}$
    end
    $\alpha :=$ Minimum$(z(\mathbf{x}_n + \alpha(\mathbf{y} - \mathbf{x}_n)), 0, 1)$    line search using Brent's
                                                                method
    $\mathbf{x}_{n+1} := (1 - \alpha)\mathbf{x}_n + \alpha\mathbf{y}$

    foreach $(r, s) \in \mathcal{O} \times \mathcal{D}$
    begin
      foreach $p \in \mathcal{P}^{rs}_n$
```

---

[6] Library of **E**fficient **D**ata types and **A**lgorithms, developed at the Max-Planck Institut für Informatik, Saarbrücken. **LEDA** is available at http://www.mpi-sb.mpg.de/LEDA/.

```
                $\mathbf{f}^{rs}[p] := (1-\alpha)\mathbf{f}^{rs}[p]$
            foreach $p \in \mathcal{P}^{rs}_{n+1}$
                $\mathbf{f}^{rs}[p] := \alpha d_{rs} + \mathbf{f}^{rs}[p]$
            $\mathcal{P}^{rs}_{n+1} := \mathcal{P}^{rs}_{n+1} \cup \mathcal{P}^{rs}_{n}$
        end
        $ub := z(\mathbf{x}_{n+1})$                          upper bound on objective function value
        $lb := z(\mathbf{x}_n) + (\mathbf{y} - \mathbf{x}_n) \cdot \mathbf{c}$    lower bound on objective function value
        $n := n + 1$
    until $\frac{ub - lb}{ub + lb} < \varepsilon$                  convergence check
end
```

2.3 Dynamic Assignment

The major shortcomings of static traffic assignment, namely the failure to describe congestion correctly and the fact that for large networks the link flows are overestimated, can only be resolved by dynamic, i.e. time dependent, models. Such models have been proposed by several authors, among them Merchant and Nemhauser [61, 62], Carey [14], Mahmassani [59], and Friesz et al. [29, 28]. These models, which are formulated as either nonlinear programming problems, optimal control problems or variational inequalities, differ in the way the dynamics of traffic are described. The models by Merchant and Nemhauser and by Carey model only system optimal route choice. Furthermore, the dynamics of the flow variables are not consistent with the travel times, i.e. the travel times corresponding to the link costs do not coincide with the velocity with which the flow propagates through the network. Among the first model with a consistent description of flow propagation are the models by Friesz et al. [28] and Ran, Boyce and LeBlanc [76]. Serwill proposed DRUM, a modeling approach using successive static assignment steps for every time step and adding 'unfinished trips' to the OD-matrix for the next time step [83]. A comprehensive overview of dynamic network models can be found in [77], so we restrict the discussion of dynamic assignment models to a minimum, and focus on the complexity of the problem. The notation in this section follows [84] and [77].

2.3.1 Network Model

We will consider a fixed time period $[0, T]$ which should be long enough to allow travelers to reach their destination. The period will typically be either a whole day or a peak period.

We will replace the flow variables x_a by functions $x_a(t)$, which is not longer the flow on link a but *the number of vehicles traveling on link a at time t*.

To describe the propagation of vehicles correctly, we also have to distinguish vehicles by their destination and route. Therefore, we have to introduce $x_{ap}^{rs}(t)$, the number of vehicles traveling on link a over route p from r to s at time t. The $x_a(t)$ are related to the $x_{ap}^{rs}(t)$ by

$$\sum_{(r,s)\in\mathcal{O}\times\mathcal{D}}\sum_{p\in\mathcal{P}} x_{ap}^{rs}(t) = x_a(t). \tag{2.30}$$

To describe the dynamics of $x_a(t)$, we introduce the inflow and outflow rates $u_a(t)$ and $v_a(t)$ and corresponding variables $u_{ap}^{rs}(t)$ and $v_{ap}^{rs}(t)$, respectively. The latter variables are called *control variables* since they 'control' the dynamics of the state variables $x_a(t)$. These are related to $x_{ap}^{rs}(t)$ by the state equation

$$\frac{dx_{ap}^{rs}(t)}{dt} = u_{ap}^{rs}(t) - v_{ap}^{rs}(t). \tag{2.31}$$

Usually we will assume the initial condition

$$x_{ap}^{rs}(0) = 0. \tag{2.32}$$

Of course, all these state and control variables must be nonnegative:

$$x_{ap}^{rs}(0) \geq 0, \quad u_{ap}^{rs}(0) \geq 0, \quad v_{ap}^{rs}(0) \geq 0. \tag{2.33}$$

Flow Conservation Constraints

At each node $v \in \mathcal{N} \setminus (\mathcal{O} \cup \mathcal{D})$, flow conservation implies

$$\sum_{a\in A(v)} v_{ap}^{rs}(t) = \sum_{a\in B(v)} u_{ap}^{rs}(t) \qquad \forall v \notin \{r, s\}, \tag{2.34}$$

where $A(v) = \{(v, w) \in \mathcal{A}\}$ denotes the set of all links going out of node v and $B(v) = \{(w, v) \in \mathcal{A}\}$ denotes the set of all incoming links at v.

Let $f^{rs}(t)$ denote the flow departing at origin node r to destination node s at time t, and denote the flow arriving at destination s from origin r at time t as $e^{rs}(t)$. Then the flow conservation constraints at the origin and destination nodes can be written as

$$\sum_{a\in A(r)}\sum_{p\in\mathcal{P}_{rs}} u_{ap}^{rs}(t) = f^{rs}(t) \tag{2.35a}$$

$$\sum_{a\in B(s)}\sum_{p\in\mathcal{P}_{rs}} v_{ap}^{rs}(t) = e^{rs}(t). \tag{2.35b}$$

Further we have the nonnegativity constraints

$$f^{rs}(t) \geq 0 \tag{2.36a}$$

$$e^{rs}(t) \geq 0. \tag{2.36b}$$

Flow Propagation Constraints

Contrary to the static assignment models, the dynamic assignment models describe the propagation of the flow through the network via the time-dependent inflow and outflow variables. However, it is necessary to add constraints to ensure these flow variables are consistent with the travel times on the links.

These flow propagation constraints can be formulated in different ways, depending on which variables are used to express the constraints and whether or not dispersion is included [77].

One way to formulate these constraints is by observing the fact that the vehicles using route p which are on link a at time t are at time $t + \tau_a(t)$ either on some downstream link which is part of the subroute $\tilde{p}$ of p starting at a, or have arrived at the destination.

Using

$$E_p^{rs}(t) = \int_0^t e_p^{rs}(\tau)d\tau$$

this fact can be written as

$$x_{ap}^{rs}(t) = E_p^{rs}(t + \tau_a(t)) - E_p^{rs}(t) + \sum_{b \in \tilde{p}} x_{bp}^{rs}(t + \tau_a(t)) - x_{bp}^{rs}(t). \tag{2.37}$$

The main problem with the flow propagation constraints is that they contain the travel times $\tau_a(t)$ which are unknown until the problem is solved. This is no problem for the theoretical formulation of the equilibrium conditions, but in a solution algorithm this problem has to be solved by the *relaxation technique*: The constraints (2.37) are formulated using estimated travel times $\overline{\tau}_a(t)$, and the model is solved to obtain new estimates for the travel times. This process is iterated until the travel times converge. This iteration process increases the complexity of the dynamic assignment problem drastically compared to the static assignment problem.

Travel Times

As in the static models, we assume that the travel time over a link only depends on the state of the link, i.e.

$$\tau_a(t) = c_a\left(x_a(t), u_a(t), v_a(t)\right). \tag{2.38}$$

For the decision of the travelers, two concepts of route travel times may be considered. One is the *instantaneous travel time* $\psi_p^{rs}(t)$, which is the time needed to travel along route p if the traffic conditions at time t prevailed during the whole trip. It follows that

$$\psi_p^{rs}(t) = \sum_{a \in p} \tau_a(t). \tag{2.39}$$

It is reasonable to assume that drivers would choose their route according to the instantaneous travel time if they would have perfect information on the current network state but no knowledge on the future evolution of the network state.

Another criterion might be the *actual travel time* $\eta_p^{rs}(t)$, which is the time a traveler needs to travel along route p if he starts at time t. To express $\eta_p^{rs}(t)$ in terms of the link travel times, we assume that $p = (r, v_1, v_2, \ldots, v_i, \ldots, s)$ and recursively define

$$\eta_p^{rr}(t) = 0, \tag{2.40}$$

$$\eta_p^{rv_i}(t) = \eta_p^{rv_{i-1}}(t) + \tau_{(v_{i-1},v_i)}(t + \eta_p^{rv_{i-1}}(t)). \tag{2.41}$$

The actual travel time would be a reasonable decision criteria for the travelers in a day-to-day setup, since by trying different routes a traveler learns the actual travel time of different routes.

We define

$$\sigma^{rs}(t) = \min_{p\in\mathcal{P}_{rs}} \psi_p^{rs}(t) \tag{2.42}$$

$$\pi^{rs}(t) = \min_{p\in\mathcal{P}_{rs}} \eta_p^{rs}(t). \tag{2.43}$$

2.3.2 Dynamic User Equilibrium Conditions and Variational Inequality Formulation

The goal of this section is the variational inequality formulation of the dynamic user equilibrium (DUE) conditions. The main reason for a variational inequality formulation instead of a nonlinear program formulation which we have used in the static case is the fact that the flow propagation constraint (2.37) contains the link travel times which are not known until the problem is solved.

DUE Conditions

The extension of Wardrop's first principle to the dynamic case using either the instantaneous or actual travel times is straightforward. One should, however, keep in mind that using either of both decision criteria implies an assumption on the type and quality of knowledge the travelers have of the network state.

Since the focus of this thesis is a day-to-day setup, i.e. we want to model the route choice of a set of travelers in a daily recurring situation like commuting under the assumption that the travelers have 'learned' about the dynamic network state, we will use the actual travel time in the sequel. Using the notation of the preceding sections, we can express the dynamic user equilibrium conditions as a *nonlinear complementarity problem*:

$$\eta_p^{rs}(t) - \pi^{rs}(t) \geq 0 \tag{2.44a}$$

$$f_p^{rs}(t) \cdot \left(\eta_p^{rs}(t) - \pi^{rs}(t)\right) = 0 \tag{2.44b}$$

$$f_p^{rs}(t) \geq 0. \tag{2.44c}$$

The conditions (2.44) are expressed in terms of the route flows $f_p^{rs}(t)$. The drawback of this formulation is that an efficient solution algorithm must not enumerate all routes.

Therefore, the following link based formulation is preferable. To shorten the notation in the following lemma, we define for each link $a = (v, w)$ the reduced costs

$$\Omega_a^{rv}(t) = \pi^{rv}(t) + \tau_{(v,w)}(t + \pi^{rv}) - \pi^{rw}(t). \tag{2.45}$$

Lemma 2.3.1 *The DUE conditions (2.44) on the route flows are equivalent to the following nonlinear complementarity problem on the link inflow rates* $u_a^{rs}(t)$ *for each link* $a = (v, w)$*:*

$$\Omega_a^{rv}(t) \geq 0 \tag{2.46a}$$
$$u_a^{rs}(t + \pi^{rv}) \cdot \Omega_a^{rv}(t) = 0 \tag{2.46b}$$
$$u_a^{rs}(t + \pi^{rv}) \geq 0. \tag{2.46c}$$

A proof of lemma 2.3.1 can be found in [77]. Readers with a background in network flow problems may recall the fact that the reduced costs of a link are zero iff it is part of a minimum cost route, which directly implies the lemma.

Summary of the Network Constraints

This section gives a short summary of the network constraints for further reference.

State equations:

$$\frac{dx_{ap}^{rs}(t)}{dt} = u_{ap}^{rs}(t) - v_{ap}^{rs}(t) \tag{2.47a}$$
$$\frac{dE_{ap}^{rs}(t)}{dt} = e_p^{rs}(t) \tag{2.47b}$$

Flow conservation:

$$\sum_{a \in A(v)} v_{ap}^{rs}(t) = \sum_{a \in B(v)} u_{ap}^{rs}(t) \tag{2.47c}$$
$$f^{rs}(t) = \sum_{a \in A(r)} \sum_{p \in \mathcal{P}_{rs}} u_{ap}^{rs}(t) \tag{2.47d}$$
$$e^{rs}(t) = \sum_{a \in B(s)} \sum_{p \in \mathcal{P}_{rs}} v_{ap}^{rs}(t) \tag{2.47e}$$

Flow propagation:

$$x_{ap}^{rs}(t) = E_p^{rs}(t + \tau_a(t)) - E_p^{rs}(t) + \sum_{b \in \tilde{p}} x_{bp}^{rs}(t + \tau_a(t)) - x_{bp}^{rs}(t) \tag{2.47f}$$

Definitional constraints:

$$x_a(t) = \sum_{r,s,p} x_{ap}^{rs}(t), \quad u_a(t) = \sum_{r,s,p} u_{ap}^{rs}(t), \quad v_a(t) = \sum_{r,s,p} v_{ap}^{rs}(t) \tag{2.47g}$$

Nonnegativity constraints:

$$x_{ap}^{rs}(0) \geq 0, \quad u_{ap}^{rs}(0) \geq 0, \quad v_{ap}^{rs}(0) \geq 0 \tag{2.47h}$$

$$e_p^{rs}(t) \geq 0, \quad E_p^{rs}(t) \geq 0 \tag{2.47i}$$

Initial conditions:

$$E_p^{rs}(0) = 0 \tag{2.47j}$$

$$x_{ap}^{rs}(0) = 0 \tag{2.47k}$$

2.3.3 Formulation as Variational Inequality Problem

The following theorem gives a formulation of the DUE conditions (2.46) as a variational inequality (VI) problem. Although this VI problem usually cannot be solved directly, this formulation is very convenient from a theoretical standpoint, mainly because it provides a natural framework for the incorporation of flow propagation constraints which explicitly contain the link travel times.

Theorem 2.3.2 *A dynamic traffic flow pattern $u_a^{rs\star}(t)$ satisfying the network constraints (2.47) is a DUE state iff it satisfies the variational inequality*

$$\int_0^T \sum_{r,s} \sum_{a=(v,w)} \Omega_a^{rw\star}(t) \cdot \left(u_a^{rs}\left(t + \pi^{rv\star}(t)\right) - u_a^{rs\star}\left(t + \pi^{rv\star}(t)\right)\right) dt \geq 0, \tag{2.48}$$

where all starred variables are calculated with respect to $u_a^{rs\star}(t)$.

A proof of theorem 2.3.2 is given in [77].

2.3.4 Relaxation Procedure

The variational inequality (2.48) provides an elegant formulation of the DUE problem. However, for an actual algorithmical implementation, a NLP is more convenient. To make the problem finite-dimensional, we first discretize time in $N = \lfloor \frac{T}{\Delta t} \rfloor$ intervals of length Δt and replace the interval $[0, T]$ by the set

$$\{t_n := n\Delta t \mid n = 0 \ldots N\} \tag{2.49}$$

All functions of time are therefore replaced by N-dimensional vectors. Of course, Δt has to be chosen sufficiently small to resolve the smallest link travel time, i.e. Δt must be less than the length of the shortest link divided by the maximum possible velocity.

In each iteration of the relaxation procedure, we fix the link travel times $\tau_a(t_n)$ in the flow propagation constraints as $\overline{\tau}_a(t_n)$. Note that these estimated link travel times must be multiples of Δt since $x_a(t)$ and $E_a(t)$ are only defined for $t = n\Delta t$. Furthermore, the optimal travel times $\pi^{rv\star}(t_n)$ are fixed as $\overline{\pi}^{rv\star}(t_n)$.

Under relaxation, the variational inequality problem (2.48) is equivalent to the NLP (see [77], chapter 6)

$$\min_{\mathbf{u},\mathbf{v},\mathbf{x},\mathbf{E}} Z(\mathbf{u},\mathbf{v},\mathbf{x},\mathbf{E}) = \sum_{n=0}^{N} \sum_{a=(v,w)} \Bigg\{ \int_0^{u_a(t_k)} \tau_a(x_a(t_n), u, v_a(t_n))du + \sum_r u_a^r(t_n) \left(\overline{\pi}^{rv}(t_{k(n)}) - \overline{\pi}^{rw}(t_{k(n)}) \right) \Bigg\} \tag{2.50a}$$

subject to

$$x_{ap}^{rs}(t_{n+1}) = x_{ap}^{rs}(t_n) + u_{ap}^{rs}(t_n) - v_{ap}^{rs}(t_n) \tag{2.50b}$$

$$E^{rs}(t_{n+1}) = E^{rs}(t_n) + \sum_{a\in B(s)} \sum_p v_{ap}^{rs}(t_n) \tag{2.50c}$$

$$f^{rs}(t_n) = \sum_{a\in A(r)} \sum_p u_{ap}^{rs}(t_n) \tag{2.50d}$$

$$\sum_{a\in B(v)} v_{ap}^{rs}(t_n) = \sum_{a\in A(v)} u_{ap}^{rs}(t_n) \tag{2.50e}$$

$$x_{ap}^{rs}(t_n) = \sum_{b\in\tilde{p}} \left\{ x_{bp}^{rs}(t_n + \overline{\tau}(t_n)) - x_{bp}^{rs}(t_n) \right\} + E^{rs}(t_n + \overline{\tau}(t_n)) - E^{rs}(t_n) \tag{2.50f}$$

$$x_{ap}^{rs}(t_{n+1}) \geq 0, \quad u_{ap}^{rs}(t_n) \geq 0, \quad v_{ap}^{rs}(t_n) \geq 0 \tag{2.50g}$$

$$E^{rs}(t_{n+1}) \geq 0 \tag{2.50h}$$

$$x_{ap}^{rs}(t_0) = 0 \tag{2.50i}$$

$$E^{rs}(t_0) = 0 \tag{2.50j}$$

where $k(n)$ is defined by

$$t_n = t_{k(n)} + \overline{\pi}^{rv}(t_{k(n)}), \tag{2.51}$$

i.e. $t_{k(n)}$ is the departure time for a traveler starting from node r arriving at node v at time t_n. Since

$$\begin{aligned} \frac{\partial Z}{\partial u_a^{rs}(t_{k(n)} + \overline{\pi}^{rv}(t_{k(n)}))} &= \frac{\partial Z}{\partial u_a^{rs}(t_n)} \\ &= \tau_a(t_n) + \overline{\pi}^{rv}(t_{k(n)}) - \overline{\pi}^{rw}(t_{k(n)}) \\ &= \tau_a(t_{k(n)} + \overline{\pi}^{rv}(t_{k(n)})) + \overline{\pi}^{rv}(t_{k(n)}) - \overline{\pi}^{rw}(t_{k(n)}) \\ &= \overline{\Omega}_a^{rv}(t_{k(n)}) \end{aligned} \tag{2.52}$$

the NLP (2.50a) is equivalent to (2.48).

2.3.5 Solution Method

The NLP (2.50a) can be solved in a similar fashion as (2.5) using the Frank-Wolfe method (see A.2). We will see that, like in the static case, the linear subproblem can be solved as a shortest path problem. However, the network will not be the original one as it was in the static case, but will consist of many 'copies' of the original network (see below).

The Linear Subproblem

We denote the variables of the linear subproblem by '$\widehat{\ }$' so that the linear subproblem (A.14) reads[7]

$$\min_{\widehat{\mathbf{u}},\widehat{\mathbf{v}},\widehat{\mathbf{x}},\widehat{E}} \widehat{Z}(\widehat{\mathbf{u}},\widehat{\mathbf{v}},\widehat{\mathbf{x}},\widehat{E}) = d_{(\mathbf{u_0},\mathbf{v_0},\mathbf{x_0},E_0)} Z(\widehat{\mathbf{u}},\widehat{\mathbf{v}},\widehat{\mathbf{x}},\widehat{E}) \tag{2.53}$$

subject to the constraints (2.50b)–(2.50j) for $\widehat{\mathbf{u}}$, $\widehat{\mathbf{v}}$, $\widehat{\mathbf{x}}$ and $\widehat{E}$. The constraints (2.50b)–(2.50d) can be viewed as flow conservation constraints for new, additional nodes in the network $\widehat{\mathcal{G}} = (\widehat{\mathcal{N}}, \widehat{\mathcal{A}})$, which is shown in figure 2.9. Therefore, (2.53) can be viewed as a shortest path problem for every OD-pair with an additional flow propagation constraint (2.50f). This flow propagation constraint can be included by setting the costs of 'infeasible' links to infinity (see [77], section 6.2). Note that the shortest path problems for each OD-pair and departure time are independent and could be solved in parallel.

2.3.6 Complexity

In the previous section we have seen that the DUE problem (2.48) can be solved using the relaxation technique and the Frank-Wolfe algorithm, where the linear subproblems can be treated as a set of decoupled shortest path problems. However, the complexity of the problem increases drastically compared to the static case.

Counting the number of nodes and links in the auxiliary network (see figure 2.9) gives

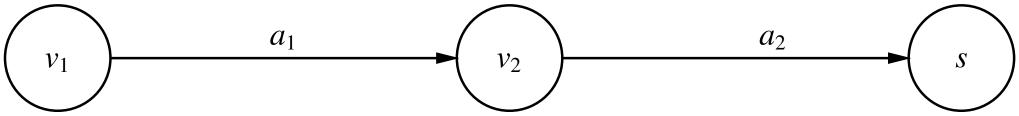
$$|\widehat{\mathcal{N}}| = (|\mathcal{N}| + |\mathcal{A}|)N + 1 \tag{2.54}$$

$$|\widehat{\mathcal{A}}| = (3|\mathcal{A}| + 1)N. \tag{2.55}$$

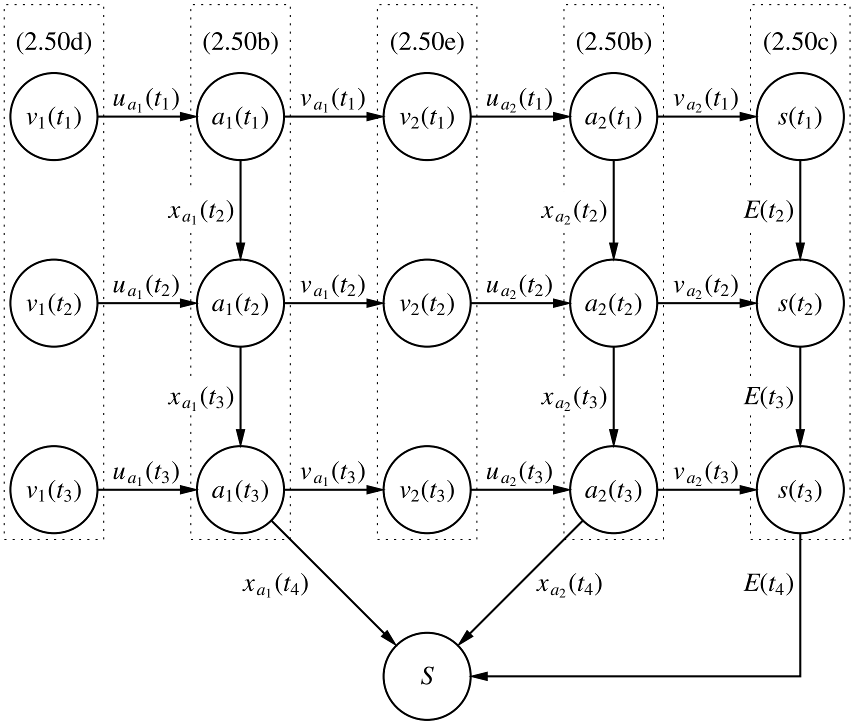
Using Dijkstra's algorithm, each shortest path problem takes $O(|\widehat{\mathcal{A}}| + |\widehat{\mathcal{N}}| \log |\widehat{\mathcal{N}}|)$ time [23, 60]. Since the number of shortest path problem increases by a factor of N compared to the static case, the time for solving one linear subproblem increases by a factor of about $4N^2$ compared to the static case. Since our goal is to simulate the traffic of a whole day and the typical time needed to traverse an arc in an urban road network would require a time step of about 1 minute, the time needed to solve the linear subproblem would increase by a factor of $8 \cdot 10^6$. Even if we would only use a time step of 30 minutes, the factor would still be 9000.

---

[7] With $d_p f$ we denote the differential of f at the point p, so that $d_p f(x) = \mathbf{x} \cdot \nabla f(x)$.



(a) Original network



(b) Auxiliary network

Figure 2.9 Auxiliary network $\widehat{\mathcal{G}} = (\widehat{\mathcal{N}}, \widehat{\mathcal{A}})$ to solve the linear subproblem (2.53) as a shortest path problem. The dashed boxes indicate the corresponding network constraint. For each link and every time step, an additional node corresponding to the state equation is added. Further, a 'super destination node' has to be added to convert the problem into a shortest path problem for every time step. The cost functions on the links are given by the corresponding components of $\nabla\widehat{Z}$.

Chapter
Three

Traffic Simulation Models

This section gives an overview of traffic simulation models. Of course, a complete treatment of all the available traffic simulation models is beyond the scope of thesis[1]. Instead we will focus on points related to traffic assignment, i.e. the calculation of user equilibria within these models.

3.1 Introduction

From the point of view of traffic assignment, traffic simulation models can be divided into two major classes: vehicle-oriented (*microscopic*) models describing the movement of individual vehicles through the network[2] and flow-based (*macroscopic*) models which do not discern individual vehicles but describe traffic as a kind of 'fluid'. For example, the constraints (2.47a)–(2.47f) define an — albeit primitive — traffic simulation model.

3.2 Macroscopic Models

Traffic assignment relies on the fact that route choice can be described in a model. In this respect, macroscopic model have the drawback that for each variable in the model, e.g. the velocity field $v(x, t)$ and the density $\rho(x, t)$, we have to introduce corresponding variables for each path. This is the reason why the analytical DTA models usually use a rather simple model of traffic flow. Nevertheless, we give a short overview of the most important macroscopic traffic models.

The first fluid-dynamical description of traffic flow was the Lighthill-Whitham model [56], which consists of the continuity equation

$$\frac{\partial}{\partial t}\rho + \frac{\partial}{\partial x}\rho v = 0 \tag{3.1}$$

---

[1] The European Community has funded a project to review the state-of-the-art traffic simulation models. See [39].

[2] From the point of view of traffic flow theory, vehicle-oriented models which do not model vehicle-vehicle interactions but use 'mean field' description of vehicle dynamics (like DYNEMO [82, 81]) are discerned as a third class called *mesoscopic*.

together with a velocity density relationship $v = V(\rho)$, giving the kinematic wave equation

$$\frac{\partial}{\partial t}\rho + c(\rho)\frac{\partial}{\partial x}\rho = 0 \tag{3.2a}$$

where the velocity $c(\rho)$ of the kinematic waves is given by

$$c(\rho) = \frac{d}{d\rho}\rho V(\rho). \tag{3.2b}$$

The major problem with the Lighthill-Whitham model is that traffic is not always in equilibrium but drivers have to *react* by accommodating their acceleration. This problem can be solved by stating an equation for the time derivative of the local velocity, i.e. the acceleration:

$$\frac{d}{dt}v(x,t) \equiv \frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} = \frac{V(\rho) - v}{\tau} - \frac{c^2}{\rho}\frac{\partial \rho}{\partial x}. \tag{3.3}$$

On the right hand side, the first term describes a relaxation to the equilibrium velocity while the second term, called *anticipation term* describes the fact that a driver slows down if the density ahead of him increases. However, the solutions of (3.3) tend to develop discontinuous shock waves. Therefore, Kühne [53] has proposed adding a viscous term to prevent the formation of discontinuities:

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} = \frac{V(\rho) - v}{\tau} - \frac{c^2}{\rho}\frac{\partial \rho}{\partial x} + \frac{1}{\rho}\frac{\partial}{\partial x}\left(\mu\frac{\partial v}{\partial x}\right). \tag{3.4}$$

Similar models have been proposed by Kerner and Konhäuser [43] and Helbing [35].

However, using one of these models as an underlying state equation for dynamic traffic assignment seems infeasible even with state of the art computers.

3.3 Microscopic Models

Microscopic, i.e. vehicle-oriented, models have the advantage that additional information needed by the DTA algorithm, e.g. origin, destination, route and departure time, can be added to the data structures of the model easily.

Although the distinction between car-following models and mesoscopic models is not directly relevant to DTA, we will adopt it for this section.

3.3.1 Car-Following Models

Despite the fact most of us know how to drive a car, there are still no generally accepted ‘first principles’ from which one can derive a unique model of car-following behavior. Some facts we can derive from everyday experience are

1. Most of the time the dynamics are collision-free.

2. Maximum velocity, acceleration and deceleration are bounded by physical limitations of the cars and the drivers.
3. Most of the time we only look forward, and in 'first order approximation' we only react to the car directly in front of us.
4. Drivers need some time to react.

However, this facts still allow many different modeling approaches. In the sequel, we assume that there are N cars on a single-lane road which are numbered such that car $i + 1$ is the car in front of car i.

Delayed Differential Equations

The fact that the reaction of drivers is delayed by their reaction time Δt suggests modeling driving behavior as a delayed differential equation

$$\frac{dv_i}{dt} = f(v_i, v_{i+1}, \Delta x_i)\big|_{t-\Delta t} , \tag{3.5}$$

where $\Delta x_i = x_{i+1} - x_i$. In fact, many such models have been proposed in the fifties.

Gazis, Herman and Rothery [33] proposed a model of the form

$$\frac{dv_i}{dt} = \alpha v_i^m(t) \frac{v_{i+1} - v_i}{(x_{i+1} - x_i)^l}\Bigg|_{t-\Delta t} , \tag{3.6}$$

where α, l and m are parameters of the model. The advantage of these models is that the model equations can be integrated directly to determine the speed-density relation.

However, these models lack any foundation in the study of human behavior. Wiedemann [90] proposed a delayed differential equation model based on *perception thresholds*, i.e. physiological limitations on the perceptions of distances and velocity differences. This approach has been developed further to sophisticated and rather detailed models of the driver-vehicle system like PELOPS [57].

For DTA applications, however, these models are computationally too costly. For example, the maximum number of vehicles which can be simulated in real-time on state of the art hardware with PELOPS is about 2000 [70], which is insufficient for the simulation of complex urban networks.

Cellular Automata and Coupled Maps

The major drawback of the delayed differential equation models is the poor computational performance. A common approach to overcome this problem is a discretization of such models with a fixed time step Δt which is usually chosen as the reaction time, i.e. in the order of one second. This yields a set of coupled maps, i.e. an update rule of the form

$$v_i(t + \Delta t) = f(v_i(t), x_i(t), v_{i+i}(t), x_{i+1}(t)) \tag{3.7}$$

which is applied to all cars simultaneously. However, instead of discretizing a given differential equation, one can also design such coupled maps directly.

While the idea behind the delayed differential equation models is to make the description of the vehicle dynamics as detailed as possible, the interesting question for the coupled map models is how minimalistic a model can be while still maintaining the fundamental features of traffic flow.

Perhaps the most minimalistic approaches are cellular automata, i.e. models in which time, space and internal state are discrete. While the first such models were proposed by Cremer and Ludwig [18] and Schütt [80], the most prominent cellular automaton model is the Nagel-Schreckenberg model [69], which has been thoroughly studied by many authors [65, 68, 79, 73].

In the Nagel-Schreckenberg model, the street is discretized in cells of length Δx. Measurements of the car density in dense jams (about 130 cars per kilometer) imply that $\Delta x = 7.5m$ is a reasonable value. In the sequel we will omit the 'natural' units Δx and Δt. Note that also the velocity is discretized in units of $\Delta x/\Delta t$. The update rules are

acceleration: $v_i(t+\Delta t) := \min\{v_{\max}, v_i(t)+1\}$

deceleration: $v_i(t+\Delta t) := \min\{v_i(t+\Delta t), x_{i+1}(t)-x_i(t)\}$

'dawdling': with probability p_{brake} set $v_i(t+\Delta t) := \max\{v_i(t+\Delta t)-1, 0\}$

translation: $x_i(t+\Delta t) := x_i(t)+\Delta t v_i(t+1)$

which are executed in *parallel* for each vehicle. Since this model uses integer arithmetic and can be even implemented using *single-bit coding* [65], it is very efficient computationally. In fact, with modern hardware it is capable of simulating about 10^6 vehicles in real-time [78].

It is very surprising that despite its very simple description of driving behavior the model reproduces the fundamental properties of traffic flow:

- Above a certain density, traffic jams occur spontaneously,
- the density-flow relation is qualitatively correct[3],
- the time interval between two cars passing a traffic light — the crucial parameter for the capacity of signalized intersections — is modeled correctly.

This is especially surprising since one of the basic properties of cars, namely the finite deceleration, is not modeled at all.

There are two shortcomings of the Nagel-Schreckenberg model. The first one is that due to the discrete nature of the model one cannot model the emissions of pollutants easily. The second one is that the Nagel-Schreckenberg model does not produce metastable states and other types of complex behavior which have been observed in measurements [44, 45, 46].

For this reasons, Krauß [50, 51] has proposed a model which can be viewed as a minimal model satisfying the four conditions stated at the beginning of this section and the assumption that imperfections in driving behavior can be modeled as stochastic fluctuations of the velocity. Taking into account the maximum deceleration b, the *safe velocity*

---

[3] Using multi-lane rules and different types of cars, the density-flow relation of the model can even be calibrated quantitatively (see [73]).

can be expressed as (see [51])

$$v_{\text{safe}} = v_{i+1} + \frac{x_{i+1} - x_i - v_{i+1}\Delta t}{\frac{v_{i+1}+v_i}{2b} + \Delta t}. \tag{3.8}$$

Using this notion of a safe velocity, Krauß extended the update rules of the Nagel-Schreckenberg model to

speed update: $v_i(t+\Delta t) := \min\{v_{\max}, v_i(t) + a, v_{\text{safe}}\}$

'dawdling': $v_i(t+\Delta t) := \max\{v_i(t+\Delta t) - a\varepsilon\xi, 0\}$

translation: $x_i(t+\Delta t) := x_i(t) + \Delta t v_i(t+1)$,

where $\xi \in [0,1]$ is a uniformly distributed random variable and ε is a parameter controlling the amplitude of the 'noise' in the acceleration. As in the Nagel-Schreckenberg model, the time step Δt is usually set to 1.

Krauß showed that depending on the maximum acceleration and maximum deceleration, the model divides into three subclasses with different types of structure formation [51, 40]. One class shows similar behavior as the Nagel-Schreckenberg model, one class also shows metastable states which are observed in real-world data, and one class shows no structure formation at all. The model with realistic parameters[4] belongs to the class which shows metastable states.

Since the Krauß model uses floating point arithmetic and involves a division, it is not as fast as the Nagel-Schreckenberg model. However, the loss in computational performance is only about 50%, comparing two optimized single-lane implementations [51]. Since in network simulations about half of the CPU time is needed by the lane-changing rules and IO operations, the difference between both models is only about 25% for simulations of real networks with PLANSIM-T(see Appendix B.1).

Both the Nagel-Schreckenberg model and the Krauß model are fast enough for simulations of large networks like the Wuppertal network, and due to their microscopic nature they are suited for the simulation-based traffic assignment algorithm described in chapter 4. In fact, iterated simulations using the Nagel-Schreckenberg model have been used for DTA purposes by the TRANSIMS group [67, 66]. However, for doing multiple iterations of the traffic of a whole day these models would consume much computing time and the use would be restricted to high performance computers. The models described in the next section can reduce the needed computing resources significantly.

3.3.2 Mesoscopic Models

The main advantage of driver oriented models for DTA is the ability to track individual drivers and to associate routing information with them. While the description of car-following behavior is interesting from the theoretical point of view and for some applications like the modeling of emissions, it is not needed for DTA. The mesoscopic models which are driver oriented but do not model car following behavior are therefore the ideal models for use in DTA applications.

---

[4] Typical cars have $a \approx 0.8ms^{-2}$ and $b \approx 4.5ms^{-2}$, and the typical speed on German highways is $v_{\max} \approx 36ms^{-1}$. Like in the Nagel-Schreckenberg model, we choose $\Delta x = 7.5m$ and $\Delta t = 1s$.

Many different approaches to mesoscopic modeling of traffic flow have been proposed. Schwerdtfeger has proposed DYNEMO [82, 81], a kind of 'mean field' model in which the velocity of a car depends on the traffic density on a link. Daganzo [19, 20, 15] has proposed a cell transmission model which can be viewed as a spatial discretization of an underlying fluid-dynamical model. The simplest mesoscopic model, however, is to model each link in a network as a queue with some capacity. Since such a model was developed and implemented for the DTA algorithm described in section 4, we will describe the model more detailed in the next section.

3.4 *FASTLANE*: A Queuing Model

This section describes FASTLANE[5], a mesoscopic traffic simulation tool based on a very simple queuing model. The basic idea of this model is that the travel time on a link is the sum of the time needed to travel along the length of the link and the time which may be spent waiting in a queue, and that the characteristics of a link can be described by its length, its capacity and by the number of cars which fit into the link if the link is jammed. In contrast to the cell transmission model mentioned in the previous section the links are not divided into smaller parts.

We will see that despite its simplistic approach, this model provides a good approximation of the travel times in the Nagel-Schreckenberg and Krauß models.

3.4.1 Model Description

Links

In the FASTLANE model, each link in a road network each link is described by its capacity $q_{\max}$ (maximum flow), the length l (to calculate the travel times) and the maximum number of cars which fit into the link.

When a vehicle enters a link, a travel time t_{travel} is calculated from the length and the current state of the link, i.e. the current number n of cars on the link, and the desired velocity v_0 of the vehicle. The vehicle is the stored until the travel time has elapsed, or — technically speaking — the vehicle is put into a priority queue with a priority corresponding to the calculated time of arrival at the end of the link.

However, if we would only use this travel time to move vehicles across a link, we would neglect important effects like queuing and spill-back. Therefore, the number of vehicles which may leave the link after they have arrived at the end of the link is limited by both by the capacity of the link and the number of vehicles which fit into the next link. The total number of cars which fit into the link is usually calculated as the maximum traffic density $\rho_{\max}$ times the length l. However, this could cause capacity bottlenecks if there are short links[6] since not even $q_{\max}\Delta t$ vehicles might fit on the link. For this reason, FASTLANE by default enforces this number to be at least $2q_{\max}\Delta t$.

---

[5] **Fa**st **S**imulator for **T**raffic in **La**rge **Ne**tworks

[6] For example, the 'Seestrauch' network contains 'artificial' links of length zero to model turning restrictions.

If there is more than one outgoing link, each vehicle chooses its next link according to its associated route plan and the capacity restrains are handled separately for each outgoing link[7]. If a link has more than one in-going link and the total number of vehicles which may enter the link exceed the capacity of the link, the available capacity is shared proportionally according to priorities associated with the links. In this way, right of way and green-time shares can be modeled.

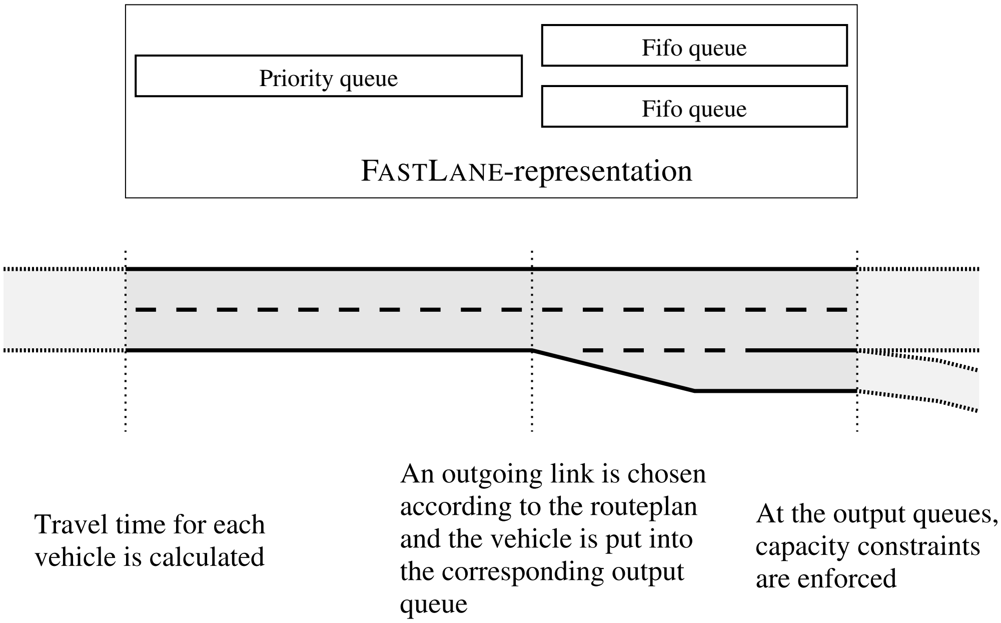

Fifo queue
Priority queue
Fifo queue
FASTLANE-representation
Travel time for each vehicle is calculated
An outgoing link is chosen according to the routeplan and the vehicle is put into the corresponding output queue
At the output queues, capacity constraints are enforced

Figure 3.1 Update algorithm of FASTLANE: For each vehicle entering a link, the free flow travel time is calculated and the vehicle is stored in a priority queue with a priority equal to the expected time of arrival at the end of the link. At each time step, cars which have 'arrived' at the end are put into the output queue corresponding to their destination. The output queues have a fixed capacity (vehicles per second) and cars can only leave an output queue if there is sufficient space on the next link.

Both in reality and in simulation models, the outflow of a link is not deterministic but fluctuates stochastically, resulting in a fluctuating jam length and thus a fluctuating travel time. Since these fluctuations might be important for the DTA algorithm, the capacity of the links is not constant but modeled as a random variable, i.e. the number of vehicles which are taken from the output queues and put into the next link is determined by a random number generator (see 3.4.3).

An important parameter of the link model is the function to determine the expected travel time. In principle, any relation between traffic density and velocity could be used for this purpose. However, it turned out in the calibration process that this is not even

---

[7] This corresponds to infinitely large turn pockets. However, realistic turn pockets can be modeled by adding an additional link for each direction.

necessary. Just setting $t_{\text{travel}} = l/v_0$, i.e. neglecting any density dependence of the expected travel time, reproduces the travel times of the Nagel-Schreckenberg model with an error of less than five per cent.

Nodes

The nodes of a network can be used to implement signalized and unsignalized intersections. However, the current version of FASTLANE completely ignores delays at the nodes other than the queuing delays on the links resulting from the capacity constraints. The only way to model right of way in the current version is to specify the portion of the available capacity which is assigned to a link if the number of vehicles which want to enter a link exceeds the capacity of the link.

Temporal Resolution

The performance of a simulation model is usually related to the size of the time step of the model. Therefore, one would like to choose the time step as large as possible. However, this might result in systematic errors in the travel time due to the fact that the time to traverse an edge is always a multiple of the time step. Especially, we have to ensure that the rule used to round the travel times is not biased. This is achieved by using a stochastic rounding rule

$$[x]_{\text{stoch}} = \begin{cases} \lceil x \rceil & \text{with probability } x - \lfloor x \rfloor \\ \lfloor x \rfloor & \text{with probability } \lceil x \rceil - x, \end{cases} \tag{3.9}$$

which does not change the expectation value of x, to round $\frac{t_{\text{travel}}}{\Delta t}$.

3.4.2 Implementation

FASTLANE was implemented in C++ using the LEDA library (c.f. [60]), a C++ class library providing all common data structures and very useful data structures to represent graphs and to associate data with links and edges.

FASTLANE is much less complex than PLANSIM-T. While PLANSIM-T has grown to more than 20000 lines of code, FASTLANE consists of less than 3000 lines of code. Details on the format of the input files and a short introduction on how FASTLANE is used for DTA can be found in appendix B.2.

3.4.3 Calibration and Validation of the link model

Since the main objective is to calculate the user-optimal routes for a given time-dependent ODM as an input for a car-following model, the travel times in the queuing model should be a good estimate for the travel times for the respective car-following model. In the sequel it is shown that the parameters of the queuing model can be chosen such that it reproduces the travel times of the Nagel-Schreckenberg and Krauß models even in situations with a non-trivial time-dependence of the queue lengths.

Calibration of a Single Bottleneck

Since the capacity of a single link can be measured in a straightforward way, the simplest nontrivial network for the calibration process is a network with a single bottleneck.

In the car following models (Krauß and Nagel-Schreckenberg) the bottleneck was represented by a merging of two lanes into one lane. In the Nagel-Schreckenberg model, we used the standard time and length scales of $\tau = 1s$ resp. $\lambda = 7.5m$, a maximum velocity of $v_{\max} = 5\lambda/\tau$ and a deceleration probability of $p_{\text{brake}} = 0.2$. For the Krauß model, the same time and length scales were used and the parameters were chosen as $a = b = 0.4$ and $\varepsilon = 0.5$.

The input flow q to the bottleneck was changed periodically according to

$$q(t) = A - 0.5 \left(\cos \frac{2\pi t}{\text{day}} - \cos \frac{4\pi t}{\text{day}} \right)$$

which shows the typical structure of two ‘rush-hours’ per day. The parameter A was chosen in a way that the queue does not fully dissolve between the rush hours. This was achieved by setting $A = 0.5$ for the Nagel-Schreckenberg model and $A = 0.35$ for the queuing model. For all models the results were averaged over 400 simulation runs.

For the queuing model, the time step was set to $30s$ and the capacity of the link was modeled by a normal distribution[8] whose parameters were set to the values measured in the Nagel-Schreckenberg model.

Figure 3.2 compares the travel time and the number of vehicles as a function of time between the queuing model and the Nagel-Schreckenberg model. Both mean values and standard deviation of flow and travel time are reproduced very well. Even the decrease of the queue length at noon and the increase in the afternoon is reproduced by the queuing model very well, although the car following behavior is neglected completely.

Figure 3.3 shows the same comparison for the Krauß model and FASTLANE. In this case, during the rush hour the travel times measured in both models differ by up to 12%. This is due to the fact that the Krauß model exhibits metastable states with a flow exceeding the capacity. Due to this metastability, the flow in the Krauß model is greater than the flow in the queuing model for a certain time period (see figure 3.4), resulting in a shorter queue length and shorter travel time compared to the queuing model.

Network Calibration

Even if the parameters for the links are calibrated, the travel times of vehicles in large networks might differ between the queuing model and the car following models since interactions at nodes are neglected in the queuing model. Further, turning restrictions in urban networks are modeled by auxiliary nodes and links of length zero in PLANSIM-T. While vehicles can travel through these links without loss of time in PLANSIM-T, the time to travel trough a link in the queuing model is at least one time step.

Figures 3.5 and 3.6 compare the travel time distribution[9] of both models for the ‘Seestrauch’ network (c.f. C.1), which includes both highways and urban roads. For a

---

[8] Negative capacity values were cut off.

[9] More precisely: The distribution of mean travel times, sampled over 120 seconds.

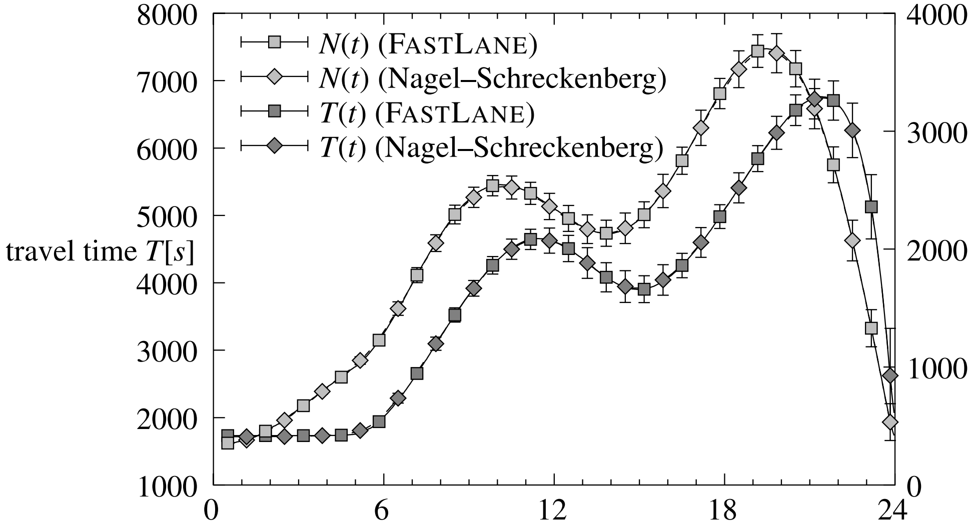


Figure 3.2 Comparison of queuing model and the Nagel-Schreckenberg model: The curves show the travel time and the number of vehicles in the system, i.e. the queue length, as a function of time for both models. The 'errorbars' indicate the standard deviation measured in 400 simulation runs. The correspondence of the curves of both models is so good that the curves are nearly indiscernible. Even the standard deviations of the travel time and the queue length are reproduced very well.

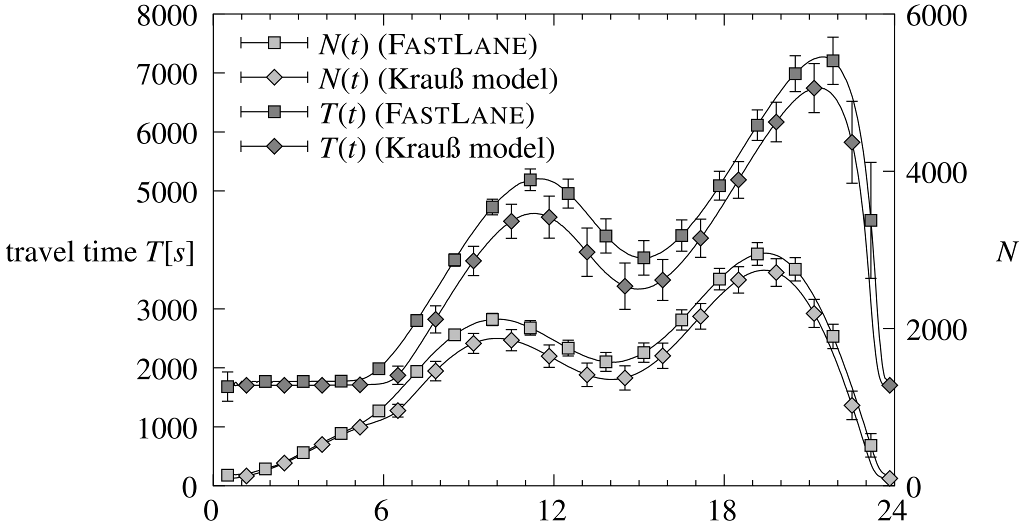


Figure 3.3 The same comparison between FastLane and the Krauß model. In this case, the deviations of the travel times are much larger (up to 12%). These deviations are due the metastable states with a flow greater than the capacity (see figure 3.4).

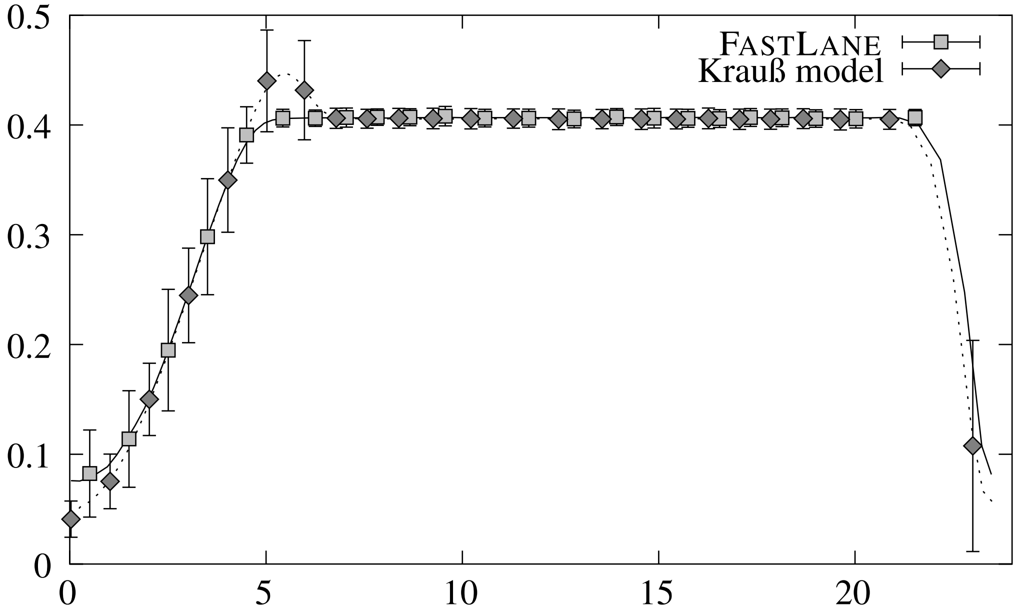


Figure 3.4 Comparison of the flows in the queuing model and the Krauß model. At the beginning of the 'rush hour', the flow in the Krauß model temporarily exceeds the capacity (defined as the long-term sustainable flow). This metastability results in shorter travel times of the Krauß model compared to the queuing model.

single origin destination pair (figure 3.5), the travel times differ only slightly by about 2%. Looking at all origin destination pairs, the travel time difference is still less than 4%, but the distribution of the travel times is much narrower in FASTLANE. Presumably, this is due to the fact that FASTLANE neglects interactions at intersections.

3.4.4 Computational Performance

The major advantage of the queuing model is that each vehicle has to be 'touched' only three times during its travel through a link — first it has to be stored in the priority queue, then it has to be removed from the priority queue and put in an output queue, and finally it has to be removed from the output queue and handed over to the next link. This is quite different from car following models, where the computational burden imposed by a single car is usually proportional to the travel time and hence the length of the link.

Since the output queues are simply stacks, a car can be added and removed from the output queue in $O(1)$ time. The only part of the update with a complexity depending on the characteristics of the link is the storage of the car in the priority queue, which takes $O(logn)$ time, where n is the number of cars already stored in the priority queue[10]. Since the number of cars traveling on a link can be bounded by the maximum density $\rho_{\max}$ of a link times the length of the link, the complexity for putting a car on a link grows at most

---

[10] FASTLANE uses the priority queues of the LEDA library which are implemented using Fibonacci heaps [60].

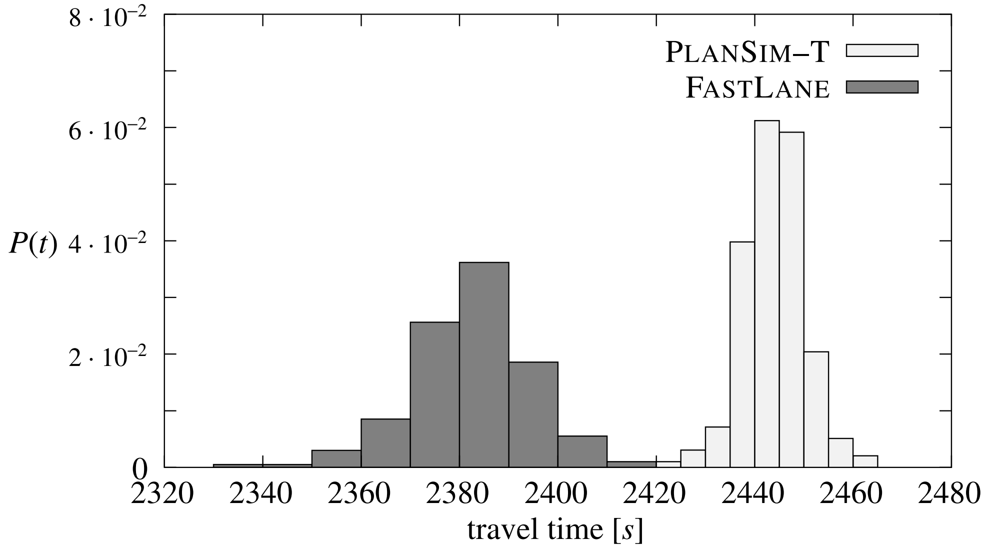


Figure 3.5 Distribution of travel times (sampled over 120 seconds) for a single origin destination pair of the 'Seestrauch' network for PLANSIM-T (using the Nagel-Schreckenberg model) and FASTLANE (with a time step of 10 seconds). The mean travel time differs by about 60 seconds or 2%.

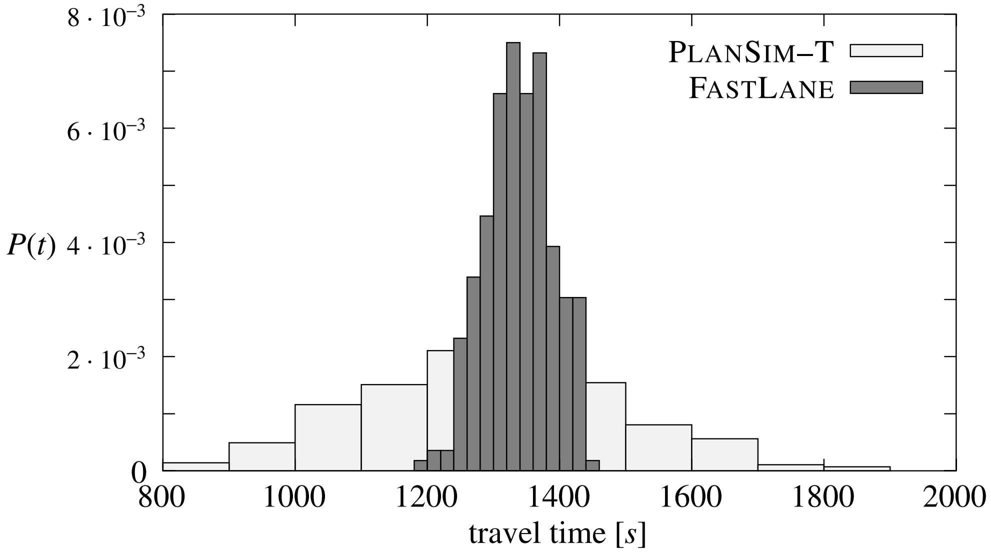


Figure 3.6 Distribution of mean travel times for all origin destination pairs in the 'Seestrauch' network (sampled over 120 seconds) for PLANSIM-T and FASTLANE. The mean travel times in both models differ less than 4%. However, the variance is much smaller in FASTLANE, presumably due to the fact that interactions at intersections are neglected.

logarithmically with the length of the link. Figure 3.7 shows that the computing time indeed increases approximately logarithmically with the length of a link.

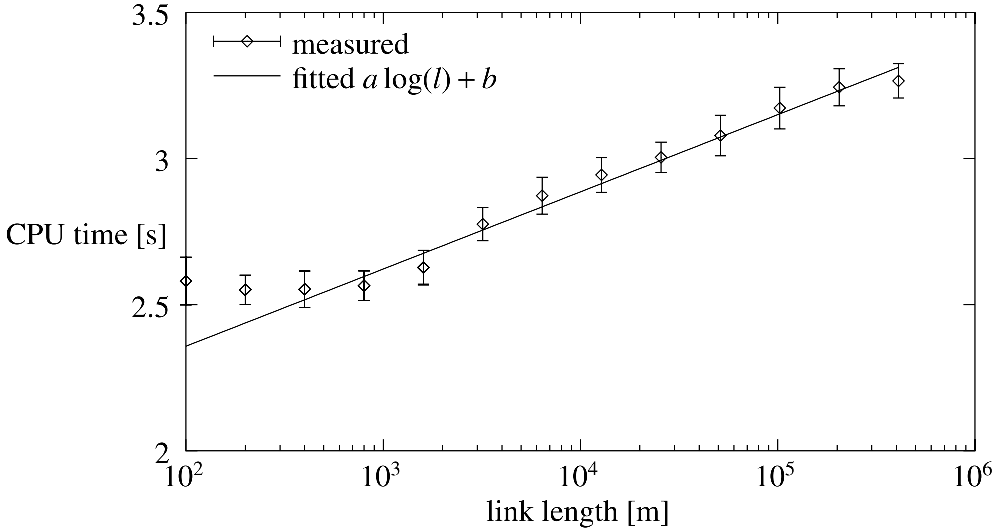


Figure 3.7 Influence of the link length on the computing time needed to simulate the travel of a fixed number of cars through a single link. Since the number of vehicles on a link in a steady-state situation increases linear with the length of the link, we expect the computing time to grow logarithmically with the link length. The fitted curve of the form $a\log(x) + b$ shows that this is indeed approximately true. The deviations for small link lengths may be due to the fact that the 'effective' length $n_{\max}/\rho_{\max}$ of a link is at least $2q_{\max}\Delta t/\rho_{\max}$, which is about $250m$ in this example.

If we neglect the overhead to execute a time step regardless of whether cars are updated or not, the complexity should not depend on the size of the time step of the model at all. Therefore we expect the computing time to decrease as $O(1/\Delta t)$ since the overhead is proportional to the number of time steps. However, this is not entirely true since the number of cars which are put into the priority queue per time step — and therefore the mean cost of storing a vehicle in the priority queue — increases with the size of the time step. Figure 3.8 shows how the computing time depends on the size of the time step empirically.

Table 3.1 compares the computational performance of the queuing model and the Nagel-Schreckenberg model. Depending on the implementation, the computation time for a single link in the Nagel-Schreckenberg model grows linear either with the length l of the link or the number $n = \rho l$ of cars on the link, where ρ is the average density of cars. The computation time in the queuing model consists of a part which does not depend on the link length but only the flow through the link (i.e. calculation of arrival times) and the time to insert the vehicles in the priority queue, which grows logarithmically with the number of vehicles in the queue.

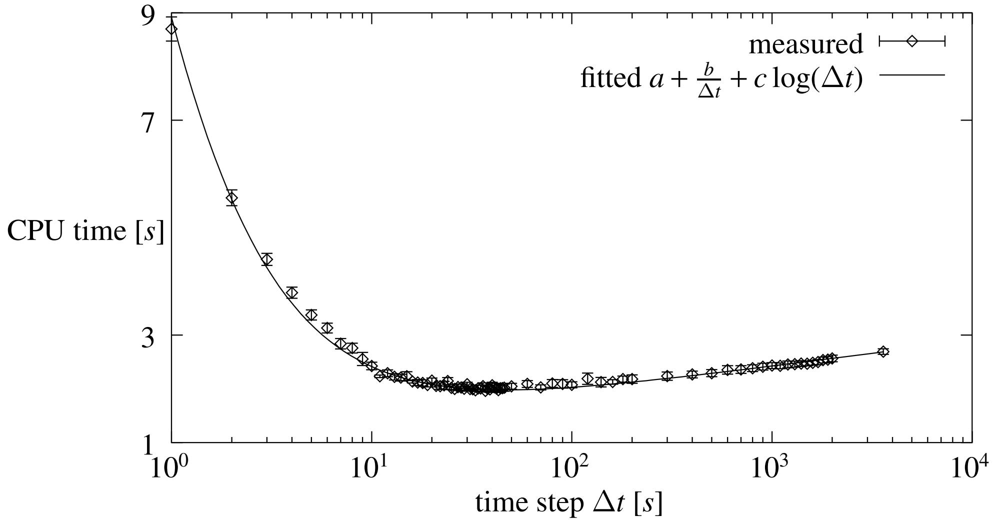


Figure 3.8 Influence of the time step size on the computing time. Each time steps adds a constant overhead, summing up to a total overhead proportional to $\frac{1}{\Delta t}$. Since the mean time for a vehicle update consists of the time to put a vehicle in the priority queue (growing logarithmically with the expected number of cars already in the queue, which is proportional to Δt) and some constant time, we would expect the total CPU time to be of the form $a + \frac{b}{\Delta t} + c \log \Delta t$, which fits the data quite well.

				av. link	CPU time [s]		
Network	links	nodes	trips	length [km]	NaSc	Queue	factor
Bottleneck	2	3	43000	0.4	430	3	143
Wuppertal	16769	9098	$1.6 \cdot 10^6$	0.3	32000	3700	8.6
NRW	990	487	$4.3 \cdot 10^6$	2.4	47780	559	85

Table 3.1 Comparison of the computational performance of the Nagel-Schreckenberg model and the queuing model for different networks. Network 'Wuppertal' refers to an urban network, 'NRW' to a highway network. 'Bottleneck' is the calibration network with only a single bottleneck. All simulations were run on a 166MHz UltraSPARC processor.

Chapter Four

Simulation-based Traffic Assignment

As we have seen in chapter 2, the analytical assignment models are either not capable of modeling congested networks or too complex to be solved for large networks. When calculating the route choices for use in a traffic simulation model, these analytical models have also the disadvantage that the cost function used in the assignment model may not be consistent with the travel times in the simulation model. Further, a realistic modeling of the propagation of flow and of the interactions at intersections is difficult within the analytical models.

Therefore, it is not surprising that traffic assignment models based on traffic simulation models have been proposed by several authors. Prominent models are CONTRAM [55], a model at the Transportation Research Laboratory, INTEGRATION [87, 75] developed by van Aerde et al. at Queen's University, Canada, DYNASMART [58] developed by Mahmassani et al. at the University of Texas at Austin, and DynaMIT [6, 5, 16] developed by Ben-Akiva et al. at the MIT. However, these models have not been applied to networks with more than a few hundred links due to performance reasons.

In this chapter we present an approach to the assignment model which is based on iterated simulation. Although on first sight this approach seems to be even more computationally complex than the analytical models, it has successfully been applied to determine the dynamic user equilibrium for a whole day in a large network with more than 15000 links (the urban network of Wuppertal[1]) using the queuing model described in section 3.4. In fact, using a similar queuing model, the TRANSIMS group has solved the dynamic traffic assignment problem for the morning peak period in an even larger network.

One of the major drawbacks of the simulation based assignment models is that studying the convergence and the stability of these models is difficult. Although the existence and uniqueness of fixed points of iterative dynamic routing/assignment methods have been studied in a general context [41], these results are not applicable to travel time functions which result from a simulation model. For example, one of the key assumption, namely that the travel time is a continuous function of the departure time, does not hold in a microscopic simulation model which includes signalized intersections. Therefore, the discussion of the stability of the algorithm in section 4.2 is limited to rather simple types of instabilities.

---

[1] Wuppertal, a city with about 400,000 inhabitants in Nordrhein-Westfalen, was the study area of the FVU project.

In the sequel we will always assume that the departure time of each traveler is fixed. Including departure time choice would be straightforward, but would slow down the convergence of the algorithm since the space of possible choices would become much larger. Further we will focus on automobile traffic, and will often use the terms ‘traveler’ and ‘driver’ synonymously.

4.1 Probabilistic Modeling of Route Choice

4.1.1 Introduction

A pragmatic way to find the time-dependent user equilibrium would be an iterative simulation:

1. Initialize the route of each driver by the optimal route in the empty network.
2. Calculate the time dependent costs of the links by simulation.
3. Recalculate the optimal routes of a certain portion p of the drivers using the time dependent costs from step 2.
4. If routes have changed in step 3, go to step 2.

In fact, a modified version of this approach is being used by the TRANSIMS group [85] to calculate the route choices in large networks [66, 42, 67].

However, in many cases this process tends to produce oscillations in the route choices or is unstable [17]. Even in the case of a network with only two identical parallel links the obvious equilibrium, namely 50% of the drivers using either route, will not be stable since even small fluctuations in travel time will lead to a state where in every iteration the portion p of drivers will change to the route which was used by less drivers in the previous iteration.

For this reason, we use a slightly different approach where each driver d knows a set $\mathcal{P}_d$ of routes. The route choice behavior is modeled by a probability distribution on this set of routes which is used to pick a random route from $\mathcal{P}_d$ in each iteration. In the example network with only two parallel routes, the user equilibrium would correspond to the state in which for each of the two possible routes the probability of being chosen is 0.5. We will see in section 4.2 that if the parameters of the update rules for the probability distribution are properly chosen, this state will be a stable fixed point of the algorithm.

4.1.2 Traveler Model

The first step in our approach is to disaggregate the time-dependent OD-matrix into a set of drivers with a fixed departure time. Each driver d is described by the following variables:

- an origin O_d, a destination D_d and a departure time t_d,
- a set $\mathcal{P}_d$ of routes from O_d to D_d,
- a probability distribution $p_d : \mathcal{P}_d \to \mathbb{R}_+$ with $\sum_{r\in\mathcal{P}_d} p_d(r) = 1$,

- ‘learned travel times’ $\tau_d : \mathcal{P}_d \to \mathbb{R}_+$.

Of course, no efficient algorithm can enumerate all routes in a network, so $\mathcal{P}_d$ can only contain a small subset of all possible routes. This problem can be handled in different ways:

1. Precalculate a ‘feasible’ set of routes, for example the k shortest paths which do not use any node more than once [65, 72, 26].
2. Start with only a single route, i.e. the shortest path, for each driver. After each iteration of the algorithm below, check if the portion of each drivers travel time spent in queues exceeds some limit. If so, calculate the shortest path according to the time dependent link cost functions generated by the last simulation run and add this route to $\mathcal{P}_d$.

4.1.3 Update Rules

This section describes the update rules of our model.

Update of Travel Times

In each simulation, a driver d chooses a route r from $\mathcal{P}_d$ according to the probability distribution p_d. After the simulation, the travel times $\tau_d(r)$ are updated according to

$$\tau'_d(r) = \tau_s(r) \tag{4.1a}$$

$$\tau'_d(s) = \beta\tau_g(s) + (1-\beta)\tau_d(s) \quad \forall s \in R \setminus \{r\} \tag{4.1b}$$

where $\tau_s(r)$ is the travel time of route r measured in the simulation and $\tau_g(s)$ is the travel time of route s calculated from the time dependent link costs generated by the simulation. Rule 4.1b prevents drivers from ‘remembering’ the travel times of routes they have not used for a long time. In the simulations done for this thesis, β was set to 0.05.

Update of $\mathbf{p_d}$

The update rule for p_d has to increase the probability of choosing a route which has a low travel time and has to decrease the probability of choosing a route which has a high travel time.

More specific, for each pair $(r, s) \in \mathcal{P}_d$ the relative cost difference

$$\delta_{rs} = \frac{\tau_d(s) - \tau_d(r)}{\tau_d(s) + \tau_d(r)} \in [-1, 1] \tag{4.2}$$

is calculated, which is a good measure to compare two routes and has the advantage of being bounded to a fixed interval.

The update rules for the probabilities $p_d(r)$ are formulated as

$$p'_d(r) = f(\delta_{rs}), \tag{4.3a}$$

$$p'_d(s) = p_d(r) + p_d(s) - p'_d(r) \tag{4.3b}$$

where $f : [-1,1] \to [0, p_d(r) + p_d(s)]$ is a monotonic differentiable function with the properties

$$f(-1) \geq 0, \tag{4.4a}$$
$$f(1) \leq p_d(r) + p_d(s), \tag{4.4b}$$
$$f(0) = p_d(r), \tag{4.4c}$$

which ensure that p'_d is a probability distribution.

Furthermore, we want to ensure that symmetric fluctuations of δ_{rs} around zero are mapped to symmetric fluctuations around $p_d(r)$. Otherwise, stochastic fluctuations of the travel times at the equilibrium would result in a systematic drift away from the equilibrium. It is easy to see that this implies that $f''(0)$ should be near zero, i.e. f should be approximately linear around zero.

The function

$$f(x) = \frac{p_d(r)\,(p_d(r) + p_d(s))\,g(x)}{p_d(r)g(x) + p_d(s)}, \tag{4.5}$$

with

$$g(x) = \exp\left(\frac{ax}{1-x^2}\right) \tag{4.6}$$

satisfies the conditions above. Figure 4.1 shows f for $a = 1$, $p_d(r) = 0.8$ and $p_d(s) = 0.2$.

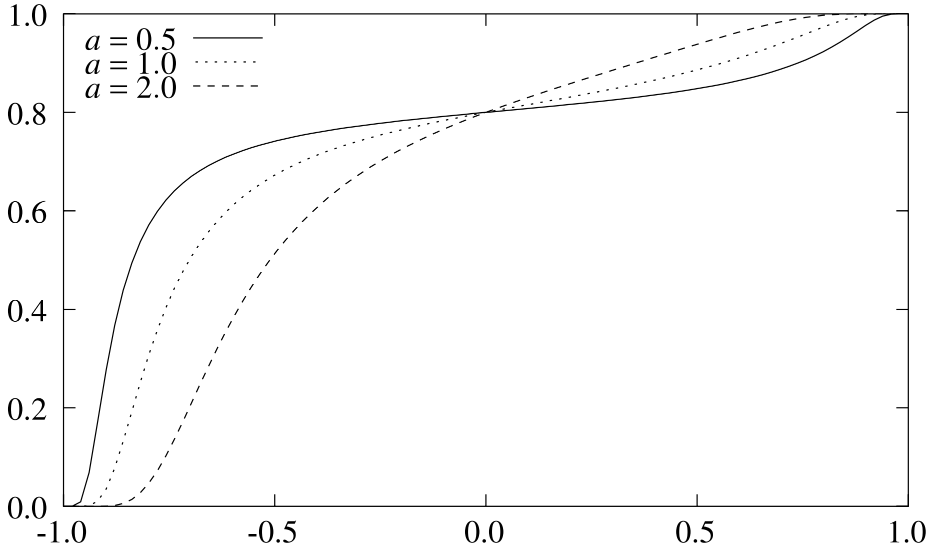


Figure 4.1 Graph of function f (see eq. (4.5)) for $p_d(r) = 0.8$ and $p_d(s) = 0.2$ and different values of a.

4.2 Stability Analysis

The simplest network to analyze the stability of the algorithm is a net with only two nodes O and D and two routes r_1 and r_2 of equal length from O to D which have only one bottleneck at the end of each route. For simplicity, a time continuous formulation is used in the sequel although the simulation model uses discrete time steps.

Let $d(t)$ denote the demand, v the velocity of the cars, C_i and l_i the capacity respectively the length of route r_i, $Q_i(t)$ the number of cars in the queue of r_i at time t and $T_i(t) = \frac{l_i}{v} + \frac{Q_i(t+l_i/v)}{C_i}$ the travel time for vehicles which use route r_i and start at time t.

Wardrop's first principle $T_1(t) = T_2(t)$ in this case is simply

$$\frac{l_1}{v} + \frac{Q_1(t+l_1/v)}{C_1} = \frac{l_2}{v} + \frac{Q_2(t+l_2/v)}{C_2}. \tag{4.7}$$

The number of vehicles entering route i per unit time at time t is $d(t)p_i(t)$ and these vehicles enter the bottleneck queue at time $t+l_i/v$. During an infinitesimal interval dt, the number of vehicles leaving the queue is $C_i dt$ provided there are vehicles in the queue. So the dynamics of the queues are given by

$$\dot{Q}_i(t+l_i/v) = p_i(t)d(t) - \begin{cases} C_i & \text{if } Q_i(t+l_i/v) > 0 \\ 0 & \text{if } Q_i(t+l_i/v) = 0. \end{cases} \tag{4.8}$$

For the 'jammed case' with $p_i d > C_i$, the equilibrium condition $\dot{T}_1(t) = \dot{T}_2(t)$ yields

$$p_i(t) = \frac{C_i}{C_1 + C_2} \tag{4.9}$$

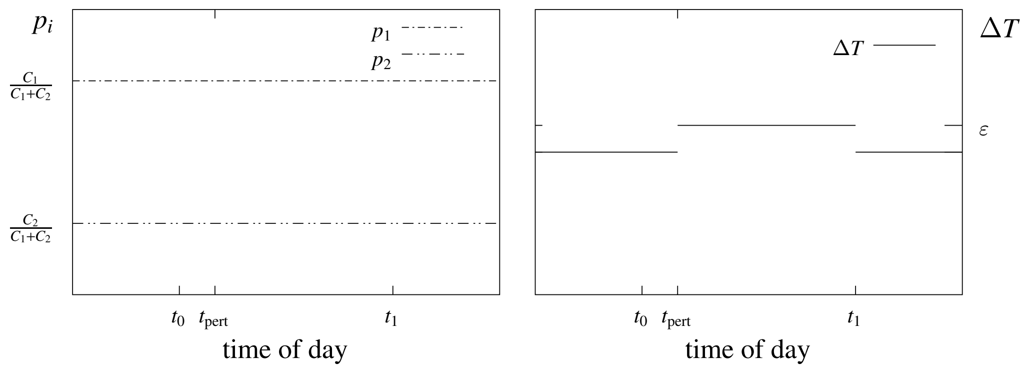
as expected.

Let us assume there is a congestion period starting at t_0 and ending at $t_1 = t_0 + \Delta t$ and at some time t_{pert} during this period a perturbation of the user equilibrium leads to a travel time difference $\Delta T(t_{\text{pert}}) = T_2(t_{\text{pert}}) - T_1(t_{\text{pert}}) = \varepsilon$. Since the p_i fulfill the equilibrium condition $\dot{T}_1(t) = \dot{T}_2(t)$, this travel time difference remains constant until the queue dissolves. According to the update rules, all drivers starting during the congestion period after t_{pert} will change p_1 by the amount

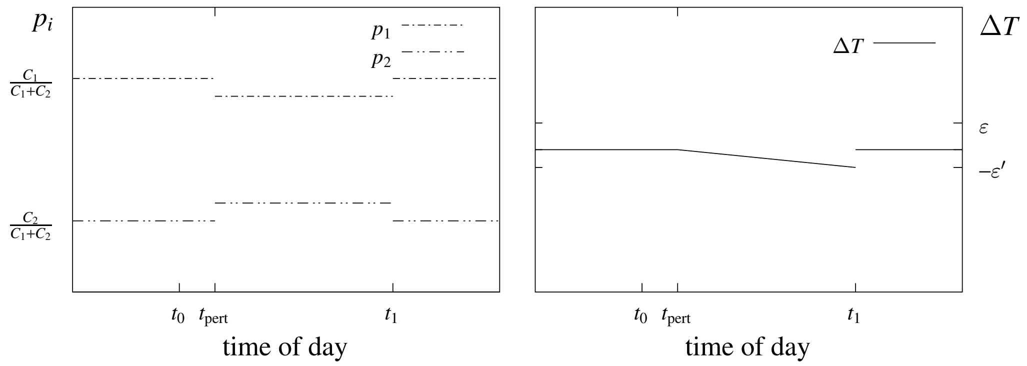
$$\Delta p(t) = p_1(t) - p_1'(t) = -\left(p_2(t) - p_2'(t)\right) = p_1(t) - f\left(\frac{\varepsilon}{T_1(t) + T_2(t)}\right) \tag{4.10}$$

where p_i' denotes the updated probability of choosing route r_i.

During the next iteration, this perturbation of p_1 leads to different travel times for the drivers which start after t_{pert} during the congestion period, resulting in a travel time



iteration n



iteration $n + 1$

Figure 4.2 Consequences of a perturbation of the travel times in a *congested* network with only two parallel links, assuming that the route choice probabilities satisfy the equilibrium conditions (4.9). If a perturbation at some time t_{pert} leads to a difference of size ε in the travel times, this difference will — due to the equilibrium condition $\dot{T}_1(t) = \dot{T}_2(t)$ — remain constant until the end of the congestion period when the queues on both links dissolve. The update rule will shift the p_i away from their equilibrium values. In the next iteration, this leads to more traffic on one link, so the queue on that link will grow faster, giving the travel time difference $\Delta T'(t)$. The stability of the DTA algorithm is ensured if the maximum ε' of $|\Delta T'(t)|$ is smaller than ε.

difference $\Delta T(t)$. We have

$$\begin{aligned}\frac{d}{dt}(\Delta T(t)) &= \frac{\dot{Q}_2(t+l_2/v)}{C_2} - \frac{\dot{Q}_1(t+l_1/v)}{C_1}\\ &= \frac{p_2'(t)d(t)-C_2}{C_2} - \frac{p_1'(t)d(t)-C_1}{C_1}\\ &= \frac{(p_2(t)+\Delta p(t))\,d(t)-C_2}{C_2} - \frac{(p_1(t)-\Delta p(t))\,d(t)-C_1}{C_1}\\ &= d(t)\left(\frac{1}{C_1}+\frac{1}{C_2}\right)\Delta p(t)\end{aligned} \tag{4.11}$$

and therefore

$$\begin{aligned}\Delta T(t) &= \int_{t_{\text{pert}}}^{t} d(\tau)\left(\frac{1}{C_1}+\frac{1}{C_2}\right)\Delta p(\tau)d\tau\\ &\le \max_{t\in[t_0,t_1]} d(t)\Delta t\left(\frac{1}{C_1}+\frac{1}{C_2}\right)\Delta p(t).\end{aligned} \tag{4.12}$$

The stability of algorithm is ensured if

$$\max_{t\in[t_0,t_1]}|\Delta T(t)| < \varepsilon. \tag{4.13}$$

Taylor expansion using (4.10) yields

$$\begin{aligned}\max_{t\in[t_0,t_1]}|\Delta T(t)| &\le \max_{t\in[t_0,t_1]} d(t)\Delta t\left(\frac{1}{C_1}+\frac{1}{C_2}\right)|\Delta p(t)|\\ &\le \max\frac{d(t)\Delta t}{T_1(t)+T_2(t)}\left(\frac{1}{C_1}+\frac{1}{C_2}\right)\left(\varepsilon f'(0)+\frac{\varepsilon^2}{2}\left|f''(\theta\varepsilon)\right|\right).\end{aligned} \tag{4.14}$$

Combining this with (4.13) yields the stability condition

$$f'(0) < \frac{C_1C_2}{C_1+C_2}\frac{1}{\Delta t}\min_{t\in[t_0,t_1]}\frac{T_1(t)+T_2(t)}{d(t)} \tag{4.15}$$

on $f'(0)$ and an expression for the domain of stability

$$\varepsilon < \min_{\theta\in]0,1[}\frac{1}{|f''(\theta\varepsilon)|}\left(\frac{C_1C_2}{C_1+C_2}\frac{1}{\Delta t}\min_{t\in[t_0,t_1]}\frac{T_1(t)+T_2(t)}{d(t)} - f'(0)\right). \tag{4.16}$$

For the special update rule (4.5) condition (4.15) is equivalent to

$$a < \frac{C_1+C_2}{\Delta t}\min_{t\in[t_0,t_1]}\frac{T_1(t)+T_2(t)}{d(t)}. \tag{4.17}$$

From the point of view of performance, a larger value of a is desirable. Equation (4.17) gives a theoretical limit on the value of a above which we cannot expect the algorithm to converge. This condition has a plausible explanation: The higher the capacity compared to the demand and the longer the travel times compared to the duration of the rush hour, i.e. the less jammed the network is and the less important the queuing delays are compared to the travel times, the less susceptible to perturbations is the algorithm.

Of course, the above discussion covers only a simple type of instability. However, it turns out that a larger number of available routes and a larger number of origin destination pairs tend to *improve* the stability of the algorithm. This is not surprising since the variance of a sum of random variables decreases with the number of summands. Another indication of the type of instability discussed in this section being indeed the most important one is the fact that the iteration process produces oscillating route choice probabilities. Figure 4.3 shows a typical example. These oscillations are produced by the same

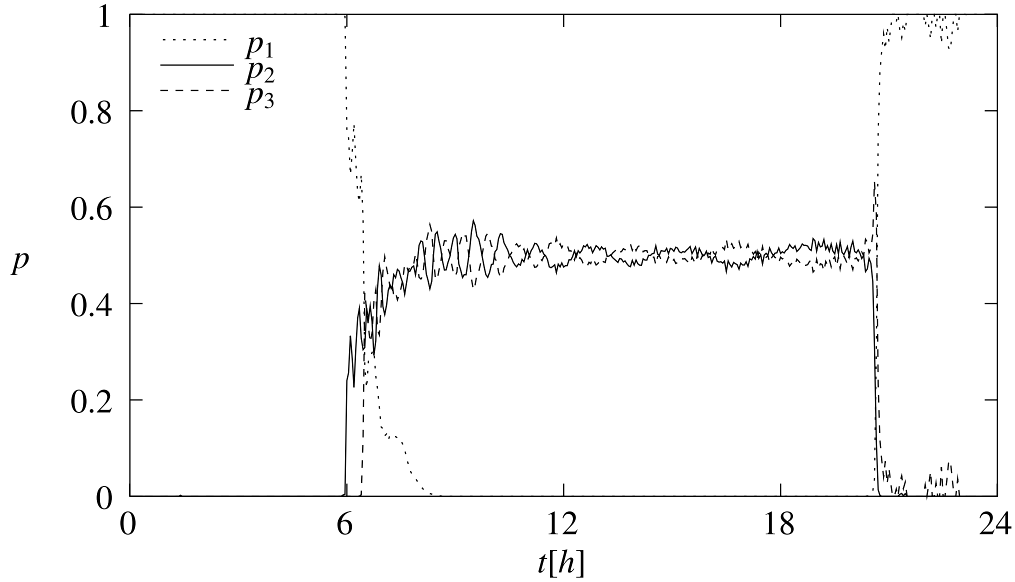


Figure 4.3 Typical route choice probabilities before the iteration has converged. This example is taken from the dynamic version of Braess's paradox discussed in section 5.1. During the period where p_1 is zero, the equilibrium is $p_2 = p_3 = 0.5$ due to the symmetry of the network. The iteration process produces route choice probabilities which are oscillating over the day. These oscillations are caused by the fact that the algorithm adjusts the route choice probabilities for every driver independently, but in fact each drivers travel time is affected by all the drivers with an earlier starting time. Due to this coupling, an 'over-reaction' at some time t_0 leads to an over-compensation at later times.

mechanism as discussed in the stability analysis. Longer travel times on route r_i at some instant t_0 lead to a decrease in the route choice probability p_i at this time. This affect all drivers starting later, and these drivers will in turn *increase* p_i. This coupling over time, together with the restoring force generated by the update rule for the p_i, cause the p_i to behave like a string.

4.2.1 Convergence

Unfortunately, it is difficult to study the convergence of the algorithm. Strictly speaking, the algorithm will usually not converge at all, since the underlying simulation model will be stochastic. The stochastic fluctuations of the simulation model will always cause a shift of the probabilities, and the only thing we can hope for is that the averages of the route choice probabilities taken over several iterations will converge, or that the distribution of the route choice probabilities will become stationary for each driver.

The discussion in the previous section has shown that once the algorithm has reached a state sufficiently close to an equilibrium, it will stay there and stochastic fluctuations of the travel times in the underlying simulation model will only cause stochastic fluctuations of the route choice probabilities. However, in general we cannot tell how many equilibria exist, since for the underlying 'cost function', i.e. the simulation model, the conditions for the uniqueness of the equilibrium — separability and convexity — will usually not hold[2].

Nevertheless, the simulations done for this thesis showed no example where the algorithm did not converge. Further, the convergence was actually slowest for *small* networks with few OD pairs. For example, in the case of the Braess networks (c.f. section 5.1) the DTA algorithm took several thousand iterations until the route choice probabilities became stationary, while for the Wuppertal network about 40 iterations were sufficient.

4.2.2 Numerical example

In this section the stability condition is numerically tested for an example network with only two nodes O and D and two routes r_1 and r_2 joining them. The duration of one simulation period T is 3 hours, and the capacities and the demand (both in units of vehicles per second) are given by

$$C_1 = 1, \qquad C_2 = 2 \tag{4.18}$$

$$d(t) = 2 - 2\cos\frac{2\pi t}{T}. \tag{4.19}$$

The equilibrium p_i are

$$p_1 = \frac{1}{3}, \qquad p_2 = \frac{2}{3}. \tag{4.20}$$

The time step Δt of the simualtion was set to $30s$. The lengths of the routes are chosen such that the travel time in the uncongested network is about $460s$ while the maximum travel time is around $1200s$. The duration of the jammed period is about $6000s$ and the minimum value of $(T_1 + T_2)/d$ is approximately $300s^2$. According to condition (4.17), the algorithm should be stable for $a < 0.15$.

---

[2] The cost function is said to be *separable* if the cost on link a only depends on link x_a, i.e. if the Jacobian is diagonal. For example, the cost function will not be separable if the simulation models includes prioritized intersections.

To test the stability of the algorithm, we define

$$\delta_{12}^2 = \frac{1}{T} \sum_{i=0}^{T/\Delta t} \frac{(T_1(t_i) - T_2(t_i))^2}{(T_1(t_i) + T_2(t_i))^2} \tag{4.21}$$

where we have used the abbreviation $t_i = i\Delta t$. δ_{12}^2 is a measure of how far the system is off the equilibrium. The average value $\langle \delta_{12}^2 \rangle$ of δ_{12}^2 over several iterations starting at the equilibrium indicates whether the systems stays at the equilibrium or is drifted away by the algorithm.

Figure 4.4 shows $\langle \delta_{12}^2 \rangle$ and the average travel time as a function of a. Averages were taken over 1000 iterations of the algorithm. The value $a = 0$ is included to measure the fluctuations due to the stochastic behavior of the individual drivers.

To test the convergence of the algorithm, the iteration was run with $a = 0.1$ and randomly distributed p_i. Figure 4.5 shows that after about 160 iterations $\langle \delta_{12}^2 \rangle$ has reached the equilibrium value showing that the algorithm has converged. By using a higher starting value ($a = 0.4$ in this example) and decreasing it during the iteration to a value for which the algorithm is stable, the convergence of the algorithm can be drastically speeded up.

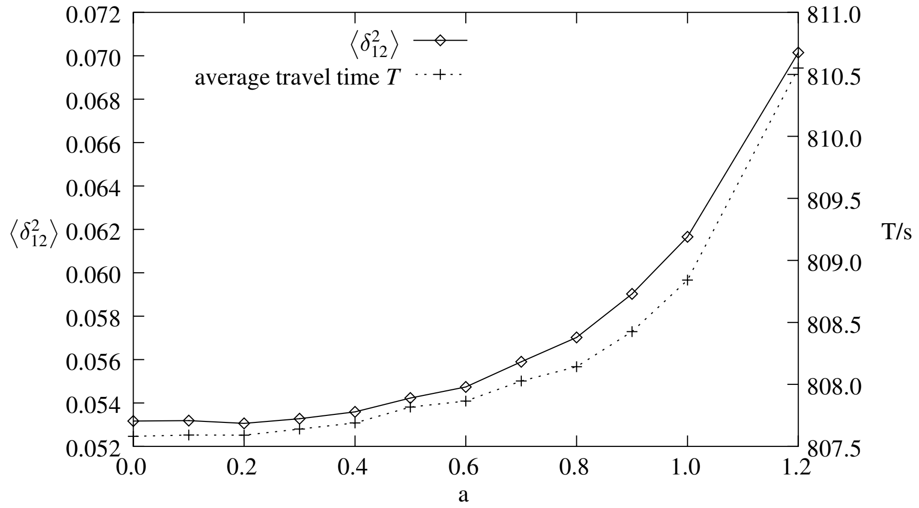


Figure 4.4 Average values of δ_{12}^2 and average travel times for different values of a sampled over 5000 iterations of the algorithm starting at the equilibrium point. The value for $a = 0$ (i.e. the algorithm doesn't change the route choice probabilities at all) is not zero because of the fluctuations of the queue lengths due to the stochastic route choice of the individual drivers. The stability condition (4.17) for this example is $a < 0.15$, but in this numerical example the algorithm is even stable stable for $a < 0.3$.

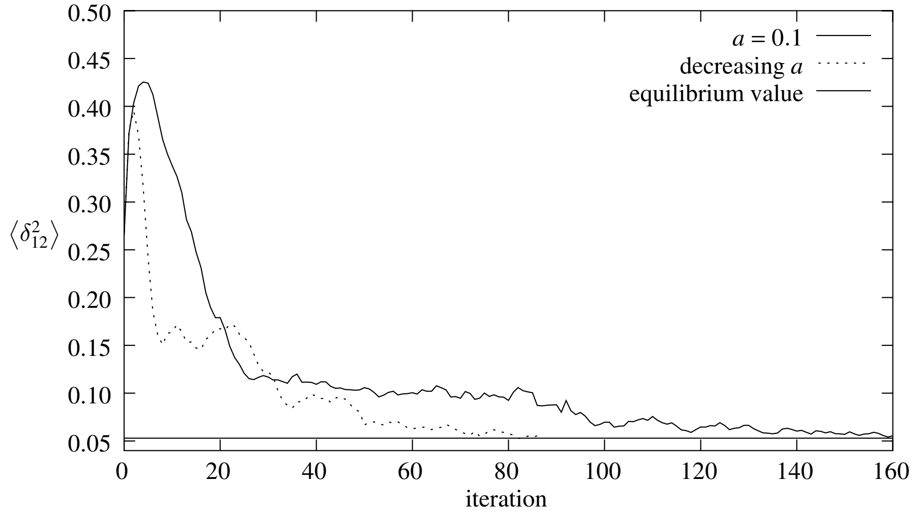


Figure 4.5 $\langle \delta_{12}^2 \rangle$ as a function of the number of iterations for $a = 0.1$, starting with random values of p_i. With $a = 0.1$ the equilibrium value of $\langle \delta_{12}^2 \rangle$ is reached after 160 iterations. Similar to the simulated annealing approach, the convergence can be speeded up by starting with a higher value of a and decreasing it during the iteration process. Using a starting value of $a = 0.4$ which is linearly decreased to $a = 0.1$ in the first 60 iterations, the equilibrium value of $\langle \delta_{12}^2 \rangle$ is reached after 80 iterations.

Chapter

Five

Applications

5.1 A Dynamic Version of Braess's Paradox

In section 2.2.6 we have discussed Braess's paradox of static traffic assignment, which demonstrates the counter-intuitive consequences Wardrop's first principle can have.

We also saw that the effect of increasing the travel time by adding a link to the network strongly depends on the demand, i.e. the traffic volume (see figure 2.7 on page 18). The interesting question is what will happen if the demand is time-dependent and has a strong peak period like the typical demand patterns observed in real-world data.

A simple minded, quasi-static approach would be to calculate the travel times as a function of the time-dependent traffic volumes using the static relation shown in figure 2.7. This would mean that

- before the peak period, the travelers would be better off with the additional link,
- during a transition phase the link would increase the travel time,
- and during the peak period the additional link would not matter at all.

Depending on the parameters, it might well be that in this model the network with the additional link has a better overall performance.

We will see in this section that this approach is indeed too simple minded and that the additional link increases the travel time during the whole peak period. This dynamic version of Braess's paradox is therefore also a good example to show that static assignment fails to describe dynamic situations correctly.

Figure 5.1 shows the four networks considered in the sequel. The lengths of the individual segments are chosen such that the length of routes A and B (see figure 5.1) is $120km$ and the length of route C is $80km$. Networks one and two have an optimal throughput of 16 (flow on routes $q_A = q_B = 8s^{-1}$), while the capacity of the third and fourth network is 18 ($q_A = q_B = 8s^{-1}$, $q_C = 2s^{-1}$). For demands $d < 8s^{-1}$, the optimal route in the networks two and three is C because this route is shorter than A and B. The same holds for network four when the demand is less than $2s^{-1}$.

Braess's Paradox occurs in the dynamic case when the demand rises above the capacity of route C. In the sequel, we will assume a demand of the form

$$d(t) = 10s^{-1} - 7s^{-1} \cos \omega t, \qquad \omega = \frac{2\pi}{T}, \tag{5.1}$$

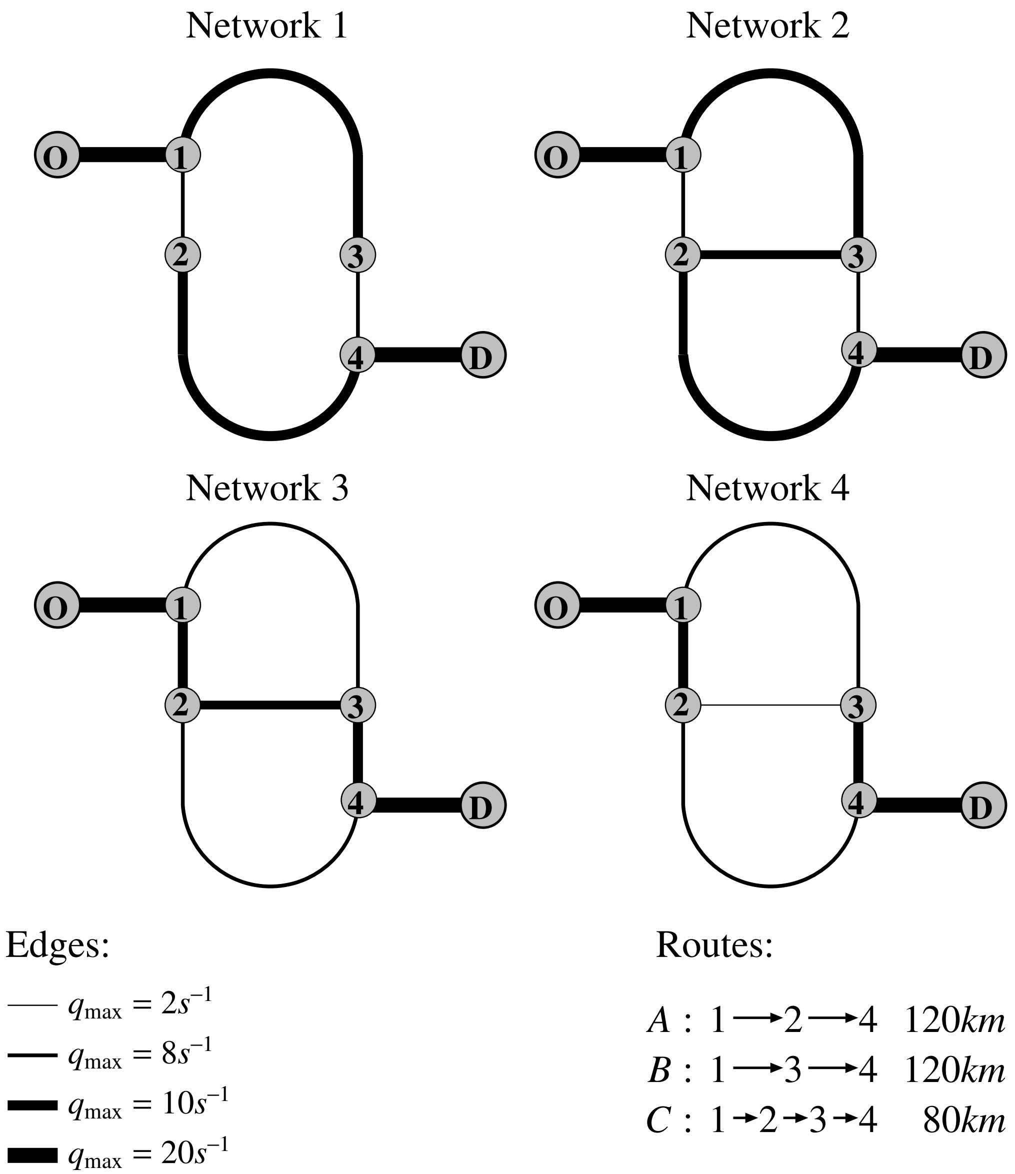

Network 1
Network 2
Network 3
Network 4
O
1
2
3
4
D
Edges:
$q_{\max} = 2s^{-1}$
$q_{\max} = 8s^{-1}$
$q_{\max} = 10s^{-1}$
$q_{\max} = 20s^{-1}$
Routes:
$A : 1 \rightarrow 2 \rightarrow 4$ $120km$
$B : 1 \rightarrow 3 \rightarrow 4$ $120km$
$C : 1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ $80km$

Figure 5.1 Dynamic version of Braess's paradox. We consider four different networks. The first two networks are the 'usual' Braess networks. The third and fourth network are examples where the additional link improves the network capacity. The system optimal capacity of networks 1 and 2 is $16s^{-1}$ ($q_A = q_B = 8s^{-1}$), whereas the system optimal capacity of networks 3 and 4 is $18s^{-1}$ ($q_A = q_B = 8s^{-1}, q_C = 2s^{-1}$). The reader is invited to guess which network has the lowest average travel time and in which network the highest travel times occur.

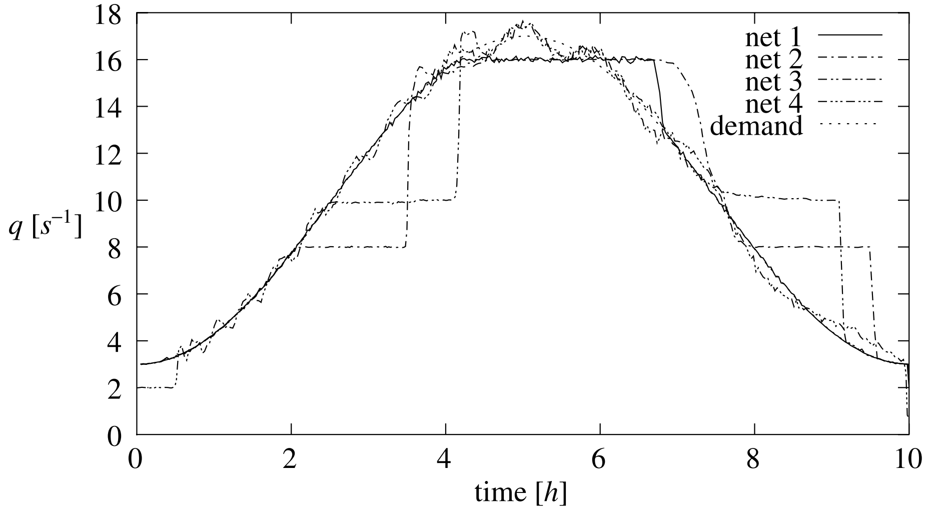


Figure 5.2 Flows in the different versions of Braess's network. In networks 3 and 4, small oscillations of the flow remain.

where we chose $T = 36000s$. The reader may wonder why the demand and capacity are chosen rather high compared to flows observed on real roads. The reason for this lies in the way the simulation based traffic assignment algorithm works. In each iteration, the drivers have to choose their route according to their probability distribution p_d. Since this is done independently for each driver, the stochastic fluctuations in the number n of drivers choosing a given route are of order $\sqrt{n}$. In networks with many OD-relations, these fluctuations would average out, but in our case these fluctuations generate noise disturbing the iteration process, especially since these networks have very sensitive user equilibria. By using large flows, and thereby increasing the number of drivers, these fluctuations are damped.

In network 1, the optimal solution is always $f(A) = f(B)$ due to the symmetry of the network.

For network 2, the situation is more complicated and the optimal route choice depends on the demand. While d is less than $8s^{-1}$, C is of course optimal. This corresponds to the case $d_{OD} \leq 80/31$ in the static version of Braess's paradox (see figure 2.7 on page 18). C remains optimal until the queue length in C compensates for the different lengths of the routes, which corresponds to $80/31 < d_{OD} < 40/11$ in the static case.

The important point is that during this intermediate period, where the demand has exceeded the capacity of route C but C is still optimal, the flow through the network is bounded by the capacity of C, and therefore suboptimal. This is the reason for the plateaus in figure 5.2.

After the queue on route C has grown to a length such that the travel time on route C is equal to the travel time on routes A and B, the route choice probabilities will change and A and B will be used again. The effect of this in the different networks is different. In network 2, nobody will use route C after a short time, which corresponds to the case $d_{OD} > 80/9$ in the static version of Braess's paradox. This also implies that the total flow

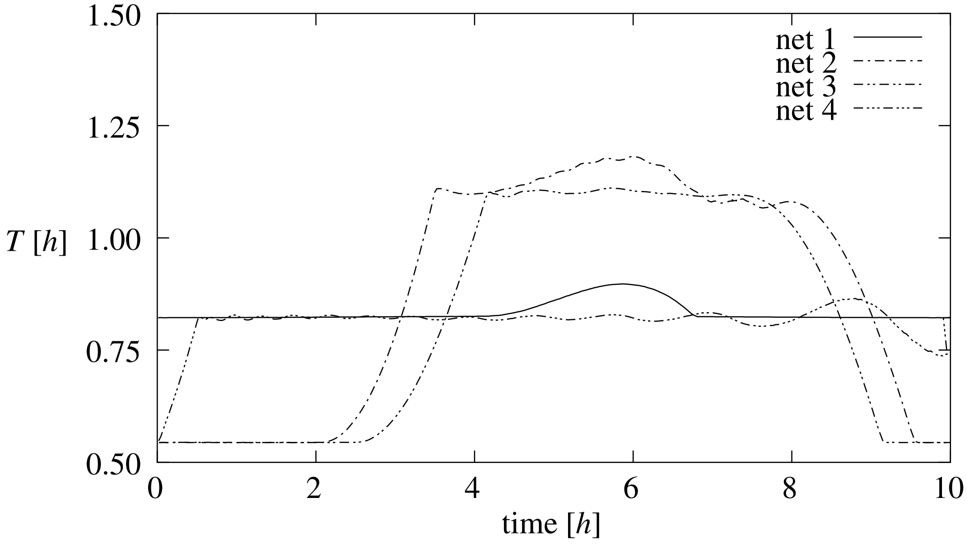


Figure 5.3 Travel times in the different versions of Braess's network.

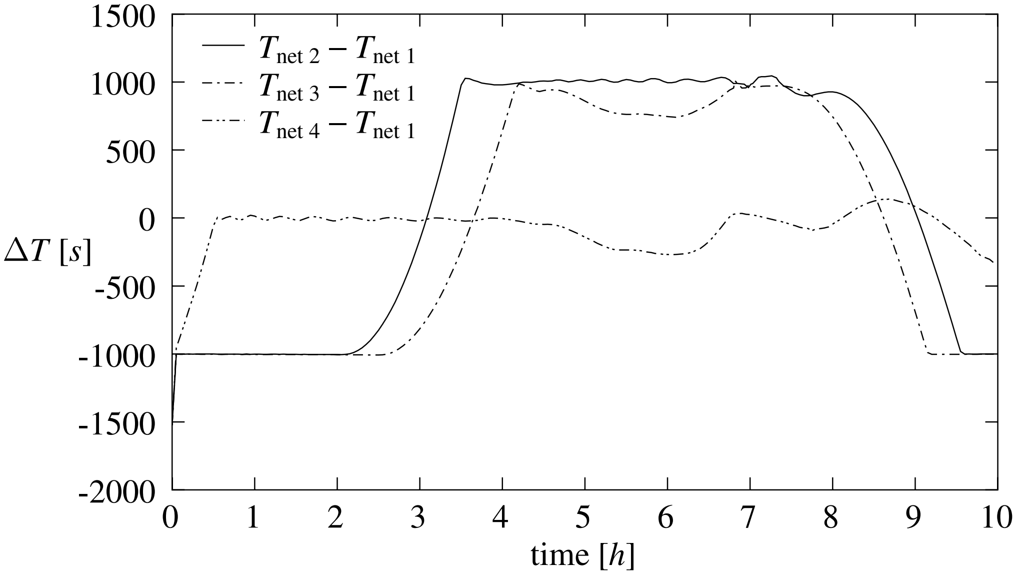


Figure 5.4 Travel time differences between network 1 and the other networks.

through the network reaches the optimal value of $16s^{-1}$, i.e. the drivers behave like in network 1. However, this does *not* imply that the travel times in both networks are equal. Due to the fact that in the intermediate period the flow through the network was only $8s^{-1}$, the queues in network 2 have grown longer and until the 'rush-hour' is over, i.e. when the queues have dissolved, the travel time in network 2 is larger by a constant amount of about 1000 seconds (see figure 5.4).

In network 3, the situation is a bit different. The optimal capacity of this network is $18s^{-1}$, which is more than the maximum of the demand. Nevertheless, the same effect as in network 2 occurs. As long as the travel time on route C is less than the travel time on the other routes, everybody will use route C, with the effect that the total flow through network 3 is *less* than the flow through network 1. Only after the travel time on route C is equal to the travel time on the other routes, the route choices will change and the capacity will increase. During the hour with the highest demand, the total flow through network 3 is finally higher than the flow in network 1, but nevertheless, the travel time in network 3 is *higher* than in network 1.

On the average, the travel times in networks 2 and 3 are higher than in network 1 since the flow, i.e. the number of travelers, during the peak period in which network 1 performs better is higher than during the time when networks 2 and 3 perform better.

The network with the best performance is actually network 4, which also has a capacity of $18s^{-1}$. In network 4, the capacity of the 'shortcut' is chosen such that the use of route C can never adversely effect the other routes, since the capacity of the link $3 \longrightarrow 4$ is equal to the sum of the capacities of links $1 \longrightarrow 3$ and $2 \longrightarrow 3$. Therefore, it can never perform worse than network 1, but during the peak-period it performs better due to its higher capacity. Comparing networks 4 and 3, we see that sometimes reducing the capacity of a link can *improve* the performance of a network, which is an effect slightly different from the one in the original version of Braess's paradox.

5.2 The Highway Network of Nordrhein-Westfalen

The highway network of Nordrhein-Westfalen was the first network studied by our group for which a time-dependent OD-matrix was available. This matrix was provided by the Institut für Stadtbauwesen (ISB) of the Technische Universität Aachen, and was generated by taking a static ODM for NRW, computing a heuristic solution of the static assignment problem for a network including secondary roads, tracing all trips using the highway network, and finally 'dynamicalizing' the resulting matrix for the highway network using typical time-of-day dependencies of the flows, which were extracted from induction-loop data provided by the Landschaftsverband Rheinland. Taking into account the complexity of this process, it is quite surprising that the daily averages of the flows resulting from this matrix (see figure 5.5) coincide rather well with measurement data (see figure 5.6).

Since differences between static and dynamic assignment can hardly be seen in figure 5.5, the distribution of the relative differences $(q_{\text{static}} - q_{\text{dynamic}})/(q_{\text{static}} + q_{\text{dynamic}})$ is shown in figure 5.7. If these differences were caused by some purely random fluctuations, one would expect these differences to be normal-distributed. Figure 5.7 indicates that this is not the case, instead the data fit very well to a distribution of the form $a\exp(-bx^c)$ with $c = 0.7$ and $b = 19.6$.

	network 1	network 2	network 3	network 4
capacity	$16s^{-1}$	$16s^{-1}$	$18s^{-1}$	$18s^{-1}$
average travel time	$3034s$	$3545s$	$3285s$	$2951s$

Table 5.1 Summary of Braess's paradox

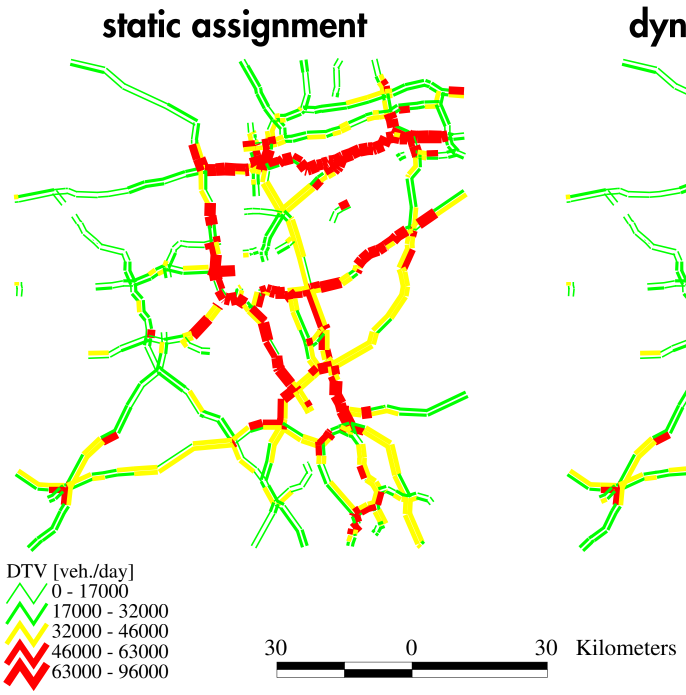


Figure 5.5 Comparison between static and dynamic assignment for the highway network of NRW. The plot shows the average daily traffic flows, calculated by summing up the results of 24 individual static assignments for each hour (left) and by the dynamic assignment algorithm (right). A more detailed comparison is shown in figure 5.7.

Still, it is surprising how small the difference between the flows resulting from static and dynamic assignment are. Figure 5.10 shows the reason for this: For most OD-relations, only one route is used (see figures 5.8 and 5.10), effecting the flow for this OD-relation to be assigned to the same route in both the static and dynamic assignment.

The reason for this is shown in figure 5.9: The average length of the trips in the OD-matrix is only $13km$. Since the highway network is rather sparse, in most case there is only one reasonable route for trips of this short length, even if the network is congested. Given the structure of the OD-matrix, it is therefore not surprising that the differences between the static and dynamic assignment are so small. Whether this short average trip length is realistic or an artificial effect caused by the algorithm used to generate the matrix is hard to say. An indication for the latter possibility might be that the average duration of a trip in the highway network of NRW is only about 440 seconds, while the average duration of a trip in the network of Wuppertal — which is mostly an urban network — is about 2100 seconds (see figure 5.12).

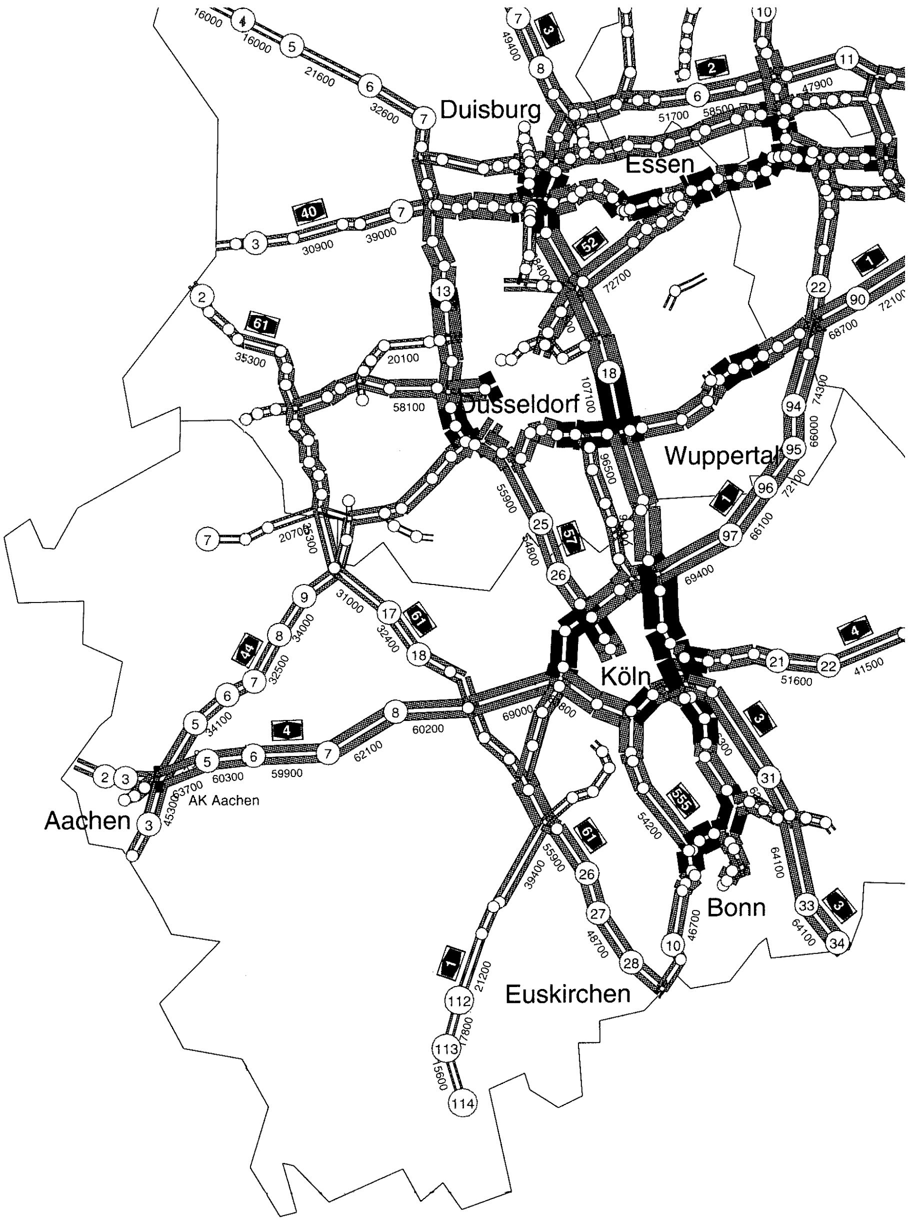


Figure 5.6 Average daily traffic flows for the NRW highway network. This figure was taken from [63].

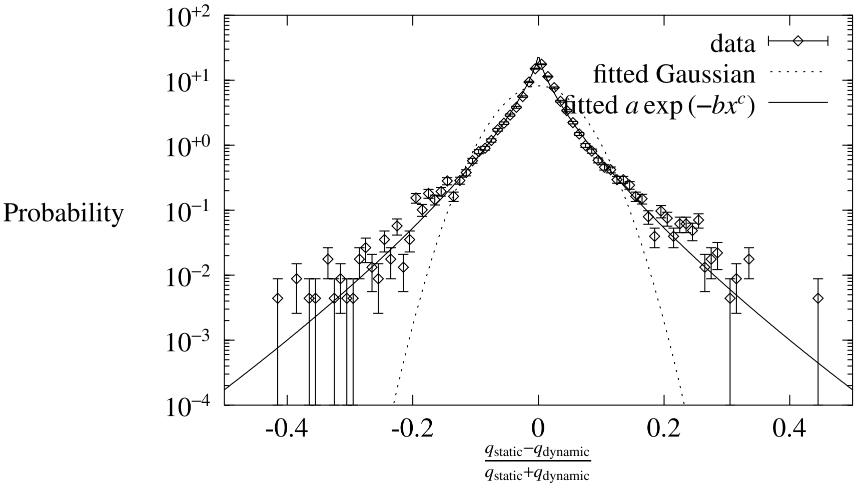


Figure 5.7 Comparison between static and dynamic assignment. The figure shows the distribution of the relative error $(q_{\text{static}} - q_{\text{dynamic}})/(q_{\text{static}} + q_{\text{dynamic}})$. If the deviations would be purely stochastic, one would expect this quantity to be normal distributed. The plot shows that the data set fits only very badly to a normal distribution with the same mean ($7.8 \cdot 10^{-4}$) and standard deviation ($4.9 \cdot 10^{-2}$), but is fitted very well by a distribution of the form $a\exp(-bx^c)$ with $c = 0.7$ and $b = 19.6$.

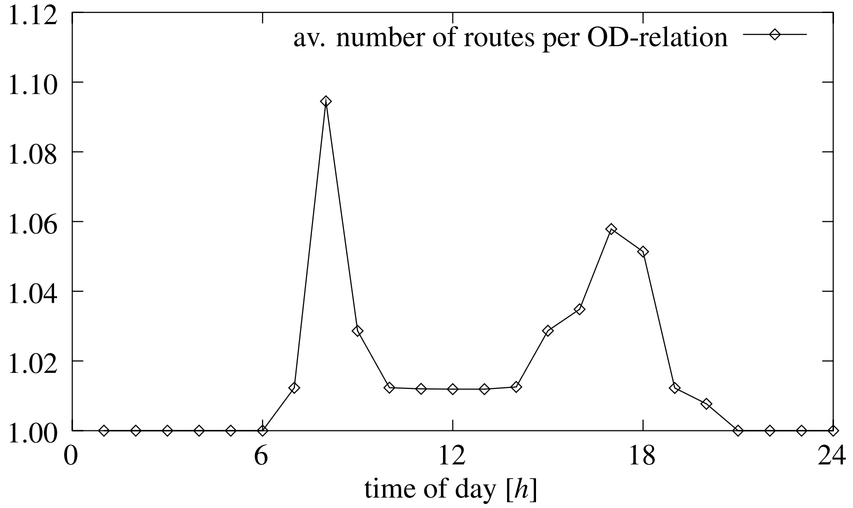


Figure 5.8 Average number of routes per OD-relations generated by the static assignment for the NRW highway network. Even during the morning and afternoon peaks, the number of generated routes exceeds the number of OD-relations only by 10 percent.

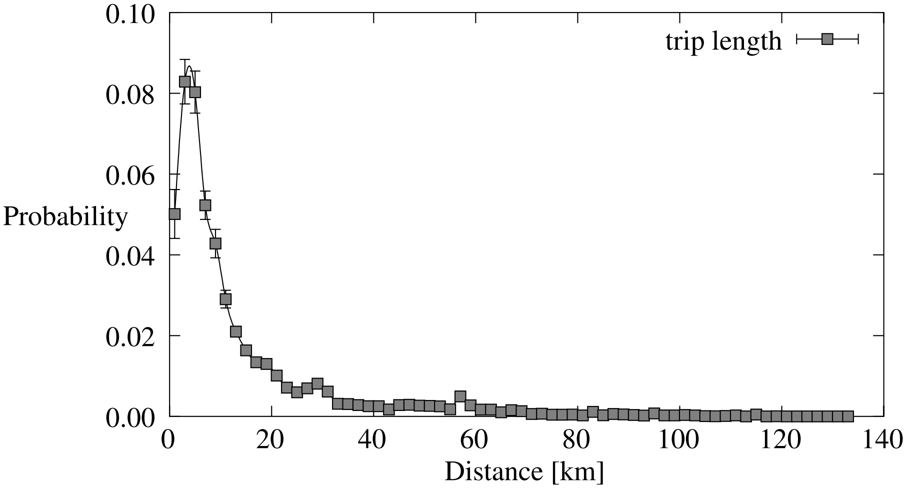


Figure 5.9 Distribution of the trip lengths, calculated using the OD-matrix provided by the ISB. Surprisingly, the distribution of the trip lengths is sharply peaked at short distances, and the average trip length is only $13km$.

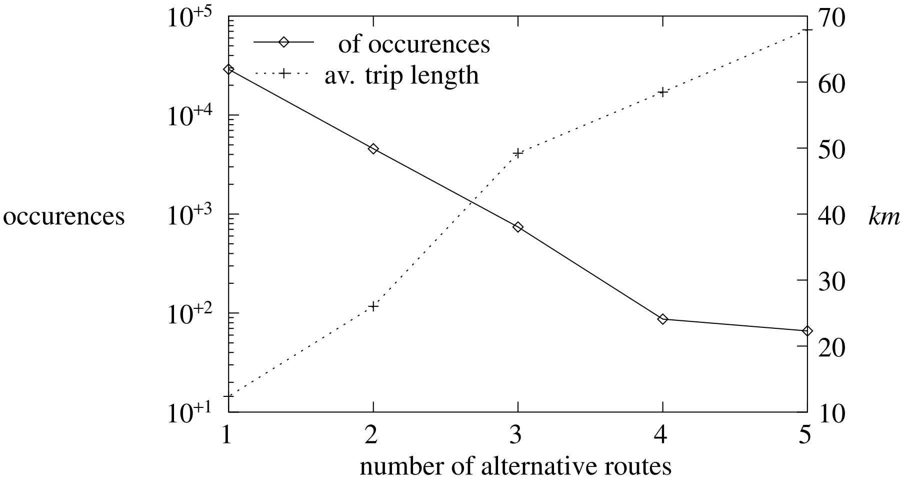


Figure 5.10 The plot shows both the number of cases where the algorithm has generated a given number of routes, and the average length of the corresponding trips. As one would expect, there are only few OD-pairs for which many routes are generated, and that these routes are generated for rather long trips. Since the OD-matrix includes only very few long trips (see figure 5.9), it is not surprising that for most OD-pairs there is no alternative route. This also explains why the results of the static and the dynamic assignment agree so well.

5.3 Wuppertal

To demonstrate that the different parts of the FVU (see figure 1.1 on page 2) are interoperable and can be used together to simulate fairly large networks, a study area within NRW had to be chosen which should be interesting for all groups participating in the FVU, for which all necessary data should be available, and the simulation of which should be a challenge and should demonstrate the performance of the methods employed by the FVU.

The city of Wuppertal fulfilled all these conditions. Due to its situation in the v-shaped valley of the river Wupper at the border of the Ruhrgebiet, this city of about 400,000 inhabitants is an interesting object of study for both transportation researchers and meteorologists. The city proved to be very cooperative and provided the necessary data for the groups modeling the demand for transportation and evaluating the impacts of traffic on the population.

Consisting of 16769 links and 9098 nodes, the street network of Wuppertal is a challenging object of study for traffic simulation. Even doing a static assignment on this network is time-consuming: On a 336MHz-clocked UltraSPARC-II processor, 70 iterations of the Frank-Wolfe algorithm[1] took 93 hours of CPU-time. Figure 5.11 shows the convergence-rate of the Frank-Wolfe algorithm for the Wuppertal network, using the BPR cost function (see section 2.2.5) and the OD-matrix for 11am. After 29 iterations, the relative accuracy, i.e. $\frac{ub-lb}{ub+lb}$ where ub is the upper bound on the objective function value and lb the lower bound, is better than 10^{-6}.

To provide the input data for a microscopic simulation with PLANSIM-T, a dynamic traffic assignment had to be done to calculate the route choices. How much CPU time would be needed to do this using the dynamical traffic assignment models discussed in section 2.3? In section 2.3.6 we have shown that taking into account only the linear subproblems, the time to solve such a dynamic assignment model would be greater by a factor of about $4N^2$, where N is the number of time steps. Due to the requirements of the meteorological groups, a time resolution of at least 15 minutes had to be provided. The time needed per iteration of the Frank-Wolfe algorithm would therefore be about 37000 times higher than in the static case. Given the fact that a single iteration in the static case needs about 1.3 hours, this would result in 2048 days per iteration in the dynamic case. Therefore, the dynamical assignment models based on mathematical programming methods could not be applied for our purpose.

Compared to the dynamic assignment models, one iteration of the simulation-based DTA algorithm is quite fast: The simulation of Wuppertal network using FASTLANE takes less than one hour of CPU time on a UltraSPARC-II processor clocked with 336MHz[2], and the update of the route choice probabilities takes about 6.5 hours of CPU time on the same hardware. Figure 5.12 indicates that about 40 iterations are needed until the travel times converge.

The total CPU time needed by the simulation-based DTA algorithm to compute the

---

[1] implemented using the LEDA-library, see section 2.2.7

[2] See also table 3.1 on page 44 for measurements made on a 166MHz UltraSPARC processor. The fact that the speedup is less than expected from comparing the clock-rates indicates that the simulation is memory or I/O-bound.

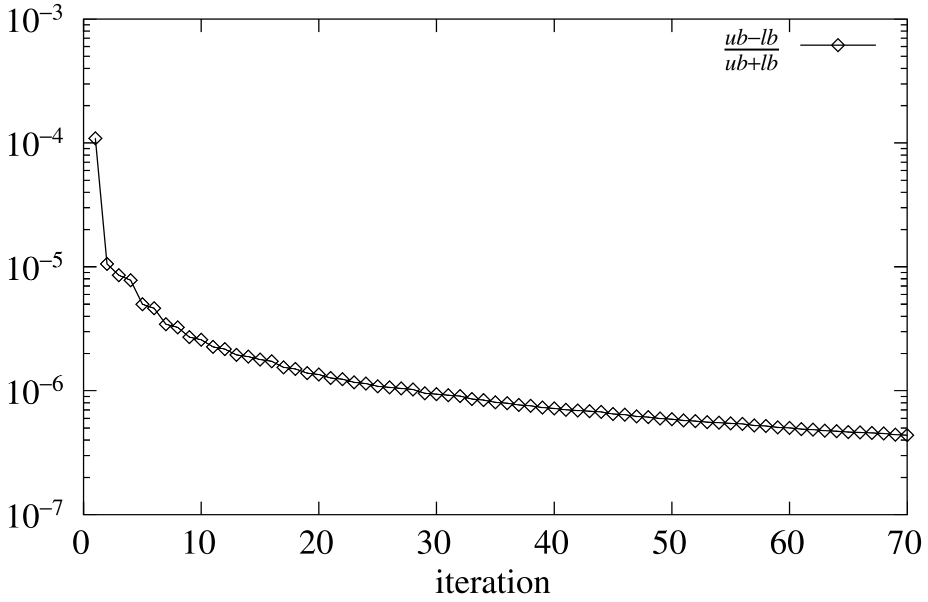


Figure 5.11 Convergence rate of the Frank-Wolfe algorithm for the solution of the static assignment problem for Wuppertal. An accuracy of 10^{-6} is reached after 29 iterations.

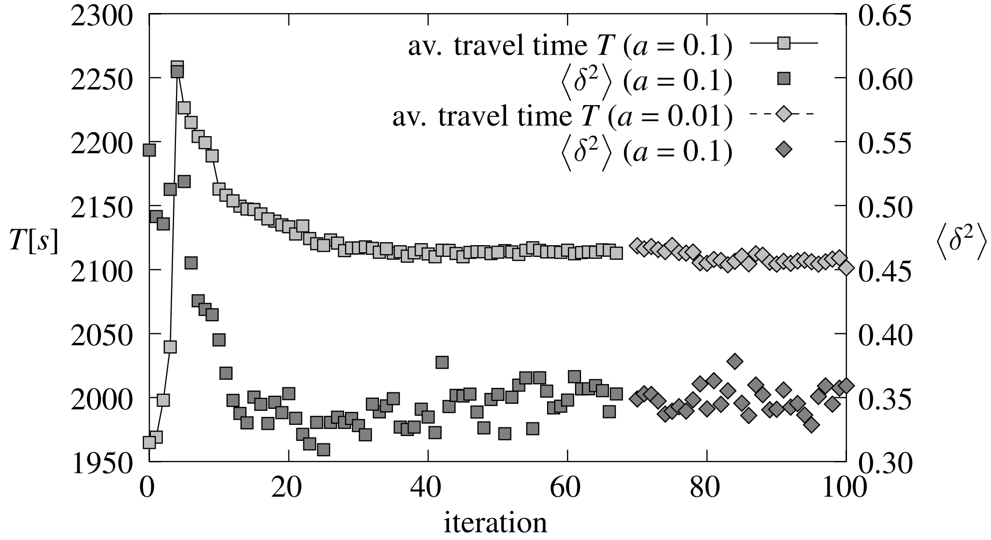


Figure 5.12 Convergence of the simulation-based DTA algorithm for the Wuppertal network. The plot shows both the mean travel time and the mean of the squared relative travel-time difference $\langle\delta^2\rangle$ (c.f. equation (4.21)). After 70 iterations with $a = 0.1$, additional iterations with $a = 0.01$ were performed to check if a was chosen small enough to ensure the stability of the algorithm.

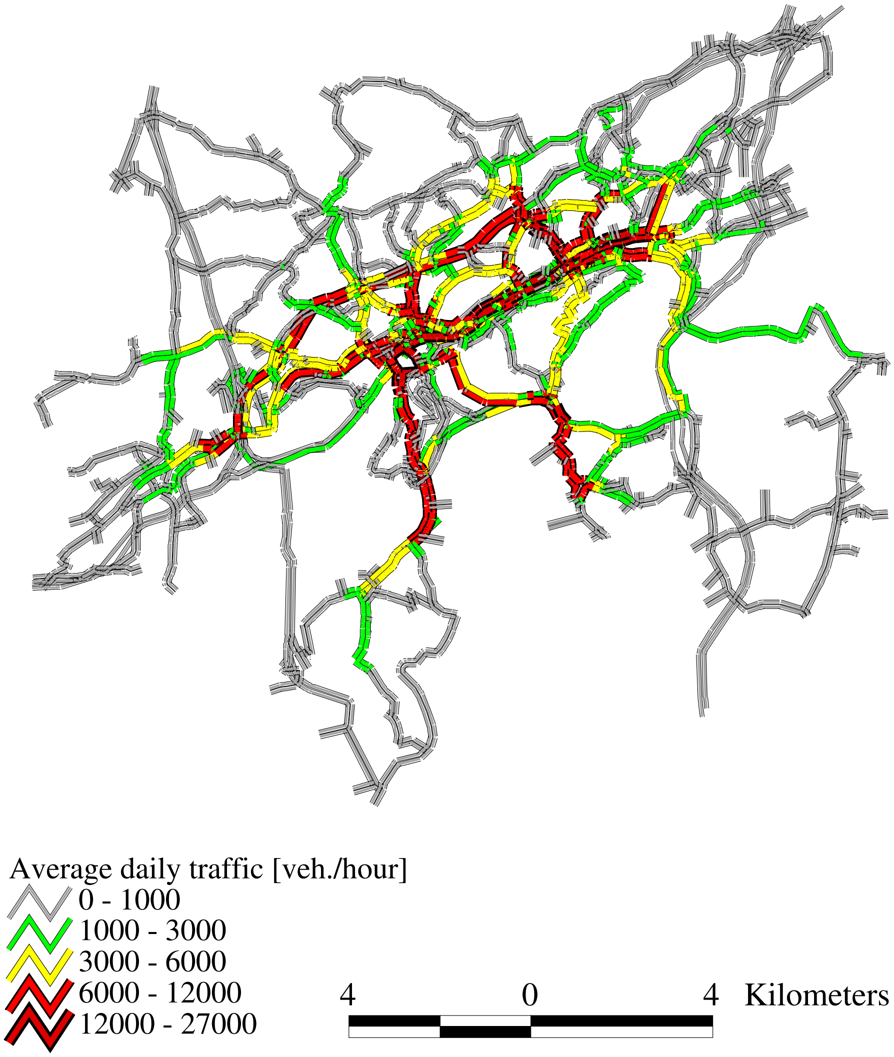
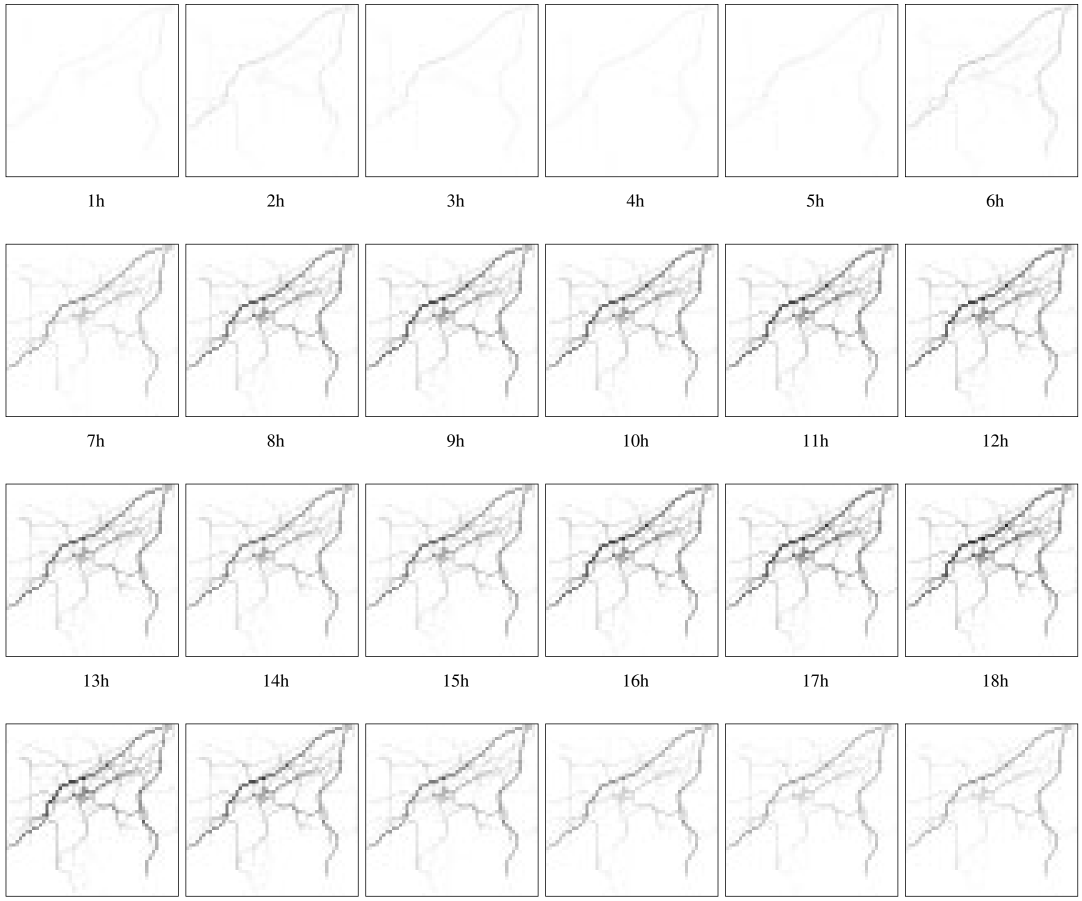


Figure 5.13 Average daily traffic in Wuppertal, calculated using the simulation-based DTA algorithm and the dynamic OD-matrix provided by the ISB.



1h 2h 3h 4h 5h 6h

7h 8h 9h 10h 11h 12h

13h 14h 15h 16h 17h 18h

19h 20h 21h 22h 23h 24h

Figure 5.14 Time-dependence of NO_x emissions in Wuppertal computed with PlanSim-T, using the routes calculated with the simulation-based DTA algorithm. The highways bypassing Wuppertal in the north (A46) and east (A1) are clearly visible. Since the simulation starts with an empty network, the emissions calculated for the first hour are probably too low.

dynamic user equilibrium for the Wuppertal network is therefore about 300 hours or 12.5 days. This is still a lot, but by using parallel computers — the time-consuming update of the route choice probabilities can be parallelized in a straightforward way — the time needed to do these computations could be reduced.

Figure 5.13 shows the resulting average daily traffic flows after the algorithm has converged. The time-dependent traffic flows were used by the Lehrstuhl für umweltverträgliche Infrastrukturplanung (LUIS) as input data for their GIS-based evaluation tool ‘LUISE’ [37], which checks if the environmental impacts of traffic (noise, pollution) are in accordance with German regulations, which specify valuation criteria depending on both the type of the area (residential, industrial) and the time of day (especially in the case of noise).

When comparing figures 5.13 and 5.6, it becomes obvious that the OD-matrix does not contain all of the traffic bypassing Wuppertal on the highways A46 and A1. While for the data needed by LUISE, which concentrates on the urban area, this is not a serious problem, this fact would make the calculated emissions unusable for the meteorological groups. Therefore, in the ensuing calculation of pollutant-emissions using PLANSIM-T, this effect was corrected by adding additional traffic on these higways. Figure 5.14 shows the resulting NOx emissions as a function of time, which were used by the meteorological groups as input data for the simulation of ozon-immissions [11, 47].

Chapter

Six

Summary and Outlook

In this work we have presented a simulation-based approach to the dynamic traffic assignment problem. Using the simulation model FASTLANE, which uses a queuing dynamic to describe the traveling of vehicles on a link in a street network and neglects completely car-following behavior, we have been able to perform a dynamic user equilibrium traffic assignment for the street network of Wuppertal, consisting of 16769 links and 9098 nodes. This assignment was needed to calculate realistic route choice for an ensuing microscopic simulation of the traffic in Wuppertal, which was performed in the context of the FVU to provide data on both traffic flow and emissions of pollutants with a high time resolution.

On a Sun Enterprise 10000, this computation took about two weeks of CPU time. Using an analytical dynamic assignment model, the computation of such an user equilibrium would not have finished yet even if it were started at the beginning of the FVU in 1995.

In the meanwhile, other groups have started to apply similar simulation models on even larger networks, like the network of Portland, Oregon. Although the requirements on computing resources do not yet allow this dynamic assignment method to be routinely employed in transportation planning applications, this may change in the near future as the costs of computing resources decrease.

In chapter 2 we have introduced the subject with a short review of the static traffic assignment problem, which is well understood both from the mathematical and from the algorithmical point of view. We have discussed analytical models for the dynamic traffic assignment problem and have compared their complexity with the complexity of the static traffic assignment problem, assuming the use of the Frank-Wolfe algorithm, which can be used to solve both the static and dynamic problem.

In chapter 3 we have reviewed different classes of traffic simulation models, focusing on their usability for traffic assignment based on iterated simulations. We saw that models describing individual vehicles are suited best for this purpose. We have also described FASTLANE, a queuing model developed by the author, and demonstrated that it can be used to approximate the travel times of more detailed car-following models like the Nagel-Schreckenberg model and its continuous extension by Krauß.

In chapter 4 we have presented an algorithm to solve the dynamic user equilibrium problem using iterated simulations. This approach is based on describing route choice of driver d by means of a probability distribution p_d on the set of available routes. After each simulation, this probability distribution is updated according to the travel times measured in the simulation. The advantage of using a probability distribution instead of assigning a single route to each driver is the ability to study the stability of the algorithm analytically — albeit only in simple cases. Note, however, that in these cases the equilibrium is *unstable* with respect to the algorithms using only one route per driver and re-routing some fraction of the drivers after each simulation.

Finally, we have presented three example applications in chapter 5. The first is a dynamic version Braess's paradox, which also demonstrates that static and dynamic treatment of the same situation may yield different results. Especially, static descriptions of situations with time varying demand showing a pronounced peak are flawed. The other applications are the study cases of the FVU, the simulation of the highway network of NRW and the simulation of Wuppertal mentioned above, which would not have been possibly using the analytical models of the traffic assignment problem.

Appendix

A

Mathematical Prerequisites

A.1 First-order Optimality Conditions with Inequality Constraints

In this section, we give a short overview of the first-order optimality conditions for optimization problems with inequality constraints, since these conditions are less well-known than their counterpart for equality constraints. Although the proof of these optimality conditions is beyond the scope of this thesis, we give a simple geometrical explanation of the additional positivity constraint on the Lagrange multipliers which is useful as a mnemonic. Despite the fact that constrained optimization problems have already been studied very early in the history of real analysis by J.-L. Lagrange in the context of theoretical mechanics, the case of inequality constraints was first studied systematically by Kuhn and Tucker in the sixties[52].

A.1.1 Kuhn-Tucker Conditions

Consider an optimization problem of the form

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) && \text{(A.1a)} \\ &\text{subject to} && \mathbf{h}(\mathbf{x}) = \mathbf{0} && \text{(A.1b)} \\ & && \mathbf{g}(\mathbf{x}) \leq \mathbf{0}, && \text{(A.1c)} \end{aligned}$$

with $f \in C^1(\mathbb{R}^n, \mathbb{R})$, $\mathbf{h} \in C^1(\mathbb{R}^n, \mathbb{R}^m)$ and $\mathbf{g} \in C^1(\mathbb{R}^n, \mathbb{R}^p)$.

Definition A.1.1 *A point* $\mathbf{x} \in \mathbb{R}^n$ *satisfying the constraints*

$$\mathbf{h}(\mathbf{x}) = \mathbf{0}, \qquad \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \tag{A.2}$$

is said to be a regular point *of the constraints (A.2) if the set*

$$\bigcup_{1 \leq i \leq m} \{\nabla h_i(\mathbf{x})\} \cup \bigcup_{i \in \{j | g_j(\mathbf{x}) = 0\}} \{\nabla g_i(\mathbf{x})\} \tag{A.3}$$

is linear independent.

The following theorem extends the well-known Lagrange multiplier theorem of constrained optimization to the case of inequality constraints[1].

Theorem A.1.1 (Kuhn-Tucker Conditions) *Let* $\mathbf{x}^*$ *be a relative minimum of problem (A.1) and suppose* $\mathbf{x}^*$ *is a regular point of the constraints. Then there are* $\lambda_i \in \mathbb{R}$, $1 \leq i \leq m$ *and* $\mu_j \in \mathbb{R}_{\geq 0}$, $1 \leq j \leq p$ *such that*

$$\boldsymbol{\nabla} f(\mathbf{x}^*) + \sum_{j=1}^{p} \mu_j \boldsymbol{\nabla} g_j(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i \boldsymbol{\nabla} h_i(\mathbf{x}^*) = \mathbf{0} \tag{A.4a}$$

$$\mu_j g_j(\mathbf{x}^*) = 0 \qquad \forall 1 \leq j \leq p. \tag{A.4b}$$

A proof of theorem A.1.1 can be found in [3] or [22] (example 26.4), where the latter states a result which extends to Banach spaces.

Remark A.1.1 *The condition of* $\mathbf{x}^*$ *being a regular point of the constraints cannot be dropped. Consider the problem*

$$\textit{minimize} \quad f(x) = x \qquad \textit{subject to} \quad g(x) = x^2 \leq 0.$$

Since $x^* = 0$ *is the only feasible point, it is optimal, but*

$$f'(0) + \mu g'(0) = 1 \neq 0 \qquad \forall \mu \in \mathbb{R}_{\geq 0}.$$

For readers familiar with the case of equality constraints, we give a short explanation of the additional conditions for the Lagrange multipliers μ_j for the inequality constraints.

First of all, we should notice that condition (A.4b) just states that μ_j is only playing a role in condition (A.4a) if $g_j(\mathbf{x}^*) = 0$, i.e. if the constraint is 'active'. Figure A.1 shows this situation where all inequality constraints are 'active' and there are no equality constraints. In this case, the main difference to the case of an equality constraint is that it is not sufficient for $\boldsymbol{\nabla} f(\mathbf{x}^*)$ to be perpendicular to the tangent space of the manifold defined by the constraints. Instead, we must also ensure that we cannot improve the objective function by moving into the feasible region. In the case of (A.1) this means that $\boldsymbol{\nabla} f(\mathbf{x}^*)$ has to point into the feasible region, i.e. $-\boldsymbol{\nabla} f(\mathbf{x}^*)$ has to be a linear combination of the $\boldsymbol{\nabla} g_j(\mathbf{x}^*)$ with *positive* coefficients, which is exactly the statement of (A.4a).

Remark A.1.2 *If we replace (A.1c) by*

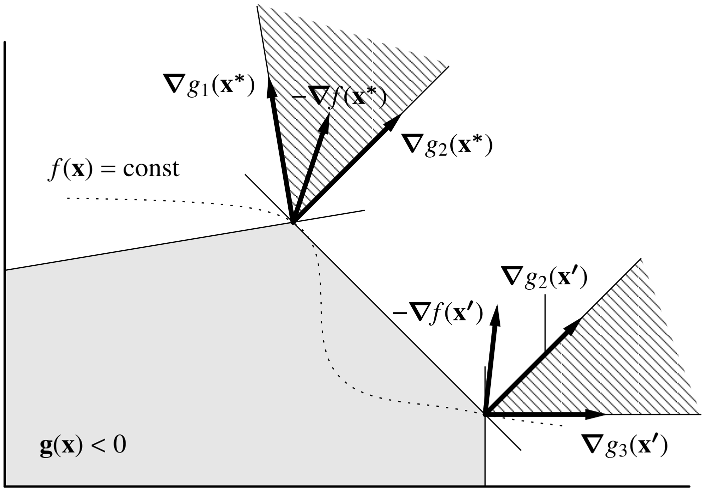
$$\mathbf{g}(\mathbf{x}) \geq \mathbf{0}, \tag{A.1c'}$$

(A.4a) has to be replaced by

$$\boldsymbol{\nabla} f(\mathbf{x}^*) - \sum_{j=1}^{p} \mu_j \boldsymbol{\nabla} g_j(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i \boldsymbol{\nabla} h_i(\mathbf{x}^*) = \mathbf{0}. \tag{A.4a'}$$

---

[1] The conditions on the differentiability of the constraints can actually be weakened, however at the cost of complicating the notation. For a stronger version of theorem A.1.1 see [3]



$\nabla g_1(\mathbf{x}^*)$
$-\nabla f(\mathbf{x}^*)$
$\nabla g_2(\mathbf{x}^*)$
$f(\mathbf{x}) = \text{const}$
$\nabla g_2(\mathbf{x}')$
$-\nabla f(\mathbf{x}')$
$\mathbf{g}(\mathbf{x}) < 0$
$\nabla g_3(\mathbf{x}')$

Figure A.1 Optimality condition under inequality constraints. For $\mathbf{x}^*$ with some $g_i(\mathbf{x}^*) = 0$ to be a local minimum of f under the constraint $\mathbf{g}(\mathbf{x}) \leq 0$, the gradient $\nabla f(\mathbf{x}^*)$ has not only to be perpendicular to the tangent plane of the manifold defined by $g_i(\mathbf{x}) = 0$ at $\mathbf{x}^*$ (this would be sufficient in the case of the condition $\mathbf{g}(\mathbf{x}) = 0$), but also has to be a linear combination of the $\nabla g_i(\mathbf{x}^*)$ with positive coefficients, i.e. lie in the positive cone generated by the $\nabla g_i(\mathbf{x}^*)$. At the point $\mathbf{x}'$, which is not a minimum, this condition is violated.

Lagrangian Formulation

Just as in the case of equality constraints, the optimality conditions (A.4) and the constraints of (A.1) can be expressed in terms of the Lagrangian

$$L(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \boldsymbol{\mu}\mathbf{g}(\mathbf{x}) - \boldsymbol{\lambda}\mathbf{h}(\mathbf{x}) \tag{A.6}$$

by taking the partial derivatives with respect to all variables.

Corollary A.1.2 *Let* $\mathbf{x}^*$ *be a relative minimum for problem (A.1) and suppose* $\mathbf{x}^*$ *is a regular point of the constraints. Then there are* $\boldsymbol{\lambda}^* \in \mathbb{R}^m$ *and* $\boldsymbol{\mu}^* \in \mathbb{R}^p_{\geq 0}$ *such that*

$$\frac{\partial L(\mathbf{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*)}{\partial x_i} = 0 \qquad \forall 1 \leq i \leq n \tag{A.7a}$$

$$\frac{\partial L(\mathbf{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*)}{\partial \lambda_i} = 0 \qquad \forall 1 \leq i \leq m \tag{A.7b}$$

$$\mu_i \frac{\partial L(\mathbf{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*)}{\partial \mu_i} = 0 \qquad \forall 1 \leq i \leq p \tag{A.7c}$$

$$\frac{\partial L(\mathbf{x}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*)}{\partial \mu_i} \geq 0 \qquad \forall 1 \leq i \leq p. \tag{A.7d}$$

A.1.2 Non-negativity and Affine Constraints

In traffic assignment applications, problems of the form

$$\min z(\mathbf{x}) \tag{A.8a}$$

subject to

$$\sum_i h_{ij} x_i = b_j \qquad \forall 1 \le j \le m \tag{A.8b}$$

$$x_i \ge 0 \qquad \forall 1 \le i \le n \tag{A.8c}$$

often arise.

Unfortunately, theorem A.1.1 seems not to be applicable to problem (A.8) on first sight since the $m + n$ gradients of the constraints cannot be linear independent. However, at a closer look we see that each x_i can only be 'active' in one of both constraints: If x_i is zero, it can be omitted from the affine constraints, and if x_i is greater than zero, the corresponding Lagrange multiplier is zero anyway. So if the matrix $\mathbf{H} = (h_{ij})$ has rank $m \le n$, we can in fact apply theorem A.1.1. Furthermore, we can eliminate the Lagrange multipliers related to constraint (A.8c).

Theorem A.1.3 *Let* $\mathbf{x}^*$ *be a local minimum of (A.8) with* $\mathbf{H}$ *having full rank* $m \le n$*. Then there are* λ_j*,* $1 \le j \le m$ *with*

$$x_i^* \cdot \left(\frac{\partial z(\mathbf{x}^*)}{\partial x_i} + \sum_j \lambda_j h_{ij} \right) = 0 \qquad \forall 1 \le i \le n \tag{A.9a}$$

$$\frac{\partial z(\mathbf{x}^*)}{\partial x_i} + \sum_j \lambda_j h_{ij} \ge 0 \qquad \forall 1 \le i \le n. \tag{A.9b}$$

Proof. As we stated above, we can apply theorem A.1.1 despite the fact that the $m+n$ gradients of all constraints are linear dependent since the gradients of the active constraints are linear independent. Therefore there are Lagrange multipliers λ_j, $1 \le j \le m$ and μ_i, $1 \le i \le n$ with $\mu_i \ge 0$ and

$$\frac{\partial z(\mathbf{x}^*)}{\partial x_i} + \sum_j \lambda_j h_{ij} - \mu_i = 0 \qquad \forall 1 \le i \le n \tag{A.10a}$$

$$\mu_i x_i^* = 0 \qquad \forall 1 \le i \le n. \tag{A.10b}$$

Condition (A.9a) follows from the fact that if x_i is greater than zero, μ_i has to be zero due to (A.10b) and in this case (A.10a) implies that the second factor in (A.9a) vanishes. Since $\mu_i \ge 0$, (A.10b) implies also (A.9a). □

As a corollary, we can express the optimality conditions (A.9) and the constraints of (A.8) in terms of the Lagrangian

$$L(\mathbf{x}, \boldsymbol{\lambda}) = z(\mathbf{x}) + \sum_j \lambda_j \left(\sum_i h_{ij} x_i - b_j \right). \tag{A.11}$$

Corollary A.1.4 *Let $\mathbf{x}^*$ be a local minimum of (A.8) with $\mathbf{H}$ having full rank $m \leq n$. Then there exist a $\boldsymbol{\lambda}^* \in \mathbb{R}^m$ with*

$$x_i^* \frac{\partial L(\mathbf{x}^*, \boldsymbol{\lambda}^*)}{\partial x_i} = 0 \qquad \forall 1 \leq i \leq n \tag{A.12a}$$

$$\frac{\partial L(\mathbf{x}^*, \boldsymbol{\lambda}^*)}{\partial x_i} \geq 0 \qquad \forall 1 \leq i \leq n \tag{A.12b}$$

$$\frac{\partial L(\mathbf{x}^*, \boldsymbol{\lambda}^*)}{\partial \lambda_j} = 0 \qquad \forall 1 \leq j \leq m. \tag{A.12c}$$

A.2 The Frank-Wolfe Algorithm

In 1956, Frank and Wolfe proposed an algorithm for solving quadratic optimization problems with linear constraints [27]. This algorithm is also well-suited for the convex optimization problems which arise in traffic assignment.

Consider the convex optimization problem[2]

$$\min z(\mathbf{x}) \tag{A.13a}$$

$$\text{subject to} \quad \sum_i h_{ij} x_i \geq b_j. \tag{A.13b}$$

Contrary to most descent methods which chose a descent direction based on the projection of the negative gradient of z on the set of feasible directions, the Frank Wolfe method considers also how far it is possible to move along a direction, as shown in figure A.2. This is done by solving the linear program

$$\min \mathbf{x} \cdot \boldsymbol{\nabla} z(\mathbf{x}_n) \tag{A.14a}$$

$$\text{subject to} \quad \sum_i h_{ij} x_i \geq b_j, \tag{A.14b}$$

i.e. the linearization of (A.13). This method of finding the descent direction has the additional advantage that we know how far we can proceed in this search direction without leaving the feasible set, since the optimal solution of a linear program is always situated at the boundary of the feasible set.

Let $\mathbf{x}^\star$ be the solution of (A.14). We know that all convex combinations $\lambda \mathbf{x}_n + (1 - \lambda)\mathbf{x}^\star$, $\lambda \in [0, 1]$ of $\mathbf{x}_n$ and $\mathbf{x}^\star$ are feasible. To find the optimal step size, we only have to solve the one-dimensional optimization problem

$$\min z\left(\lambda \mathbf{x}_n + (1 - \lambda)\mathbf{x}^\star\right) \tag{A.15a}$$

$$\text{subject to} \quad 0 \leq \lambda \leq 1. \tag{A.15b}$$

For our implementation we have used Brent's parabolic interpolation method as described in [74].

---

[2] Note that the restriction to inequality constraints in this section is only for shortness of notation. Equality constraints can still be included as two inequality constraints.

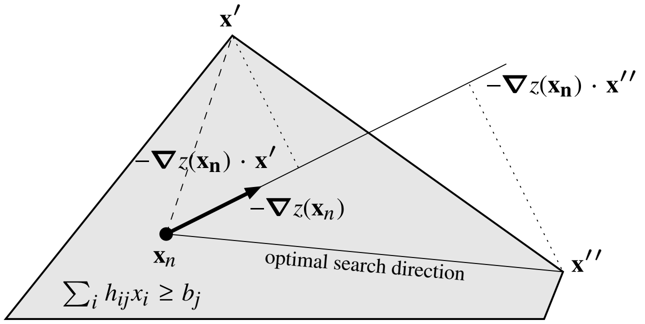

$\mathbf{x}'$
$-\nabla z(\mathbf{x_n}) \cdot \mathbf{x}''$
$-\nabla z(\mathbf{x_n}) \cdot \mathbf{x}'$
$-\nabla z(\mathbf{x}_n)$
$\mathbf{x}_n$
optimal search direction
$\mathbf{x}''$
$\sum_i h_{ij} x_i \geq b_j$

Figure A.2 The search direction of the Frank-Wolfe algorithm is chosen as the direction which gives the maximum improvement of the objective function $z(\mathbf{x})$ when taking into account not only the gradient of z at the current point $\mathbf{x}_n$ but also the constraints. The figure shows an example where, due to the constraints, the negative gradient is *not* the optimal descent direction of the linearized problem. Instead, x'' is the optimum solution of the linearized problem.

A further advantage of the Frank-Wolfe method is that it gives also a lower bound on the value of the objective function: Since z is convex, we know

$$z(\mathbf{x}) \geq z(\mathbf{x}_n) + (\mathbf{x} - \mathbf{x}_n) \cdot \nabla z(\mathbf{x}_n), \tag{A.16}$$

so we also have

$$\begin{aligned} \min_{h_{ij}x_i \geq b_j} z(\mathbf{x}) &\geq z(\mathbf{x}_n) + \min_{h_{ij}x_i \geq b_j} (\mathbf{x} - \mathbf{x}_n) \cdot \nabla z(\mathbf{x}_n) \\ &= z(\mathbf{x}_n) + (\mathbf{x}^\star - \mathbf{x}_n) \cdot \nabla z(\mathbf{x}_n) \\ &=: \underline{z}(\mathbf{x_n}), \end{aligned} \tag{A.17}$$

where $\mathbf{x}^\star$ is again the solution of (A.14). Having a lower bound $\underline{z}(\mathbf{x_n})$ on the objective function value is especially valuable in multidimensional optimization, since the position of the minimum cannot be bracketed like in the one-dimensional case.

The linear approximation step and the one-dimensional are iterated until the relative difference between upper and lower bound falls below some given limit. The whole algorithm reads as follows:

```
procedure Frank-Wolfe
begin
  $n := 0$
  $x_0 :=$ some feasible point
  repeat
    $\mathbf{y} :=$ Simplex$(\nabla z(\mathbf{x}_n), \mathbf{h}, \mathbf{b})$
    $\alpha := \arg\min_{\beta \in [0,1]} z((1-\beta)\mathbf{x}_n + \beta\mathbf{y})$
    $x_{n+1} := (1-\alpha)\mathbf{x}_n + \alpha y$
    $ub := z(\mathbf{x}_{n+1})$
```

```
        $lb := z(\mathbf{x}_n) + (\mathbf{y} - \mathbf{x}_n)\nabla z(\mathbf{x}_n)$
        $n := n + 1$
    until $\frac{ub - lb}{|ub + lb|} < \varepsilon$
end
```

Appendix

B

Details on *PLANSIM-T* and *FASTLANE*

B.1 The *PLANSIM-T*raffic Simulator

In the past years, different microscopic traffic simulation models have been studied at the ZPR, namely the models by Nagel and Schreckenberg, Krauß, Wiedemann [90] and the TOCA-model by Wu [10]. Although the car following behavior of all these models can be implemented rather easily, building a simulator for complex networks based on these models is a lot of programming work.

Except for the Wiedemann model, these models are all coupled map models (c.f. chapter 3), i.e. the velocity of each car is updated based on the current velocity of that car, the velocity of the car in front and the distance to the car in front. The lane change rules of these models furthermore take into account the velocities and distances of the cars on the neighboring lanes.

Since all these models need the same data to update a car, the obvious idea is to build a network simulation tool which can simulate arbitrary road networks based on either of these models. This idea resulted in the simulation tool PLANSIM-T. The basic idea of PLANSIM-T is that every complicated road network can be built up from basic components like

- links,
- crossings,
- traffic lights,
- yield- and stop-signs,
- on- and off-ramps.

All of these components were implemented in an object-oriented framework using C++. For example, the class EDGE provides all methods to determine the distance to the car in front and the velocity of the car in front. EDGES are connected by means of CONNECTIONS, which take care of the route choice and the priorities of crossing roads. Different models with different car following behavior and lane changing rules are implemented as subclasses of EDGES (e.g. CAEDGE for the Nagel-Schreckenberg model) which overload the methods describing the behavior of vehicles, like MOVE for the car-following and LANECHANGE for the lane-changing rules.

PLANSIM-T was developed jointly by Bartosz Borowik, Stefan Krauß, Peter Oertel, Felix Pütsch, Christian Rössel, Peter Wagner, and the author of this thesis. Figure B.1 shows an example of how more complicated structures can be built up using the building blocks provided by PLANSIM-T. A detailed documentation can be found in [8]. Of course, for networks of the size of the Wuppertal network building and maintaining the input files for PLANSIM-T is infeasible. Therefore, tools have been built to generate PLANSIM-T input files from various graph formats.

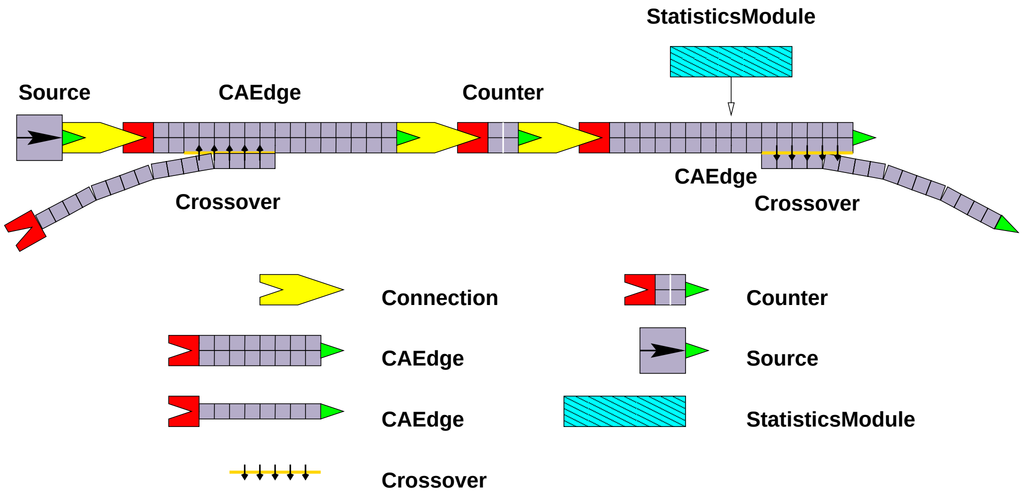


Figure B.1 An example of a network in PLANSIM-T

B.2 *FASTLANE*

FASTLANE is a software package for simulation-based dynamic traffic assignment developed by the author of this thesis. It consists of

- a traffic simulator based on the queuing model described in section 3.4,
- a program implementing the update rules described in section 4,
- a program to generate a set of individual drivers from a time-dependent OD matrix,
- a driver script to do the iterations of the simulation/update process and some additional tools described later.

All of these programs rely on a representation of the road network as a graph or a line graph (actually, this distinction is only important for the simulator). FASTLANE was implemented in C++ using the LEDA library (c.f. [60]), which provides all necessary graph-related data structures. Figure B.2 shows how the data flow between the various programs during the iterations of the DTA. The formats of the data files are described in the next section.

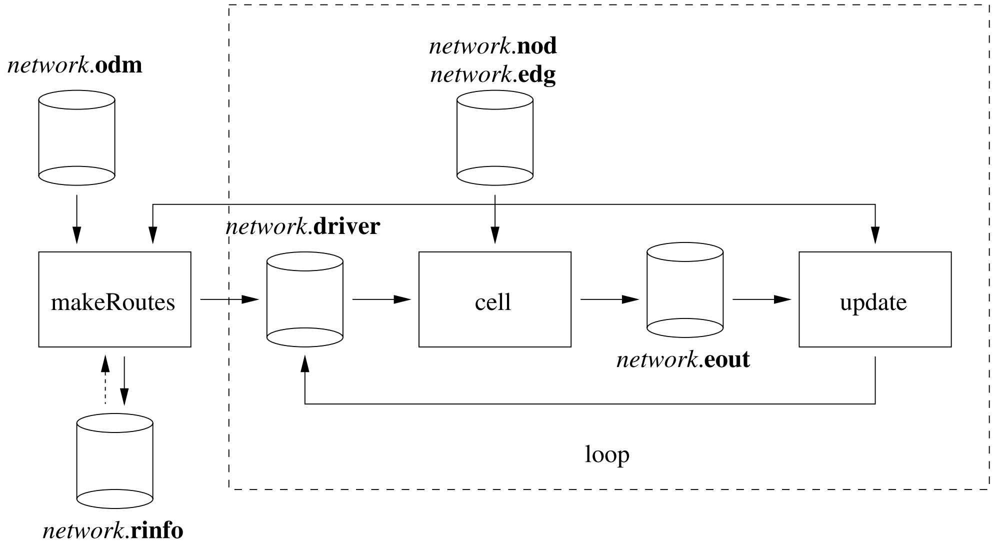


Figure B.2 Flow chart of the DTA algorithm

B.2.1 Input Formats

To allow an easy exchange of data with other programs, most of the input files of FASTLANE use very simple ASCII formats. Only the driver set is stored in a binary format for performance reasons.

The Graph

The graph underlying the network is stored in the files *network*.**`nod`** and *network*.**`edg`**, the former specifying the nodes, the latter specifying the edges.

The .**`nod`** file uses the simple format

id   x-coordinate   y-coordinate

The coordinates are only used for the (optional) graphics and to calculate the length of edges if the length is not specified in the **`.edg`** file. For the latter purpose, the coordinates should be specified in meters (like UTM or Gauß-Krüger coordinates [36]).

The format of the **`.edg`** file is a little bit more complicated since all important attributes of the queuing model (capacity $q_{\max}$, maximum number of cars $n_{\max}$) are associated with edges in the graph[1]:

id   from-node   to-node   length   [options]

The possible options are listed in table B.1.

The OD Matrix

The time-dependent OD matrix can be specified either by Fourier coefficients in the form

---

[1] We use the standard Backus-Naur notation [x] to specify that x can occur zero or more times

Option	Type	Default	Meaning
-c $q_{\max}$	`float`	$3.0s^{-1}$	Capacity of the link. The default value corresponds to a three-lane freeway.
-n $n_{\max}$	`float`	$0.4m^{-1} \cdot l$	Maximum number of vehicles on the link. The default value corresponds to a three-lane freeway. This value will always be increased by $2q_{\max}\Delta t$
-v $v_{\max}$	`float`	$43.5ms^{-1}$	Corresponds to the mean velocity in the Nagel-Schreckenberg model with $v_{\max} = 6$ and $p_{\text{brake}} = 0.2$.
-s σ_q	`float`	0	Sigma of the capacity. See section 3.4.3.
-l l	`float`		This option overrides the length specified in column three.

Table B.1 Possible options to be specified in the `.edg` file

from-node *to-node* $d_0[\ [a_i b_i]\ ]$

or by giving a value for each time period in the form

from-node *to-node* $d_0\ [d_i]$

In the former case, the demand between the nodes will be calculated as

$$d_0 + \sum_{i=1}^{n} (a_i \sin i\omega t + b_i \cos i\omega t)\,, \tag{B.1}$$

while in the latter case the demand in time period $i = nt/T$, where n is the number of values specified and T is the length of the simulation period, will be d_i. Unless specified, all programs assume $T = 1\text{day} = 86400s$.

B.2.2 The Driver File

All the variables associated with a driver (c.f. section 4), i.e.

- the origin O_d, the destination D_d and the departure time t_d,
- the set $\mathcal{P}_d$ of routes from O_d to D_d,
- the probability distribution p_d,
- the 'learned travel times' τ_d,

are generated by the program **`makeRoutes`** and stored in the binary file *network*`.driver`. This has the disadvantage of making this file machine-dependent and unreadable, but saves a lot of space and speeds up the input of this file a lot — which is critical for the very simple queuing model. The file is accessed by mapping it into the address space of the program using the **`mmap`** system call of the UNIX operating system. Since there may be many drivers sharing a single route, not the route but only the index of a route in a route file is stored.

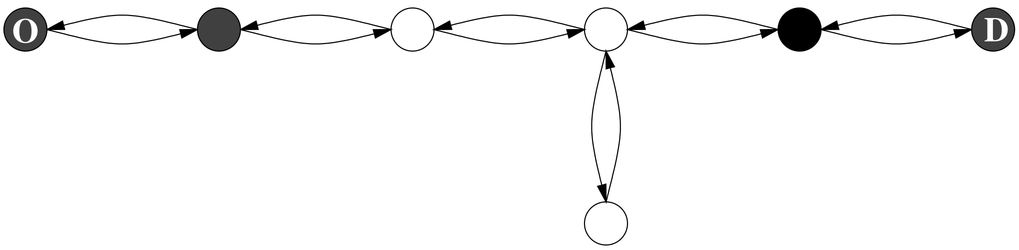


Figure B.3 Compression of routes. Except for the origin, the destination and the first node of a route, only nodes which are necessary to reconstruct the route are stored. These are nodes w with (v, w) being part of the route and with either outdegree(v) > 2 or outdegree(v) $= 2$ and $(w, v) \notin \mathcal{A}$, i.e. nodes following a node where there are at least two other possible directions other than turning around. The black node is an example of such a node.

```
0  200  200
1  100  100
2 -100  100
3  100 -100
4 -100 -100
5  400  400
```

An example `.nod` file.

```
0>1 0 1 2000 -c 5.0
1>2 1 2 1000 -v 40 -c 0.8
1>3 1 3 8000 -v 10 -c 1.2
2>3 2 3 1000 -v 40 -c 1.2
2>4 2 4 8000 -v 10 -c 1.2
3>4 3 4 1000 -v 40 -c 0.8
```

An example `.edg` file

Figure B.4 Example graph file for the dynamic version of Braess's paradox (see section 5.1).

B.2.3 The Route File

The set of all routes in the simulation is stored in the *network*`.rinfo`. As for the `.driver` file, a binary format would be preferable for performance reasons. But since the routes have to be read only once at the beginning of the simulation, we decided that the advantages of a human-readable format outweigh the performance disadvantages. Furthermore, a binary format would rely on the numbering of nodes in the `.node` file, so adding a single node would make the whole route file unusable.

The routes are represented as lists of nodes, therefore the input format is

origin $[v_i]$ *destination*

and the index of a route is just the line number in the `.rinfo` file. To save both disk space and main memory, only the nodes necessary to reconstruct the path in the network are stored (see figure B.3). The route file does not have to be generated by hand. The `makeRoutes` program will generate it or will add additional routes if necessary.

```
0 4 1.1 [0 -1]
```

Figure B.5 An example `.odm` file for the Braess network, specifying a demand of the form $1.1 - \cos(\omega t)$ between the nodes 0 and 4.

```
0 1 2 3 4
0 1 3 4
0 1 2 4
```

Figure B.6 An example `.rinfo` file for the Braess network, specifying the three possible routes in the network.

B.2.4 The Programs and their Output Formats

Generating Individual Drivers: `makeRoutes`

For the simulation based DTA algorithm, a population of individual drivers which is consistent with the time-dependent OD matrix has to be generated. This is done using the program **`makeRoutes`**.

Syntax:

`makeRoutes` [*options*] *network*

Table B.2 lists the options understood by **`makeRoutes`**.

Option	Type	Default	Meaning
-t $t_{\max}$	**`int`**	$86400s$	Time period for which to generate drivers. N.B: This does not effect the period T of the OD matrix.
-T T	**`int`**	$86400s$	Period of the OD matrix, i.e. the smallest $\Delta t > 0$ with $d(t+\Delta t) = d(t)$. Affects both OD matrices defined by Fourier series and by individual demand values.
-q Δt	**`int`**	$30s$	Time step. Should be less than the time step of the subsequent simulation, i.e. should be 1 if PlanSim-T is used. Otherwise, cars would enter the net as bulks every Δt seconds.
-f f	**`float`**	1	Demand is multiplied by f.
-l			Specifies that the graph should be interpreted as a line graph.
-o ω	**`float`**	$2\pi/T$	Sets ω for the Fourier series.
-m *mode*	**`int`**	1	If *mode* is 1, the demand is distributed randomly on all available routes. If *mode* is 2, the demand is put on the first available route only.

Table B.2 Options understood by **`makeRoutes`**.

Input Files: `makeRoutes` uses the following input files: *network*.`nod`, *network*.`edg`, *network*.`odm` and — optional — *network*.`rinfo`

Output Files: `makeRoutes` generates *network*.`driver` and writes *network*.`rinfo` if new routes have been generated. If necessary, new routes are generated as shortest paths between the origin and destination nodes.

The Simulator: `cell`

`cell` is the implementation of the queuing model discussed in section 3.4.

Syntax:

`cell` [*options*] *network*

Table B.3 lists the options understood by `cell`.

Option	Type	Default	Meaning
-t $t_{\max}$	`int`	infinity	Specifies the simulation period.
-q Δt	`int`	$30s$	Specifies the time step of the simulation.
-l			Specifies that the graph is a line graph.
-g			Enables a very simple visualization.
-R			Read-only mode. Do not generate driver output files (see below).
-Q			Quiet mode. Disables 'progress meter'.

Table B.3 Options are understood by `cell`

Input Files: `cell` uses the following input files: *network*.`nod`, *network*.`edg`, *network*.`rinfo` and *network*.`driver`. If the latter file is missing, looks for an *network*.`odm` and generates drivers directly from the OD matrix.

Output Files: To discern the output files from various runs, `cell` names all output files in the form *network_pid*.`xxx`, where *pid* is the process id under which `cell` is running. The following output files are important:

***network_pid_i*.`driver`** The drivers in the *network*.`driver` file have to be sorted by departure time. However, in general drivers will not arrive at their destination in order of departure time. In the first version of the DTA algorithm, the update rules were implemented directly in `cell` and the updated drivers were written back to the *network*.`driver` file which was mapped into the address space in read/write mode. However, the resulting non-sequential write accesses led to a large I/O overhead since one cannot control the time when a 'dirty' page, i.e. a memory page containing modifications, is written back to the file, so that each page possibly have to be written many times. For this reason, the drivers are now written sequentially in order of arrival. To minimize the cost of sorting this file afterwards, the drivers are put into buckets, i.e. different files, depending on their departure interval. By default, 24 files are created, one for each hour.

***network_pid*.`out`** Global information on the network state is written to this file. The format is

$t \quad N \quad q_{in} \quad q_{out} \quad \langle t\rangle \quad d_{total}$

where t is the time, N is the number of drivers currently in the network, q_{in} is the flow into the network, q_{out} is the flow out of the network, $\langle t\rangle$ is the mean travel time of cars which have arrived in the current time interval, and d_{total} is the total distance traveled by all drivers which have arrived so far.

***network_pid*.`eout`** In this file, the costs of traveling along each edge are stored in a binary format. This file is used by **`update`** to calculate the shortest path with respect to the actual travel times in the simulation. The data are written in intervals of 1800s.

Updating the Route-Choice Probabilities: `update`

The update rules of the DTA algorithm described in section 4 are implemented in **`update`**. Additionally, **`update`** sorts the drivers which are written by **`cell`** in order of arrival by departure time.

Syntax:

`update` [*options*] *network*

The following option is understood by **`update`**:

Option	Type	Default	Meaning
-A a	**`float`**	0.1	Sets the parameter a in the DTA algorithm (c.f. equation (4.5)).
-B β	**`float`**	0	Sets the parameter β in the DTA algorithm (c.f. equation (4.1b)).
-C c	**`float`**	0	Add random number uniformly distributed between $[-c, c]$ to δ_{rs} in equation (4.2).

Input Files: Except for the graph files, **`update`** reads the drivers from *network_i*.**`driver`** and the time-dependent edge weights from *network*.**`eout`**.

Output Files: **`update`** writes the file *network*.**`driver`** and — if routes have been added — *network*.**`rinfo`**.

Extracting Statistics: `rStat`

Since the driver file is written in a binary format, it is difficult to gather statistics about the route choice probabilities for a single origin destination pair by means of the standard

UNIX tools. `rStat` is a tool written for this purpose. Further, `rStat` can be used to visualize the routes with ArcView [24].

Syntax:

`rStat` [*options*] *network*

`rStat` recognizes the following options:

Option	Type	Default	Meaning
-o *origin*	`string`		Specify the origin node
-d *destination*	`string`		Specify the destination node
-q Δt	`int`	3600*s*	Specify the time interval which is sampled over.
-A			ArcView mode. Generate a table suitable to visualize a single Route with ArcView.
-n *route*	`int`		Only for ArcView mode: Specify route to draw.

Input Files: `rStat` reads the graph files, *network*.`driver` and *network*.`rinfo`.

Output Files: `rStat` writes to the standart output. By default, the following data are written:

- A header consisting of two lines, each starting with a #. The first line gives the total number of drivers. The second line gives a list of all routes between the specified origin and destination from the *network*.`rinfo` file.
- The following lines are of the form
  $t \quad n \quad [p_i T_i]$ where t is the time, n is the number of drivers with this OD relation starting between t and $t + \Delta t$. For each route i between the origin and destination, p_i is the average probability of this route, and T_i is the mean travel time of this route as 'learned' by the drivers.

In ArcView mode, a table in format suitable for ArcView is generated. In this table, all edges of the graph belonging to the specified route are marked, so that the route can be drawn with ArcView.

Doing Multiple Iterations: `loop`

Doing multiple iterations of `cell` and `update` by hand would be tedious. Furthermore, the output files of `cell` have to be renamed for `update` (the *pid* part has to be removed). To make this more comfortable, the shell script `loop` is provided.

Syntax:

`loop` [*options*] *network*

In addition to the options understood by **`cell`**, **`update`**, and **`rStat`**, **`loop`** recognizes the following options:

Option	Type	Default	Meaning
-n *iter*	**`int`**	5	Do *iter* iterations.
-I *i*	**`int`**	0	Start at iteration *i*.
-r			Resume. Checks if output files of a previous run exist and skips corresponding iterations.

Input Files: To run properly, **`loop`** requires the graph files, *network*.**`driver`** and *network*.**`rinfo`**.

Output Files: The output files of **`cell`** are renamed *network*.**`out_`*`i`***, where *i* is the iteration number.

Repairing Drivers: `salvage`

Due to rounding errors, the probability distributions of drivers may become unnormalized. **`salvage`** checks the driver file for unnormalized or illegal route choice probabilities and also for illegal, i.e. unknown, route numbers. If possible, these errors are corrected.

Syntax:

`salvage` *network*

Input Files: *network*.**`driver`**, *network*.**`rinfo`**

Output Files: *network*.**`driver`**

Appendix C

Networks

This appendix summarizes the three ‘real world’ road networks which were used for the simulations presented in this thesis.

C.1 The Network of ‘Seestrauch’

The ‘Seestrauch’ network (see figure C.1) is the road network of a small city in Nordrhein–Westfalen and was used by courtesy of the Institut für Stadtbauwesen (ISB) der Technischen Universität Aachen, which requested us not to reveal the real name of ‘Seestrauch’. It consist of 115 nodes and 278 links, of which 38 nodes are origin and destination nodes and 21 links are highway links. To represent the turning restrictions in the urban part, additional nodes and links have been inserted, resulting in a total of 432 nodes and 646 edges. The average link length is 160*m*, the total length is 103*km*.

C.2 The Highway Network of the Landschaftsverband Rheinland

This network is the major part of the highway network of Nordrhein–Westfalen. This network was provided by the Landschaftsverband Rheinland and the ISB, which also provided the time-dependent origin destination matrix for this network. The network consists of 481 nodes and 996 links, the origin destination matrix has a time resolution of one hour. The average link length is 2.4*km*, the total length is 2 415*km*.

C.3 The Network of Wuppertal

This is by far the most complex network considered in this thesis, consisting of 9098 nodes and 16769 links with a total length of 5 760*km*. The data for this network was provided by the administration of Wuppertal, by the Lehrstuhl für Umweltverträgliche Infrastrukturplanung (LUIS) of the Bergische Universität/Gesamthochschule Wuppertal

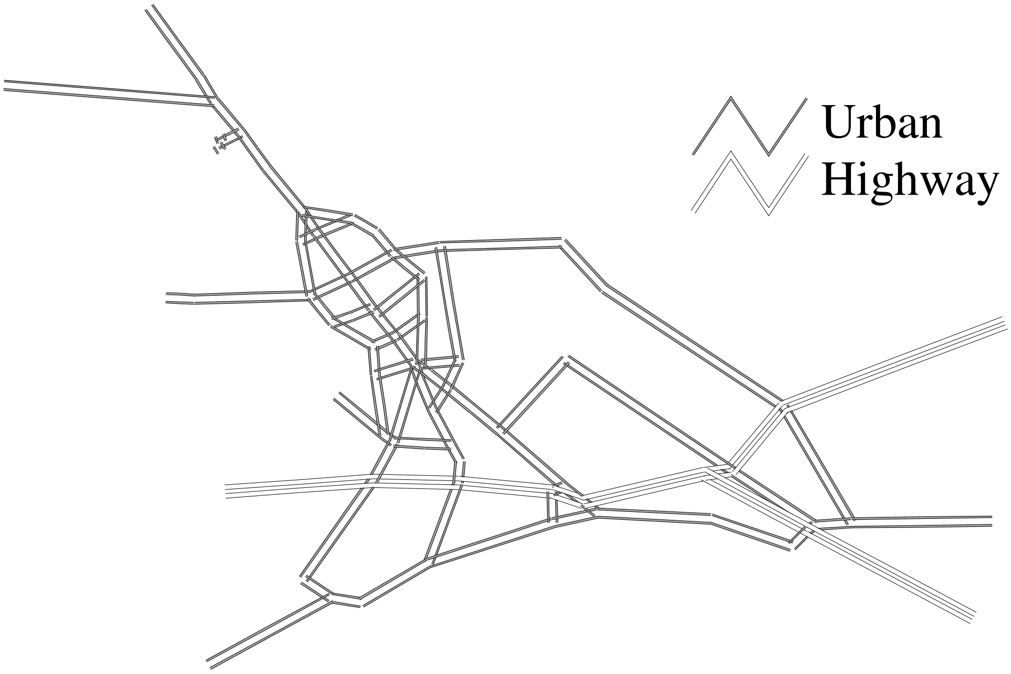


Figure C.1 The ‘Seestrauch’ network

and partly taken from a geographical database by Navigation Technologies. A time dependent origin destination matrix with a temporal resolution of one hour between 462 OD cells was provided by the ISB.

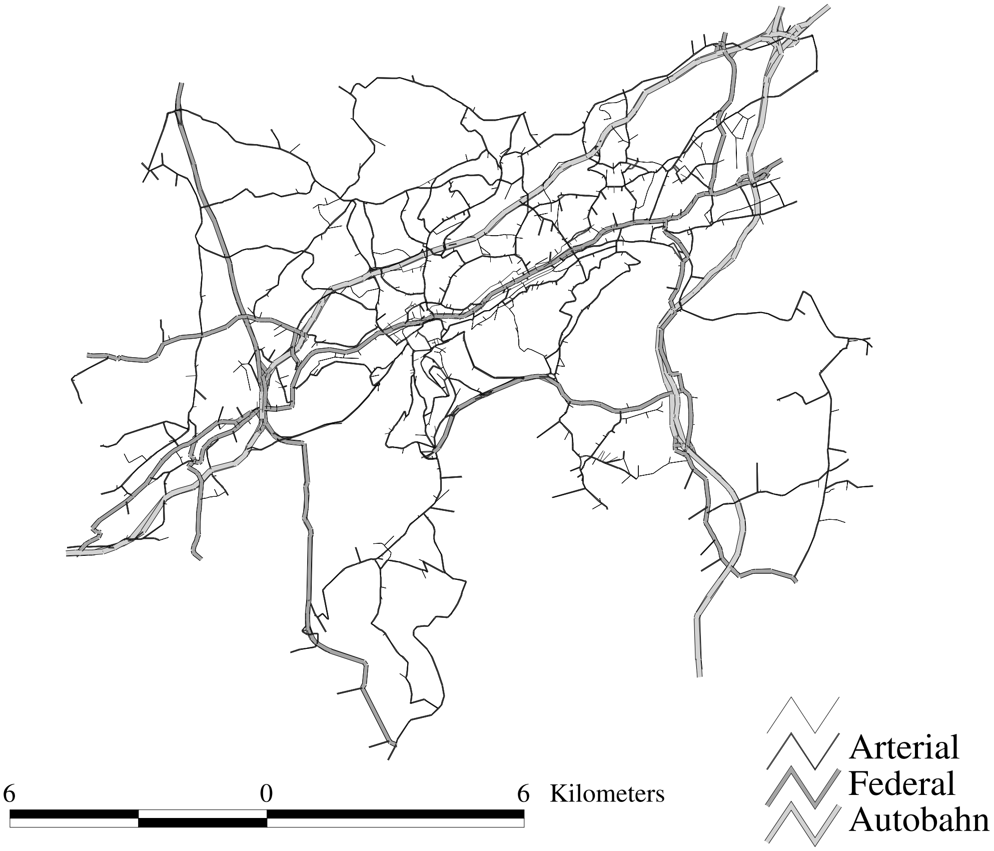


Figure C.2 The Wuppertal network

Appendix D

Glossary of Notation

Symbol	Meaning	See page
$\mathcal{G}$	Graph representing the traffic network	8
$\mathcal{A}$	Set of arcs in the network	8
$\mathcal{N}$	Set of nodes in the network	8
$\mathcal{O}$	Set of origin nodes	8
$\mathcal{D}$	Set of destination nodes	8
x_a	Flow on link a	8
τ_a	Travel time on link a	8
$\mathcal{P}_{rs}$	Set of paths from r to s	8
d_{rs}	Travel demand from r to s	8
f_p^{rs}	Flow from r to s on path p	8
τ_p^{rs}	Cost of traveling from r to s on path p	8
$\delta_{a,p}^{rs}$	Arc-path incidence indicator variable	8
$x_a(t)$	Number of vehicles traveling on link a at time t	24
$x_{ap}^{rs}(t)$	Number of vehicles traveling on link a over route p from r to s at time t	24
$u_a(t)$	Inflow rate on link a at time t	25
$u_{ap}^{rs}(t)$	Dto. with respect to route p from r to s	25
$v_a(t)$	Outflow rate off link a at time t	25
$v_{ap}^{rs}(t)$	Dto. with respect to route p from r to s	25
$A(v)$	Set of links going out of v	25
$B(v)$	Set of links going into v	25
$f_a^{rs}(t)$	Rate of vehicles departing at node r to destination s at time t	25
$e_a^{rs}(t)$	Rate of vehicles arriving at node s from origin r at time t	25
$\psi_p^{rs}(t)$	Instantaneous travel time over route p at time t	26
$\eta_p^{rs}(t)$	Actual travel time over route p at time t	27
$\sigma^{rs}(t)$	Minimum instantaneous travel time from r to s at time t	27
$\pi^{rs}(t)$	Minimum actual travel time from r to s at time t	27
O_d	Origin of driver d	53

D_d	Destination of driver d	53
t_d	Departure time of driver d	53
$\mathcal{P}_d$	Set of Routes known by driver d	53
p_d	Probability distribution according to which driver d chooses her route	53

Bibliography

[1] R. K. Ahuja, T. L. Magnati, and J. B. Orlin. *Network Flows*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[2] H. Baum et al. Research Cooperative for the Simulation of Traffic and Environmental Impacts. In M. Schreckenberg and D. E. Wolf, editors, *Traffic and Granular Flow '97*, pages 229–236, Singapore, 1998. Springer Verlag.

[3] M. S. Bazaraa and C. M. Shetty. *Nonlinear Programming*. John Wiley & sons, New York, 1979.

[4] M. Beckmann, C. B. McGuire, and C. B. Winsten. *Studies in the Economics of Transportation*. Yale University Press, New Haven, 1956.

[5] M. Ben-Akiva, M. Bierlaire, H. N. Koutsopoulos, and R. Mishalani. DynaMIT: A Simulation-Based System for Traffic Prediction and Guidance Generation. Presented at TRISTAN III, San Juan, Porto Rico. Available at http://its.mit.edu/, June 1998.

[6] M. E. Ben-Akiva, M. Bierlaire, J. Bottom, H. N. Koutsopoulos, and R. Mishalani. Development of a Route Guidance Generation System for Real-Time Application. In *Proceedings of the 8th IFAC/IFIP/IFORS Symposium on Transportation Systems*, Chania, Greece, 1997.

[7] D. H. Bernstein and S. A. Gabriel. The Traffic Equilibrium Problem with Nonadditive Path Costs. *Transp. Sc.* **31**:324–336, 1997.

[8] B. Borowik, C. Gawron, S. Krauß, P. Oertel, F. Pütsch, C. Rössel, and P. Wagner. The PLANSIM-Traffic Simulation. Working Paper 338, ZPR, 1998. Available at http://www.zpr.uni-koeln.de/~paper.

[9] D. Braess. Über ein Paradoxon der Verkehrsplanung. *Unternehmensforschung* **12**:256–268, 1968. In German.

[10] W. Brilon and N. Wu. Evaluation of Cellular Automata for Traffic Flow Simulation on Freeways and Urban Streets. In K. J. Beckmann, editor, *Tagungsband zum Ergebnis-Workshop Verkehr und Mobilität*, number 66 in Stadt Region Land, pages 111–117. Institut für Stadtbauwesen der Technischen Universität Aachen, Aachen, 1998.

[11] W. Brücher, C. Kessler, A. Ebel, and M. J. Kerschgens. Regionale und lokale Immissionsmodellierung auf der Basis von dynamischen Verkehrssimulationen. In K. J. Beckmann, editor, *Tagungsband zum Ergebnis-Workshop Verkehr und Mobilität*, number 66 in Stadt Region Land, pages 147–154. Institut für Stadtbauwesen der Technischen Universität Aachen, Aachen, 1998.

[12] Forschung Straßenbau und Verkehrstechnik, Heft 490. Bundesminister für Verkehr, Abt. Straßenbau, Bonn, 1986.

[13] Bundesverkehrsministerium, editor. *Verkehr in Zahlen*. Deutsches Institut für Wirtschaftsforschung, 1996.

[14] M. Carey. A Constraint Qualification for a Dynamic Traffic Assignment Model. *Transp. Sc.* **20**:55–58, 1986.

[15] R. Cayford, W.-H. Lin, and C. F. Daganzo. The NETCELL simulation package: Technical description. California PATH Research Report UCB-ITS-PRR-97-23, Institute of Transportation Studies, University of California, Berkeley, CA, May 1997. Available at http://www.ce.berkeley.edu/Programs/Transportation/Daganzo/publications.html.

[16] I. Chabini and Y. He. A Flow-Based Approach to Dynamic Traffic Assignment. Presented at the CORS/INFORMS Meeting, Montreal, Canada, Apr. 1998. See also [34].

[17] R. G. Clegg, M. O. Ghali, and M. J. Smith. The Importance of Route Choice Methods in Dynamic Micro-simulation/Assignment Traffic Modelling. Available at http://gridlock.york.ac.uk/htdocs/papers.html [1997, July 21], 1997.

[18] M. Cremer and J. Ludwig. A Fast Simulation Model for Traffic Flow on the Basis of Boolean Operations. *Mathematics and Computers in Simulation* **28**:297–303, 1986.

[19] C. F. Daganzo. The Cell transmission Model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transp. Res.* **28B**(4):269–287, 1994.

[20] C. F. Daganzo. The cell transmission model. Part II: Network traffic. *Transp. Res.* **29B**(2):79–93, 1995.

[21] K. B. Davidson. A Flow-Travel Time Relationship for Use in Transportation Planning. In *Proceedings of the Australian Road Research Board*, volume 3, pages 183–194, Melbourne, 1966.

[22] K. Deimling. *Nonlinear Functional Analysis*, chapter 9, pages 319–377. Springer Verlag, Berlin Heidelberg New York, 1980.

[23] E. W. Dijkstra. A Note on two Problems in Connection with Graphs. *Numerische Mathematik* **1**:269, 1959.

[24] Environmental Systems Research Institute, Redlands, CA. *Using ArcView GIS*. See also http://www.esri.com.

[25] C. Fisk. Some Developments in Equilibrium Traffic Assignment. *Transp. Res.* **14B**:243–255, June 1980.

[26] B. L. Fox. Data Structures and Computer Science Techniques in Operations Research. *Op. Res.* **26**:686–717, 1978.

[27] M. Frank and P. Wolfe. An Algorithm for Quadratic Programming. *Naval Research Logistics Quarterly* **3**:95–110, 1956.

[28] T. L. Friesz, D. Bernstein, T. E. Smith, R. L. Tobin, and B. W. Wie. A Variational Inequality Formulation of the Dynamic Network User Equilibrium Problem. *Op. Res.* **41**(1):179–191, 1993.

[29] T. L. Friesz, J. Luque, R. L. Tobin, and B.-W. Wie. Dynamic Network Traffic Assignment Considered as a Continuous Time Optimal Control Problem. *Op. Res.* **37**(6):893–901, 1989.

[30] Homepage of the research cooperative 'Simulation of Traffic and Environmental Impacts' (FVU). http://www.zpr.Uni-Koeln.DE/Forschungsverbund-Verkehr-NRW/ .

[31] C. Gawron. An Iterative Algorithm to Determine the Dynamic User Equilibrium in a Traffic Simulation Model. *Int. Jrn. Mod. Phys. C* **9**(3):393–407, 1998.

[32] C. Gawron, S. Krauß, and P. Wagner. Dynamic User Equilibria in Traffic Simulation Models. In M. Schreckenberg and D. E. Wolf, editors, *Traffic and Granular Flow '97*, pages 469–473, Singapore, 1998. Springer Verlag.

[33] D. C. Gazis, R. Herman, and R. W. Rothery. Nonlinear Follow-the-Leader Models of Traffic Flow. *Op. Res.* **9**:545–567, 1961.

[34] Y. He. A Flow-Based Approach to the Dynamic Traffic Assignment Problem: Formulations, Algorithms and Computer Implementations. Master's thesis, Center for Transportation Studies, MIT, June 1997.

[35] D. Helbing. Improved Fluid-Dynamic Model for Vehicular Traffic. *Phys. Rev. E* **51**:3164–3169, 1995.

[36] J. Hoschek. *Mathematische Grundlagen der Kartographie*. B.I.-Wissenschaftsverlag, 2nd edition, 1984. In German.

[37] F. Huber and S. Kaufmann. Zeitteilige Bewertung verkehrlicher Wirkungen auf der Basis von dynamischer Verkehrssimulation. In K. J. Beckmann, editor, *Tagungsband zum Ergebnis-Workshop Verkehr und Mobilität*, number 66 in Stadt Region Land, pages 163–170. Institut für Stadtbauwesen der Technischen Universität Aachen, Aachen, 1998.

[38] Institut für Stadtbauwesen der TU Braunschweig. *Planungshandbuch Umwelt und Verkehr*, 1986. published in [12].

[39] Institute for Transport Studies, University of Leeds. *SMARTEST (Simulation Modelling Applied to Road Transportation European Scheme Test*. See http://www.its.leeds.ac.uk/smartest/.

[40] S. Janz. Mikroskopische Minimalmodelle des Straßenverkehrs. Diploma thesis, University of Cologne, 1998. In German.

[41] D. E. Kaufman, R. L. Smith, and K. E. Wunderlich. User-equilibrium Properties of Fixed Points in Iterative Dynamic Routing/Assignment Methods. Available at http://www.ecs.umass.edu/ieor/faculty/kaufman/ [1997, December 2]. Submitted to Transp. Res. C.

[42] T. Kelly and K. Nagel. Relaxation criteria for iterated traffic simulations. *Int. Jrn. Mod. Phys. C* **9**(1):113–132, 1998.

[43] B. S. Kerner and P. Konhäuser. Cluster Effect in Initially Homogeneous Traffic. *Phys. Rev. E* **48**(4):R2335–R2338, 1993.

[44] B. S. Kerner and H. Rehborn. Experimental Features and Characteristics of Traffic Jams. *Phys. Rev. E* **53**:1297–1300, 1996.

[45] B. S. Kerner and H. Rehborn. Experimental Properties of Complexity in Traffic Flow. *Phys. Rev. E* **53**:4275–4278, 1996.

[46] B. S. Kerner and H. Rehborn. Experimental Properties of Phase Transition in Traffic Flow. *Phys. Rev. Letters* **79**:4030, 1997.

[47] C. Kessler, W. Brücher, M. J. Kerschgens, and A. Ebel. Wechselwirkung von verkehrsbedingten und anderen anthropogenen Emissionen bei der Schadstoffausbreitung in belasteten Räumen und ihrer Umgebung. In K. J. Beckmann, editor, *Tagungsband zum Ergebnis-Workshop Verkehr und Mobilität*, number 66 in Stadt Region Land, pages 147–154. Institut für Stadtbauwesen der Technischen Universität Aachen, Aachen, 1998.

[48] L. Kleinrock. *Queueing Systems*. John Wiley & sons, New York, 1975.

[49] D. E. Knuth. *The TEXBook*. Addison-Wesley, Reading, Massachusetts, 1984.

[50] S. Krauß. Towards a Unified View of Microscopic Traffic Flow Theories. In M. Papageorgiou and A. Pouliezos, editors, *Proceedings of the 8th IFAC/IFIP/IFORS Symposium, Chania, Greece, 16-18 June 1997*, volume 2. Elsevier Science, 1997.

[51] S. Krauß. *Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics*. PhD thesis, University of Cologne, 1998.

[52] H. W. Kuhn and A. W. Tucker. Nonlinear Programming. In J. Neyman, editor, *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley and Los Angeles, California, 1961. University of California Press.

[53] R. D. Kühne. Macroscopic Freeway Model for Dense Traffic — Stop-Start Waves and Incident Detection. In J. Volmuller and R. Hamerslag, editors, *Proceedings of the Ninth International Symposium on Transportation and Traffic Theory*, Utrecht, The Netherlands, 1984. VSP.

[54] T. Larsson and M. Patriksson. Simplicial Decomposition with Disaggregated Representation for the Traffic Assignment Problem. *Transp. Sc.* **26**:4–17, 1992. available at http://www.math.chalmers.se/~mipat/vita.html [1998, October 6].

[55] D. R. Leonard, P. Gower, and N. B. Taylor. CONTRAM: Structure of the Model. Research Report 178, Transportation Research Laboratory, Crowthorne, UK, 1989.

[56] M. J. Lighthill and G. B. Whitham. On Kinematic Waves (Part II): A Theory of Traffic Flow on Crowded Roads. *Proceedings of the Royal Society* **A 229**:317–345, 1955.

[57] J. Ludmann, D. Neunzig, and M. Weilkes. Traffic Simulation with Consideration of Driver Models — Theory and Examples. In *Proceedings of the EUROMOTOR-Seminar on Telematics, Vehicle and Environment*, Aachen, 1996.

[58] H. Mahmassani, T.-Y. Hu, and R. Jayakrishnan. Dynamic Traffic Assignment and Simulation for Advanced Network Informatics (DYNASMART). In N. H. Gartner and G. Improta, editors, *Urban Traffic Networks: Dynamic Flow Modeling and Control*. Springer Verlag, Berlin Heidelberg New York, 1995.

[59] H. S. Mahmassani and R. Herrman. Dynamic User Equilibrium Departure Time and Route Choice on Idealized Traffic Arterials. *Transp. Sc.* **18**:362–384, 1984.

[60] K. Mehlhorn and S. Näher. *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press, to appear in January, 1999. More information on LEDA is available at http://www.mpi-sb.mpg.de/LEDA/.

[61] D. K. Merchant and G. L. Nemhauser. A Model and an Algorithm for the Dynamic Traffic Assignment Problem. *Transp. Sc.* **12**:183–199, 1978.

[62] D. K. Merchant and G. L. Nemhauser. Optimality Conditions for a Dynamic Traffic Assignment Model. *Transp. Sc.* **12**:200–207, 1978.

[63] Ministerium für Stadtentwicklung und Verkehr des Landes Nordrhein-Westfalen, editor. *Daten und Fakten zum NRW-Verkehr*. WAZ-Druck, Duisburg, 1994. In German.

[64] J. D. Murchland. Braess's Paradox of Traffic Flow. *Transp. Res.* **4**:391–394, 1970.

[65] K. Nagel. *High-speed microsimulations of traffic flow*. PhD thesis, Universität zu Köln, 1995.

[66] K. Nagel. Experiences with iterated traffic microsimulations in Dallas. In D. E. Wolf and M. Schreckenberg, editors, *Traffic and Granular Flow*. Springer, 1998. in press.

[67] K. Nagel, R. Frye, R. Jakob, M. Rickert, and P. Stretz. Dynamic traffic assignment on parallel computers. Unclassified Report LA-UR 98-2944, LANL, 1998. Submitted to ISCOPE 98. Available at http://www-transims.tsasa.lanl.gov/documents-research98.html.

[68] K. Nagel and M. Paczuski. Emergent Traffic Jams. *Phys. Rev. E* **51**:2909, 1995.

[69] K. Nagel and M. Schreckenberg. A Cellular Automaton Model for Traffic Flow. *J. Physique I* **2**:2221, 1992.

[70] D. Neunzig. personal communication.

[71] M. Patriksson. *The Traffic Assignment Problem*. Topics in Transportation. VSP, Utrecht, The Netherlands, 1994.

[72] A. Perko. Implementation of Algorithms for *K* Shortest Loopless Paths. *Networks* **16**:149–160, 1986.

[73] M. Ponzlet and P. Wagner. Validation of a CA-Model for Traffic Simulation of the Northrhine-Westphalia Motorway Network. In *The 24th European Transport Forum, Proceedings of Seminar D&E*, volume P 404–1, 1996.

[74] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2nd edition, 1992.

[75] H. A. Rakha and M. van Aerde. *Comparison of Simulation Modules of TRANSYT and INTEGRATION Models*. Number 1566 in Transportation Research Record. Transportation Research Board, 1996.

[76] B. Ran, D. Boyce, and L. J. LeBlanc. A New Class of Instantanious Dynamic User-Optimal Traffic Assignment Models. *Op. Res.* **41**:192–202, 1993.

[77] B. Ran and D. E. Boyce. *Modeling Dynamic Transportation Networks*. Springer, 2 edition, 1996.

[78] M. Rickert. *Traffic Simulation on Distributed Memory Computers*. PhD thesis, University of Cologne, Jan. 1998. Available at http://www.zpr.uni-koeln.de/~mr/dissertation.

[79] A. Schadschneider and M. Schreckenberg. Cellular Automaton Models and Traffic Flow. *Journal of Physics A* **26**:L679, 1993.

[80] H. Schütt. Entwicklung eines sehr schnellen, bitorientierten Verkehrssimulationssystems für Straßennetze. Schriftenreihe der ag automatisierungstechnik, TU Hamburg Harburg, 1991. In German.

[81] T. Schwerdtfeger. DYNEMO: A Model for the Simulation of Traffic Flow in Motorway Networks. In J. Volmuller and R. Hamerslag, editors, *Proceedings of the Ninth International Symposium on Transportation and Traffic Theory*, Utrecht, The Netherlands, 1984. VSP.

[82] T. Schwerdtfeger. *Makroskopisches Simulationsmodell für Schnellstraßennetze mit Berücksichtigung von Einzelfahrzeugen (DYNEMO)*. PhD thesis, Universität Karlsruhe, 1987. In German. See also [81].

[83] D. Serwill. *DRUM — Modellkonzept zur dynamischen Routensuche und Umlegung*. PhD thesis, RWTH Aachen, 1994. In German.

[84] Y. Sheffi. *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*. Prentice Hall, Englewood Cliffs, New Jersey, 1985.

[85] Homepage of the TRANSIMS project. http://www-transims.tsasa.lanl.gov/.

[86] U.S. Bureau of Public Roads, editor. *Traffic Assignment Manual*. U.S. Department of Commerce, Washington, D.C., 1964. See [84], page 358.

[87] M. van Aerde, B. Hellinga, M. Baker, and H. Rakha. INTEGRATION: Overview of Simulation Features. Presented at the 1996 Transportation Research Board. To appear in *Transportation Research Record*. Available at http://sunburn.uwaterloo.ca/~bhelling/page8.html.

[88] A. Vildósola Engelmayer. Berechnung des Benutzeroptimums in Verkehrsnetzen mit Hilfe des Frank-Wolfe-Algorithmus. Diploma thesis, University of Cologne, Oct. 1998.

[89] J. G. Wardrop. Some Theoretical Aspects of Road Traffic Research. In *Proceedings of the Institute of Civil Engineers*, volume 1, pages 325–378, 1952.

[90] R. Wiedemann. Simulation des Straßenverkehrsflusses. Schriftenreihe des Instituts für Verkehrswesen Heft 8, Universität Karlsruhe, 1974.

[91] S. Yagar. Traffic Assignment Models. In N. H. Gartner, C. J. Messer, and A. J. Rathi, editors, *Traffic Flow Theory*, chapter 11. Federal Highway Administration, 1996. Available at http://www.tfhrc.gov/its/tft/tft.htm.

Deutsche Zusammenfassung

Im Rahmen des Forschungsverbundes „Verkehrssimulation und Umweltwirkungen“ (FVU) [2, 30] hatte die Arbeitsgruppe „Verkehr“ des ZPR/ZAIK die Aufgabe, den Verkehr in großen Straßennetzen, wie etwa dem der Stadt Wuppertal mit knapp 17000 einzelnen Straßenabschnitten, zu simulieren. Die Arbeitsgruppe brachte dabei verschiedene, sehr leistungsfähige Verkehrssimulationsmodelle [65, 78, 51, 8] ein, mit denen es möglich ist, etwa den Verkehrs auf dem gesamten deutschen Autobahnnetz mit ca. 10^6 Fahrzeugen in Echtzeit zu simulieren.

Solche Modelle benötigen sehr detaillierte Eingangsdaten zur Verkehrsnachfrage, nämlich eine Liste aller Fahrten mit Startzeit und Route. Dagegen standen als Ausgangsdaten nur zeitlich und räumlich aggregierte Verkehrsnachfragedaten in Form einer sogenannten Start-Ziel-Matrix *ohne* Spezifikation der Routen zur Verfügung.

Ziel der vorliegenden Arbeit ist es, ein Verfahren zu entwickeln, das diese Lücke schließt, d. h. das aus räumlich und zeitlich aggregierten Verkehrsnachfragedaten individuelle Routen berechnet.

Wieso ist dies problematisch? Die grundlegende Idee hinter allen Routenwahlmodellen ist sehr einfach und wurde schon 1952 von Wardrop [89] formuliert:

> Alle benutzten Routen zu einem gegebenen Paar von Start- und Zielknoten haben die gleichen Kosten, und die Kosten aller unbenutzten Routen sind mindestens genauso groß.

Mit anderen Worten: Jeder Fahrer versucht, seine Reisezeit individuell zu optimieren. Einen Zustand, in dem dies für jeden „Benutzer“ des Netzes erfüllt ist, nennt man *Benutzergleichgewicht* (englisch: user equilibrium).

Die statische Version dieses Problems, *statische Routenumlegung* genannt, ist mathematisch und algorithmisch gut verstanden (s. Kapitel 2).

Ziel des FVU war jedoch eine *zeitabhängige* Beschreibung des Verkehrsgeschehens. Da jede Simulation nur so gut sein kann wie ihre Ausgangsdaten, ist eine zeitlich hochaufgelöste Simulation auf der Basis von Daten, die unter Vernachlässigung der Zeitabhängigkeit gewonnen wurden, wenig sinnvoll. Die bekannten Verfahren zur *dynamischen Routenumlegung*, d. h. zur Bestimmung des Benutzergleichgewichts unter Berücksichtigung der Zeitabhängigkeit der Verkehrsnachfrage (s. Kapitel 2.3), sind jedoch konzeptionell und algorithmisch deutlich komplexer als die statischen Verfahren.

Zwar lassen sich diese Modelle ähnlich wie im statischen Fall mit dem Frank-Wolfe-Algorithmus lösen, aber die auftretenden linearen Teilprobleme sind deutlich komplexer als im statischen Fall (s. Kapitel 2.3.6). Die Laufzeit für eine Iteration des Frank-Wolfe-Algorithmus für das statische Problem beträgt im Fall des Straßennetzes der Stadt Wuppertal, die als Beispielstadt für eine Demonstration der Modellkette des FVU ausgewählt wurde, ca. 1,3 Stunden auf einem mit 336 MHz getakteten UltraSPARC-II Prozessor. Legt man die Anforderungen der anderen am FVU beteiligten Gruppen an die Zeitauflösung unserer Simulationen zugrunde, so ergäbe sich im dynamischen Fall für eine *einzige* Iteration des Frank-Wolfe-Algorithmus eine Laufzeit von ca. 2048 Tagen. Selbst wenn man berücksichtigt, daß die Teilprobleme unabhängig voneinander sind und daher parallel gelöst werden können, ist dieser Aufwand für eine praktische Anwendung nicht vertretbar.

Zur Lösung der gestellten Aufgabe konnte daher nicht auf die bekannten dynamischen Routenumlegungsmodelle zurückgegriffen werden. Die in dieser Arbeit verfolgte Strategie war es deshalb, das Benutzergleichgewicht direkt im Simulationsmodell zu bestimmen, d. h. für eine gegebene Menge von „Fahrern“ mit festem Startzeitpunkt Routen zu bestimmen, die das Wardrop'sche Kriterium erfüllen.

Die einfachste Idee wäre, jedem Fahrer zunächst den geometrisch kürzesten Weg zuzuweisen, mit Hilfe dieser Routen einen Simulationslauf durchzuführen, anschließend aufgrund der in der Simulation bestimmten Reisezeiten für einen gewissen Teil der Fahrer neue Routen zu bestimmen und diesen Prozeß zu iterieren, bis die Reisezeiten konvergieren und kein Fahrer seine Reisezeit durch Wahl einer anderen Route verbessern kann. Allerdings kann man sehr leicht Beispiele konstruieren kann, in denen dieser Prozeß nicht konvergiert und in denen das Gleichgewicht instabil ist.

In Kapitel 4 wird ein Ansatz entwickelt, der darauf beruht, daß ein Fahrer mehr als eine Route kennen kann. Ein Fahrer d wird in diesem Modell durch folgende Größen beschrieben:

- Den Startknoten O_d und den Zielknoten D_d,
- die Startzeit t_d,
- eine Menge $\mathcal{P}_d$ von Routen von O_d nach D_d,
- eine Wahrscheinlichkeitsverteilung $p_d : \mathcal{P}_d \to \mathbb{R}_+$ mit $\sum_{r \in \mathcal{P}_d} p_d(r) = 1$, die die Routenwahl beschreibt,
- und „gelernte“ Reisezeiten $\tau_d : \mathcal{P}_d \to \mathbb{R}_+$.

In jeder Simulation wählt der Fahrer gemäß p_d zufällig eine Route aus $\mathcal{P}_d$ aus. Nach der Simulation werden die Reisezeiten τ_d geändert und die Wahrscheinlichkeitsverteilung p_d wird so verschoben, daß günstigere Routen häufiger gewählt werden. Der Vorteil dieses Ansatzes ist, daß zumindest in einfachen Fällen und für das unten beschriebene Warteschlangenmodell die Stabilität des Verfahrens gezeigt werden kann.

Problematisch bei solch einem Iterationsprozeß ist die Rechenzeit des Simulationsmodells. Selbst die am ZPR/ZAIK entwickelten, sehr leistungsfähigen Fahrzeugfolgemodelle sind nicht so schnell, daß eine solche Iteration mit ihnen in angemessener Zeit durchgeführt werden könnte.

Eine Simulation des Verkehrs im Straßennetz von Wuppertal über eine Zeitraum von 24 Stunden würde mit dem Nagel-Schreckenberg-Modell ca. elf Stunden Rechenzeit auf einem 366 MHz getakteten UltraSPARC-II Prozessor benötigen. Mit Hilfe eines vom Autor entwickelten Warteschlangenmodells konnte diese Zeit auf etwa eine Stunde reduziert werden[1]. Damit war es möglich, die Berechnung des dynamischen Benutzergleichgewichts für das Straßennetz von Wuppertal in 300 CPU-Stunden durchzuführen. Damit wurde die Simulation des Verkehrs in Wuppertal mit realistischen, dem Wardrop'schen Kriterium entsprechenden Routen möglich. Dies war das Hauptziel der vorliegenden Arbeit.

Das oben erwähnte Warteschlangenmodell FASTLANE wird in Kapitel 3 vorgestellt. In diesem Modell wird die Fahrzeugfolgedynamik auf einem Straßenabschnitt vernachlässigt und statt dessen jeder Straßenabschnitt als Warteschlange mit vorgegebener Länge, Kapazität[2] und vorgegebenem Fassungsvermögen beschrieben. Die Reisezeit besteht aus der Summe der Zeit, die benötigt wird, um die Länge der Kante zu durchfahren, und der Zeit, die in der Warteschlange verbracht wird. Obwohl dieses Modell den Verkehrsfluß sehr stark vereinfacht, werden die Reisezeiten, die in den am ZPR/ZAIK verwendeten Fahrzeugfolgemodellen gemessen wurden, sehr gut reproduziert[3].

Neben der Berechnung des dynamischen Benutzergleichgewichts für Wuppertal sind in Kapitel 5 zwei weitere Anwendungen beschrieben. Besonders interessant ist sicher die Diskussion einer dynamischen Variante des Braess'schen Paradoxons [9], die fundamentale Unterschiede zwischen der statischen und der dynamischen Umlegung zeigt. Im statischen Fall „verschwindet“ das Paradoxon oberhalb eines gewissen Flusses (s. Abbildung 2.7 auf Seite 18). Eine quasistatische Analyse, also die Lösung einzelner, statischer Umlegungen für jeden Zeitschritt, ließe erwarten, daß im Falle einer typischen, zeitabhängigen Verkehrsnachfrage mit einer ausgeprägten Spitzenstunde die Verschlechterung des Netzes durch Hinzufügen einer Kante nur vorübergehend ist und der Effekt während der Belastungsspitze wieder veschwindet. Die dynamische Simulation zeigt, daß dies nicht der Fall ist, und das Hinzufügen der „Braess-Kante“ die Reisezeit solange erhöht, bis der in der Spitzenzeit entstandene Stau sich wieder aufgelöst hat. Dies ist ein Beispiel dafür, daß die statische Untersuchung der Leistungsfähigkeit eines Verkehrsnetzes unter Umständen zu einer falschen Beurteilung führen kann.

Für eine routinemäßige Anwendung in der Praxis sind die benötigten 300 Stunden Rechenzeit für eine Anwendung auf ein Verkehrsnetz wie das der Stadt Wuppertal natürlich nicht akzeptabel. Der zeitaufwendigere Teil, nämlich die Neuberechnung der Routenwahlwahrscheinlichkeiten, ließe sich jedoch in trivialer Weise parallelisieren, da die Fahrer unabhängig voneinander betrachtet werden. Die Verfügbarkeit entsprechend leistungsfähiger Rechner vorausgesetzt — eine Mehrprozessor-Workstation würde genügen — ließe sich das Verfahren auch in der Praxis der Verkehrsplanung anwenden.

---

[1] Bei der Simulation von Straßennetzes mit einer größeren durchschnittlichen Kantenlänge wird der Geschwindigkeitsvorteil sogar noch größer, beim Autobahnnetz von NRW beträgt er etwa zwei Größenordnungen.

[2] in der einschlägigen Literatur oft auch als „Servicerate“ bezeichnet

[3] Wobei die Übereinstimmung mit dem Nagel-Schreckeberg-Modell besser ist als mit dem Modell von Krauß. Dies liegt wohl in erster Linie an der Existenz metastabiler Zustände im Modell von Krauß.

Danksagung

Bei der Erstellung dieser Arbeit erhielt ich von verschiedenen Seiten Unterstützung. Mein besonderer Dank gilt

Prof. Dr. Rainer Schrader und Prof. Dr. Achim Bachim für die Betreuung dieser Arbeit und dafür, daß sie es mir ermöglicht haben, mich in ein neues Fachgebiet einzuarbeiten,

Peter Wagner und Rolf Böning, die mich stets unterstützt haben, mir viele Impulse gaben und mir über Durststrecken hinweggeholfen haben,

Stefan Krauß, Markus Rickert und allen anderen Kollegen der „Verkehrsgruppen“ des ZPR/ZAIK und des DLR für die gute Zusammenarbeit und fruchtbare Diskussionen,

Stephan Hasselberg für seine Ratschläge bei allem, was mit Graphen zu tun hat, und für die Bereitstellung seiner umfangreichen Privatbibliothek,

Andrea Vildósola Engelmayer für ihre Hilfe beim Studium der statischen Umlegungsverfahren,

allen Mitarbeitern des ZPR/ZAIK für die tolle Arbeitsatmosphäre, die ich sicher noch vermissen werde,

allen, die bei der Entwicklung von PlanSim-T mitgeholfen haben, ganz besonders Peter Oertel und Felix Pütsch,

Kai Nagel, der mein Interesse für diese Arbeit überhaupt erst geweckt hat,

Guido Rindsfüser, dem Landschaftsverband Rheinland und allen anderen Partnern im Forschungsverbund „Verkehrssimulation und Umweltwirkungen“, die mir Daten zur Verfügung gestellt haben,

dem Ministerium für Wissenschaft und Forschung des Landes Nordrhein-Westfalen, das diese Arbeit gefördert hat,

den Entwicklern der LEDA-Bibliothek am Max-Planck-Institut für Informatik,

Monika Reimpell für viel Verständnis und zahlreiche Korrekturen.

Further I wish to thank all authors of free software which I used while working on this thesis, among them Colin Kelley, Donald E. Knuth, Leslie Lamport, Brian V. Smith, Richard Stallman, Supoj Sutanthavibul, Larry Wall, and Thomas Williams.

Erklärung

Ich versichere, daß ich die von mir vorgelegte Dissertation selbständig angefertigt, die benutzten Quellen und Hilfsmittel vollständig angegeben und die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken im Wortlaut oder dem Sinn nach entnommen sind, in jedem Einzelfall als Entlehnung kenntlich gemacht habe; daß diese Dissertation noch keiner anderen Fakultät oder Universität zur Prüfung vorgelegen hat; daß sie – abgesehen von unten angegebenen Teilpublikationen – noch nicht veröffentlicht worden ist sowie, daß ich eine solche Veröffentlichung vor Abschluß des Promotionsverfahrens nicht vornehmen werde. Die Bestimmungen dieser Promotionsordnung sind mir bekannt. Die von mir vorgelegte Dissertation ist von Prof. Dr. R. Schrader betreut worden.

Teilpublikationen

C. Gawron. An Iterative Algorithm to Determine the Dynamic User Equilibrium in a Traffic Simulation Model. *Int. Jrn. Mod. Phys. C* **9**(3):393–407, 1998.

C. Gawron, S. Krauß, and P. Wagner. Dynamic User Equilibria in Traffic Simulation Models. In M. Schreckenberg and D. E. Wolf, editors, *Traffic and Granular Flow '97*, pages 469–473, Singapore, 1998. Springer Verlag.

Lebenslauf

Persönliche Daten

Name	Christian Gawron
geboren am	01.03.1968 in Köln
Eltern	Gisela Gawron, geb. Freiberg, Heribert Gawron
Staatsangehörigkeit	deutsch
Familienstand	ledig

Schulbildung

1974 – 1978	Grundschule in Bad Honnef und Köln
1978 – 1987	Gymnasium in Köln

Zivildienst

1987 – 1989	beim Arbeiter–Samariter–Bund Deutschland e. V.

Studium

1989 – 1991	Grundstudium der Physik an der Universität zu Köln
1991	Vordiplom
1991 – 1995	Hauptstudium an der Rheinischen Friedrich–Wilhelms–Universität Bonn
1992 – 1995	Studentische Hilfskraft am Rechenzentrum der Universität zu Köln
1993 – 1995	Studentische Hilfskraft am Physikalischen Institut und am Institut für Theoretische Kernphysik der Universität Bonn
1995	Diplom in Physik
1995 – 1998	Wissenschaftlicher Mitarbeiter am Zentrum für Paralleles Rechnen der Universität zu Köln