The Quickest Multicommodity Flow Problem*

Lisa Fleischer[1] and Martin Skutella[2]

[1] Graduate School of Industrial Administration, Carnegie Mellon University
Pittsburgh, PA 15213, USA
lkf@andrew.cmu.edu
http://www.andrew.cmu.edu/user/lkf/

[2] Institut für Mathematik, MA 6–1, Technische Universität Berlin
Straße des 17. Juni 136, 10623 Berlin, Germany
skutella@math.tu-berlin.de
http://www.math.tu-berlin.de/~skutella/

Abstract. Traditionally, flows over time are solved in time-expanded networks which contain one copy of the original network for each discrete time step. While this method makes available the whole algorithmic toolbox developed for static flows, its main and often fatal drawback is the enormous size of the time-expanded network. In particular, this approach usually does not lead to efficient algorithms with running time polynomial in the input size since the size of the time-expanded network is only pseudo-polynomial.

We present two different approaches for coping with this difficulty. Firstly, inspired by the work of Ford and Fulkerson on maximal s-t-flows over time (or 'maximal dynamic s-t-flows'), we show that static, length-bounded flows lead to provably good multicommodity flows over time. These solutions not only feature a simple structure but can also be computed very efficiently in polynomial time.

Secondly, we investigate 'condensed' time-expanded networks which rely on a rougher discretization of time. Unfortunately, there is a natural tradeoff between the roughness of the discretization and the quality of the achievable solutions. However, we prove that a solution of arbitrary precision can be computed in polynomial time through an appropriate discretization leading to a condensed time expanded network of polynomial size. In particular, this approach yields a fully polynomial time approximation scheme for the quickest multicommodity flow problem and also for more general problems.

1 Introduction

We consider flow problems in networks with fixed capacities and transit times on the arcs. The transit time of an arc specifies the amount of time it takes for flow to travel from the tail to the head of that arc. In contrast to the classical case of static flows, a *flow over time* in such a network specifies a flow rate entering

* Extended abstract; information on the full version of the paper can be obtained via the authors' WWW-pages.

an arc for each point in time. In this setting, the capacity of an arc limits the rate of flow into the arc at each point in time.

Flows over time may be applied to various areas of operations research and have many real-world applications such as traffic control, evacuation plans, production systems, communication networks, and financial flows. Examples and further application can be found in the survey articles of Aronson [1] and Powell, Jaillet, and Odoni [18].

Flows over time have been introduced about forty years ago by Ford and Fulkerson [5, 6]. They consider the problem of sending the maximal possible amount of flow from a source node s to a sink node t within a given time T. This problem can efficiently be solved by essentially one min-cost flow computation on the given network where transit times of arcs are interpreted as costs per unit of flow. Ford and Fulkerson show that an optimal solution to this min-cost flow problem can be turned into a maximal flow over time by first decomposing it into flows on paths. The corresponding flow over time starts to send flow on each path at time zero, and repeats each so long as there is enough time left in the T time units for the flow along the path to arrive at the sink. A flow over time featuring this structure is called *temporally repeated*.

A problem closely related to the problem of computing a maximal flow over time is the *quickest flow problem*: Send a given amount of flow from the source to the sink in the shortest possible time. This problem can be solved in polynomial time by incorporating the algorithm of Ford and Fulkerson in a binary search framework. Burkard, Dlaska, and Klinz [2] present a faster algorithm which even solves the quickest flow problem in strongly polynomial time.

A natural generalization of the quickest flow problem and the maximal flow problem considered by Ford and Fulkerson can be defined on networks with additional costs on the arcs. Klinz and Woeginger [14] show that the search for a quickest or a maximal flow over time with minimal cost cannot be restricted to the class of temporally repeated flows. In fact, adding costs has also a considerable impact on the complexity of these problems. Klinz and Woeginger prove NP-hardness results even for the special case of series parallel graphs. Moreover, they show that the problem of computing a maximal temporally repeated flow with minimal cost is strongly NP-hard.

Another generalization is the quickest transshipment problem: Given a vector of supplies and demands at the nodes, the task is to find a flow over time that zeroes all supplies and demands within minimal time. Hoppe and Tardos [13] show that this problem can still be solved in polynomial time. They introduce the use of *chain decomposable flows* which generalize the class of temporally repeated flows and can also be compactly encoded as a collection of paths. However, in contrast to temporally repeated flows, these paths may also contain backward arcs. Therefore, a careful analysis is necessary to show feasibility of the resulting flows over time.

All results mentioned so far work with a discrete time model, that is, time is discretized into steps of unit length. In each step, flow can be sent from a node v through an arc (v, w) to the adjacent node w, where it arrives $\tau_{(v,w)}$ time steps later; here, $\tau_{(v,w)}$ denotes the given integral transit time of arc (v,w). In particular, the time-dependent flow on an arc is represented by a time-indexed vector in this model. In contrast to this, in the continuous time model the flow on an arc e is a function $f_e : \mathbb{R}^+ \to \mathbb{R}^+$. Fleischer and Tardos [4] point out a strong connection between the two models. They show that many results and algorithms which have been developed for the discrete time model can be carried over to the continuous time model.

In the discrete time model, flows over time can be described and computed in *time-expanded networks* which were introduced by Ford and Fulkerson [5, 6]. A time expanded network contains a copy of the node set of the underlying 'static' network for each discrete time step. Moreover, for every arc e in the static network with integral transit time τ_e, there is a copy between all pairs of time layers with distance τ_e in the time-expanded network. Unfortunately, due to the time expansion, the size of the resulting network is in general exponential in the input size of the problem. This difficulty has already been pointed out by Ford and Fulkerson[3]. On the other hand, the advantage of this approach is that it turns the problem of determining an optimal flow over time into a classical 'static' network flow problem on the time-expanded network. This problem can then be solved by well-known network flow algorithms. A main contribution of this paper is to provide a possibility to overcome the difficulties caused by the size of time-expanded networks while still maintaining the latter advantage.

A straightforward idea is to reduce the size of time-expanded networks by replacing the time steps of unit length by larger steps. In other words, applying a sufficiently rough discretization of time leads to a *condensed* time-expanded network of polynomial size. However, there is a tradeoff between the necessity to reduce the size of the time-expanded network and the desire to limit the loss of precision of the resulting flow model since the latter results in a loss of quality of achievable solutions.

We show that there is a satisfactory solution to this tradeoff problem. An appropriate choice of the step length leads to a condensed time-expanded network of polynomial size which still allows a $(1+\varepsilon)$-approximate precision in time, for any $\varepsilon > 0$. This observation has potential applications for many problems involving flows over time. In particular, it yields a fully polynomial time approximation scheme (FPTAS) for the quickest multicommodity flow problem.

To the best of our knowledge, this is the first approximation result for the general time-dependent multicommodity flow problem. Moreover, the complexity of the quickest multicommodity flow problem is still open. It has neither been proved to be NP-hard nor is it known to be solvable in polynomial time. Apart from this, we believe that our result is also of interest for flow problems, like the quickest transshipment problem, which are known to be solvable in polynomial time. While the algorithm of Hoppe and Tardos [13] for the quickest transshipment problem relies on submodular function minimization, the use of condensed

[3] They use the time-expanded network only in the analysis of their algorithm. The optimality of the computed flow over time is proven by a cut in the time-expanded network whose capacity equals the value of the flow.

time-expanded networks leads to an FPTAS which simply consists of a series of max-flow computations.

Moreover, our approach also works in the setting with costs and we can give a bicriteria FPTAS[4] for the min-cost quickest multicommodity flow problem. Notice that already the single-commodity version of this problem is known to be NP-hard [14].

We next introduce a geometrically-condensed time-expanded network satisfying the following property: For any point in time θ, every unit of flow which arrives at its destination in a given flow at time θ, arrives there before time $(1+\varepsilon)\theta$ in a corresponding solution for this network. We use this to give the first FPTAS for the *earliest arrival flow problem* when there are multiple sources and a single sink. In contrast to Hoppe and Tardos' FPTAS for the single-source, single-sink problem where the amount of flow is approximately optimal at every moment of time [12], we obtain optimal flows in approximately optimal time.

While our analysis shows that condensed time-expanded networks lead to theoretically efficient (polynomial time) algorithms with provably good worst case performance, these algorithms can certainly not compete with methods, like the algorithm of Ford and Fulkerson, which solely work on the underlying static network. On the other hand, such methods are only known for restricted problems with one single commodity. For more general problems, like multicommodity flows over time, it is not even clear how to encode optimal solutions efficiently. Complex time dependency seems to be an inherent 'defect' of optimal solutions to these problems. Against this background, it is interesting to ask for provably good solutions featuring a reasonably simple structure.

Inspired by the work of Ford and Fulkerson on maximal s-t-flows over time, we show that static, *length-bounded* flows in the underlying static network lead to provably good multicommodity flows over time which, in addition, can be computed very efficiently. A length-bounded flow has a path decomposition where the length of each flow-carrying path is bounded. Based on such a path decomposition, we can construct a temporally repeated flow over time. Moreover, if we start with a maximal length-bounded flow, the resulting flow over time needs at most twice as long as a quickest flow over time. If one allows a $(1+\varepsilon)$-violation of the length bound, a maximal length-bounded flow can be computed efficiently in polynomial time. Therefore, this approach yields a $(2+\varepsilon)$-approximation algorithm for the quickest multicommodity flow problem with costs. In this context it is interesting to remember that the problem of computing a quickest temporally repeated flow with bounded cost is strongly NP-hard [14] and therefore does not allow an FPTAS, unless P=NP. We also present pathological instances which imply a lower bound on the worst case performance of our algorithm and show that the given analysis is tight.

The paper is organized as follows. In Sect. 2 we give a precise description of the problem under consideration and state basic properties of static (length-bounded) flows and flows over time. In Sect. 3 we present a $(2+\varepsilon)$-approximation

[4] A family of approximation algorithms with simultaneous performance ratio $1+\varepsilon$ for time and cost, for any $\varepsilon > 0$.

algorithm based on length-bounded flows. In Sect. 4 we introduce time-expanded networks and discuss the interconnection between flows over time and (static) flows in time-expanded networks. Our main result on condensed time expanded networks is presented in Sect. 5. Finally, in Sect. 6 and 7 we discuss extensions of this result to the corresponding min-cost flow problems and to earliest arrival flows. In this extended abstract, we omit all details in the last two sections due to space restrictions.

2 Preliminaries

We consider routing problems on a network $\mathcal{N} = (V, A)$. Each arc $e \in A$ has an associated *transit time* or *length* τ_e and a capacity u_e. An arc e from node v to node w is sometimes also denoted (v, w); in this case, we write $\mathrm{head}(e) = w$ and $\mathrm{tail}(e) = v$. There is a set of commodities $K = \{1, \dots, k\}$ that must be routed through network $\mathcal{N}$. Each commodity is defined by a source-sink pair $(s_i, t_i) \in V \times V$, $i \in K$, and it is required to send a specified amount of flow d_i from s_i to t_i, called the demand. In the setting with costs, each arc e has associated cost coefficients $c_{e,i}$, $i \in K$, which determine the per unit cost for sending flow of commodity i through the arc.

2.1 Static Flows

A *static (multicommodity) flow* x on $\mathcal{N}$ assigns every arc-commodity pair (e, i) a non-negative flow value $x_{e,i}$ such that *flow conservation constraints*

$$\sum_{e\in\delta^-(v)} x_{e,i} \;-\; \sum_{e\in\delta^+(v)} x_{e,i} \;=\; 0 \qquad \text{for all } v \in V \setminus \{s_i, t_i\},$$

are obeyed for any commodity $i \in K$. Here, $\delta^+(v)$ and $\delta^-(v)$ denote the set of arcs e leaving node v ($\mathrm{tail}(e) = v$) and entering node v ($\mathrm{head}(e) = v$), respectively. The static flow x *satisfies the multicommodity demands* if

$$\sum_{e\in\delta^-(t_i)} x_{e,i} \;-\; \sum_{e\in\delta^+(t_i)} x_{e,i} \;\geq\; d_i \ ,$$

for any commodity $i \in K$. Moreover, x is called *feasible* if it obeys the *capacity constraints* $x_e \leq u_e$, for all $e \in A$, where $x_e := \sum_{i\in K} x_{e,i}$ is the total flow on arc e. In the setting with costs, the cost of a static flow x is defined as

$$c(x) \;:=\; \sum_{e\in A}\sum_{i\in K} c_{e,i}x_{e,i} \ . \tag{1}$$

2.2 Flows Over Time

In many applications of flow problems, static routing of flow as discussed in Sect. 2.1 does not satisfactorily capture the real structure of the problem since not only the amount of flow to be transmitted but also the time needed for the transmission plays an essential role.

A *(multicommodity) flow over time* f on $\mathcal{N}$ with time horizon T is given by a collection of Lebesgue-measurable functions $f_{e,i} : [0,T) \to \mathbb{R}^+$ where $f_{e,i}(\theta)$ determines the rate of flow (per time unit) of commodity i entering arc e at time θ. Transit times are fixed throughout, so that flow on arc e progresses at a uniform rate. In particular, the flow $f_{e,i}(\theta)$ of commodity i entering arc e at time θ arrives at head(e) at time $\theta + \tau_e$. Thus, in order to obey the time horizon T, we require that $f_{e,i}(\theta) = 0$ for $\theta \in [T - \tau_e, T)$.

In our model, we allow intermediate storage of flow at nodes. This corresponds to holding inventory at a node before sending it onward. Thus, the flow conservation constraints are integrated over time to prohibit deficit at any node:

$$\sum_{e\in\delta^-(v)} \int_{\tau_e}^{\xi} f_{e,i}(\theta - \tau_e)\ d\theta \ - \sum_{e\in\delta^+(v)} \int_0^{\xi} f_{e,i}(\theta)\ d\theta \ \geq\ 0\ , \tag{2}$$

for all $\xi \in [0,T)$, $v \in V \setminus \{s_i\}$, and $i \in K$. Moreover, we require that equality holds in (2) for $\xi = T$ and $v \in V \setminus \{s_i, t_i\}$, meaning that no flow should remain in the network after time T.

The flow over time f satisfies the multicommodity demands if

$$\sum_{e\in\delta^-(t_i)} \int_{\tau_e}^{T} f_{e,i}(\theta - \tau_e)\ d\theta \ - \sum_{e\in\delta^+(t_i)} \int_0^{T} f_{e,i}(\theta)\ d\theta \ \geq\ d_i\ , \tag{3}$$

for any commodity i. Moreover, f is called feasible if it obeys the capacity constraints. 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. Thus, the capacity constraints are $f_e(\theta) \leq u_e$, for all $\theta \in [0,T)$ and $e \in A$, where $f_e(\theta) := \sum_{i\in K} f_{e,i}(\theta)$ is the total flow into arc e at time θ.

For flows over time, a natural objective is to minimize the *makespan*: the time T necessary to satisfy all demands. In the *quickest (multicommodity) flow problem*, we are looking for a feasible flow over time with minimal time horizon T that satisfies the multicommodity demands.

In the setting with costs, the cost of a flow over time f is defined as

$$c(f) \ := \ \sum_{e\in A} \sum_{i\in K} c_{e,i} \int_0^T f_{e,i}(\theta)\ d\theta\ . \tag{4}$$

The *quickest (multicommodity) flow problem with costs* is to find a feasible flow over time f with minimal time horizon T that satisfies the multicommodity demands and whose cost is bounded from above by a given *budget* B. A natural variant of this problem is to bound the cost for every single commodity i by a budget B_i, that is, $\sum_{e\in A} c_{e,i} \int_0^T f_{e,i}(\theta)\ d\theta \leq B_i$, for all $i \in K$. All of our results on the quickest multicommodity flow problem with costs work in this setting also.

2.3 Length-Bounded Static Flows

While static flows are not defined with reference to transit times, we are interested in static flows that suggest reasonable routes with respect to transit times. To account for this, we consider decompositions of static flows into paths.

It is well known that in a static flow x the flow $(x_{e,i})_{e\in A}$ of any commodity $i \in K$ can be decomposed into the sum of flows on a set of s_i-t_i-paths and flows on cycles. We denote the set of all s_i-t_i-paths by $\mathcal{P}_i$ and the flow value of commodity i on path $P \in \mathcal{P}_i$ is denoted by $x_{P,i}$. Then, for each arc $e \in A$,

$$x_{e,i} \;=\; \sum_{\substack{P\in\mathcal{P}_i:\\ e\in P}} x_{P,i} \;.$$

We assume without loss of generality that there are no cycles in the flow decomposition; otherwise, the solution x can be modified by decreasing flow on those cycles.

The static flow x is called *T-length-bounded* if the flow of every commodity $i \in K$ can be decomposed into the sum of flows on s_i-t_i-paths such that the length $\tau(P) := \sum_{e\in P} \tau_e$ of any path $P \in \mathcal{P}_i$ with $x_{P,i} > 0$ is at most T.

While the problem of computing a feasible static flow that satisfies the multicommodity demands can be solved efficiently, it is NP-hard to find such a flow which is in addition T-length-bounded, even for the special case of a single commodity. This follows by a straightforward reduction from the NP-complete problem PARTITION. On the other hand, the length-bounded flow problem can be approximated within arbitrary precision in polynomial time. More precisely, if there exists a feasible T-length-bounded static flow x which satisfies the multicommodity demands, then, for any $\varepsilon > 0$, we can compute a feasible $(1+\varepsilon)T$-length-bounded static flow x' of cost $c(x') \le c(x)$ satisfying all demands with running time polynomial in the input size and $1/\varepsilon$.

In order to prove this, we first formulate the problem of finding a feasible T-length-bounded static flow as a linear program in path-variables. Let

$$\mathcal{P}_i^T \;:=\; \{P \in \mathcal{P}_i \mid \tau(P) \le T\}$$

be the set of all s_i-t_i-paths whose lengths are bounded from above by T. The cost of path $P \in \mathcal{P}_i$ is defined as $c_i(P) := \sum_{e\in P} c_{e,i}$. The length bounded flow problem can then be written as:

$$\begin{aligned}
\min \quad & \sum_{i\in K}\sum_{P\in\mathcal{P}_i^T} c_i(P)\,x_{P,i} & & \\
\text{s. t.} \quad & \sum_{P\in\mathcal{P}_i^T} x_{P,i} \;\ge\; d_i & & \text{for all } i \in K,\\
& \sum_{i\in K}\sum_{\substack{P\in\mathcal{P}_i^T:\\ e\in P}} x_{P,i} \;\le\; u_e & & \text{for all } e \in A,\\
& x_{P,i} \;\ge\; 0 & & \text{for all } i \in K,\ P \in \mathcal{P}_i^T.
\end{aligned}$$

Unfortunately, the number of paths in $\mathcal{P}_i^T$ and thus the number of variables in this linear program are in general exponential in the size of the underlying network $\mathcal{N}$. If we dualize the program we get:

$$\begin{array}{lll} \max & \sum_{i\in K} d_i z_i \; - \; \sum_{e\in A} u_e y_e & \\ \text{s. t.} & \sum_{e\in P} (y_e + c_{e,i}) \;\geq\; z_i & \text{for all } i \in K,\, P \in \mathcal{P}_i^T, \\ & z_i, y_e \;\geq\; 0 & \text{for all } i \in K,\, e \in A. \end{array}$$

The corresponding separation problem can be formulated as a length-bounded shortest path problem: Find a shortest s_i-t_i-path P with respect to the arc weights $y_e + c_{e,i}$ whose length $\tau(P)$ is at most T, that is, $P \in \mathcal{P}_i^T$. While this problem is NP-hard [10], it can be solved approximately in the following sense: For any $\varepsilon > 0$, one can find in time polynomial in the size of the network $\mathcal{N}$ and $1/\varepsilon$ an s_i-t_i-path P with $\tau(P) \leq (1+\varepsilon)T$ whose length with respect to the arc weights $y_e + c_{e,i}$ is bounded from above by the length of a shortest path in $\mathcal{P}_i^T$ [11, 15, 17]. Using the equivalence of optimization and separation [9], this means for our problem that we can find in polynomial time an optimal solution to a modified dual program which contains additional constraints corresponding to paths of length at most $(1+\varepsilon)T$. From this dual solution we get a primal solution which uses additional variables corresponding to those paths of length at most $(1+\varepsilon)T$.

Notice that the method described above relies on the ellipsoid method and is therefore of rather restricted relevance for solving length-bounded flow problems in practice. However, the FPTASes developed in [8, 3] for multicommodity flow problems can be generalized to the case of length-bounded flows: Those algorithms iteratively send flow on shortest paths with respect to some length function. In order to get a length-bounded solution, these shortest paths must be replaced by (up to a factor of $(1+\varepsilon)$) length-bounded shortest paths.

3 A Simple $(2+\varepsilon)$-Approximation Algorithm

In this section we generalize the basic approach of Ford and Fulkerson [5, 6] to the case of multiple commodities and costs. However, in contrast to the algorithm of Ford and Fulkerson which is based on a (static) min-cost flow computation, the method we propose employs length-bounded static flows.

Any feasible flow over time f with time horizon T and cost at most B naturally induces a feasible static flow x on the underlying network $\mathcal{N}$ by averaging the flow on every arc over time, that is,

$$x_{e,i} \;:=\; \frac{1}{T}\int_0^T f_{e,i}(\theta)\; d\theta$$

for all $e \in A$ and $i \in K$. By construction, the static flow x is feasible and it satisfies the following three properties:

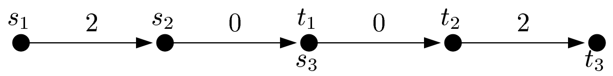

Fig. 1. An instance of the quickest multicommodity flow problem containing three commodities; commodities 1 and 3 have demand value 1, commodity 2 has demand value 2. The numbers at the arcs indicate the transit times; all arcs have unit capacity. A quickest flow with waiting at intermediate nodes allowed takes 3 time units and stores one unit of commodity 2 at the intermediate node $t_1 = s_3$ for two time units. However, if flow cannot be stored at intermediate nodes, an optimal solution takes time 4.

(i) it is T-length-bounded;
(ii) it satisfies a fraction of $\frac{1}{T}$ of the demands covered by the flow over time f;
(iii) $c(x) = c(f)/T$.

Due to the fixed time horizon T, flow can only travel on paths of length at most T in f such that property (i) is fulfilled. Property (ii) follows from (3). Finally, property (iii) is a consequence of (1) and (4).

On the other hand, given an arbitrary feasible static flow x meeting requirements (i),(ii), and (iii), it can easily be turned into a feasible flow over time g meeting the same demands at the same cost as f within time horizon $2T$: Pump flow into every s_i-t_i-path P given by the length-bounded path decomposition of x at the corresponding flow rate $x_{P,i}$ for T time units; then wait for at most T additional time units until all flow has arrived at its destination. In particular, no flow is stored at intermediate nodes in this solution.

Lemma 1. *Allowing the storage of flow at intermediate nodes in $\mathcal{N}$ saves at most a factor of 2 in the optimal makespan. On the other hand, there are instances where the optimal makespan without intermediate storage is 4/3 times the optimal makespan with intermediate storage.*

Proof. The bound of 2 follows from the discussion above. In Fig. 1 we give an instance with a gap of 4/3 between the optimal makespan without storing and the optimal makespan with storing at intermediate nodes. □

Notice that the gap of 4/3 is not an artifact of the small numbers in the instance depicted in Fig. 1. It holds for more general demands and transit times as well: For instance, scale all transit times and capacities of arcs by a factor of q and multiply all pairwise demands by a factor of q^2. The ratio of optimal makespans for the problems without to with storage is still 4/3.

In contrast to Ford and Fulkerson's temporally repeated flows, the flows over time resulting from length-bounded static flows described above, do not necessarily use flow-carrying paths as long as possible. However, we can easily enforce this property by scaling the flow rate $x_{P,i}$ on any path P by a factor $T/(2T - \tau(P)) \leq 1$ and sending flow into the path at this modified rate during the time interval $[0, 2T - \tau(P))$.

We can now state the main result of this section.

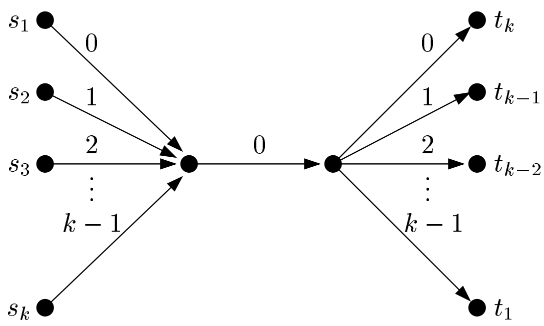

Fig. 2. An instance with k commodities showing that the analysis in the proof of Theorem 1 is tight. All arcs have unit capacity and transit times as depicted above. The demand value of every commodity is 1. A quickest flow needs $T^* = k$ time units. However, any static flow can satisfy at most a fraction of $1/k$ of the demands. In particular, the makespan of the resulting flow over time is at least $2k - 1$.

Theorem 1. *For the quickest multicommodity flow problem with costs, there exists a polynomial time algorithm that, for any $\varepsilon > 0$, finds a solution of the same cost as optimal with makespan at most $2 + \varepsilon$ times the optimal makespan.*

Proof. Using binary search, we can guess the optimal makespan with precision $1+\varepsilon/4$, that is, we get T with $T^* \leq T \leq (1+\varepsilon/4)T^*$. If we relax property (i) to allow flow on paths of length at most $(1+\varepsilon/4)T \leq (1+3\varepsilon/4)T^*$, a feasible static flow meeting properties (i) to (iii) can be computed in polynomial time; see Sect. 2.3. This static flow can then be turned into a flow over time with makespan $(1+3\varepsilon/4)T^* + (1+\varepsilon/4)T^* = (2+\varepsilon)T^*$ as described above. □

In Fig. 2 we present an instance which shows that the analysis in the proof of Theorem 1 is tight, that is, the performance guarantee of the discussed approximation algorithm is not better than 2.

4 Flows Over Time and Time-Expanded Networks

Traditionally, flows over time are solved in a time-expanded network. Given a network $\mathcal{N} = (V, A)$ with integral transit times on the arcs and an integral time horizon T, the *T-time-expanded network* of $\mathcal{N}$, denoted $\mathcal{N}^T$ is obtained by creating T copies of V, labeled V_0 through V_{T-1}, with the θ^{th} copy of node v denoted v_θ, $\theta = 0, \dots, T-1$. For every arc $e = (v, w)$ in A and $0 \leq \theta < T - \tau_e$, there is an arc e_θ from v_θ to $w_{\theta+\tau_e}$ with the same capacity and cost as arc e. In addition, there is a *holdover arc* from v_θ to $v_{\theta+1}$, for all $v \in V$ and $0 \leq \theta < T-1$, which models the possibility to hold flow at node v.

Any flow in this time-expanded network may be taken by a flow over time of equal cost: interpret the flow on arc e_θ as the flow through arc $e = (v, w)$ that starts at node v in the time interval $[\theta, \theta+1)$. Similarly, any flow over time completing by time T corresponds to a flow in $\mathcal{N}^T$ of the same value and cost obtained by mapping the total flow starting on e in time interval $[\theta, \theta+1)$ to flow on arc e_θ. More details can be found below in Lemma 2 (set $\Delta := 1$). Thus, we may solve any flow-over-time problem by solving the corresponding static flow problem in the time-expanded graph.

One problem with this approach is that the size of $\mathcal{N}^T$ depends linearly on T, so that if T is not bounded by a polynomial in the input size, this is not a polynomial-time method of obtaining the required flow over time. However, if all arc lengths are a multiple of $\Delta > 0$ such that $\lceil T/\Delta \rceil$ is bounded by a polynomial in the input size, then instead of using the T-time-expanded graph, we may rescale time and use a condensed time-expanded network that contains only $\lceil T/\Delta \rceil$ copies of V. Since in this setting every arc corresponds to a time interval of length Δ, capacities are multiplied by Δ. We denote this condensed time-expanded network by $\mathcal{N}^T/\Delta$, and the copies of V in this network by $V_{\rho\Delta}$ for $\rho = 0, \ldots, \lceil T/\Delta \rceil - 1$.

Lemma 2. *Suppose that all arc lengths are multiples of Δ and T/Δ is an integer. Then, any flow over time that completes by time T corresponds to a static flow of equal cost in $\mathcal{N}^T/\Delta$, and any flow in $\mathcal{N}^T/\Delta$ corresponds to a flow over time of equal cost that completes by time T.*

Proof. Given an arbitrary flow over time, a modified flow over time of equal value and cost can be obtained by averaging the flow value on any arc in each time interval $[\rho\Delta, (\rho+1)\Delta)$, $\rho = 0, \ldots, T/\Delta - 1$. This modified flow over time defines a static flow in $\mathcal{N}^T/\Delta$ in a canonical way. Notice that the capacity constraints are obeyed since the total flow starting on arc e in interval $[\rho\Delta, (\rho+1)\Delta)$ is bounded by Δu_e. The flow values on the holdover arcs are defined in such a way that flow conservation is obeyed in every node of $\mathcal{N}^T/\Delta$.

On the other hand, a static flow on $\mathcal{N}^T/\Delta$ can easily be turned into a flow over time. The static flow on an arc with tail in $V_{\rho\Delta}$ is divided by Δ and sent for Δ time units starting at time $\rho\Delta$. If the head of the arc is in $V_{\sigma\Delta}$ for $\sigma \geq \rho$, then the length of the arc is $(\sigma-\rho)\Delta$, and the last flow (sent before time $(\rho+1)\Delta$) arrives before time $(\sigma+1)\Delta$. Note that if costs are assigned to arcs of $\mathcal{N}^T/\Delta$ in the natural way, then the cost of the flow over time is the same as the cost of the corresponding flow in the time-expanded graph. □

If we drop the condition that T/Δ is integral, we get the following slightly weaker result.

Corollary 1. *Suppose that all arc lengths are multiples of Δ. Then, any flow over time that completes by time T corresponds to a static flow of equal value and cost in $\mathcal{N}^T/\Delta$, and any flow in $\mathcal{N}^T/\Delta$ corresponds to a flow over time of equal value that completes by time $T + \Delta$.*

5 An FPTAS for Multicommodity Flow Over Time

Our FPTAS for flow over time will use a graph $\mathcal{N}^T/\Delta$ for an appropriately defined Δ. We show below that, even when all arc lengths are not multiples of Δ, for an appropriate choice of Δ that depends on ε we may round the lengths up to the nearest multiple of Δ, and suffer only a $1+\varepsilon$ factor increase in the makespan of our flow. Thus, our algorithm is simply to first round the arc lengths, construct the corresponding condensed time-expanded network $\mathcal{N}^T/\Delta$, solve the flow problem in this time-expanded graph, and then translate this solution into a flow over time. We show below that this natural algorithm yields an FPTAS for minimizing the makespan of flow over time problems.

In the last step of the sketched algorithm, we make use of the following straightforward observation which will also be employed at several points during the analysis of the algorithm.

Observation 1. *Any flow over time with time horizon T in a network with elongated arc lengths induces a flow over time satisfying the same demands within the same time horizon in the original network.*

5.1 Increasing the Transit Times

Our analysis starts with an optimal flow that completes by time T^*, and then shows how to modify this flow so that it completes by time $(1+\varepsilon)T^*$ in a network with elongated arc lengths. Throughout the proof we often use the following *freezing technique* for modifying a given flow over time in a fixed network $\mathcal{N}$: At some point in time θ, we '*freeze*' the flow in progress and '*unfreeze*' it later at time $\theta+\delta$, thereby increasing the completion time of the flow by at most δ. More formally, freezing a flow in progress during a time interval $[\theta, \theta+\delta)$ means that no new flow is sent onto any arc during this interval; instead, at time θ, every unit of flow which is currently traveling on some arc continues on its path at its regular pace until it reaches the next node and then rests there for δ time units before it continues its journey. The motivation for introducing such a freezing period is that it provides free capacity which can be used to send additional flow on an arc during the time interval $[\theta, \theta+\delta)$.

Let f^* be an optimal flow over time for network $\mathcal{N}$ with m arcs, n nodes, and with commodities $1 \le i \le k$ associated with source-sink node pairs (s_i, t_i) and flow demands d_i. Let T^* be the completion time of this flow. The following theorem contains the key result for the construction of our FPTAS.

Theorem 2. *Let $\varepsilon > 0$ and $\Delta \le \frac{\varepsilon^2}{4k^2m^4}T^*$. Increasing the transit time of every arc by an additive factor of at most Δ increases the minimum time required to satisfy the multicommodity demands by at most εT^*.*

The rough idea of the proof is that, for each elongated arc e, its original transit time can be emulated by holding ready additional units of flow in a buffer at the head of the arc. Since the required amount of flow in this buffer depends on the total flow that is traveling on the arc at any moment of time, we first show that we can bound the maximal rate of flow into an arc, without too much increase in the makespan.

For an arbitrary flow over time f, consider the flow on arc e of commodity i. This flow travels from s_i to t_i on a set of paths $\mathcal{P}_{e,i}(f)$, all containing arc e. For

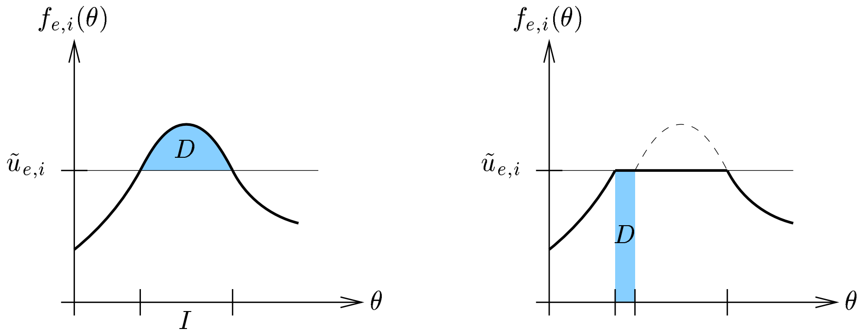

Fig. 3. Modification of the flow over time f: The flow of commodity i on arc e is truncated at $\tilde{u}_{e,i}$. In the modified solution, the extra D flow units (shaded area on the left hand side) are sent onto arc e while the flow in the remaining network is frozen (shaded area on the right hand side).

any path P, let $u(P)$ denote the minimum capacity of an arc $e' \in P$ and $|P|$ the number of arcs on P. Define $u_{e,i}(f) := \max_{P \in \mathcal{P}_{e,i}(f)} u(P)$.

Lemma 3. *For any $\varepsilon > 0$, there is a flow over time f in the network $\mathcal{N}$ that fulfills the following properties:*

(i) *It satisfies all demands by time $(1 + \varepsilon/2)T^*$.*
(ii) *For any arc e and any commodity i, the rate of flow onto arc e of commodity i is bounded by*

$$\tilde{u}_{e,i}(f) \;:=\; \frac{2km^2}{\varepsilon}\, u_{e,i}(f)$$

at any time.

Proof. The optimal flow over time f^* obviously satisfies property (i). We start with the flow $f := f^*$ and carefully modify it until it also satisfies (ii). During this modification we only delay but never reroute flow such that $u_{e,i}(f)$ and $\tilde{u}_{e,i}(f)$ stay fixed and are therefore denoted by $u_{e,i}$ and $\tilde{u}_{e,i}$, respectively.

An illustration of the following modification is given in Fig. 3. In any time interval[5] I when f sends more than $\tilde{u}_{e,i}$ units of flow of commodity i onto arc e, we truncate the flow at $\tilde{u}_{e,i}$. In order to compensate for the resulting loss of

$$D \;:=\; \int_I (f_{e,i}(\theta) - \tilde{u}_{e,i})\; d\theta$$

flow units, we freeze the flow in $\mathcal{N}$ for $D/\tilde{u}_{e,i}$ time units at the beginning of time interval I and push D units of flow onto arc e (see the shaded area on the

[5] It follows from the discussion in Sect. 4 (see Lemma 2) that there are only finitely many such intervals since we can assume without loss of generality that $f^*_{e,i}$ is a step function.

right hand side of Fig. 3). Afterwards, the flow is unfrozen again. Due to this freeze-unfreeze process, the D flow units arrive early (compared to the remaining flow) at the head of arc e and are stored there until it is time to send them onto another arc.

We repeat this freeze-unfreeze step for every commodity-arc pair. By construction, the resulting flow f fulfills property (ii) and satisfies all demands. It thus remains to show that f completes before time $(1+\varepsilon/2)T^*$. Notice that the increase in the completion time of f compared to f^* is exactly the additional time added to the total flow schedule when the flow gets frozen due to some arc e and some commodity i. In the following we show that the total freezing time caused by arc e and commodity i is at most $\frac{\varepsilon}{2mk}T^*$, for any e and i.

Whenever the flow is frozen due to arc e and commodity i, the flow rate of commodity i onto arc e is exactly $\tilde{u}_{e,i}$; see Fig. 3. It therefore suffices to show that a total of at most $\tilde{u}_{e,i}\frac{\varepsilon}{2mk}T^* = mu_{e,i}T^*$ flow of commodity i is sent through arc e in the optimal solution f^* and thus also in f. Consider all flow of commodity i on paths in $\mathcal{P}_{e,i}$. If we choose for each path $P \in P_{e,i}$ an arc with capacity $u(P) \leq u_{e,i}$, the total flow through the chosen arcs bounds the total flow of commodity i through arc e. Since we can choose at most m arcs and the makespan of f^* is T^*, this gives the desired bound of $mu_{e,i}T^*$ on the total flow of commodity i sent through arc e. This completes the proof. □

With the aid of Lemma 3 we can now prove Theorem 2.

Proof (of Theorem 2). Let f be a flow over time in network $\mathcal{N}$ (with original transit times). We start by modifying f as described in the proof of Lemma 3 so that it fulfills properties (i) and (ii). Let $\mathcal{N}_\Delta$ be $\mathcal{N}$ modified so that the transit time of every arc is rounded up to an integral multiple of Δ. We show how the flow over time f can be modified to satisfy all demands in $\mathcal{N}_\Delta$ by time $(1+\varepsilon)T^*$. Although some flow units will be rerouted during this modification, the set of paths $\mathcal{P}_{e,i} := \mathcal{P}_{e,i}(f)$ remains unchanged, for any arc e and any commodity i. In particular, $u_{e,i}(f)$ and $\tilde{u}_{e,i}(f)$ stay fixed and are therefore denoted by $u_{e,i}$ and $\tilde{u}_{e,i}$, respectively.

The modification is done in m steps such that in each step the transit time of only one arc e is increased at the cost of increasing the makespan by at most $\frac{\varepsilon}{2m}T^*$. Thus, the total increase of the makespan of f after m steps is at most $\frac{\varepsilon}{2}T^*$. Together with the prior modifications discussed in Lemma 3, this implies that the resulting flow completes by time $(1+\varepsilon)T^*$.

Each step has two phases. In the first phase, the transit time of arc e remains unchanged but the demand satisfied by f is increased to $d_i + \Delta\tilde{u}_{e,i}$, for all commodities i. Then, in the second phase, the extra $\Delta\tilde{u}_{e,i}$ units of flow from the first phase are used to emulate the original transit time τ_e on the elongated arc e of length $\tau_e + \Delta$. (It follows from Observation 1 that it suffices to consider the extreme case of increasing the transit time by exactly Δ).

Phase 1: For each commodity i, let $P_{e,i} \in \mathcal{P}_{e,i}$ be an s_i-t_i-path with capacity $u_{e,i}$. The additional $\Delta\tilde{u}_{e,i}$ units of flow are routed through path $P_{e,i}$: At time 0, we freeze the current flow throughout the network for $\Delta\frac{2km^2}{\varepsilon}$ time units and pump an extra $\Delta\frac{2km^2}{\varepsilon}u_{e,i} = \Delta\tilde{u}_{e,i}$ units of flow of commodity i into the first arc on this path. When this flow arrives at the next arc on path $P_{e,i}$, we again freeze the current flow for $\Delta\frac{2km^2}{\varepsilon}$ time units and send this flow onto the next arc, and so on.

Notice that the extra flow does not violate the capacity constraints since the capacity of any arc on path $P_{e,i}$ is at least $u_{e,i}$. Moreover, the extra units of flow arrive at their destination t_i before time

$$\tau(P_{e,i}) + \Delta\frac{2km^2}{\varepsilon} \;\le\; T^* + \Delta\frac{2km^2}{\varepsilon} \enspace .$$

We add this flow to f. The makespan of f is increased by

$$|P_{e,i}|\,\Delta\frac{2km^2}{\varepsilon} \;\le\; \Delta\frac{2km^3}{\varepsilon} \;\le\; \frac{\varepsilon}{2km}T^* \enspace .$$

Repeating this for all commodities i increases the makespan by at most $\frac{\varepsilon}{2m}T^*$.

Phase 2: Now, we increase the length of arc e to $\tau_e + \Delta$ and modify the current flow over time f as follows. The point on arc e that is distance τ_e from tail(e) is called apex(e). For any commodity i, the first $\Delta\tilde{u}_{e,i}$ units of flow of commodity i traveling across arc e are stored in a special buffer as soon as they arrive at head(e). The flow stored in the buffer at head(e) can then be used to emulate the original transit time on arc e for any further unit of flow of commodity i.

An illustration of the following argument is given in Fig. 4. Any further unit of flow of commodity i arriving at apex(e) is instantly 'replaced' by a unit of flow from the buffer. Then, after Δ time units, when the flow has traveled from apex(e) to head(e), the buffer is refilled again. In other words, the flow in the buffer is treated as newly arriving flow at head(e), and the flow reaching the head enters the buffer. Thus, due to property (ii) from Lemma 3, the choice of buffer size $\Delta\tilde{u}_{e,i}$ ensures that the buffer is never empty.

Notice that this modification of f does not increase its makespan. The result of the modification is that exactly d_i units of flow of commodity i arrive at destination t_i and $\Delta\tilde{u}_{e,i}$ units of flow remain in the buffer at the head of arc e in the end. Since the latter effect is undesired, we revoke it as follows: The $\Delta\tilde{u}_{e,i}$ units of flow in the buffer are exactly those that arrive last at head(e). We can simply delete them from the entire solution, that is, we never send them from s_i into the network. □

5.2 The FPTAS

As a consequence of Theorem 2 we can now give an FPTAS for the problem of computing a quickest multicommodity flow over time:

Input: A directed network $\mathcal{N}$ with non-negative integral transit times and capacities on the arcs, and commodities $1 \le i \le k$ associated with source-sink node pairs (s_i, t_i) and flow demands d_i; a number $\varepsilon > 0$.

Output: A multicommodity flow over time satisfying all demands.

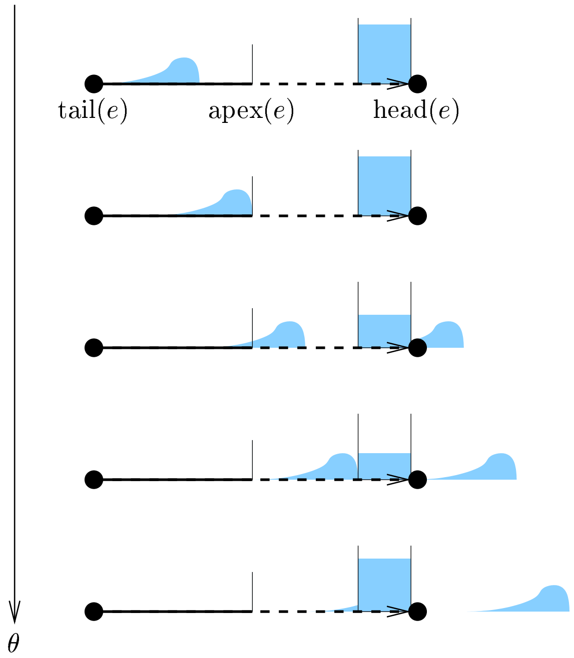

Fig. 4. In Phase 2, the flow in the buffer at the head of the elongated arc e is used to emulate its original length τ_e.

Step 1. Compute a good lower bound L on the optimal makespan T^* such that L is at least a constant fraction of T^*.

Step 2. Set $\Delta := \frac{\varepsilon^2}{4m^4k^2}L$ and round the transit times up to the nearest multiple of Δ.

Step 3. Find the minimal $T = \ell\Delta$, $\ell \in \mathbb{N}$, such that there exists a feasible flow satisfying all demands in the condensed time-expanded network $\mathcal{N}^T/\Delta$.

Step 4. Output the flow over time that corresponds to the flow in $\mathcal{N}^T/\Delta$ from Step 3 (see Lemma 2 and Observation 1).

It follows from Theorem 2 and Corollary 1 that the above algorithm computes a solution with makespan $T \leq (1+\varepsilon)T^* + \Delta \leq (1+2\varepsilon)T^*$. Moreover, it can be implemented to run in time polynomial in n, m, and $1/\varepsilon$: Step 1 can be done in polynomial time using the constant factor approximation algorithm from Sect. 3 (see Theorem 1). Step 2 is trivial and Step 3 can be done in polynomial time by binary search since $\ell \in O(k^2m^4/\varepsilon^2)$ by Theorem 2 and choice of Δ. Here, Δ is chosen so that the size of the condensed time-expanded network $\mathcal{N}^T/\Delta$ is polynomial. Thus, Step 4 can also be done in polynomial time.

Theorem 3. *There is an FPTAS for the problem of computing a quickest multicommodity flow over time.*

6 Problems with Costs

Unfortunately, Theorem 3 cannot directly be generalized to the quickest multicommodity flow problem with costs. The reason is that in our analysis we have to reroute flow in order to show that there exists a reasonably good solution in the condensed time-expanded network $\mathcal{N}^T/\Delta$ (see proof of Theorem 2). Since the thick paths $P_{e,i}$, which are used to route additional units of flow, might be relatively expensive, the modified flow in the network with increased transit times can possibly violate the given budget B. However, we can prove the following bicriteria result. We omit the proof in this extended abstract.

Theorem 4. *Given an instance of the quickest multicommodity flow problem with costs and $\varepsilon > 0$, one can compute in time polynomial in the input size and $1/\varepsilon$ a flow over time of cost at most $(1+\varepsilon)B$ whose makespan is within a factor of $1+\varepsilon$ of the optimal makespan.*

7 Earliest Arrival Flows

For a single commodity problem, an *earliest arrival flow* is a flow that simultaneously maximizes the amount of flow arriving at the sink before time θ, for all $\theta = 0, \dots, T$. The existence of such a flow was first observed by Gale [7] for the case of a single source. Both Wilkinson [19] and Minieka [16] give equivalent pseudo-polynomial time algorithms for this case, and Hoppe and Tardos [12] describe an FPTAS for the problem. For multiple sources, an earliest arrival flow over time can be computed in the discrete time model by using lexicographically maximal flows in the time-expanded network [16]. However, due to the exponential size of the time-expanded network, this does not lead to an efficient algorithm for the problem.

Unfortunately, lexicographically maximal flows in *condensed* time-expanded networks do not necessarily yield approximate earliest arrival flows. One problem is that, in our analysis, the first units of flow on an arc are always used to fill the buffer at the head of the arc and are therefore 'lost'. As a consequence, the first units of flow that actually arrive at the sink might be pretty late.

Another problem arises due to the discretization of time itself. Although we can interpret a static flow in a time-expanded network as a continuous flow over time, in doing so, we only get solutions where the rate of flow arriving at the sink is constant (i. e., averaged) within each discrete time interval. While this effect is negligible for late intervals in time, it might well cause problems within the first time intervals. In the full version of this paper, we introduce a *geometrically-condensed time-expanded network* to surmount this difficulty and obtain the following result.

Theorem 5. *For any $\varepsilon > 0$, a $(1+\varepsilon)$-approximate earliest arrival flow in a network with multiple sources and a single sink can be computed in time polynomial in the input size and $1/\varepsilon$ by computing a lexicographically maximal flow in an appropriate geometrically condensed time-expanded network.*

References

1. J. E. Aronson. A survey of dynamic network flows. *Annals of Operations Research*, 20:1–66, 1989.
2. R. E. Burkard, K. Dlaska, and B. Klinz. The quickest flow problem. *ZOR — Methods and Models of Operations Research*, 37:31–58, 1993.
3. L. K. Fleischer. Approximating fractional multicommodity flows independent of the number of commodities. *SIAM Journal on Discrete Mathematics*, 13:505–520, 2000.
4. L. K. Fleischer and É Tardos. Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, 23:71–80, 1998.
5. L. R. Ford and D. R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6:419–433, 1958.
6. L. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, Princeton, NJ, 1962.
7. D. Gale. Transient flows in networks. *Michigan Mathematical Journal*, 6:59–63, 1959.
8. N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 300–309, Palo Alto, CA, 1998.
9. M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics.* Springer, Berlin, 1988.
10. G. Handler and I. Zang. A dual algorithm for the constrained shortest path problem. *Networks*, 10:293–310, 1980.
11. R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17:36–42, 1992.
12. B. Hoppe and É Tardos. Polynomial time algorithms for some evacuation problems. In *Proceedings of the 5th Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 433–441, Arlington, VA, 1994.
13. B. Hoppe and É Tardos. The quickest transshipment problem. *Mathematics of Operations Research*, 25:36–62, 2000.
14. B. Klinz and G. J. Woeginger. Minimum cost dynamic flows: The series-parallel case. In E. Balas and J. Clausen, editors, *Integer Programming and Combinatorial Optimization*, volume 920 of *Lecture Notes in Computer Science*, pages 329–343. Springer, Berlin, 1995.
15. D. H. Lorenz and D. Raz. A simple efficient approximation scheme for the restricted shortest path problem. *Operations Research Letters*, 28:213–219, 2001.
16. E. Minieka. Maximal, lexicographic, and dynamic network flows. *Operations Research*, 21:517–527, 1973.
17. C. A. Phillips. The network inhibition problem. In *Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pages 776–785, San Diego, CA, 1993.
18. W. B. Powell, P. Jaillet, and A. Odoni. Stochastic and dynamic networks and routing. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, chapter 3, pages 141–295. North–Holland, Amsterdam, The Netherlands, 1995.
19. W. L. Wilkinson. An algorithm for universal maximal dynamic flows in a network. *Operations Research*, 19:1602–1612, 1971.