Traffic Networks and Flows over Time*

Ekkehard Köhler[1], Rolf H. Möhring[2], and Martin Skutella[2]

[1] Brandenburgische Technische Universität Cottbus,
Fakultät I — Mathematik, Naturwissenschaften und Informatik,
Mathematisches Institut, PSF 10 13 44, 03013 Cottbus, Germany
ekoehler@math.tu-cottbus.de
http://www.math.tu-cottbus.de/INSTITUT/lsgdi
[2] Technische Universität Berlin,
Fakultät II — Mathematik und Naturwissenschaften,
Institut für Mathematik, Sekr. MA 5–1,
Straße des 17. Juni 136, D–10623 Berlin, Germany
{moehring,skutella}@math.tu-berlin.de
http://www.math.tu-berlin.de/~{moehring,skutella}

1 Introduction

In view of the steadily growing car traffic and the limited capacity of our street networks, we are facing a situation where methods for better traffic management are becoming more and more important. Studies [92] show that an individual "blind" choice of routes leads to travel times that are between 6% and 19% longer than necessary. On the other hand, telematics and sensory devices are providing or will shortly provide detailed information about the actual traffic flows, thus making available the necessary data to employ better means of traffic management.

Traffic management and route guidance are optimization problems by nature. We want to utilize the available street network in such a way that the total network "load" is minimized or the "throughput" is maximized. This article deals with the mathematical aspects of these optimization problems from the viewpoint of network flow theory. This is, in our opinion, currently the most promising method to get structural insights into the behavior of traffic flows in large, real-life networks. It also provides a bridge between traffic simulation [12,77] or models based on fluid dynamics [80] and variational inequalities [23], which are two other common approaches to study traffic problems. While simulation is a powerful tool to evaluate traffic scenarios, it misses the optimization potential. On the other hand, fluid models and other models based on differential equations capture very well the dynamical behavior of traffic as a continuous quantity, but can currently not handle large networks.

Traffic flows have two important features that make them difficult to study mathematically. One is 'congestion', and the other is 'time'. 'Congestion' captures the fact that travel times increase with the amount of flow on the streets, while 'time' refers to the movement of cars along a path as a "flow over time".

* A preliminary version of this paper has appeared as [59]. Some parts have been taken from the following papers: [9,29,41,51,58,61].

J. Lerner, D. Wagner, and K.A. Zweig (Eds.): Algorithmics, LNCS 5515, pp. 166–196, 2009.

We will therefore study several mathematical models of traffic flows that become increasingly more difficult as they capture more and more of these two features.

Section 2 deals with flows without congestion. The first part gives an introduction to the classical static network flow theory and discusses basic results in this area. The presented insights are also at the core of more complex models and algorithms discussed in later sections. The main part of the section then introduces the temporal dimension and studies flows over time without congestion. Although of indisputable practical relevance, flows over time have never attracted as much attention among researchers as their classical static counterparts. We discuss some more recent results in this area which raise hope and can be seen as first steps towards the development of methods that can cope with flows over time as they occur in large real-life traffic networks. We conclude this section with a short discussion of flows over time as they occur in evacuation planning.

In Section 3 we add congestion to the static flow model. This is already a realistic traffic model for rush-hour traffic where the effect of flow changing over time is secondary compared to the immense impact of delays due to congestion. We consider algorithms for centralized route guidance and discuss fairness aspects for the individual user resulting from optimal route guidance policies. Finally, we review fast algorithms for shortest path problems which are at the core of more complex algorithms for traffic networks and whose running time is therefore of utmost importance.

Section 4 then combines flows over time with congestion. Compared to the models discussed in earlier sections, this setting is considerably closer to real-world traffic flows. Thus, it hardly surprises that it also leads to several new mathematical difficulties. One main reason is that the technique of time expansion which still worked for the problems of Section 2 is no longer fully available. Nevertheless, it is possible to derive approximation algorithms for reasonable models of congestion, but the full complexity of this problem is by far not understood yet.

Despite the partial results and insights discussed in this article, the development of suitable mathematical tools and algorithms which can deal with large real-world traffic networks remains an open problem. The inherent complexity of many traffic flow problems constitutes a major challenge for future research.

2 Static Flows and Flows over Time

An exhaustive mathematical treatment of network flow theory started around the middle of the last century with the ground-breaking work of Ford and Fulkerson [35]. Historically, the study of network flows mainly originates from problems related to transportation of materials or goods; see, e. g., [46,54,63]. For a detailed survey of the history of network flow problems we refer to the recent work of Schrijver [87].

2.1 Static Flow Problems

Usually, flows are defined on networks (directed graphs) $G = (V, E)$ with *capacities* $u_e \geq 0$ and, in some settings, also *costs* c_e on the arcs $e \in E$. The set of

nodes V is partitioned into *source* nodes, *intermediate* nodes, and *sink* nodes. On an intuitive level, flow emerges from the source nodes, travels through the arcs of the network via intermediate nodes to the sinks, where it is finally absorbed. More precisely, a *static flow* x assigns a value $0 \leq x_e \leq u_e$ to every arc $e \in E$ of the network such that for every node $v \in V$

$$\sum_{e \in \delta^-(v)} x_e \; - \sum_{e \in \delta^+(v)} x_e \begin{cases} \leq 0 & \text{if } v \text{ is a source,} \\ = 0 & \text{if } v \text{ is an intermediate node,} \\ \geq 0 & \text{if } v \text{ is a sink.} \end{cases} \tag{1}$$

Here, $\delta^+(v)$ and $\delta^-(v)$ denote the set of arcs e leaving node v and entering node v, respectively. Thus, the left hand side of (1) is the net amount of flow entering node v. For intermediate nodes this quantity must obviously be zero; this requirement is usually refered to as *flow conservation constraint*. A flow x is said to *satisfy the demands and supplies* d_v, $v \in V$, if the left hand side of (1) is equal to d_v for every node $v \in V$. In this setting, nodes v with negative demand d_v (i. e., positive supply $-d_v$) are sources and nodes with positive demand are sinks. A necessary condition for such a flow to exist is $\sum_{v \in V} d_v = 0$. Observe that the sum of the left hand side of (1) over all $v \in V$ is always equal to 0.

A flow problem in a network $G = (V, E)$ with several sources $S \subset V$ and several sinks $T \subset V$ can easily be reduced to a problem with a single source and a single sink: Introduce a new *super-source* s and a new *super-sink* t. Then add an arc (s, v) of capacity $u_{(s,v)} := -d_v$ for every source $v \in S$ and an arc (w, t) of capacity $u_{(w,t)} := d_w$ for every sink $w \in T$. In the resulting network with node set $V \cup \{s, t\}$ all nodes in V are intermediate nodes and we set $d_s := \sum_{v \in S} d_v$ and $d_t := \sum_{w \in T} d_w$.

For the case of one single source s and one single sink t, the flow x is called an *s-t-flow* and the left hand side of (1) for $v = t$ is the *s-t-flow value* which we denote by $|x|$. Due to flow conservation constraints, the absolute value of the left hand side of (1) for $v = s$ also equals the flow value. In other words, all flow leaving the source must finally arrive at the sink. Ford and Fulkerson [33,35] and independently Elias, Feinstein, and Shannon [24] show in their so-called 'Max-Flow-Min-Cut Theorem' that the maximum s-t-flow value is equal to the minimum capacity of an s-t-cut. An *s-t-cut* $\delta^+(S)$ is given by a subset of vertices $S \subseteq V \setminus \{t\}$, $s \in S$, and defined as the set of arcs going from S to its complement $V \setminus S$, i. e., $\delta^+(S) := (\bigcup_{v \in S} \delta^+(v)) \setminus (\bigcup_{v \in S} \delta^-(v))$. The *capacity* of an s-t-cut is the sum of the capacities of all arcs in the cut. We give an example of a maximum s-t-flow and a matching minimum cut in Figure 1. Ford and Fulkerson [35] also observe that the Max-Flow-Min-Cut Theorem can be interpreted as a special case of linear programming duality.

Today, a variety of efficient (i. e., polynomial time) algorithms are known for computing an s-t-flow of maximum value and a corresponding minimum capacity cut. We refer to the standard textbook by Ahuja, Magnanti, and Orlin [1] for a detailed account of results and algorithms in this area. However, we mention one further structural result for s-t-flows by Ford and Fulkerson [35]. Any given

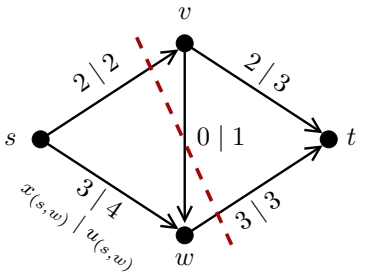

Fig. 1. A maximum s-t-flow together with a matching minimum cut in a network. The first number at each arc denotes its flow value and the second number its capacity. The depicted s-t-flow has value 5 which equals the capacity of the depicted s-t-cut $\delta^+(\{s,w\}) = \{(s,v),\,(w,t)\}$.

s-t-flow $x = (x_e)_{e\in E}$ can be decomposed into the sum of flows of value x_P on certain s-t-paths $P \in \mathcal{P}$ and flows of value x_C on certain cycles $C \in \mathcal{C}$, that is,

$$x_e \;=\; \sum_{\substack{P\in\mathcal{P}\\ e\in P}} x_P \;+\; \sum_{\substack{C\in\mathcal{C}\\ e\in C}} x_C \qquad \text{for all } e \in E.$$

Moreover, the number of paths and cycles in this decomposition can be bounded by the number of arcs, i. e., $|\mathcal{P}| + |\mathcal{C}| \le |E|$.

In the setting with costs on the arcs, the cost of a static flow x is defined as $c(x) := \sum_{e\in E} c_e\, x_e$. For given demands and supplies d_v, $v \in V$, the *minimum cost flow problem* asks for a flow x with minimum cost $c(x)$ satisfying these demands and supplies. As for the maximum flow problem discussed above, various efficient algorithms are known for this problem and a variety of structural characterizations of optimal solutions has been derived. Again, we refer to [1] for details.

A static flow problem which turns out to be considerably harder than the maximum flow and the minimum cost flow problem is the *multicommodity flow problem*. Every *commodity* $i \in K$ is given by a source-sink pair $s_i, t_i \in V$ and a prescribed flow value d_i. The task is to find an s_i-t_i-flow x_i of value d_i for every commodity $i \in K$ such that the sum of these flows obeys the arc capacities, i. e., $\sum_{i\in K}(x_i)_e \le u_e$ for all $e \in E$. So far, no combinatorial algorithm has been developed which solves this problem efficiently. On the other hand, there exist polynomial time algorithms which, however, rely on a linear programming formulation of the problem and thus on general linear programming techniques (such as the ellipsoid method or interior point algorithms); see e. g. [1].

2.2 Flows over Time

Flow variation over time is an important feature in network flow problems arising in various applications such as road or air traffic control, production systems, communication networks (e. g., the Internet), and financial flows. This characteristic is obviously not captured by the static flow models discussed in the

previous section. Moreover, apart from the effect that flow values on arcs may change over time, there is a second temporal dimension in these applications: Usually, flow does not travel instantaneously through a network but requires a certain amount of time to travel through each arc. Thus, not only the amount of flow to be transmitted but also the time needed for the transmission plays an essential role.

Definition of flows over time. Ford and Fulkerson [34,35] introduce the notion of 'dynamic flows' or 'flows over time' which comprises both temporal features. They consider networks G with capacities u_e and transit times $\tau_e \in \mathbb{Z}_+$ on the arcs $e \in E$. A *flow over time* f on G with *time horizon* T is given by a collection of functions $f_e : [0, T) \to [0, u_e]$ where $f_e(\theta)$ determines the rate of flow (per time unit) entering arc e at time θ. Here, capacity u_e is interpreted as an upper bound on the rate of flow entering arc e, i. e., a capacity per unit time. Transit times are fixed throughout, so that flow on arc e progresses at a uniform rate. In particular, the flow $f_e(\theta)$ entering the tail of arc e at time θ arrives at the head of e at time $\theta + \tau_e$. In order to get an intuitive understanding of flows over time, one can associate the arcs of the network with pipes in a pipeline system for transporting some kind of fluid. The length of each pipeline determines the transit time of the corresponding arc while the width determines its capacity.

Continuous vs. discrete time. In fact, Ford and Fulkerson introduce a slightly different model of flows over time where time is discretized into intervals of unit size $[0, 1), [1, 2), \ldots, [T - 1, T)$, and the function f_e maps every such interval to the amount of flow which is sent within this interval into arc e. We call such a flow over time *discrete*. Discrete flows over time can for instance be illustrated by associating each arc with a conveyor belt which can grasp one packet per time unit. The maximal admissible packet size determines the capacity of the corresponding arc and the speed and length of the conveyor belt define its transit time.

Fleischer and Tardos [31] point out a strong connection between the discrete time model and the continuous time model described above. They show that most results and algorithms which have been developed for the discrete time model can be carried over to the continuous time model and vice versa. Loosely speaking, the two models can be considered to be equivalent.

Flow conservation constraints. When considering flows over time, the flow conservation constraints (1) must be integrated over time to prohibit deficit at any node v which is not a source. Hence, for all $\xi \in [0, T)$

$$\sum_{e \in \delta^-(v)} \int_{\tau_e}^{\xi} f_e(\theta - \tau_e)\, d\theta \;-\; \sum_{e \in \delta^+(v)} \int_{0}^{\xi} f_e(\theta)\, d\theta \;\geq\; 0 \;. \tag{2}$$

The first term on the left hand side of (2) is the total inflow into node v until time ξ and the second term is the total outflow out of v until time ξ. Notice that since the left hand side of (2) might be positive, temporary storage of flow at intermediate nodes is allowed. This corresponds to holding inventory at a node

before sending it onward. However, we require equality in (2) for $\xi = T$ and any intermediate node v in order to enforce flow conservation in the end. If storage of flow at intermediate nodes is undesired, it can be prohibited by requiring equality in (2) for all $\xi \in [0, T)$.

As in the case of static flows, a flow over time f satisfies the demands and supplies d_v, $v \in V$, if the left hand side of (2) for $\xi = T$ is equal to d_v for every node $v \in V$. If there is only one single sink t, the left hand side of (2) for $\xi = T$ and $v = t$ is the flow value of f.

Time-expanded networks. Ford and Fulkerson [34,35] observe that flow-over-time problems in a given network with transit times on the arcs can be transformed into equivalent static flow problems in the corresponding *time-expanded network*. The time-expanded network contains one copy of the node set of the underlying 'static' network for each discrete time step θ (building a *time layer*). Moreover, for each arc e with transit time τ_e in the given network, there is a copy between each pair of time layers of distance τ_e in the time-expanded network. An illustration of this construction is given in Figure 2. Thus, a discrete flow over time in the given network can be interpreted as a static flow in the corresponding time-expanded network. Since this interrelation works in both directions, the concept of time-expanded networks allows to solve a variety of time-dependent flow problems by applying algorithmic techniques developed for static network flows; see e. g. [31]. Notice, however, that one has to pay for this simplification of the considered flow problem in terms of an enormous increase in the size of the network. In particular, the size of the time-expanded network is only pseudo-polynomial in the input size and thus does not directly lead to efficient algorithms for computing flows over time. Nevertheless, time-expanded networks are often used to solve problems in practice.

Maximum s-t-flows over time. Ford and Fulkerson [34,35] give an efficient algorithm for the problem of sending the maximum possible amount of flow from

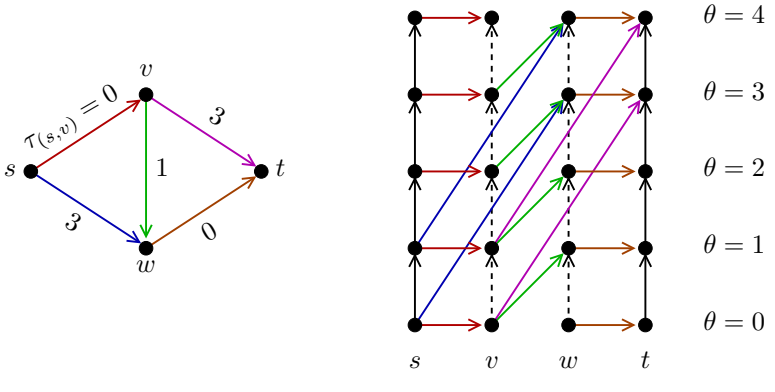

Fig. 2. A network with transit times on the arcs (left hand side) and the corresponding time-expanded network (right hand side). The vertical black arcs model the possibility to store flow at nodes.

one source to one sink within a given time horizon T. They show that this problem can be solved by one minimum-cost flow computation in the given network, where transit times of arcs are interpreted as cost coefficients. Their algorithm is based on the concept of *temporally repeated flows*. Let x be a feasible static s-t-flow in G with path decomposition $(x_P)_{P\in\mathcal{P}}$ where $\mathcal{P}$ is a set of s-t-paths. If the transit time $\tau_P := \sum_{e\in P} \tau_e$ of every path $P \in \mathcal{P}$ is bounded from above by T, the static s-t-flow x can be turned into a *temporally repeated* s-t-flow f as follows. Starting at time zero, f sends flow at constant rate x_P into path $P \in \mathcal{P}$ until time $T - \tau_P$, thus ensuring that the last unit of flow arrives at the sink before time T. An illustration of temporally repeated flows is given in Figure 3.

Feasibility of f with respect to capacity constraints immediately follows from the feasibility of the underlying static flow x. Notice that the flow value $|f|$ of f, i. e., the amount of flow sent from s to t is given by

$$|f| \;=\; \sum_{P\in\mathcal{P}} (T-\tau_P)\, x_P \;=\; T\,|x| - \sum_{e\in E} \tau_e\, x_e \;. \tag{3}$$

The algorithm of Ford and Fulkerson computes a static s-t-flow x maximizing the right hand side of (3), then determines a path decomposition of x, and finally returns the corresponding temporally repeated flow f. Ford and Fulkerson show that this temporally repeated flow is in fact a maximum s-t-flow over time by presenting a matching minimum cut in the corresponding time-expanded network. This cut in the time-expanded network can be interpreted as a *cut over time* in the given network, that is, a cut wandering through the network over time from the source to the sink.

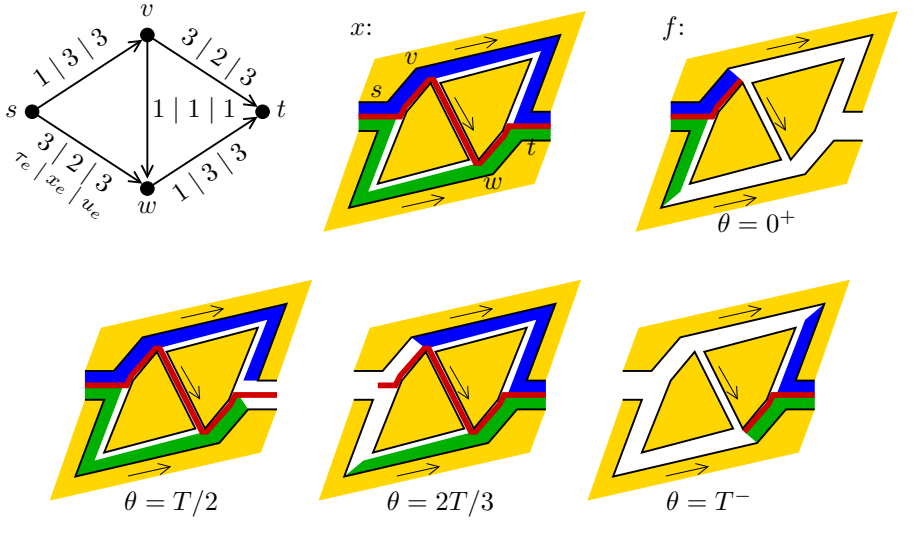

Fig. 3. A static s-t-flow x and 4 snapshots of the corresponding temporally repeated s-t-flow f

A problem closely related to the one considered by Ford and Fulkerson is the *quickest s-t-flow problem.* Here, instead of fixing the time horizon T and asking for a flow over time of maximal value, the value of the flow (demand) is fixed and T is to be minimized. This problem can be solved in polynomial time by incorporating the algorithm of Ford and Fulkerson into a binary search framework. Burkard, Dlaska, and Klinz [15] observed that the quickest flow problem can even be solved in strongly polynomial time.

More complex flow over time problems. As mentioned above, a static flow problem with multiple sources and sinks can easily be reduced to one with a single source and a single sink. Notice, however, that this approach no longer works for flows over time. It is in general not possible to control the amount of flow which is sent on the arcs connecting the super-source and super-sink to the original nodes of the network. Hoppe and Tardos [49] study the *quickest transshipment problem* which asks for a flow over time satisfying given demands and supplies within minimum time. Surprisingly, this problem turns out to be much harder than the special case with a single source and sink. Hoppe and Tardos give a polynomial time algorithm for the problem, but this algorithm relies on submodular function minimization and is thus much less efficient than for example the algorithm of Ford and Fulkerson for maximum s-t-flows over time.

Even more surprising, Klinz and Woeginger [55] show that the problem of computing a minimum cost s-t-flow over time with prespecified value and time horizon is NP-hard. This means that under the widely believed conjecture P≠NP, there does not exist an efficient algorithm for solving this problem. The cost of a flow over time f is defined as

$$c(f) \;:=\; \sum_{e \in E} c_e \int_0^T f_e(\theta)\; d\theta \;\; .$$

In fact, Klinz and Woeginger prove that the problem is already NP-hard on very special series-parallel networks. The same hardness result holds for the *quickest multicommodity flow problem*; see Hall, Hippler, and Skutella [41]. On the other hand, Hall et al. develop polynomial time algorithms for solving the problem on tree networks. An overview of the complexity landscape of flows over time is given in Table 1.

Approximation results. Fleischer and Skutella [27] generalize the temporally repeated flow approach of Ford and Fulkerson in order to get an approximation algorithm for the quickest multicommodity flow problem with bounded cost. A key insight in their result is that by taking the temporal average of an optimal multicommodity flow over time with time horizon T, one gets a static flow in the given network which is T-length-bounded, i. e., it can be decomposed into flows on paths whose transit time is bounded by T. While it is initially not clear how to compute a good multicommodity flow over time efficiently, a length-bounded static flow mimicking the static 'average flow' can be obtained in polynomial time (if the length bound is relaxed by a factor of $1 + \epsilon$). The computed static

Table 1. The complexity landscape of flows over time in comparison to the corresponding static flow problems. The third column 'transshipment' refers to single-commodity flows with several source and sink nodes. The quoted approximation results hold for the corresponding quickest flow problems.

	s-t-flow	transshipment	min-cost flow	multicommodity flow
(static) flow	poly	poly ($\simeq$ s-t-flow)	poly	poly ($\simeq$ LP)
flow over time with storage	poly [34] ($\simeq$ min-cost flow)	poly [49] ($\simeq$ subm. func.)	pseudo-poly NP-hard [55] FPTAS [28,29]	pseudo-poly NP-hard [41] FPTAS [28,29]
flow over time without storage				strongly NP-hard [41] 2-approx. [27,29]

multicommodity flow can then be turned into a temporally repeated flow in a similar way as in the algorithm of Ford and Fulkerson. Moreover, the time horizon of this temporally repeated flow is at most $(2+\epsilon)\,T$ and therefore within a factor of $2+\epsilon$ of the optimum. Martens and Skutella [70] make use of this approach to approximate s-t-flows over time with a bound on the number of flow-carrying paths (*k-splittable flows over time*).

Fleischer and Skutella [27,28] also show that better performance ratios can be obtained by using a variant of time-expanded networks. In *condensed* time-expanded networks, a rougher discretization of time is used in order to get networks of polynomial size. However, in order to discretize time into steps of larger length $L > 1$, transit times of arcs have to be rounded up to multiples of L. This causes severe problems in the analysis of the quality of solutions computed in condensed time-expanded networks. Nevertheless, one can show that the solution space described by a condensed time-expanded network is only slightly degraded. This yields a general framework to obtain fully polynomial time approximation schemes for various flow-over-time problems like, for example, the quickest multicommodity flow problem with costs. Moreover, this approach also yields more efficient algorithms for solving flow over time problems in practical applications.

For the case that arc costs are proportional to transit times, Fleischer and Skutella [28] describe a very simple fully polyomial-time approximation scheme based on capacity scaling for the minimum cost s-t-flow over time problem.

For a broad overview of flows over time we refer to the survey papers by Aronson [3], Powell, Jaillet, and Odoni [79], Kotnyek [64], and Lovetskii and Melamed [68]. We also refer to the PhD thesis of Hoppe [47] for an easily accessible and detailed treatment of the topic based on the *discrete time model*. Skutella [90] presents a treatment of flows over time for the purpose of teaching in an advanced course.

2.3 Earliest Arrival Flows

In typical evacuation situations, the most important task is to get people out of an endangered building or area as fast as possible. Since it is usually not known how long a building can withstand a fire before it collapses or how long a

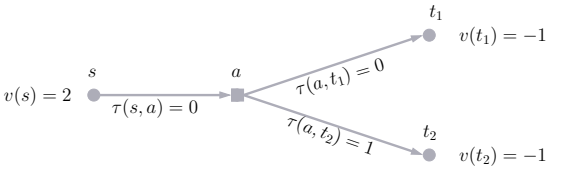

Fig. 4. A network with one source and two sinks with unit demands for which an earliest arrival flow does not exist. All arcs have unit capacity and the transit times are given in the drawing. Notice that one unit of flow can reach sink t_1 by time 1; in this case, the second unit of flow reaches sink t_2 only by time 3. Alternatively we can send one unit of flow into sinks t_1 and t_2 simultaneously by time 2. It is impossible to fulfill the requirement, that both flow units must have reached their sink by time 2 and one of them must have already arrived at time 1.

dam can resist a flood before it breaks, it is advisable to organize an evacuation such that as much as possible is saved no matter when the inferno will actually happen. In the setting of flows over time, the latter requirement is captured by so-called earliest arrival flows.

Shortly after Ford and Fulkerson introduced flows over time, the more elaborate *earliest arrival s-t-flow problem* was studied by Gale [37]. Here the goal is to find a single s-t-flow over time that simultaneously maximizes the amount of flow reaching the sink t up to any time $\theta \geq 0$. A flow over time fulfilling this requirement is said to have the *earliest arrival property* and is called *earliest arrival s-t-flow*. Gale [37] showed that earliest arrival s-t-flows always exist. Minieka [75] and Wilkinson [94] both gave pseudopolynomial-time algorithms for computing earliest arrival s-t-flows based on the Successive Shortest Path Algorithm [53,50,16]. Hoppe and Tardos [48] present a fully polynomial-time approximation scheme for the earliest arrival s-t-flow problem that is based on a clever scaling trick.

In a network with several sources and sinks with given supplies and demands, flows over time having the earliest arrival property do not necessarily exist [30]. Baumann and Skutella [9] give a simple counterexample with one source and two sinks; see Figure 4.

For the case of several sources with given supplies and a single sink, however, earliest arrival transshipments do always exist. This follows, for example, from the existence of lexicographically maximal flows in time-expanded networks; see, e.g., [75]. We refer to this problem as the *earliest arrival transshipment problem*. Hajek and Ogier [40] give the first polynomial time algorithm for the earliest arrival transshipment problem with zero transit times. Fleischer [30] gives an algorithm with improved running time. Fleischer and Skutella [29] use condensed time-expanded networks to approximate the earliest arrival transshipment problem for the case of arbitrary transit times. They give an FPTAS that approximates the time delay as follows: For every time $\theta \geq 0$ the amount of flow that should have reached the sink in an earliest arrival transshipment by time θ, reaches the sink at latest at time $(1 + \varepsilon)\theta$. Tjandra [91] shows how to compute

earliest arrival transshipments in networks with time dependent supplies and capacities in time polynomial in the time horizon and the total supply at sources. The resulting running time is thus only pseudopolynomial in the input size.

Earliest arrival flows and transshipments are motivated by applications related to evacuation. In the context of emergency evacuation from buildings, Berlin [13] and Chalmet et al. [20] study the quickest transshipment problem in networks with multiple sources and a single sink. Jarvis and Ratliff [52] show that three different objectives of this optimization problem can be achieved simultaneously: (1) Minimizing the total time needed to send the supplies of all sources to the sink, (2) fulfilling the earliest arrival property, and (3) minimizing the average time for all flow needed to reach the sink. Hamacher and Tufecki [45] study an evacuation problem and propose solutions which further prevent unnecessary movement within a building.

Baumann and Skutella [9] (see also [7]) present a polynomial time algorithm for computing earliest arrival transshipments in the general multiple-source single-sink setting. All previous algorithms rely on time expansion of the network into exponentially many time layers. Baumann and Skutella first recursively construct the earliest arrival pattern, that is, the piece-wise linear function that describes the time-dependent maximum flow value obeying supplies and demands. The algorithm employs submodular function minimization within the parametric search framework of Megiddo [71,72]. As a by-product, a new proof for the existence of earliest arrival transshipments is obtained that does not rely on time expansion. It can finally be shown that the earliest arrival pattern can be turned into an earliest arrival transshipment based on the quickest transshipment algorithm of Hoppe and Tardos [49].

The running time of the obtained algorithm is polynomial in the input size plus the number of breakpoints of the earliest arrival pattern. Since the earliest arrival pattern is more or less explicitly part of the output of the earliest arrival transshipment problem, the running time of the algorithm is polynomially bounded in the input plus output size.

3 Static Traffic Flows with Congestion

In the previous section we have considered flows over time where transit times on the arcs are fixed. The latter assumption is no longer true in situations where congestion does occur. Congestion means that the transit time τ_e on an arc e is no longer constant, but a monotonically increasing, convex function $\tau_e(x_e)$ of the flow value x_e on that arc e. Congestion is inherent to car traffic, but also occurs in evacuation planning, production systems, and communication networks. In all of these applications the amount of time needed to traverse an arc of the underlying network increases as the arc becomes more congested.

Congestion in static traffic networks has been studied for a long time by traffic engineers, see e. g. [89]. It models the rush hour situation in which flow between different origins and destinations is sent over a longer period of time. The usual objective to be optimized from a global system-oriented point of view is the overall road usage, which can be viewed as the sum of all individual transit times.

3.1 User Equilibrium and System Optimum

From a macroscopic point of view, such a static traffic network with congestion can be modeled by a multicommodity flow problem, in which each commodity $i \in K$ represents two locations in the network, between which d_i "cars" are to be routed per unit of time. The data (s_i, t_i, d_i), $i \in K$, form the so-called *origin-destination matrix*. Feasible routings are given by flows x, and the "cost" $c(x)$ of flow x is the total travel time spent in the network. On arc e, the flow x then induces a transit time $\tau_e(x_e)$, which is observed by all x_e flow units using that arc e. So the total travel time may be written as $c(x) = \sum_{e \in E} x_e \, \tau_e(x_e)$.

It is usually more convenient to think of a feasible routings in terms of *flows on paths* instead of *flows on arcs*. Then, for every commodity i, the d_i amounts of flow sent between s_i and t_i are routed along certain paths P from the set $\mathcal{P}_i$ of possible paths between s_i and t_i. These paths P can be seen as the choice of routes that users take who want to drive from s_i to t_i. On the other hand, any solution x in path formulation can be seen as a route recommendation to the users of the traffic network, and thus as a route guidance strategy. Therefore, it is reasonable to study properties of flows as route recommendations and investigate their quality within this model.

Current route guidance system are very simple in this respect and usually restrict to recommend shortest or quickest paths without considering the effect on the congestion that their recommendation will have. Simulations show that users of such a simple route guidance system will—due to the congestion caused by the recommendation—experience longer transit times when the percentage of users of such a system gets larger.

Without any guidance, without capacity restrictions, but full information about the traffic situation, users will try to choose a fastest route and achieve a Nash equilibrium, a so-called *user equilibrium*. Such an equilibrium is defined by the property that no driver can get a faster path through the network when everybody else stays with his route. One can show that in such an equilibrium state all flow carrying paths P in each $\mathcal{P}_i$ have the same transit time $\tau_P = \sum_{e \in P} x_e \, \tau_e(x_e)$, and, furthermore, if all τ_e are strictly increasing and twice differentiable, then the value of the user equilibrium is unique. This is no longer true in capacitated networks [21].

So in uncapaciated networks, the user equilibrium is characterized by a certain fairness, since all users of the same origin destination pair have the same transit time. Therefore, is has been widely used as a reasonable solution to the static traffic problem with congestion. This is supported by recent results showing that, under reasonable assumption about obtaining traffic information, a simple replication-exploration strategy will indeed converge to the Nash equilibrium and the convergence (the number of rounds in which users choose new routes) is polynomial in the representation length of the transit time functions [26]. So users of a static traffic system can learn the user equilibrium efficiently.

The advantage of the user equilibrium lies in the fact that it can be achieved without traffic control (it only needs selfish users) and that it is fair to all users

with the same origin and destination. However, it does not necessarily optimize the total travel time. Roughgarden and Tardos [83] investigate the difference that can occur between the user equilibrium and the system optimum, which is defined as a flow x minimizing the total travel time $\sum_{e \in E} x_e \, \tau_e(x_e)$. The ratio $c(UE)/c(SO)$ of the total travel time $c(UE)$ in the user equilibrium over that in the system optimum is referred to as *the price of anarchy*. In general, it may become arbitrarily large than in the system optimum, but it is never more than the total travel time incurred by optimally routing twice as much traffic [83]. Improved bounds on the price of anarchy can be given for special classes of transit time functions [83,86].

Another unfavorable property of the user equilibrium is a non-monotonicity property with respect to network expansion. This is illustrated by the *Braess paradox*, where adding a new road to a network with fixed demands actually *increases* the overall road usage obtained in the updated user equilibrium [14,89,39].

So one may ask what can be achieved by centralized traffic management. Of course, the system optimum would provide the best possible solution by definition. But it may have long routes for individual users and thus misses fairness, which makes it unsuited as a route guidance policy (unless routes are enforced upon users by pricing and/or central guidance). To quantify the phenomenon of long routes in the system optimum, Roughgarden [82] introduces a measure of unfairness. The *unfairness* is the maximum ratio between the transit time along a route in the system optimum and that of a route in the user equilibrium (between the same origin-destination pair). In principle, the unfairness may become arbitrarily large, but like the price of anarchy, it can be bounded by a parameter $\gamma(\mathcal{C})$ of the class $\mathcal{C}$ of underlying transit time functions [86]. For instance, $\gamma\{\text{polynomials of degree } p \text{ with nonnegative coefficients}\} = p + 1$.

In computational respect, user equilibrium and system optimum are quite related problems in networks without capacities on the arcs. In fact, given convex and twice differentiable transit time functions $\tau_e(x_e)$, the system optimum is the user equilibrium for the transit time functions $\bar{\tau}_e(x_e) = \tau_e(x_e) + x_e \frac{d\tau_e(x_e)}{dx_e}$, and the system optimum is the unique user equilibrium for the transit time functions $\hat{\tau}_e(x_e) = \frac{1}{x_e} \int_0^{x_e} \tau_e(t)dt$, see [11].

So they are both instances of convex optimization with convex separable objective and linear constraints. They are usually solved with the Frank-Wolfe algorithm [36], which is essentially a feasible direction method, and several variants thereof such as the Partan algorithm [32], which performs a more intelligent line search. Computing the routes of the user equilibrium is usually referred to as *traffic assignment problem*. It was originally introduced by Wardrop [93] in order to model natural driver behavior, and has been studied extensively in the literature, see [89,78]. In the absence of arc capacities, the linearized problem decomposes into separate shortest path problems for every origin-destination pair (commodity), where the length of an arc e is given by the marginal cost $\bar{\tau}_e(x_e) = \tau_e(x_e) + x_e \frac{d\tau_e(x_e)}{dx_e}$. For capacitated networks, one has to solve a multicommodity flow problem instead.

3.2 Constrained System Optimum

Can one overcome the unfairness in the system optimum by introducing constraints on the paths that may be recommended as routes? This problem has been investigated by Jahn et al. in [51] for static traffic networks with congestion. For each commodity i they consider only a subset $\bar{\mathcal{P}}_i \subseteq \mathcal{P}_i$ of *admissible paths*. In the simplest case, admissibility is defined by geographical length in the following way. Every arc e has a geographical length ℓ_e, and path $P \in \mathcal{P}_i$ is *admissible* if its geographical length $\ell_P := \sum_{e \in P} \ell_e$ does not exceed the geographical length of a shortest path between s_i and t_i by a certain, globally controlled *fairness factor* $L > 1$ (e. g., $L = 1.2$). The corresponding optimum is a *constrained system optimum* for the original problem and the total travel time increases with $1/L$. An example from [51] with real data is given in Figure 5.

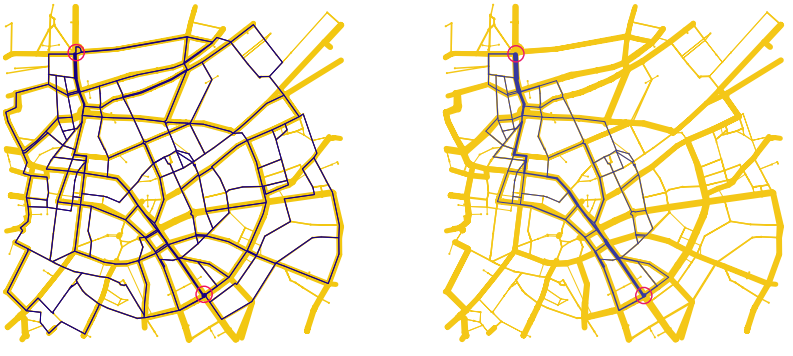

Fig. 5. System optimum with restrictions on path lengths: One commodity routed through the road network between the marked nodes. The left-hand side image displays the system optimum in which flow is distributed over the whole network in order to avoid high arc flows that would incur high arc transit times. In the right-hand side image the same amount of flow gets routed, but this time with a restriction on the geographical path lengths. Line thickness denotes arc capacity (yellow) and arc usage (blue).

More precisely, Jahn et al. introduce the concept of the *normal length* of a path, which can be either its traversal time in the uncongested network, its traversal time in user equilibrium, its geographic distance, or any other appropriate measure. The only condition imposed on the normal length of a path is that it may not depend on the actual flow on the path. Equipped with this definition, they look for a *constrained system optimum* (*CSO*) in which no path carrying positive flow between a certain origin-destination pair is allowed to exceed the normal length of a shortest path between the same origin-destination pair by more than a tolerable factor $L > 1$. By doing so, one finds solutions that are fair and efficient at the same time. This approach implements a compromise between user equilibrium and system optimum, which meets a demand expressed e. g. by

Beccaria and Bolelli [10]: The goal should be to "find the route guidance strategy which minimizes some global and community criteria with individual needs as constraints".

Computations in [51] with the street network of Berlin and networks from the *Transportation Network Test Problems* website [5] indicate that the *CSO* with transit times in the user equilibrium as normal lengths is the right concept to measure fairness. The resulting total travel time cannot exceed that of the user equilibrium, i.e., $c(SO) \leq c(CSO) \leq c(UE)$, and achieves a much better fairness than the unrestricted system optimum, while at the same time avoiding the larger road usage and non-monotonicity of the user equilibrium. An example is given in Figure 6 taken from [51].

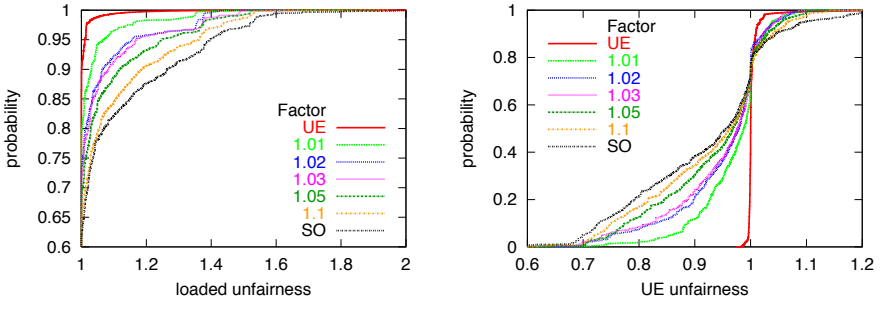

Fig. 6. Objective values and unfairness distributions for the area Neukölln in Berlin. Normal lengths are equal to transit times in user equilibrium. Total travel times are distinctively smaller than in equilibrium, while the fraction of users traveling longer than in equilibrium is substantially smaller.

Besides the unfairness with respect to the user equilibrium, Jahn et al. consider also an intrinsic unfairness of the suggested routes, which is measured as the largest ratio ℓ_{P_1}/ℓ_{P_2} of the geographical path lengths ℓ_{P_j} of two flow carrying paths P_1, P_2 for any origin-destination pair i. The intrinsic unfairness of the system optimum may go up to 3 and more, while its control via the fairness factor L also implies a strongly correlated fairness for other "length" measures such as actual transit times or transit times in the uncongested network.

3.3 Algorithmic Issues

To solve Problem *CSO*, Jahn et al. use the Partan variant of the Frank-Wolfe algorithm mentioned above. The model is path-based, where every potential path $P \in \cup_{i \in K} \bar{\mathcal{P}}_i$ is represented by a path flow variable x_P. However, these paths are only given implicitly by the length constraints $\sum_{e \in P} \ell_e \leq L\, \lambda(s_i, t_i)$ for a path between s_i and t_i, where $\lambda(s_i, t_i)$ is the normal path length between s_i and t_i. Because there may be exponentially many paths, they are only generated as needed. For that reason, the algorithm can be considered as a column generation method. This becomes even more obvious when the network is capacitated. In

that case, in order to find the best descent direction, one has to solve a linear program (a multicommodity flow problem) in path variables with column generation.

One then maintains a linear program (the *master program*) with a subset of the variables (enough to contain a basis). Checking the master program for optimality then reduces to a *constrained shortest path problem* in the given network: Compute a shortest path (where the length is influenced by the dual solution of the master program) such that the normal length constraint $\sum_{e \in P} \ell_e \leq L\,\lambda(s_i, t_i)$ is preserved. Then either optimality is established and the current solution in the master problem is optimal, or there are paths that are "too long". These paths are added to the master problem (usually in exchange for others) and one iterates.

This is the point in the algorithm where also other conditions on the paths could be considered. They can be dealt with in the current algorithm if the shortest path problem resulting from the linear programming optimality conditions can be combined with the restrictions defining the paths.

The constrained shortest path problem is NP-hard in the weak sense. There are many different approaches to solve this problem in practice. Jahn et al. implemented the label correcting algorithm of [2] because of its superior computational efficiency. The algorithm fans out from the start node s and labels each reached node $v \in V$ with labels of the form $(d_\ell(v), d_\tau(v))$. For each path from s to v that has been detected so far, $d_\tau(v)$ represents its transit time and $d_\ell(v)$ its distance. During the course of the algorithm, several labels may have to be stored for each node v, namely the Pareto-optimal labels of all paths that have reached it. This labeling algorithm can be interpreted as a special kind of branch-and-bound with a search strategy similar to breadth-first search. Starting from a certain label of v, one obtains lower bounds for the remaining paths from v to t by separately computing ordinary shortest path distances from v to t with respect to transit times τ_e and lengths ℓ_e, respectively. If one of these bounds is too large, the label can be dismissed. The algorithm reduces essentially to a repeated application of Dijkstra's algorithm and runs polynomial in the maximum number of labels generated at a node.

The non-linear part of the algorithm is dealt with as in the Frank-Wolfe algorithm. One first computes a feasible direction $y - x = \operatorname{argmin}\{\nabla c(x)^T y \mid y \text{ feasible flow}\}$ of the objective function c at the current flow x and then a stepsize λ by line search. The next solution is then obtained as $x := x + \lambda(y - x)$. At first glance, this natural approach seems to be infeasible since the objective function $c(x)$ involves all of the many path variables x_P. But the scalar product $\nabla c(x)^T y$ can equivalently be expressed in arc variable vectors, thus reducing the number of variables to the number of arcs in the network. Also the line search can be reduced to the number of paths that actually carry flow, which does not get too large.

Figure 7 illustrates the nesting of the different steps in the algorithm. The second step (solving the linear program) is only necessary for capacitated networks. It becomes clear that algorithms for shortest paths and constrained shortest

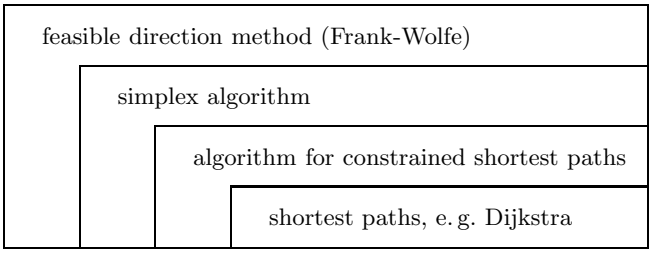

Fig. 7. Steps in computing the constrained system optimum. The second step (solving the linear program) is only necessary for capacitated networks.

paths are at the core of the algorithm and that their run time largely determines the efficiency of the whole algorithm.

The algorithm has been tested extensively in [51]. Most notably, the time needed to compute a constrained system optimum is typically not larger than that for computing an unconstrained system optimum, and it is only somewhat larger than that for getting a user equilibrium. In fact, the problem of finding a constrained system optimum becomes computationally more costly with increasing values of the tolerance factor L. The reason is that the number of allowable paths increases. However, the constrained shortest path subproblems become easier because the normal lengths are less binding. In this trade-off situation, the total work and the number of iterations increase, but the work per iteration decreases. Generally, most of the time is spent on computing constrained shortest paths (which implies that improved algorithms for this subproblem would yield greatly improved overall performance). Instances with a few thousand nodes, arcs and commodities can be solved on an average PC within minutes. Bigger instances like Berlin take longer but can also be solved without difficulty in less than an hour.

Additional speedups are obtained by König [62] with a Lagrangian relaxation that relaxes the coupling condition between flow rates on paths and arcs, i.e., the condition that $x_e = \sum_{P:e\in P} x_P$. The problem can then be separated into commodities and determining the Lagrangian dual reduces to computing resource-constrained shortest paths and evaluation of a simple twice differentiable function and its derivatives. The actual computation of the Lagrangian dual is done with the Proximal Analytic Center Cutting Plane Method of Babonneau et al. [4]. Altogether, this leads to a speedup by a factor of 2 compared with the Partan algorithm.

3.4 Acceleration of Shortest Path Computation

Figure 7 shows that repeated (constrained) shortest path calculations form the bottleneck for computing the constrained system optimum. Similar requirements for repeated shortest path calculations arise in algorithms for navigation systems, where the network is again considered to be static (at least over certain periods).

The last few years have witnessed tremendous progress in speeding up such calculations by preprocessing the network data for more efficient repeated shortest path calculations. A good overview on this progress is given by the theses of Wilhalm [95], Schilling [85], Schultes [88] and the volume of papers of the 2006 Dimacs Challenge on Shortest Paths (in preparation) [22].

Köhler et al. [57,85] investigate a generalization of a partition-based arc labelling approach has that is referred to as the *arc-flag approach*. The basic idea of the arc-flag approach using a simple rectangular geographic partition was suggested by Lauther in [66,67] and patented in [25]. The arc-flag approach divides the graph $G = (V, A)$ into regions $r \in R$ and gathers information for each arc $a \in A$ and for each region $r \in R$ ($V = \bigcup_{r \in R} r$) on whether the arc a is on at least one shortest path leading into region r. For each arc $a \in A$ this information is stored in a flag (bit) vector f_a. The vector f_a contains a flag (`true` or `false`) for each region $r \in R$ indicating whether the arc a can contribute to answering shortest path queries for nodes in region r or not, see Figure 8. Thus, the size of each flag vector is determined by the number $|R|$ of regions and the number of flag vectors is bounded by the number $|A|$ of arcs. Since the actual number of unique flag vectors can be much smaller than the number of arcs, storing the flag vectors at one point and adding an index (or pointer) to each arc can reduce the extra amount of memory below the obvious $|A||R|$ bits. The number of regions depends on the input graph size, but can be kept to a moderate size: 200 regions already lead to considerable speedups on instances with 24M nodes and 58M edges.

The arc-flags are used in a slightly modified Dijkstra computation to avoid exploring unnecessary paths. This means that one checks the flag entry of the corresponding target region (the region where the target node t belongs to) each time before the Dijkstra algorithm tries to traverse an arc. Thus, implementing the arc-flags is one of the simplest acceleration modifications of the standard Dijkstra algorithm and therefore suggests itself for straightforward usage in existing code bases.

The choice of the underlying partition is crucial for the speedup of the arc-flag acceleration of Dijkstra's algorithm. In [57] a multi-way arc separator is suggested

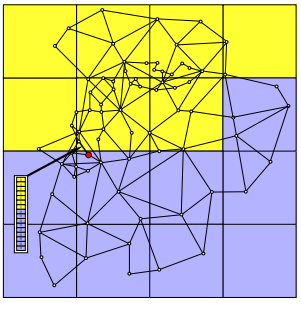

Fig. 8. Rectangular decomposition. Every arc $e = (u, v)$ carries a bit vector with a bit for every region r. The bit of region r is set to 1 iff e is on a shortest path from u to a node in r.

as an appropriate partition for the arc-flags, see Figure 9 for an example. This improvement achieved much better speedups compared to the original arc-flag version in [67]. For instance, using this one can reach acceleration factors 10 times higher than with Lauther's version of the arc-flags (on networks with up to 0.3M nodes, 0.5M arcs and 278 bits of additional information per arc). This study was further refined by Möhring et al. [76].

When combining the arc-flags with a multi-way arc separator partition and a bi-directed search, the overall performance of the method is competitive to those of other acceleration techniques such as the highway hierarchy method of Sanders and Schultes [84] but requires much more preprocessing time. Hilger et al. [58] reduce the preprocessing times significantly and also improve the efficiency of queries and the space requirements. This is achieved by a new *centralized shortest path* algorithm which computes distances simultaneously from a set of starting vertices instead of one starting vertex. Another improvement on the preprocessing time is achieved by removing small attached structures for which the arc-flags can be calculated in a much easier way. Note that this reduction is only performed during preprocessing, queries are calculated on the unmodified graph using the pre-calculated information.

On continental road networks like the US network (24M nodes, 54M edges) this method only needs a few hours to complete the pre-calculation. This approach is the first to apply the arc-flag accelerated shortest path method to networks of this size. Recently, Bauer and al. [6] have demonstrated that the arc flag method currently constitutes the best performing purely goal-directed acceleration technique and that the preprocessing can even be improved further.

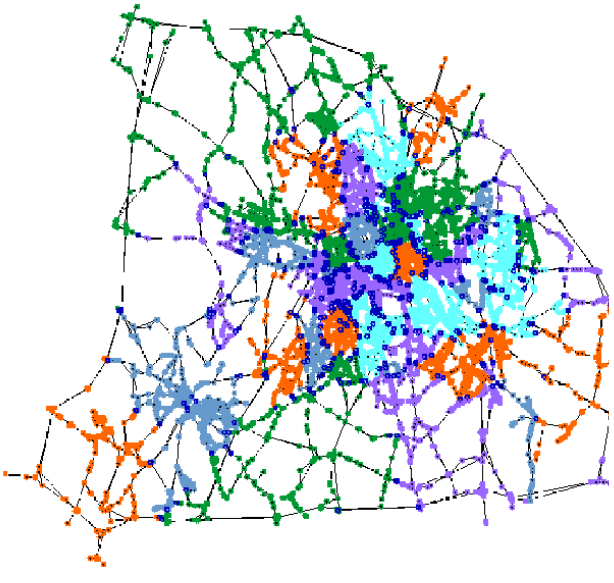

Fig. 9. Berlin divided by a hierarchical separator approach

If preprocessing is not possible, e.g. because the arc values change too often (which is the case in the algorithms for the restricted user optimum), one can still apply goal directed and bidirected search for shortest paths. Here the main underlying idea is to distort arc weights in such a way, that Dijkstra's algorithm prefers to explore nodes that are closer to the target node, thus reducing the time it takes to reach the source. Figure 10 illustrates the areas of the graph that are explored in the respective searches.

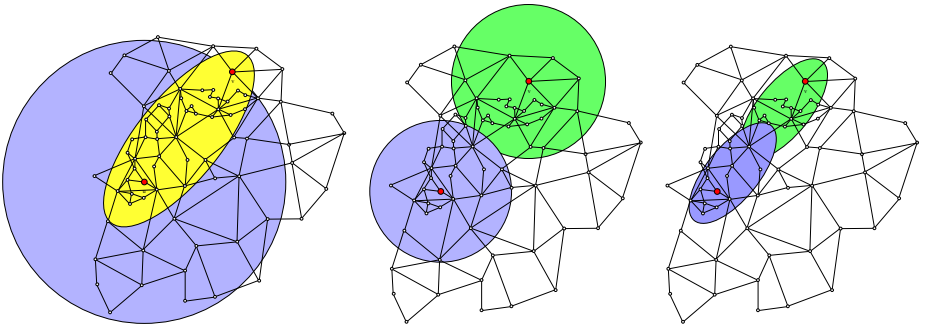

Fig. 10. Variants of simple search speedup techniques: goal directed, bidirected, and their combination (from left to right)

The goal directed approaches can also be applied to some extent to the constrained shortest path problem [57]. Here, the simple goal oriented search showed the best speedup, as the bidirectional variant suffers from the lack of a good stopping criterion.

4 Flows over Time with Flow Dependent Transit Times

In the previous sections we particularly considered flow models that either were able to capture flow variations with respect to time (Section 2.2), or transit time variations with respect to the flow situation on an arc (Section 3). Both aspects are important features that have to be taken into account when optimizing traffic systems. However, while in the previous sections we handled these features separately, we will now investigate what can be done if we have to deal with both of them at the same time. To be more precise, we study flow variations over time in networks where transit times on the arcs vary with the amount of flow on that particular arc. While both static flow theory with flow dependent transit times and the theory of flows over time with constant transit times are well studied areas of research, the field of flows over time with flow dependent transit times has attracted attention only in the last couple of years. One main reason for this seems to be the lack of well defined models for this kind of flows, which is due to the more complicated setting. Standard approaches from flows over time with constant transit times seem not easily be carried over to this case and we

will explain some of the occurring problems in the following. There is, in fact, a variety of different approaches to define an applicable model. Unfortunately, all of them are capturing only some but not all features of flows over time with flow dependent transit times. However, they hopefully will be helpful for the design of useful models and algorithms in the future.

4.1 Problems of Flow Dependent Transit Times

One significant property of flows over time in networks is that, in contrast to static flow problems, flow values on arcs may change with time. As mentioned before, in many applications one has to deal also with the phenomenon that the time taken to traverse an arc varies with the current flow situation on this arc. It is a highly nontrivial problem to map these two aspects into an appropriate and tractable mathematical network flow model and there are hardly any algorithmic techniques known which are capable of providing reasonable solutions even for networks of rather modest size. The crucial parameter for modeling temporal dynamics of time-dependent flows is the presumed dependency of the actual transit time τ_e on the current (and maybe also past) flow situation on arc e. Unfortunately, there is a tradeoff between the need of modeling this usually highly complex correlation as realistically as possible and the requirement of retaining tractability of the resulting mathematical program.

As for the static case, a fully realistic model of flow-dependent transit times on arcs must take density, speed, and flow rate evolving along the arc into consideration [38]. Unfortunately, even the solution of mathematical programs relying on simplifying assumptions is in general still impracticable, i. e., beyond the means of state-of-the-art computers, for problem instances of realistic size (as those occurring in real-world applications such as road traffic control).

When considering the correspondence between transit time and flow on a particular arc, the first question in a time dependent model is, of course, which flow or flow rate (flow units traversing an arc per time unit) do we measure? When deciding about the transit time of a given unit of flow in an arc e at a particular point in time t, what flow rate value do we consider to determine this transit time? One possibility would be, to take the inflow rate, i. e., the flow rate at the beginning of the arc e at the moment that the particular flow unit enters e. When looking at this approach in the setting of traffic networks, one realizes its drawbacks. Even if the number of cars which currently enter a street is small compared to the capacity of the street, their transit times might nevertheless be huge due to traffic congestion caused by a large number of cars which have entered the arc earlier. Another plausible choice would be to consider the whole amount of flow that is on arc e at time t, i. e., the load of e at time t. Yet, this has its disadvantages as well, since when considering all cars on a street e at some point in time t, one even counts those cars that are behind the particular car of interest.

Besides the question how to measure the flow situation on an arc, an important figure in the search for a good model is the transit time function itself. Due to the inherent complexity of the time- and flow-dependent setting, models in the

literature often rely on relatively simple transit time functions, based on those, used in static models of flow-dependent networks, e. g., the Davidson function or the function of the BPR (see [89]). While these functions $\tau_e(x_e)$ are given for a static flow value x_e, that is not varying over time, one can interpret $\tau_e(x_e)$ as the transit time on arc e for the static flow rate x_e. Of course, these functions mean a simplification of the actual flow situation in a time- and flow-dependent model. However, their use is justified already since they are a generalization of the well-established models for the static case and thus often allow to compare the quality of a computed solutions to the solutions of the static model.

When developing flow models for real-world applications, a useful model often has to satisfy certain additional requirements, which sometimes increase the complexity of this model enormously. One prominent such requirement that is present especially in traffic applications is the so-called *FIFO* or *first-in-first-out* property. This property states that no unit of flow A entering an arc e after some flow unit B exits this arc before B; in other words, overtaking of flow units is not permitted. While a violation of this causes no problems for traffic when considering single cars, it is considered not permitted by traffic scientists for large groups of cars. In a model with constant transit time this requirement is trivially satisfied since all flow through an arc travels with the same travel time. However, in a model with variable transit times on the arcs it is often very hard to guarantee.

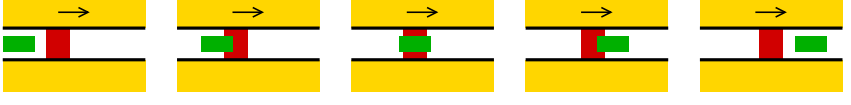

Fig. 11. Violations of the first-in-first-out property

4.2 Models and Algorithms

In the following we discuss some approaches that can be found in the literature. For a more detailed account and further references we refer to [3,69,79,81].

One of the first models for time dependent flows with flow dependent transit times has been defined by Merchant and Nemhauser [73]. They formulate a non-linear and non-convex program in which time is discretized. In their model, the outflow out of an arc in each time period solely depends on the amount of flow on that arc at the beginning of the time period. Instead of using a transit time function, as discussed above, they use for each arc e a cost function $h_{e,t}$ and an exit function g_e, where $h_{e,t}$ accounts for the transit time that is spend by the flow on arc e in period t. Now, if x is the amount of traffic on an arc e at the beginning of time period t, a cost $h_{e,t}(x)$ is incurred and during period t an amount of traffic $g_e(x)$ exits from arc e. In their setting Merchant and Nemhauser were able to consider networks of general topology and multiple origins, but only a single sink. Although this model seems to be not easily applicable to real-world applications, it has been an important stepping stone in the study of

time- and flow-dependent models. Still, the non-convexity of this model causes both analytical and computational problems.

Later, Merchant and Nemhauser [74] and Carey [17] further studied this model and described special constraint qualifications which are necessary to guarantee optimality of a solution in this model. Carey [18] introduces a slight revision of the model of Merchant and Nemhauser which transforms the non-convex problem into a convex one.

A rather different approach to modeling flows over time with flow dependent transit times was used by Carey and Subrahmanian [19]. Before we look at their model in more detail, we shortly discuss the problems occurring with the time expanded network approach of Section 2.2.

Recall that the main advantage of the time expanded network is that we can apply known algorithms for static networks to solve the corresponding time dependent problems. Unfortunately, this simple approach does not easily carry over to the case of flow dependent transit times. Consider a simple network with only two arcs. Initially, the flow value on each arc is zero and, as each arc has a certain empty transit time (in our example the transit time of each arc is one time unit), we can easily construct a time expanded network, as described earlier (see Figure 12). Again, we can consider a path P from some source s to a sink t in the original network and find the corresponding path in the time-expanded version that captures the transit times of the different arcs. However, if we now send some flow on this path P through the network, then the problem of this approach becomes evident: since the transit times of the arcs vary with the amount of flow on them, the time expanded graph changes its adjacencies and its structure. In fact, the arcs of the path corresponding to P in the original network now become incident to completely different copies of the vertices in the time-expanded network. As a consequence we cannot apply standard algorithms for static flow problems any more since they rely heavily on the fact that the underlying graph is not changing. Thus, the advantage of the time-expanded approach seems to be lost.

In spite of this daunting observation, Carey and Subrahmanian [19] were able to define a variant of the time expanded network that captures at least some features of flow dependent transit times. Their network again has fixed transit times on the arcs. But for each time period, there are several copies of an arc of the underlying 'static' network corresponding to different transit times. Now changing flow values on the arcs is to result no longer in a change of adjacencies in the graph but instead different copies of the same arc should be used for different amounts of flow. In order to enforce these flow-dependent transit times, special capacity constraints are introduced which give rise to a dependency between the flow on all copies of an arc corresponding to one time step. As a consequence of these generalized capacity constraints, the resulting static problem on the modified time-expanded graph can no longer be solved by standard network flow techniques but requires a general linear programming solver. This constitutes a serious drawback with regard to the practical efficiency and applicability of this model.

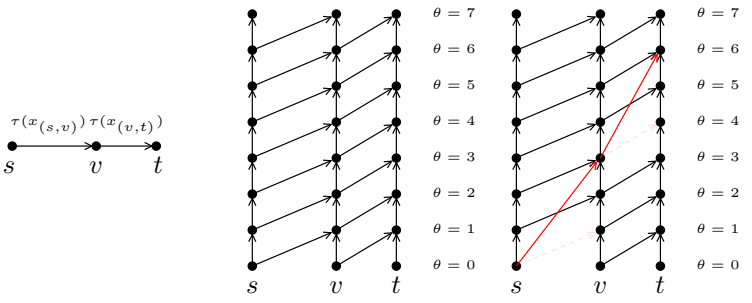

Fig. 12. A simple two-arc network together with a time-expansion for empty transit times, and the time-expansion when one unit of flow is sent from s to t, showing the problems of a time expansion for flow dependent transit times

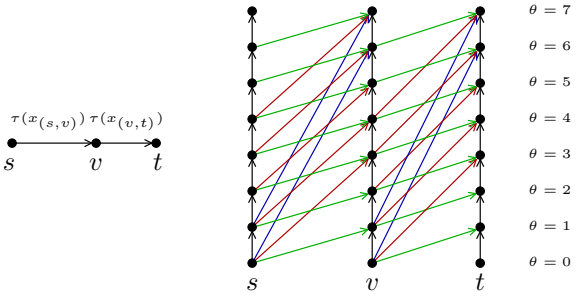

Fig. 13. Idea of expanded model of Carey and Subrahmanian; here for a simple two-arc network with transit time function $\tau(x_e)$, causing transit times 1, 3, and 6, depending on the amount of flow x_e entering arc e

Although the models mentioned up to this point mark some important developments in the study of flows over time with flow dependent transit times, they have a drawback in common; they are not suitable for efficient optimization methods. In contrast to the models for static flows and for flows over time with constant transit times, here fast standard flow algorithms cannot be applied. This is of course partially due to the complexity of these models. However, for solving real problems, also on large networks one has to have fast algorithmic tools.

The following model was suggested by KÃűhler and Skutella [60,61]. It is inspired by the earlier mentioned results of Ford and Fulkerson and of Fleischer and Skutella (see Section 2.2). Although the result of Ford and Fulkerson for computing a maximum flow over time for a fixed time horizon T cannot be generalized to the more general setting of flow-dependent transit times, it is shown in [60,61] that there exists at least a provably good temporally repeated flow for the quickest flow problem, i. e., for the problem of sending a certain amount of flow from some vertex s through the network such that it reaches the sink t as fast as possible.

As for the case of fixed transit times, and in contrast to the before mentioned models for flow dependent transit times, this flow can be determined very efficiently. Since the transit times are no longer fixed, the linear min-cost flow problem considered by Ford and Fulkerson now turns into a *convex cost* flow problem. Under very mild assumptions on the transit time functions τ_e, the resulting optimal static flow can be turned into a temporally repeated flow which needs at most twice as long as a quickest flow over time.

This result is based on the fairly general model of flow-dependent transit times where, at each point in time, the speed on an arc depends only on the amount of flow (or *load*) which is currently on that arc. This assumption captures for example the behavior of road traffic when an arc corresponds to a rather short street (notice that longer streets can be replaced by a series of short streets).

The next model that we present here is again based on the time-expanded network, known from flows over time with constant transit time. As in the model of Carey and Subrahmanian [19], KÃűhler, Langkau and Skutella [56,65] suggest a generalized time-expanded network with multiple copies of each arc for each time step. However, in contrast to the model in [19], additional 'regulating' arcs are introduced which enable to enforce flow-dependent transit times without using generalized capacity constraints. As a result, one can apply the whole algorithmic toolbox developed for static network flows to this generalized time-expanded network, also called the *fan graph* (see Figure 14).

The underlying assumption for this approach (as for the related approach of Carey and Subrahmanian [19]) is that at any moment of time the transit time on an arc solely depends on the current rate of inflow into that arc. We therefore speak of *flows over time with inflow-dependent transit times*, emphasizing the fact that transit times are considered as functions of the rate of inflow. Thus, in contrast to the load-dependent model developed by KÃűhler and Skutella [60,61], the flow units traveling on the same arc at the same time do not necessarily experience the same pace, as the transit time and thus the pace of every unit of flow is determined when entering the arc and remains fixed throughout.

As in the case of constant transit times, a drawback of this rather general time-expanded model is the size of the constructed graph. Depending on the considered time horizon and the size of a single time step, the graph grows

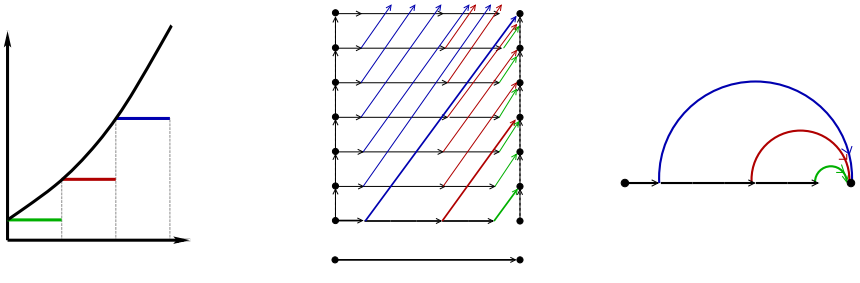

Fig. 14. Transit time function, fan-graph, and bow-graph of a single arc

rapidly. Hence, when considering large instances of networks, this model is again not applicable. However, as shown by KÃűhler, Langkau and Skutella [65,56] as well, there is also a time condensed version of this network, the so-called *bow-graph* of the given network. The idea is here to construct a graph with constant transit times on the arcs, such that the time expansion of this graph is again the fan-graph. Another way of looking at this condensed graph is, to see it as an expansion of the transit time function of the original graph, since now for each arc we have copies for each possible transit time of this arc (instead of having a copy for each time step in the time expanded graph). Again a set of regulation arcs guarantees that only a restricted amount of flow can transit an arc using a particular copy of this arc in the bow-graph. This bow-graph itself does not capture the time varying behavior of the flow over time. However, the interesting property here is that the bow-graph relates to the fan-graph in this model, as the original graph relates to the time expanded graph in the constant travel time model of Ford and Fulkerson. As shown in [65,56], this relationship can be exploited for developing a 2-approximation algorithm for the quickest flow problem in this model for flows over time with inflow-dependent transit times.

For the load-dependent model of KÃűhler and Skutella [60,61] the quickest flow problem can be shown to be APX-hard even for the case of only one source and one sink, implying that there is no polynomial-time approximation scheme unless P=NP. The inflow-dependent model is easier to approximate. Hall, Langkau and Skutella [65,42,43] combined the inflow-dependent model with the idea of condensed time-expanded networks (see Section 2.2) to design a fully polynomial time approximation scheme for the quickest multicommodity flow problem with inflow-dependent transit times.

Not only quickest flows but also earliest arrival flows have been studied in the context of flow dependent transit times. Baumann and KÃűhler [7,8] show that for the case of flow-dependent transit times earliest arrival flows do not always exist. However, an optimization version of the earliest arrival problem is introduced. This is done as follows. Find the minimum α such that there is a flow over time f that sends for each $\theta \in [0, T)$ at least as much flow into the sink t as can be sent into t up to time $\frac{\theta}{\alpha}$ by a maximum flow over time $f_{\max}(\frac{\theta}{\alpha})$ within this time horizon, i.e., $\text{value}(f, \theta) \geq \text{value}(f_{\max}(\frac{\theta}{\alpha}), \frac{\theta}{\alpha})$. Such a flow will be called an *α-earliest arrival flow.* Using any of the approximation algorithms for the quickest flow problem both for load-dependent and inflow-dependent flows one can design approximation algorithms for the α-earliest arrival flow problem.

Finally, we would like to mention the *rate-dependent* model. It was introduced by Hall and Schilling [44] and tries to overcome some of the deficiencies of the flow-dependent models outlined above. In particular one can avoid FIFO-violations (see Section 4.1) which might occur in the inflow-dependent model and undesired overtaking (see Section 4.1) that might occur in the load-dependent model. In the rate-dependent model the relationship between the flow-rate on an edge and the *pace* (which is defined to be the inverse of the velocity) is the central notion. Roughly speaking, the novel idea of this model is not to set a fixed pace for each possible flow rate (as done in the earlier models), but rather to

allow a whole interval of pace–flow rate combinations. Hence, one ends up with a less restrictive and more realistic model. Unfortunately, these advantages are obtained on the cost of efficiency: Similarly, as for the earlier models, computing a quickest flow is NP-hard. But, in contrast, no efficient approximation algorithm, e.g. for the quickest flow in this model is known. Yet, the authors show the practical relevance of their approach by presenting a heuristic algorithm for the quickest flow problem and comparing its solutions with the solutions of the inflow-dependent model.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows. Theory, Algorithms, and Applications. Prentice Hall, Englewood Cliffs (1993)
2. Aneja, Y.P., Aggarwal, V., Nair, K.P.K.: Shortest chain subject to side constraints. Networks 13, 295–302 (1983)
3. Aronson, J.E.: A survey of dynamic network flows. Annals of Operations Research 20, 1–66 (1989)
4. Babonneau, F., du Merle, O., Vial, J.-P.: Solving large-scale linear multicommodity flow problems with an active set strategy and proximal-accpm. Operations Research 54(1), 184–197 (2006)
5. Bar-Gera, H.: Transportation network test problems (2002), http://www.bgu.ac.il/~bargera/tntp/
6. Bauer, R., Delling, D., Sanders, P., Schieferdecker, D., Schultes, D., Wagner, D.: Combining hierarchical and goal-directed speed-up techniques for Dijkstra's algorithm (submitted) (2008)
7. Baumann, N.: Evacuation by Earliest Arrival Flows. Ph.D thesis, Universität Dortmund (2007)
8. Baumann, N., Köhler, E.: Approximating earliest arrival flows with flow-dependent transit times. Discrete Appl. Math. 155, 161–171 (2007)
9. Baumann, N., Skutella, M.: Solving evacuation problems efficiently: Earliest arrival flows with multiple sources. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, Berkeley, CA, pp. 399–408 (2006)
10. Beccaria, G., Bolelli, A.: Modelling and assessment of dynamic route guidance: the MARGOT project. In: Vehicle Navigation & Information Systems Conference Proceedings (VNIS 1992), pp. 117–126. IEEE, Los Alamitos (1992)
11. Beckmann, M., McGuire, C.B., Winston, C.B.: Studies in the economics of transportation. Yale University Press, New Haven (1956)
12. Ben-Akiva, M., Koutsopoulos, H., Mishalani, R., Yang, Q.: Simulation laboratory for evaluating dynamictraffic management systems. ASCE Journal of Transportation Engineering 123(4), 283–289 (1997)
13. Berlin, G.N.: The use of directed routes for assessing escape potential. National Fire Protection Association, Boston (1979)
14. Braess, D.: Über ein paradoxon aus der Verkehrsplanung. Unternehmensforschung 12, 258–268 (1968)
15. Burkard, R.E., Dlaska, K., Klinz, B.: The quickest flow problem. ZOR — Methods and Models of Operations Research 37, 31–58 (1993)
16. Busaker, R.G., Gowen, P.J.: A procedure for determining minimal-cost network flow patterns. Technical Report 15, Operational Research Office. John Hopkins University, Baltimore, MD (1961)

17. Carey, M.: A constraint qualification for a dynamic traffic assignment model. Transp. Science 20, 55–58 (1986)
18. Carey, M.: Optimal time-varying flows on congested networks. OR 35, 58–69 (1987)
19. Carey, M., Subrahmanian, E.: An approach for modelling time-varying flows on congested networks. Transportation Research B 34, 157–183 (2000)
20. Chalmet, L.G., Francis, R.L., Saunders, P.B.: Network models for building evacuation. Management Science 28, 86–105 (1982)
21. Correa, J.R., Schulz, A.S., Stier Moses, N.E.: Selfish routing in capacitated networks. Mathematics of Operations Research 29(4), 961–976 (2004)
22. DIMACS. 9th Implementation Challenge – Shortest Paths (2006)
23. Dirkse, S.P., Ferris, M.C.: Traffic modeling and variational inequalities using GAMS. In: Toint, P.L., Labbe, M., Tanczos, K., Laporte, G. (eds.) Operations Research and Decision Aid Methodologies in Traffic and Transportation Management, vol. 166, pp. 136–163. Springer, Berlin (1998)
24. Elias, P., Feinstein, A., Shannon, C.E.: Note on maximum flow through a network. IRE Transactions on Information Theory IT-2, 117–119 (1956)
25. Enders, R., Lauther, U.: Method and device for computer assisted graph processing (May 1999), http://gauss.ffii.org/PatentView/EP1027578
26. Fischer, S., Räcke, H., Vöcking, B.: Fast convergence to wardrop equilibria by adaptive sampling methods. In: Proceedings of the 38th Annual ACM Symposium on Theory of Computing, pp. 653–662 (2006)
27. Fleischer, L., Skutella, M.: The quickest multicommodity flow problem. In: Cook, W.J., Schulz, A.S. (eds.) IPCO 2002. LNCS, vol. 2337, pp. 36–53. Springer, Heidelberg (2002)
28. Fleischer, L., Skutella, M.: Minimum cost flows over time without intermediate storage. In: Proceedings of the 14th Annual ACM–SIAM Symposium on Discrete Algorithms, Baltimore, MD, pp. 66–75 (2003)
29. Fleischer, L., Skutella, M.: Quickest flows over time. SIAM Journal on Computing 36, 1600–1630 (2007)
30. Fleischer, L.K.: Faster algorithms for the quickest transshipment problem. SIAM Journal on Optimization 12, 18–35 (2001)
31. Fleischer, L.K., Tardos, É.: Efficient continuous-time dynamic network flow algorithms. Operations Research Letters 23, 71–80 (1998)
32. Florian, M., Guélat, J., Spiess, H.: An efficient implementation of the "Partan" variant of the linear approximation method for the network equilibrium problem. Networks 17, 319–339 (1987)
33. Ford, L.R., Fulkerson, D.R.: Maximal flow through a network. Canadian Journal of Mathematics 8, 399–404 (1956)
34. Ford, L.R., Fulkerson, D.R.: Constructing maximal dynamic flows from static flows. Operations Research 6, 419–433 (1958)
35. Ford, L.R., Fulkerson, D.R.: Flows in Networks. Princeton University Press, Princeton (1962)
36. Frank, M., Wolfe, P.: An algorithm for quadratic programming. Naval Research Logistics Quaterly 3, 95–110 (1956)
37. Gale, D.: Transient flows in networks. Michigan Mathematical Journal 6, 59–63 (1959)
38. Gartner, N., Messer, C.J., Rathi, A.K.: Traffic flow theory: A state of the art report (1997), http://www-cta.ornl.gov/cta/research/trb/tft.html
39. Hagstrom, J.N., Abrams, R.A.: Characterizing braess's paradox for traffic networks. In: Proceedings of IEEE 2001 Conference on Intelligent Transportation Systems, pp. 837–842 (2001)

40. Hajek, B., Ogier, R.G.: Optimal dynamic routing in communication networks with continuous traffic. Networks 14, 457–487 (1984)
41. Hall, A., Hippler, S., Skutella, M.: Multicommodity flows over time: Efficient algorithms and complexity. Theoretical Computer Science 379, 387–404 (2007)
42. Hall, A., Langkau, K., Skutella, M.: An FPTAS for quickest multicommodity flows with inflow-dependent transit times. In: Arora, S., Jansen, K., Rolim, J.D.P., Sahai, A. (eds.) RANDOM 2003 and APPROX 2003. LNCS, vol. 2764, pp. 71–82. Springer, Heidelberg (2003)
43. Hall, A., Langkau, K., Skutella, M.: An FPTAS for quickest multicommodity flows with inflow-dependent transit times. Algorithmica 47(3), 299–321 (2007)
44. Hall, A., Schilling, H.: Flows over time: Towards a more realistic and computationally tractable model. In: Proceedings of the 7th Workshop on Algorithm Engineering and Experiments (ALENEX 2005), Vancouver, Canada, pp. 55–67. SIAM, Philadelphia (2005)
45. Hamacher, H.W., Tifecki, S.: On the use of lexicographic min cost flows in evacuation modeling. Naval Research Logistics 34, 487–503 (1987)
46. Hitchcock, F.L.: The distribution of a product from several sources to numerous localities. Journal of Mathematics and Physics 20, 224–230 (1941)
47. Hoppe, B.: Efficient dynamic network flow algorithms. Ph.D thesis, Cornell University (1995)
48. Hoppe, B., Tardos, É.: Polynomial time algorithms for some evacuation problems. In: Proceedings of the 5th Annual ACM–SIAM Symposium on Discrete Algorithms, Arlington, VA, pp. 433–441 (1994)
49. Hoppe, B., Tardos, É.: The quickest transshipment problem. Mathematics of Operations Research 25, 36–62 (2000)
50. Iri, M.: A new method of solving transportation-network problems. Journal of the Operations Research Society of Japan 26, 27–87 (1960)
51. Jahn, O., Möhring, R.H., Schulz, A.S., Stier Moses, N.E.: System-optimal routing of traffic flows with user constraints in networks with congestion. Oper. Res. 53(4), 600–616 (2005)
52. Jarvis, J.J., Ratliff, H.D.: Some equivalent objectives for dynamic network flow problems. Management Science 28, 106–108 (1982)
53. Jewel, P.A.: Optimal flow through networks. Technical Report 8, Operations Research Center. MIT, Cambridge (1958)
54. Kantorovich, L.V.: On the translocation of masses. Comptes Rendus (Doklady) de l'Académie des Sciences de l'U.R.S.S. 37, 199–201 (1942)
55. Klinz, B., Woeginger, G.J.: Minimum cost dynamic flows: The series-parallel case. In: Balas, E., Clausen, J. (eds.) IPCO 1995. LNCS, vol. 920, pp. 329–343. Springer, Heidelberg (1995)
56. Köhler, E., Langkau, K., Skutella, M.: Time-expanded graphs with flow-dependent transit times. In: Möhring, R.H., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 599–611. Springer, Heidelberg (2002)
57. Köhler, E., Möhring, R.H., Schilling, H.: Acceleration of shortest path and constrained shortest path computation. In: Nikoletseas, S.E. (ed.) WEA 2005. LNCS, vol. 3503, pp. 126–138. Springer, Heidelberg (2005)
58. Köhler, E., Möhring, R.H., Schilling, H.: Fast point-to-point shortest path computations with arc-flags (February 2008); submitted to the Special Issue about the 9th DIMACS Implementation Challenge Workshop
59. Köhler, E., Möhring, R.H., Skutella, M.: Traffic networks and flows over time. In: Kramer, J. (ed.) DFG Research Center: Mathematics for Key Technologies, pp. 49–70. Berliner Mathematische Gesellschaft (2002)

60. Köhler, E., Skutella, M.: Flows over time with load-dependent transit times. In: Proceedings of the 13th Annual ACM–SIAM Symposium on Discrete Algorithms, San Francisco, CA, pp. 174–183 (2002)
61. Köhler, E., Skutella, M.: Flows over time with load-dependent transit times. SIAM Journal on Optimization 15, 1185–1202 (2005)
62. König, F.: Traffic optimization under route constraints with Lagrangian relaxation and cutting plane methods. In: Operations Research Proceedings 2006, pp. 53–59. Springer, Heidelberg (2007)
63. Koopmans, T.C.: Optimum utilization of the transportation system. Econometrica 17, 136–146 (1949)
64. Kotnyek, B.: An annotated overview of dynamic network flows. Rapport de recherche 4936, INRIA Sophia Antipolis (2003)
65. Langkau, K.: Flows Over Time with Flow-Dependent Transit Times. Ph.D thesis, TU Berlin (2003)
66. Lauther, U.: Slow preprocessing of graphs for extremely fast shortest path calculations. In: Lecture at the Workshop on Computational Integer Programming at ZIB (no documentation available) (1997)
67. Lauther, U.: An extremely fast, exact algorithm for finding shortest paths in static networks with geographical background. In: Raubal, M., Sliwinski, A., Kuhn, W. (eds.) Geoinformation und Mobilität - von der Forschung zur praktischen Anwendung, Münster, Germany. IfGI prints, vol. 22, pp. 219–230. Institut für Geoinformatik, Westfälische Wilhelms-Universität (2004)
68. Lovetskii, S.E., Melamed, I.I.: Dynamic network flows. Automation and Remote Control 48, 1417–1434 (1987); Translated from Avtomatika i Telemekhanika 11, 7–29 (1987)
69. Mahmassani, H.S., Peeta, S.: System optimal dynamic assignment for electronic route guidance in a congested traffic network. In: Gartner, N.H., Improta, G. (eds.) Urban Traffic Networks. Dynamic Flow Modelling and Control, pp. 3–37. Springer, Berlin (1995)
70. Martens, M., Skutella, M.: Length-bounded and dynamic k-splittable flows. In: Haasis, H.-D., Kopfer, H., Schönberger, J. (eds.) Operations Research Proceedings 2005, pp. 297–302. Springer, Heidelberg (2006)
71. Megiddo, N.: Combinatorial optimization with rational objective functions. Mathematics of Operations Research 4, 414–424 (1979)
72. Megiddo, N.: Applying parallel computation algorithms in the design of serial algorithms. Journal of the ACM 30, 852–865 (1983)
73. Merchant, D.K., Nemhauser, G.L.: A model and an algorithm for the dynamic traffic assignment problems. Transp. Science 12, 183–199 (1978)
74. Merchant, D.K., Nemhauser, G.L.: Optimality conditions for a dynamic traffic assignment model. Transp. Science 12, 200–207 (1978)
75. Minieka, E.: Maximal, lexicographic, and dynamic network flows. Operations Research 21, 517–527 (1973)
76. Möhring, R.H., Schilling, H., Schütz, B., Wagner, D., Willhalm, T.: Partitioning graphs to speed-up Dijkstra's algorithm. ACM Journal of Experimental Algorithmics 11, Article No. 2.8 (2006)
77. Nagel, K., Esser, J., Rickert, M.: Large-scale traffic simulations for transportation planning. In: Stauffer, D. (ed.) Annual Review of Computational Physics VII, pp. 151–202. World Scientific Publishing Company, Singapore (2000)
78. Patriksson, M.: Traffic Assignment Problems: Models and Methods. VSP International Science Publishers, Utrecht (1994)

79. Powell, W.B., Jaillet, P., Odoni, A.: Stochastic and dynamic networks and routing. In: Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (eds.) Network Routing. Handbooks in Operations Research and Management Science, ch. 3, vol. 8, pp. 141–295. North-Holland, Amsterdam (1995)
80. Prigogine, I., Herman, R.: Kinetic Theory of Vehicular Traffic. Elsevier Science B. V., Amsterdam (1971)
81. Ran, B., Boyce, D.E.: Modelling Dynamic Transportation Networks. Springer, Berlin (1996)
82. Roughgarden, T.: How unfair is optimal routing? In: Proceedings of the 13th Annual ACM–SIAM Symposium on Discrete Algorithms, San Francisco, CA, pp. 203–204 (2002)
83. Roughgarden, T., Tardos, É.: How bad is selfish routing? In. In: Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science FOCS 2000, pp. 93–102 (2000)
84. Sanders, P., Schultes, D.: Highway hierarchies hasten exact shortest path queries. In: Brodal, G.S., Leonardi, S. (eds.) ESA 2005. LNCS, vol. 3669, pp. 568–579. Springer, Heidelberg (2005)
85. Schilling, H.: Route Assignment Problems in Large Networks. Ph.D thesis, Technische Universität Berlin (2006)
86. Schluz, A.S., Stier Moses, N.E.: Efficiency and fairness of system-optimal routing with user constraints. Networks 48(4), 223–234 (2006)
87. Schrijver, A.: On the history of the transportation and maximum flow problem. Mathematical Programming 91, 437–445 (2002)
88. Schultes, D.: Route Planning in Road Networks. Ph.D thesis, Universität Karlsruhe, TH (2008)
89. Sheffi, Y.: Urban Transportation Networks. Prentice Hall, Englewood Cliffs (1985), `http://web.mit.edu/sheffi/www/urbanTransportation.html`
90. Skutella, M.: An introduction to network flows over time. In: Cook, W., Lovász, L., Vygen, J. (eds.) Research Trends in Combinatorial Optimization, pp. 451–482. Springer, Berlin (2009)
91. Tjandra, S.: Dynamic Network Optimization with Application to the Evacuation Problem. Ph.D thesis, Univeristät Kaiserslautern. Shaker Verlag, Aachen (2003)
92. Verkehrsmanagement — Eine Bestandsaufnahme. Broschüre. Erstellt von Lösch & Partner GmbH, München, in Zusammenarbeit mit diversen deutschen Automobilherstellern (1995)
93. Wardrop, J.G.: Some theoretical aspects of road traffic research. Proceedings of the Institution of Civil Engineers 1(2), 325–362 (1952)
94. Wilkinson, W.L.: An algorithm for universal maximal dynamic flows in a network. Operations Research 19, 1602–1612 (1971)
95. Willhalm, T.: Engineering Shortest Paths and Layout Algorithms for Large Graphs. Ph.D thesis, Universität Karlsruhe (TH), Karlsruhe (2005)