IBM Research Report

Quickest Flows Over Time

Lisa K. Fleischer
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Martin Skutella
Technische Universit at Berlin,
Institut fur Mathematik, MA 6–1,
Str. des 17. Juni 136, 10623 Berlin,

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

Quickest Flows Over Time[*]

Lisa Fleischer[‡]   Martin Skutella[§]

Abstract

Flows over time (also called dynamic flows) generalize standard network flows by introducing an element of time. They naturally model problems where travel and transmission are not instantaneous. Traditionally, flows over time are solved in time-expanded networks that 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. We present several 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. Secondly, we investigate 'condensed' time-expanded networks which rely on a rougher discretization of time. 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, our approach yields fully polynomial time approximation schemes for the NP-hard quickest min-cost and multicommodity flow problems. For single commodity problems, we show that storage of flow at intermediate nodes is unnecessary; and our approximation schemes do not use any.

1   Introduction

While standard network flows are useful to model a variety of optimization problems, they fail to capture a crucial element of many routing problems: routing occurs over time. In their seminal paper on the subject, Ford and Fulkerson [12, 13] introduced flows with transit times to remedy this and described a polynomial time algorithm to

---

solve the maximum flow-over-time, also called the maximum dynamic flow problem. [1] 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 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. In order to get an intuitive understanding of flows over time, one can associate arcs of the network with pipes in a pipeline system for transporting some kind of fluid. [2] The length of each pipeline determines the transit time of the corresponding arc while the width determines its capacity. A precise definition of flows over time is given later in Section 2.

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 (e.g. the Internet), and financial flows. Examples and further applications can be found in the survey articles of Aronson [2] and Powell, Jaillet, and Odoni [33]. However, flows over time are significantly harder than their standard flow counterparts. For example, both minimum cost flows over time and fractional multicommodity flows over time are NP-hard [18, 25], even for very simple series-parallel networks.

1.1 Results from the Literature

Maximum flows over time.　Ford and Fulkerson [12, 13] 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 be solved efficiently using one min-cost flow computation on the given network. 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*.

Quickest flows.　A problem closely related to the problem of computing a maximal s-t-flow over time is the *quickest s-t-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. Using Megiddo's method of parametric search [27], Burkard, Dlaska, and Klinz [3] present a faster algorithm which solves the quickest s-t-flow problem in strongly polynomial time.

---

[1] Earlier work on this topic referred to the problems as *dynamic flow* problems. Recently the term *dynamic* has been used in many algorithmic settings to refer to problems with input data that arrives online, or changes over time; and the goal of the algorithms described is to modify the current solution quickly to handle the slightly modified input. For the problem of dynamic flows, the input data is available at the start. The solution to the problem involves a describing how the optimal flow changes over time. For these reasons, we use the term "flows over time" instead of "dynamic flows" to refer to these problems.

[2] We take a purely macroscopic point of view which does not involve any fluid dynamics.

Earliest arrival flows. An *earliest arrival flow* is an s-t-flow over time which simultaneously maximizes the amount of flow arriving at the sink before time θ, for all $\theta \in [0, T)$. Gale [14] observes that these flows exist; and Wilkinson [35] and Minieka [28] give equivalent pseudo-polynomial time algorithms to find them. These algorithms are essentially using the Successive Shortest Path Algorithm. The resulting solution is also a *latest departure flow*, i.e., a flow over time which simultaneously maximizes the amount of flow departing from the source after time θ, for all $\theta \in [0, T)$ (subject to the constraint that the flow is finished by time T). A flow over time which is both an earliest arrival flow and a latest departure flow is called *universally maximal flow over time*. Hoppe and Tardos [23, 22] describe a polynomial-time approximation scheme for the universally maximal flow problem that routes a $1 - \varepsilon$ fraction of the maximum possible flow that can reach the sink t by time θ, for all $0 \leq \theta < T$. Problems with time-dependent arc capacities have been considered by Ogier [29] and Fleischer [10].

Flows over time with costs. Natural generalization of the quickest flows and maximum flows over time can be defined on networks with additional costs on the arcs. The problem can be to either find a minimum cost flow with a given time horizon, or find a quickest flow within a given cost budget. Klinz and Woeginger [25] show that the search for a quickest or a maximum s-t-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.

Orlin [30] describes a polynomial time algorithm to compute an infinite horizon, minimum cost flow over time that maximizes throughput. The infinite horizon problem does not have specified demand and is not concerned with computing how a flow starts and stops, issues that are crucial when flow demands are changing over time.

Quickest transshipments. 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. Unlike standard network flows, this multiple source, multiple sink, single commodity flow over time is not equivalent to an s-t maximum flow over time. Hoppe and Tardos describe the first polynomial time algorithm to solve this problem [24, 22]. 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. Moreover, the algorithm of Hoppe and Tardos is not practical as it requires a submodular function minimization oracle for a subroutine.

Quickest multicommodity flows over time. In many applications, there are several commodities that must be routed through the same network. While there is substantial

literature on the static multicommodity flow problem, hardly any results on multicommodity flows over time are known. Only recently, Hall, Hippler, and Skutella [18] showed that already in the setting without costs, multicommodity flows over time are NP-hard. Indeed, it is not known if there always exists an optimal solution that can be described in polynomial space.

Discrete vs. continuous time model. All results mentioned so far have originally been developed for 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 [11] 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. Since in this paper we mainly concentrate on the continuous time model, we give a more detailed discussion of the interrelation of the two models in Section 4.1.

Time-expanded networks. In the discrete time model, flows over time can be described and computed in *time-expanded networks* which were introduced by Ford and Fulkerson [12, 13]. 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. A precise description of time-expanded networks is given in Section 4.1. 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.

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, an approach which is also used in practice to solve flow over time problems. Due to the linear dependency of the size of the time-expanded network on T, such algorithms are termed 'pseudopolynomial', since the run time of the algorithm depends on T and not $\log T$. In general, the size of these networks makes the problem solution prohibitively expensive.

1.2 Contributions of this Paper

We describe approximation algorithms for flow-over-time problems. All of our algorithms approximate the minimum time horizon of an optimal flow. Thus an α-approximate solution is a flow that solves the original problem and requires at most α times the optimal time horizon to complete.

Temporally repeated solutions. Inspired by the work of Ford and Fulkerson, we show in Section 3 that static, *length-bounded* flows in the underlying static network lead to provably good multicommodity flows over time that can also be computed efficiently. The resulting approximation algorithm computes temporally repeated solutions and has performance ratio 2. For the more general problem with bounded cost this approach yields a $(2 + \varepsilon)$-approximation algorithm. 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 [25] and therefore does not allow an FPTAS, unless P=NP. The same hardness result holds for quickest multicommodity flows without intermediate node storage and simple flow paths [18]. Finally, since a temporally repeated flow does not use intermediate node storage, our result implies a bound of 2 on the 'power of intermediate node storage', that is, a quickest multicommodity flow without intermediate node storage is at most twice as long as a quickest flow which is allowed to store flow at intermediate nodes.

Approximation schemes. Another main contribution of this paper is to show that problems that can be solved exactly in the time-expanded network can be solved close to optimally by a static flow computation in a network with polynomial size. 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.

In Section 4 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 that permits a solution completing within $(1 + \varepsilon)$ times the completion of a comparable flow in the continuous-time model, any $\varepsilon > 0$. More precisely, a condensed time-expanded network achieving this precision has n/ε^2 time layers where n is the number of nodes in the given network. One can thus say that the cost of $(1 + \varepsilon)$-approximate temporal dynamics for network flow problems is a factor of n/ε^2 in the size of the network.

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 NP-hard quickest multicommodity flow problem. Since costs can easily be incorporated into time-expanded networks, our approach can be generalized to yield FPTASes for quickest multicommodity flow problems with cost constraints. Notice that already quickest s-t-flows with bounded cost are NP-hard.

Apart from NP-hard problems, 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 [24, 22] for the quickest transshipment problem relies on submodular function minimization, the use of condensed time-expanded networks leads to an FPTAS which simply consists of a series of max-flow computations.

No storage. Flows over time raise issues that do not arise in standard network flows. One issue is storage at intermediate nodes. In most applications (such as, e.g., traffic routing, evacuation planning, telecommunications), storage is limited, undesired, or even prohibited at intermediate nodes. For single commodity problems, most generally the transportation problem with costs, we prove that storage is unnecessary and our FPTAS does not use any.

Earliest arrival flows. Finally, in Section 5 we discuss a variant of time-expanded networks which are suitable for approximating earliest arrival flows. We address the following problem: given a set of sources with supplies, and a single sink, send the supplies to the sink so that the amount of flow arriving at the sink by time θ is $D_t^*(\theta)$, the maximum possible, for all $0 \leq \theta$. Instead of using a uniform discretization of time, we introduce 'geometrically condensed time-expanded networks' which rely on geometrically increasing time steps. We use this network to obtain a flow that sends $D_t^*(\theta)$ units of flow to the sink by time $\theta(1 + \varepsilon)$ for all $0 \leq \theta$.

2 Preliminaries

We consider routing problems on a network $\mathcal{N} = (V, A)$ with $n := |V|$ nodes and $m := |A|$ arcs. Each arc $e \in A$ has an associated integral *transit time* or *length* τ_e and a capacity u_e. In the setting with costs, each arc e also has a cost coefficient c_e, which determines the per unit cost for sending flow through the arc. An arc e from node v to node w is sometimes also denoted (v, w); in this case, we write $\text{head}(e) = w$ and $\text{tail}(e) = v$.

2.1 Static Flows

We start with the definition of single-commodity flows: Let $S \subseteq V$ be a set of terminals which can be partitioned into a subset of sources S^+ and sinks S^-. Every source node $v \in S^+$ has a supply $D_v \geq 0$ and every sink $v \in S^-$ has a demand $D_v \leq 0$ such that $\sum_{v \in S} D_v = 0$. We often consider the case with only one source $s \in V$ and one sink $t \in V$. In this case, we let $d := D_s = -D_t$.

A *static flow* x on $\mathcal{N}$ assigns every arc e a non-negative flow value x_e such that the *flow conservation constraints*

$$\sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = 0 \qquad \text{for all } v \in V \setminus S$$

are obeyed. Here, $\delta^+(v)$ and $\delta^-(v)$ denote the set of arcs e leaving node v ($\text{tail}(e) = v$) and entering node v ($\text{head}(e) = v$), respectively. The static flow x *satisfies the supplies and demands* if

$$\sum_{e \in \delta^+(v)} x_e - \sum_{e \in \delta^-(v)} x_e = D_v \qquad \text{for all } v \in S.$$

For the case of a single source s and a single sink t we also use the term *s-t-flow*. An s-t-flow x satisfying supply $d = D_s = -D_t$ has *value* $|x| = d$. Finally, a flow x is called *feasible* if it obeys the *capacity constraints* $x_e \leq u_e$, for all $e \in A$. The cost of a static flow x is defined as

$$c(x) := \sum_{e \in A} c_e \, x_e \ .$$

In the multiple commodity setting, there is a set of commodities $K = \{1, \ldots, k\}$, each of which is defined by a set of terminals $S_i = S_i^+ \cup S_i^- \subseteq V$ and demands and supplies $D_{v,i}$ for $v \in S_i$ and $i \in K$. 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 $x_i := (x_{e,i})_{e \in A}$ is a single-commodity flow as defined above for all $i \in K$. The multicommodity flow x satisfies the demands and supplies if x_i satisfies the demands and supplies $D_{v,i}$ for $v \in S_i$. Finally, x is called *feasible* if it obeys the *capacity constraints* $x_e := \sum_{i \in K} x_{e,i} \leq u_e$, for all $e \in A$. In the setting with costs, the cost of a static multicommodity flow x is defined as

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

where $c_{e,i}$ is the cost coefficient associated with arc e and commodity i.

2.2 Flows Over Time

In many applications of flow problems, static routing of flow as discussed in Section 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 $\mathrm{head}(e)$ 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 order to simplify notation, we sometimes use $f_{e,i}(\theta)$ for $\theta \notin [0, T)$, implicitly assuming that $f_{e,i}(\theta) = 0$ in this case.

With respect to flow conservation, there are two different models of flows over time. In the model with *storage of flow at intermediate nodes*, it is possible to hold inventory at a node which is neither a source nor a sink before sending it onward. Thus, the flow conservation constraints are integrated over time to prohibit deficit at any node:

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

for all $i \in K$, $\xi \in [0, T)$, $v \in V \setminus S_i^+$. Moreover, we require that equality holds in (2) for $i \in K$, $\xi = T$ and $v \in V \setminus S_i$, meaning that no flow should remain in the
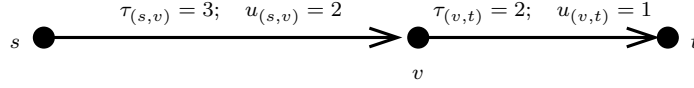
Figure 1: An instance of s-t-flows over time given by a network with transit times and capacities on the arcs.

network after time T. In the model without storage of flow at intermediate nodes we additionally require that equality holds in (2) for all $i \in K$, $\xi \in [0, T)$ and $v \in V \setminus S_i$.

The flow over time f satisfies the supplies and demands if by time T the net flow out of each terminal $v \in S_i$ of commodity i equals its supply $D_{v,i}$:

$$\int_0^T \left(\sum_{e \in \delta^+(v)} f_{e,i}(\theta) - \sum_{e \in \delta^-(v)} f_{e,i}(\theta - \tau_e) \right) d\theta = D_{v,i} \tag{3}$$

for all $i \in K$, $v \in S_i$. An s-t-*flow over time* is a single commodity flow from a single source s to a single sink t. An s-t-flow over time satisfying supply $d = D_s = -D_t$ has *value* $|f| = d$.

A flow over time f is *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 time unit. Thus, the capacity constraints are $f_e(\theta) \le u_e$, 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.

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

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

In Figure 1 we give a small illustrating example of s-t-flows over time. In order to send 2 units of flow from s to t in minimum time in the depicted network, one can choose between several alternatives. One is to send flow at rate 2 into the first arc during the time interval $[0, 1)$. Since the transit time of the first arc is 3, the two units of flow will arrive at the intermediate node v during the time interval $[3, 4)$. Thus, one can start to send flow at rate 1 into the second arc at time 3 and it will take 2 time units until time 5 before everything has been sent into the arc. Then, the flow finally arrives at the sink t within the time interval $[5, 7)$. The optimal time horizon is 7. Notice that in this solution flow is stored at the intermediate node v. An alternate solution also with time horizon 7 which avoids storing flow at v can be obtained by sending flow at rate 1 into the first arc during the time interval $[0, 2)$.

2.3 Maximum Flows Over Time and Quickest Flows

Ford and Fulkerson show how to turn an optimal solution x to the static s-t-flow problem with objective function

$$\max \quad T \, |x| - \sum_e \tau_e \, x_e \tag{5}$$

8

into a maximal s-t-flow over time: It is a well-known result from network flow theory that any static flow x in $\mathcal{N}$ can be decomposed into a sum of flows x_P on simple paths $P \in \mathcal{P}$ and flow on cycles. Without loss of generality, flow on cycles is neglected (i.e., canceled) such that x can be written as a sum of path-flows: $x_e = \sum_{P \in \mathcal{P} : e \in P} x_P$ for all $e \in A$. The resulting temporally repeated flow f sends flow at rate x_P into each path $P \in \mathcal{P}$ during the time interval $[0, T - \tau(P))$, where $\tau(P) := \sum_{e \in P} \tau_e$. In other words, f is the sum of path-flows over time f_P with $f_P(\theta) = x_P$ for $\theta \in [0, T - \tau(P))$ and $f_P(\theta) = 0$, otherwise. Feasibility of f immediately follows from feasibility of x. Moreover, the flow value is

$$|f| \;=\; \sum_{P \in \mathcal{P}} (T - \tau(P))\, x_P \;=\; T\,|x| \;-\; \sum_e \tau_e\, x_e \;. \tag{6}$$

The second equality follows since $(x_P)_{P \in \mathcal{P}}$ is a path-decomposition of x.

For flows over time, a natural objective is to minimize the *makespan*, also called *time horizon*: the time T necessary to satisfy all demands. The *quickest s-t-flow problem* asks for an s-t-flow over time with given value d and minimum time horizon T. This problem can be generalized to the setting with bounded flow cost, multiple sources and sinks (*quickest transshipment problem*), and to the case of multiple commodities. The most general problem that we consider here is the *quickest (multicommodity) transshipment problem (with bounded cost)*: Find a feasible (multicommodity) flow over time f with minimum time horizon T that satisfies given (multicommodity) demands (and whose cost is bounded by a given *budget C*). A natural variant of this problem is to bound the cost for every single commodity i by a budget C_i, that is,

$$\sum_{e \in A} c_{e,i} \int_0^T f_{e,i}(\theta)\, d\theta \;\le\; C_i$$

for all $i \in K$. All of our results also apply to these more general cost constraints.

3   A Simple Two-Approximation Algorithm

In this section we generalize the basic approach of Ford and Fulkerson [12, 13] to the case of multiple commodities (with multiple sources and sinks each) 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.

3.1   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. We denote the set of all paths starting at some source of commodity i and leading to one of its sinks by $\mathcal{P}_i$. A static (multicommodity) flow x is called T-*length-bounded* if the flow of

9

every commodity $i \in K$ can be decomposed into the sum of flows $x_{P,i}$ on paths $P \in \mathcal{P}_i$ such that the length $\tau(P)$ 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 PARTITION problem. On the other hand, the length-bounded flow problem can be approximated within arbitrary precision in polynomial time.

Lemma 3.1. *If there exists a feasible T-length-bounded static flow x which satisfies the multicommodity demands, then, for any $\varepsilon > 0$, a feasible $(1 + \varepsilon)T$-length-bounded static flow x' of cost $c(x') \leq c(x)$ satisfying all demands can be computed in time polynomial in the input size and $1/\varepsilon$.*

Proof. We first formulate the problem of finding a feasible T-length-bounded static flow as a linear program in path-variables. We assume without loss of generality that each commodity $i \in K$ has exactly one source s_i and one sink t_i with supply and demand $d_i := D_{s_i} = -D_{t_i}$; the general case with several sources and sinks can be handled by introducing one super-source and one super-sink for each commodity. Let

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

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 min-cost flow problem can then be written as:

$$\min \sum_{i \in K} \sum_{P \in \mathcal{P}_i^T} c_i(P)\, x_{P,i}$$

$$\text{s.\,t.} \quad \sum_{P \in \mathcal{P}_i^T} x_{P,i} \;\geq\; d_i \qquad\qquad \text{for all } i \in K,$$

$$\sum_{i \in K} \sum_{P \in \mathcal{P}_i^T : e \in P} x_{P,i} \;\leq\; u_e \qquad\qquad \text{for all } e \in A,$$

$$x_{P,i} \;\geq\; 0 \qquad\qquad \text{for all } i \in K,\, P \in \mathcal{P}_i^T.$$

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:

$$\max \sum_{i \in K} d_i\, z_i \;-\; \sum_{e \in A} u_e\, y_e$$

$$\text{s.\,t.} \quad \sum_{e \in P} (y_e + c_{e,i}) \;\geq\; z_i \qquad\qquad \text{for all } i \in K,\, P \in \mathcal{P}_i^T,$$

$$z_i, y_e \;\geq\; 0 \qquad\qquad \text{for all } i \in K,\, e \in A.$$

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 [20], 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 $P \in \mathcal{P}_i$ 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 $\mathcal{P}_i^T$ [21, 26, 31]. Using the equivalence of optimization and separation [16], 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 that uses additional variables corresponding to those paths of length at most $(1 + \varepsilon)T$. □

Notice that the method described in the proof 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, 15] 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]

3.2 The Approximation Algorithm

Any feasible flow over time f with time horizon T and cost at most C 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, as explained below:

  (i) it is T-length-bounded;

 (ii) it satisfies a fraction of $1/T$ of the supplies and demands covered by the flow over time f;

(iii) $c(x) = c(f)/T$.

Due to the fixed time horizon T, flow f can only travel on paths of length at most T. Thus 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 with time horizon $2T$, meeting the same supplies and demands at the same cost as f: For every commodity $i \in K$, pump flow into every path P given by the length-bounded path

---

[3]These algorithms return a solution that is at most $1 + \varepsilon$ times the minimum cost solution, that obeys capacities and serves at least $\frac{1}{1+\varepsilon}$ fraction of the demand. In the context of the approximation algorithm described in the next subsection, this approximate result is sufficient to still obtain the $2 + \varepsilon$ approximation guarantee. This is because flow can be sent just a little longer on the chosen paths to fulfill all the demand.
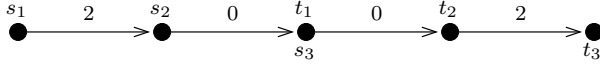
Figure 2: An instance of the quickest multicommodity flow problem containing three commodities $i = 1, 2, 3$, each with a single source s_i and a single sink t_i; commodities 1 and 3 have demand value 1, commodity 2 has demand value 2. The numbers at the arcs indicate the transit times; all arcs have 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.

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. Therefore we can state the following structural result on the power of intermediate node storage:

Lemma 3.2. *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 storage at intermediate nodes is $4/3$ times the optimal makespan with storage.*

Proof. The bound of 2 follows from the discussion above. In Figure 2 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. $\qquad\square$

Notice that the gap of $4/3$ is not an artifact of the small numbers in the instance depicted in Figure 2. 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 $4/3$.

We can now state the main result of this section.

Theorem 3.3. *For the quickest multicommodity transshipment problem with costs, there exists a polynomial time algorithm that, for any $\varepsilon > 0$, finds a solution of at most the cost of the optimal with makespan at most $2 + \varepsilon$ times the optimal makespan. Moreover, the computed solution does not store flow at intermediate nodes.*

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 Section 3.1. 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. $\qquad\square$

In Figure 3 we present an instance which shows that the analysis in the proof of Theorem 3.3 is tight. That is, the performance guarantee of the discussed approximation algorithm is not better than 2.
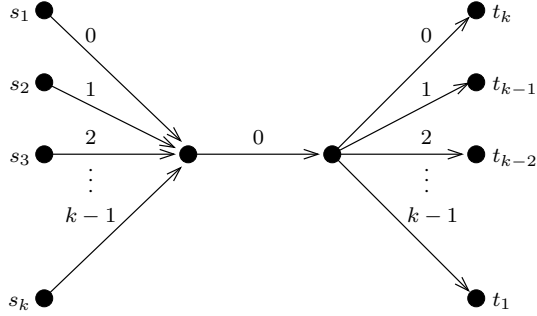
Figure 3: An instance with k commodities showing that the analysis in the proof of Theorem 3.3 is tight. All arcs have unit capacity and transit times as depicted above. The demand value of every commodity is 1. A quickest flow needs $T^* = k$ time units. However, any static flow can satisfy at most a fraction of $1/k$ of the demands. In particular, the makespan of the resulting flow over time is at least $2k - 1$.

3.3 Avoiding the Length-Bounded Flow Computation

In contrast to Ford and Fulkerson's temporally repeated flows, the flows over time resulting from T-length-bounded static flows described before Lemma 3.2, do not necessarily use flow-carrying paths as long as possible with respect to the time horizon $2T$. Instead, we stop sending flow into all paths at the same time T. However, we can easily generate a temporally repeated solution by scaling the flow rate $x_{P,i}$ on any path P to

$$\tilde{x}_{P,i} := \frac{T}{2T - \tau(P)} x_{P,i} \leq x_{P,i} .$$

Then we send flow into path P at this modified rate $\tilde{x}_{P,i}$ during the time interval $[0, 2T - \tau(P))$.

For the setting without costs, this observation also implies that the length-bounded flow computation in our algorithm can be replaced by a standard (and presumably faster) flow computation with costs, where transit times on arcs are interpreted as cost coefficients. To simplify the presentation of this result, we restrict to the case of only one source s_i and one sink t_i for every commodity $i \in K$. The scaled flow rates $\tilde{x}_{P,i}$ discussed above define a static multicommodity flow $\tilde{x}$. Let $|\tilde{x}_i|$ be the s_i-t_i-flow value of commodity i in $\tilde{x}$ and d_i the demand of commodity i. Since the temporally repeated flow with time horizon $2T$ and flow rates $\tilde{x}_{P,i}$ sends exactly d_i units of commodity i from s_i to t_i, it follows from (6) that

$$2T |\tilde{x}_i| - \sum_{e \in A} \tau_e \tilde{x}_{e,i} = d_i \qquad (7)$$

for all $i \in K$. On the other hand, any feasible static multicommodity flow $\tilde{x}$ fulfilling (7) can easily be turned into a temporally repeated flow over time satisfying all demands within time $2T$: Compute any path decomposition of $\tilde{x}$ and send flow into
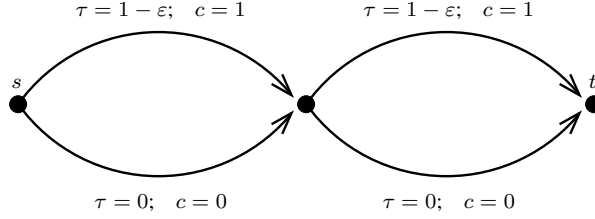
Figure 4: An instance showing that the cost of a temporally repeated flow depends on the particular path decomposition of the underlying static s-t-flow. Consider the static flow of value 2 in the depicted network with flow value 1 on every arc. The time horizon is set to 2. There are two possible decompositions of this static flow into a sum of two path-flows. One of them leads to a temporally repeated flow of cost $2 + 2\varepsilon$; the other one has cost 2ε.

every path at the corresponding flow rate as long as there is enough time left for the flow to arrive at its sink before time $2T$. This follows again from (6). We can now prove the following slightly improved approximation result which does not rely on a length-bounded static flow computation.

Theorem 3.4. *There exists a 2-approximation algorithm for the quickest multicommodity transshipment problem. Moreover, the computed solution does not store flow at intermediate nodes.*

Proof. For the sake of simplicity, we again restrict to the case of one single source s_i and one sink t_i for every commodity $i \in K$. The algorithm first solves the following flow problem

$$\min \ T$$
$$\text{s.t.} \quad 2T \, |\tilde{x}_i| \; - \; \sum_{e \in A} \tau_e \, \tilde{x}_{e,i} \; = \; d_i \qquad \text{for every } i \in K,$$
$$(\tilde{x}_{e,i})_{e \in A, \, i \in K} \text{ is a feasible multicommodity flow.}$$

It follows from (7) and the discussion above that the optimal solution value T is a lower bound on the time horizon of a quickest flow. Finally, the static flow $\tilde{x}$ can be turned into a temporally repeated flow over time with time horizon $2T$ by taking an arbitrary path decomposition of $\tilde{x}$.[4] $\square$

Unfortunately, this result cannot be generalized to the quickest multicommodity transshipment problem with costs. The reason is that the cost of a temporally repeated flow is not uniquely determined by the underlying static flow but also depends on the chosen path decomposition. An example is given in Figure 4. In fact, it can be shown by a reduction of the NP-complete problem PARTITION that finding a path decomposition of a given static flow x yielding a cheapest temporally repeated flow is NP-hard.

---

[4]If the path decomposition contains paths of length longer than $2T$, these paths contribute negatively to the right hand side of (7). Thus we can remove them along with flow on a set of shorter paths and obtain a smaller path decomposition.

4 Approximation Schemes for Quickest Flows

In this section we present a framework for obtaining fully polynomial time approximation schemes for various quickest flow problems. The backbone of this framework are special time-expanded networks of polynomial size which we introduce in Section 4.1. In Section 4.2 we present the basic idea of our approach and point out fundamental problems that need to be solved in order to make it work. Then, in Section 4.3 we discuss the special case of acyclic graphs. Under the assumption that storage of flow at intermediate nodes is allowed, acyclic graphs are amenable to a simple analysis. On the other hand, we show in Section 4.4 that optimal single commodity flows over time (with costs) do not require storage. Based on this insight, we give a fully polynomial time approximation scheme for the quickest transshipment problem with bounded cost which does not use storage at intermediate nodes in Section 4.5. We describe a generalization of our approach to the multicommodity flow setting in Section 4.6.

4.1 Condensed 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 θth copy of node v denoted v_θ, $\theta = 0, \ldots, T - 1$. The flow that passes through V_θ corresponds to flow over time in the interval $[\theta, \theta + 1)$. 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. For each terminal $v \in S_i$, $i \in K$, there is an additional infinite capacity *holdover arc* from v_θ to $v_{\theta+1}$, for all $0 \leq \theta < T - 1$, which models the possibility to hold flow at node v in the time interval $[\theta, \theta + 1)$. We assume without loss of generality that a source (sink) has no incoming (outgoing) arc in $\mathcal{N}$. Thus, a terminal is never an intermediate node on a path flow.[5] We treat the first copy v_0 of a source $v \in S_i^+$ as the corresponding source in $\mathcal{N}^T$; and treat the last copy v_{T-1} of a sink $v \in S_i^-$ as the corresponding sink in $\mathcal{N}^T$. In the model with storage of flow at intermediate nodes, we introduce holdover arcs for all nodes $v \in V$. An illustration of a time-expanded network is given in Figure 5 a) and b).

Any static (multicommodity) flow in this time-expanded network corresponds to a (multicommodity) flow over time of equal cost: for any commodity, interpret the flow on arc e_θ as the flow rate entering arc $e = (v, w)$ 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 setting the flow on e_θ to be the average flow rate into e over the interval $[\theta, \theta+1)$. More details can be found below in Lemma 4.1 (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,

---

[5]Under this assumption, the flow storage level of commodity i at $v \in S_i$ never exceeds $D_{v,i}$.
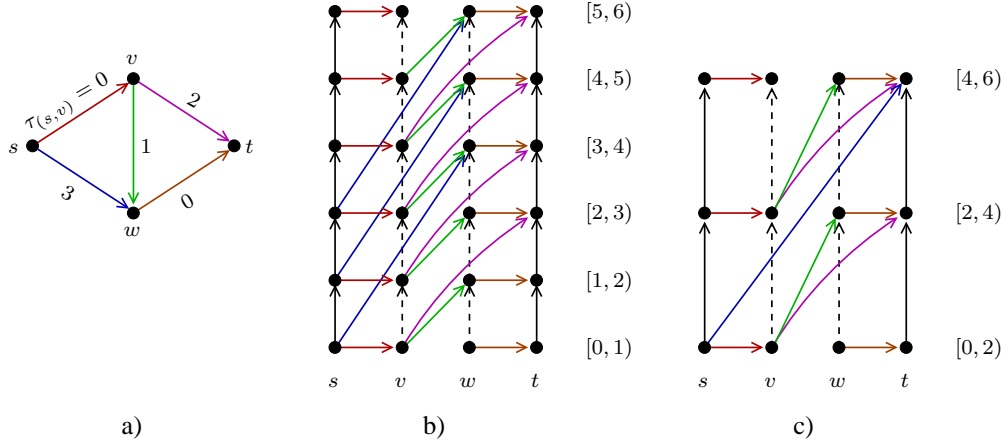
Figure 5: a) A static network $\mathcal{N}$ with transit times on the arcs, one source s and one sink t; b) the corresponding time-expanded network $\mathcal{N}^T$ with time horizon $T = 6$; c) the condensed time-expanded network $\mathcal{N}^T/\Delta$ with $\Delta = 2$; notice that the transit time of arc (s, w) has been rounded up to 4 in $\mathcal{N}^T/\Delta$ since the original transit time 3 is not a multiple of Δ (see also Section 4.2).

then instead of using the T-time-expanded network, 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 $\rho = 0, \ldots, \lceil T/\Delta \rceil - 1$. Copy $V_{\rho\Delta}$ corresponds to flow through V in the interval $[\rho\Delta, (\rho + 1)\Delta)$. An illustration is given in Figure 5 c).

Lemma 4.1. *Suppose that all arc lengths are multiples of Δ and T/Δ is an integer. Then, any (multicommodity) flow over time that completes by time T corresponds to a static (multicommodity) 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 (multicommodity) flow over time, a modified flow over time of equal value and cost can be obtained by averaging the flow value of every commodity 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 (multicommodity) 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 (multicommodity) 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
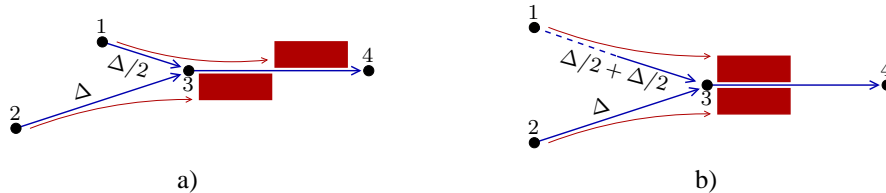
16

Figure 6: a) A flow over time in the original network. The two packets of flow originating at nodes 1 and 2 are sent one after another into arc $(3, 4)$. b) The 'same' flow over time in the network with rounded transit times causes congestion on arc $(3, 4)$ since the two packets of flow arrive simultaneously on the arc.

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 4.2. *Suppose that all arc lengths are multiples of Δ. Then, any (multicommodity) flow over time that completes by time T corresponds to a static (multicommodity) 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 before time $T + \Delta$.*

4.2 Outline of an Approximation Scheme

The basic idea of our algorithm is to round up transit times to the nearest multiple of Δ for an appropriately chosen Δ, solve the static flow problem in the corresponding Δ-condensed time-expanded network, and then translate this flow back to the setting of the original transit times. In order to obtain provably good solutions in this way, one has to make sure that the following two conditions are fulfilled:

(i) the value of an optimal solution to the instance with increased transit times (represented by the condensed time-expanded network) approximates the value of an optimal solution in the original setting;

(ii) the solution to the instance with increased transit times can be transformed into a flow over time with original arc lengths without too much loss in flow value.

Before discussing how to fulfill these conditions, we first give some simple examples to show that non-trivial problems have to be dealt with to address both (i) and (ii).

Consider first a (sub)network consisting of four nodes $\{1, 2, 3, 4\}$ and three arcs $(1, 3)$, $(2, 3)$, $(3, 4)$ with unit capacity depicted in Figure 6 a). The transit times are $\tau_{(1,3)} = \Delta/2$, $\tau_{(2,3)} = \Delta$, and $\tau_{(3,4)} = \Delta$. A flow in the graph without rounded transit times can send $\Delta/2$ units of flow in interval $[0, \Delta/2)$ on each path $P_1 = 1 \rightarrow 3 \rightarrow 4$ and $P_2 = 2 \rightarrow 3 \rightarrow 4$. Path P_1 will use arc $(3, 4)$ in interval $[\Delta/2, \Delta)$ and path P_2 will use arc $(3, 4)$ in interval $[\Delta, 3\Delta/2)$. However, if we send flow simultaneously on paths P_1 and P_2 in the network with transit times rounded up to the nearest multiple of Δ, then this will cause a bottleneck on arc $(3, 4)$; see Figure 6 b).
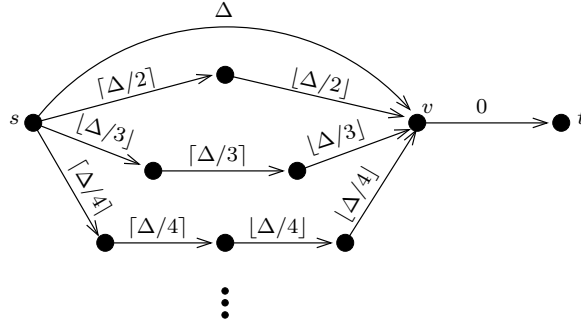
Figure 7: In this partially drawn unit capacity network, there are Δ paths from s to v. The ith path contains i arcs, each with transit time roughly Δ/i.

Now consider the unit capacity (sub)network depicted in Figure 7. If all transit times are rounded to the nearest multiple of Δ, we may send Δ units of flow simultaneously on each path from s to t, and each path will use arc (v, t) in a distinct interval of time. If we try to interpret this flow in the network with original transit times, however, each path-flow will try to use arc (v, t) in the same time interval, causing a large bottleneck.

Condition (ii) can be enforced by allowing storage of flow at nodes: If arc $e = (v, w)$ has length increased by $\Delta' \leq \Delta$, then this can be emulated in the original network by holding flow arriving at w for Δ' time units.[6] More generally, we can state the following observation.

Observation 4.1. *Consider a network $\mathcal{N} = (V, A)$ and two transit time vectors $\tau, \tau' \in \mathbb{R}_+^A$ with $\tau_e \leq \tau'_e$ for all $e \in A$. Then, a flow over time in $\mathcal{N}$ with transit times τ' can be emulated in $\mathcal{N}$ with transit times τ by introducing waiting time $\tau'_e - \tau_e$ at the head of every arc e.*

For the case of acyclic graphs, we give a simple argument in Section 4.3 to show how to uphold condition (i) when storage is allowed. In Section 4.5 we describe a more sophisticated approach that works for general graphs even when storage of flow at nodes is not allowed.

4.3 Acyclic Graphs with Storage

For acyclic graphs, the existence of a topological ordering of the nodes makes the problems illustrated above fairly easy to resolve. The algorithm is simple: for an appropriate guess of T, choose $\Delta := \frac{\varepsilon}{n} T$ and round transit times up to nearest multiple of Δ. Form the Δ-condensed time-expanded network, and compute a solution f in this network. Output the solution f' obtained by modifying f by emulating the rounded transit times as described above.

---

[6]If Δ' is large, then this requires a large amount of additional storage.

It remains to show (i), that is, there exists a feasible flow of cost at most C satisfying (multicommodity) demands D by time $T^*(1 + \varepsilon)$ in the network with transit times rounded up to the nearest multiple of Δ. Let f^* be a flow of cost at most C that satisfies demands D by time T^*. Let $\{v_0, v_1, \ldots, v_{n-1}\}$ be a topological ordering of V. Modify f^* to obtain $\hat{f}$ by setting $\hat{f}_e(\theta) = f^*(\theta - i\Delta)$ for $e = (v_i, v_j)$, and for all $\theta \in [0, \infty)$. With these modifications, flow traveling on any path that includes v_i is delayed from its original departure from i by exactly $i\Delta$ time units. Thus, flow arriving at node v_j in $\hat{f}$ arrives at most $j\Delta$ time units later than its arrival in f^*, and the time horizon of $\hat{f}$ is $T^* + \Delta(n - 1) \leq T^*(1 + \varepsilon)$. Since f^* is feasible, so is $\hat{f}$. Since flow travels on the same paths in f^* and $\hat{f}$, we have $c(f^*) = c(\hat{f})$, and $\hat{f}$ satisfies the same multicommodity demands as f^*.

Notice that $\hat{f}$ is cost and demand equivalent to a flow in the network where the transit time of an arc (v_i, v_j) is increased by $\Delta(j - i) \geq \Delta$. It is also cost and demand equivalent to a flow with transit time of (v_i, v_j) increased by $\delta_{ij} < \Delta$ to the nearest multiple of Δ, and then held at v_j for an additional $\Delta(j - i) - \delta_{ij} \geq 0$ units of time. This latter flow is a flow in the Δ-rounded network, implying the following theorem.

Theorem 4.3. *In acyclic graphs with node storage, a $(1 + \varepsilon)$-approximate solution to the quickest cost-bounded multicommodity flow problem can be obtained with a static flow computation in a network with n^2/ε nodes and nm/ε arcs.*

4.4 Minimum Cost Flows without Storage

It follows from the work of Hoppe and Tardos [24, 22] that for the quickest transshipment problem there always exists an optimal solution which does not store flow at intermediate nodes. We generalize this result to the problem with costs and also to the more general case when the flow cost on an arc is a nondecreasing, convex function of the flow rate into the arc. As mentioned in Section 4.1, when transit times are integers, the min-cost transshipment-over-time problem with or without storage at intermediate nodes can be solved by solving the corresponding static flow problem in the time-expanded network[7] $\mathcal{N}^T$.

Theorem 4.4. *For nondecreasing, convex cost functions, the cost of a minimum cost transshipment over time that does not use intermediate node storage is no more than the cost of a minimum cost transshipment over time using intermediate node storage.*

The details of this proof are not essential for understanding the remainder of the paper.

Proof. Consider a minimum cost transshipment over time with intermediate node storage and a corresponding static min-cost flow x in the time-expanded network $\mathcal{N}^T$. Notice that the set X of all min-cost solutions x is the intersection of the polytope formed by all feasible solutions with a closed convex set given by the convex cost constraint. In particular, X is convex and compact.

---

[7]Notice that the averaging argument used in the proof of Lemma 4.1 to turn an arbitrary flow over time into a static flow in the time-expanded network also works for the case of convex costs since averaging never increases cost.

For a node $z \in V$, let $x(\delta(z_\theta))$ be the net flow leaving z in the time interval $[\theta, \theta + 1)$:

$$x(\delta(z_\theta)) := \sum_{e \in \delta^+(z)} x_{e_\theta} - \sum_{e \in \delta^-(z)} x_{e_{\theta - \tau_e}} .$$

Since X is compact, there exists an $x \in X$ minimizing the convex function $F(x) := \sum_{z \in V} \sum_{\theta=0}^{T-1} |x(\delta(z_\theta))|$. We show that x does not send flow along holdover arcs of nodes in $V \setminus S$.

By contradiction, let v_φ be the earliest copy of node $v \notin S$ to send flow along a holdover arc. We have that $x(\delta(v_\varphi)) = -x_{v_\varphi, v_{\varphi+1}} < 0$. Let $[\varphi + q, \varphi + q + 1)$, for $q > 0$ and integral, be the first time interval after $[\varphi, \varphi + 1)$ in which v has more flow leaving it than entering it; that is, $x(\delta(v_{\varphi+q})) > 0$. We show in the following that $F(x)$ can be decreased by augmenting flow along a cycle in the time-expanded network $\mathcal{N}^T$. This is a contradiction to the choice of x.

Consider a time-expanded network that is infinite in both directions, $\mathcal{N}^{(-\infty,+\infty)}$. Note that $\mathcal{N}^{(-\infty,+\infty)}$ looks the same at v_φ as it does at $v_{\varphi+q}$. However, x in this network looks different at each of these copies of v. We indicate this difference by coloring the arcs of $\mathcal{N}^{(-\infty,+\infty)}$ as follows. Color transit arc $(i_{\theta - \tau_{ij}}, j_\theta)$

$$
\begin{aligned}
&\text{red} &&\text{if} &&x_{(i_{\theta-\tau_{ij}}, j_\theta)} < x_{(i_{\theta-\tau_{ij}-q}, j_{\theta-q})} \\
&\text{blue} &&\text{if} &&x_{(i_{\theta-\tau_{ij}}, j_\theta)} > x_{(i_{\theta-\tau_{ij}-q}, j_{\theta-q})} \\
&\text{no color} &&\text{if} &&x_{(i_{\theta-\tau_{ij}}, j_\theta)} = x_{(i_{\theta-\tau_{ij}-q}, j_{\theta-q})} .
\end{aligned}
$$

All holdover arcs remain colorless. Note that there are no blue arcs leaving V_θ for $\theta \geq T - 1$; and there are no red arcs entering V_θ for $\theta \leq q$.

Let P be a simple path consisting of backward red arcs and forward blue arcs from $v_{\varphi+q}$ to a node w_μ with the property that $x(\delta(w_\mu)) < x(\delta(w_{\mu-q}))$. We claim that such a P exists: Since $x(\delta(v_{\varphi+q})) - x(\delta(v_\varphi)) > 0$, node $v_{\varphi+q}$ has either a red arc entering it or a blue arc leaving it. Consider the set of all nodes which can be reached from $v_{\varphi+q}$ on a path consisting of backward red arcs and forward blue arcs. Since $x(\delta(v_{\varphi+q})) - x(\delta(v_\varphi)) > 0$, it follows from flow conservation that there must exist a node w_μ with $x(\delta(w_\mu)) - x(\delta(w_{\mu-q})) < 0$ in this set.

Note that $V(P) \subset \bigcup_{\theta=q}^{T-1} V_\theta$. We define the capacity $u(P)$ of P to be

$$u(P) := \min_{(i_\theta, j_{\theta+\tau_{ij}}) \in P} |x_{(i_\theta, j_{\theta+\tau_{ij}})} - x_{(i_{\theta-q}, j_{\theta-q+\tau_{ij}})}| .$$

We modifying x to reduce $|x(\delta(v_\varphi))|$ and $|x(\delta(v_{\varphi+q}))|$. Let

$$\kappa := \min\{u(P), -x(\delta(v_\varphi)), x(\delta(v_{\varphi+q})), x(\delta(w_{\mu-q})) - x(\delta(w_\mu))\} > 0 .$$

If an arc $(i_\theta, j_{\theta+\tau_{ij}}) \in P$ is red, then we modify x on $(i_\theta, j_{\theta+\tau_{ij}})$ and $(i_{\theta-q}, j_{\theta-q+\tau_{ij}})$ to

$$x_{(i_\theta, j_{\theta+\tau_{ij}})} := x_{(i_\theta, j_{\theta+\tau_{ij}})} + \kappa \quad \text{and} \quad x_{(i_{\theta-q}, j_{\theta-q+\tau_{ij}})} := x_{(i_{\theta-q}, j_{\theta-q+\tau_{ij}})} - \kappa .$$

If $(i_\theta, j_{\theta+\tau_{ij}}) \in P$ is blue, then

$$x_{(i_\theta, j_{\theta+\tau_{ij}})} := x_{(i_\theta, j_{\theta+\tau_{ij}})} - \kappa \quad \text{and} \quad x_{(i_{\theta-q}, j_{\theta-q+\tau_{ij}})} := x_{(i_{\theta-q}, j_{\theta-q+\tau_{ij}})} + \kappa \ .$$

Finally, we remove κ units of flow from the path of holdover arcs from v_φ to $v_{\varphi+q}$ and add κ to the path of holdover arcs from $w_{\mu-q}$ to w_μ. Notice that we have augmented flow on a cycle in $\mathcal{N}^T$ by κ. Since the domain of P is restricted to $V(P) \subset \bigcup_{\theta=q}^{T-1} V_\theta$, the flow x is still a feasible solution to our problem.

We next argue that the cost of x is not increased so that x is still in X: Since the flow augmentation transfers an equal amount of flow from one copy of an arc to a parallel copy, if flow costs are linear, this does not change the cost of our solution. Since the sum of flow on these two arcs does not change, and we simply move flow so that the flow on each is closer to the average flow on each, if our flow costs are convex and nondecreasing, then the cost of our solution does not increase.

Finally, the augmentation by κ ensures that $|x(\delta(v_\varphi))|$ and $|x(\delta(v_{\varphi+q}))|$ are each reduced by κ, and $|x(\delta(w_\mu))| + |x(\delta(w_{\mu-q}))|$ is not increased. (Either $|x(\delta(w_{\mu-q}))| > \kappa$ and $|x(\delta(w_\mu))| < -\kappa$; or, since $\kappa \leq x(\delta(w_{\mu-q})) - x(\delta(w_\mu))$, the quantities $|x(\delta(w_\mu))|$ and $|x(\delta(w_{\mu-q}))|$ exchange values). Thus, $F(x) = \sum_{z \in V} \sum_{\theta=0}^{T-1} |x(\delta(z_\theta))|$ is decreased by at least $2\kappa > 0$. This concludes the proof. $\qquad\square$

Theorem 4.4 implies that we can find a minimum cost flow over time in the time-expanded network *without* holdover arcs for intermediate nodes. We can even state the following stronger result.

Corollary 4.5. *For every instance of the minimum cost transshipment-over-time problem, when costs are nondecreasing, convex functions of the flow rate, there exists an optimal solution without intermediate node storage such that any infinitesimal unit of flow visits every node at most once.*

Proof. We first consider the case that there is no cycle of zero cost in $\mathcal{N}$. If some path flow in an optimal flow visits a node v more than once, it travels along a cycle in $\mathcal{N}$. Therefore the cost of the solution can be decreased by letting the flow wait at v. This is a contradiction to the optimality of the solution.

If there exist zero cost cycles in $\mathcal{N}$, we can increase the cost of every arc by a small amount such that an optimal solution to the modified problem always yields an optimal solution to the original problem. This eliminates cycles of zero cost and thus concludes the proof. $\qquad\square$

4.5   General Graphs without Storage

Here we describe how to adapt the outline given in Section 4.2 to yield an approximation scheme for the cost-bounded quickest transshipment problem in general graphs that does not use any intermediate node storage. There are two main steps to the approach: First, we choose Δ small enough so that we can increase the time horizon by a sufficiently large amount relative to Δ to account for bottlenecks caused by problems of type (i) (see Section 4.2). Second, we average the flow computed in the rounded network over sufficiently large intervals relative to Δ so that bottlenecks caused by

INPUT: network $\mathcal{N}$ with capacities, linear costs, transit times, demand vector D, cost bound C, and $\varepsilon > 0$;

OUTPUT: feasible flow over time f satisfying demands D at cost at most C;

1. guess T such that $T^* \leq T \leq (1 + \varepsilon) T^*$;

2. round transit times up to nearest multiple of $\Delta := \varepsilon^2 T/(2n)$;

3. construct Δ-condensed time-expanded network (without holdover arcs for intermediate nodes) with time horizon $T' = (1 + O(\varepsilon)) T$;

4. compute static flow x' in this network satisfying demands $(1 + \varepsilon) D$ at cost at most $(1 + \varepsilon) C$;

5. interpret x' as flow over time f' in the network with rounded transit times $\tilde{\mathcal{N}}$; let $(f'_P)_{P \in \mathcal{P}}$ be a path decomposition of f';

6. Define $f_P(\theta) := \frac{1}{1+\varepsilon} \frac{1}{\varepsilon T} \int_{\theta - \varepsilon T}^{\theta} f'_P(\xi) \, d\xi$ for all $P \in \mathcal{P}$ and $\theta \in [0, T' + \varepsilon T)$ as a flow in the original network $\mathcal{N}$.

Figure 8: A fully polynomial time approximation scheme. Here, T^* represents the optimal, unknown, makespan.

problems of type (ii) are averaged over a sufficiently long time to make the flow almost feasible. This second step also increases the total time horizon of the flow, but again, by careful choice of Δ, by a sufficiently small amount.

By Corollary 4.5 we can restrict to transshipments over time f that send flow only on simple paths and that never store flow at intermediate nodes. This means that f can be written as a sum of path flows over time f_P, $P \in \mathcal{P}$: Consider an arbitrary arc $e = (v, w) \in A$. The total flow into arc e at time θ is

$$f_e(\theta) = \sum_{P \in \mathcal{P}\,:\,e \in P} f_P\big(\theta - \tau(P, e)\big) \ , \tag{8}$$

where $\tau(P, e)$ denotes the length of the subpath of P which is obtained by removing arc e and all its successors. In the following, let T^* be the makespan of a quickest transshipment f^* with cost bounded by C.

A precise description of the algorithm is given in Figure 8. The exact choice of time horizon T' will be given in the analysis of the algorithm; see Lemma 4.8. By definition, the flow over time f' lives in network $\tilde{\mathcal{N}}$ which is a copy of $\mathcal{N}$ with transit times rounded up to multiples of $\Delta = \varepsilon^2 T/(2n)$. In contrast, the flow over time f is defined on $\mathcal{N}$ (with original transit times). The flows f and f' use the same set of simple paths $\mathcal{P}$. However, f flows through these paths at a faster pace than f'.

We start by discussing the running time of this algorithm. Using geometric mean binary search in step 1, we find T such that $T^* \leq T \leq (1 + \varepsilon) T^*$. We can begin with standard binary search to find lower and upper bounds on T that are within a constant multiple of each other. This requires $\log T^*$ iterations. Alternatively, the constant

factor approximation algorithm for the quickest transshipment problem with bounded cost presented in Section 3 yields a lower bound L and an upper bound U on T^* such that $U \in O(L)$. Using this, the estimate T can be obtained within $O(\log(1/\varepsilon))$ geometric mean binary search steps.

The condensed time-expanded network (without holdover arcs) constructed in step 3 contains $O(n^2/\varepsilon^2)$ nodes and $O(mn/\varepsilon^2)$ arcs. Thus the static flow x' in step 4 can be computed in polynomial time and it can be decomposed into $O(mn/\varepsilon^2)$ path flows. As a consequence thereof, the corresponding flow over time f' can be decomposed into flows over time on at most $O(mn/\varepsilon^2)$ simple paths $P \in \mathcal{P}$ such that the flow rate f'_P on each path P is zero except for an interval of length $\Delta = \varepsilon^2 T/(2n)$ where f'_P attains a positive constant value. Each path P can occur up to $O(n/\varepsilon^2)$ times in this decomposition. Thus, computing f in step 6 takes $O(mn/\varepsilon^2)$ time. To output f requires n work per path, so $O(mn^2/\varepsilon^2)$ time.

The next sequence of theorems and lemmas justify the choice of T' in the algorithm. Our key theorem says that a small perturbation of transit times has only a small effect on (time horizon, value, cost) triples of feasible flows in $\mathcal{N}$.

Theorem 4.6. *Let f be a flow over time with time horizon T in $\mathcal{N}$ with transit times τ_e satisfying demands D with cost $\leq C$ that can be decomposed into finitely many flows over time f_P on simple paths $P \in \mathcal{P}$. Let $\varepsilon > 0$ and $(\tilde{\tau}_e)_{e \in A}$ be a second set of transit times on $\mathcal{N}$ with $|\tau_e - \tilde{\tau}_e| \leq \varepsilon^2 T/(2n)$, for all $e \in A$. Then, the flow $\tilde{f}$ defined as*

$$\tilde{f}_P(\theta) := \frac{1}{1+\varepsilon} \frac{1}{\varepsilon T} \int_{\theta - \varepsilon T}^{\theta} f_P(\xi) \, d\xi$$

for all $P \in \mathcal{P}$ and $\theta \in [0, (1+\varepsilon)T)$ is a feasible flow in the network $\mathcal{N}$ with transit times $\tilde{\tau}_e$, $e \in A$, satisfying the following properties:

  (i) *the time horizon of $\tilde{f}$ is at most $(1 + \varepsilon + \varepsilon^2/2)\, T$;*

  (ii) *$\tilde{f}$ sends flow on the same set $\mathcal{P}$ of paths as f;*

 (iii) *$\tilde{f}$ satisfies demands $D/(1+\varepsilon)$.*

 (iv) *$\tilde{f}$ has cost at most $C/(1+\varepsilon)$.*

Proof. We obtain a 'smoothed' flow over time $(\hat{f}_P)_{P \in \mathcal{P}}$ on $\mathcal{N}$ with time horizon $(1 + \varepsilon)\, T$ by defining

$$\hat{f}_P(\theta) := \frac{1}{\varepsilon T} \int_{\theta - \varepsilon T}^{\theta} f_P(\xi) \, d\xi \tag{9}$$

for $\theta \in [0, (1+\varepsilon)\, T)$ and $P \in \mathcal{P}$.

An illustrative example is given in Figure 9. It is easy to check that $\hat{f}$ obeys capacity constraints and the total amount of flow sent on a path $P \in \mathcal{P}$ is the same in f and $\hat{f}$. In particular, $c(\hat{f}) = c(f) \leq C$ and $\hat{f}$ satisfies demands D.

Notice that $(\hat{f}_P)_{P \in \mathcal{P}}$ still describes a (not necessarily feasible) flow over time in $\mathcal{N}$ with transit times perturbed by at most $\varepsilon^2 T/(2n)$. Since every path $P \in \mathcal{P}$ is simple, it contains at most $n - 1$ arcs; therefore, $|\tau(P) - \tilde{\tau}(P)| \leq \varepsilon^2 T/2$ and

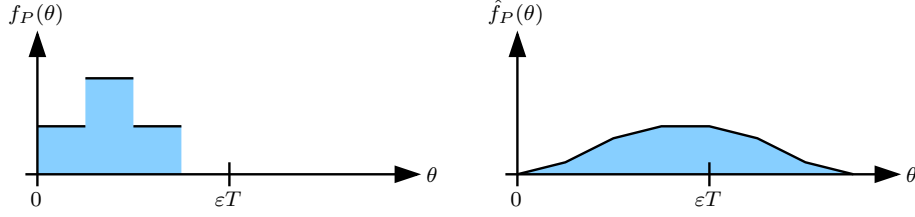$$|\tau(P, e) - \tilde{\tau}(P, e)| \leq \varepsilon^2 T/2 \tag{10}$$

Figure 9: The 'smoothed' path flow over time $\hat{f}_P$ in comparison to the original flow over time f_P sent into path P.

for all $e \in P$. Thus, with $\tilde{\tau}$ we have for all $e \in A$ and $\theta \in [0, (1 + \varepsilon + \varepsilon^2/2)\,T)$,

$$
\begin{aligned}
\hat{f}_e(\theta) \overset{(8)}{=} \; & \sum_{P \in \mathcal{P}\,:\,e \in P} \hat{f}_P(\theta - \tilde{\tau}(P, e)) \\
\overset{(9)}{=} \; & \frac{1}{\varepsilon\,T} \sum_{P \in \mathcal{P}\,:\,e \in P} \int_{\theta - \tilde{\tau}(P,e) - \varepsilon\,T}^{\theta - \tilde{\tau}(P,e)} f_P(\xi)\,d\xi \\
\overset{(10)}{\leq} \; & \frac{1}{\varepsilon\,T} \sum_{P \in \mathcal{P}\,:\,e \in P} \int_{\theta - \tau(P,e) - \varepsilon^2\,T/2 - \varepsilon\,T}^{\theta - \tau(P,e) + \varepsilon^2\,T/2} f_P(\xi)\,d\xi \qquad\qquad (11) \\
= \; & \frac{1}{\varepsilon\,T} \int_{\theta - \varepsilon^2\,T/2 - \varepsilon\,T}^{\theta + \varepsilon^2\,T/2} \sum_{P \in \mathcal{P}\,:\,e \in P} f_P\bigl(\xi - \tau(P, e)\bigr)\,d\xi \\
\overset{(8)}{=} \; & \frac{1}{\varepsilon\,T} \int_{\theta - \varepsilon^2\,T/2 - \varepsilon\,T}^{\theta + \varepsilon^2\,T/2} f_e(\xi)\,d\xi \\
\leq \; & \frac{\varepsilon^2\,T + \varepsilon\,T}{\varepsilon\,T}\, u_e \; = \; (1 + \varepsilon)\, u_e \; .
\end{aligned}
$$

(In (11), we use (10) and the fact that $f_P(\xi) \geq 0$ for all ξ.) Thus, by dividing $\hat{f}$ by $(1 + \varepsilon)$, we establish the existence of a feasible flow over time — namely $\hat{f}/(1+\varepsilon)$ — in the network $\mathcal{N}$ with transit times $\tilde{\tau}$. The time horizon of this flow is $\hat{T} = (1 + \varepsilon + \varepsilon^2/2)\,T$, its cost is $c(\hat{f})/(1 + \varepsilon) \leq C/(1 + \varepsilon)$ and it satisfies demands $D/(1 + \varepsilon)$. $\qquad\square$

The next lemma states that increasing supplies and demands by a factor of $\delta \geq 1$ requires an increase in time and cost of at most δ.

Lemma 4.7. *Let f be a flow over time with time horizon T and cost $c(f)$, satisfying a vector of supplies and demands D. Then for all $\delta \geq 1$, there exists a flow over time f' satisfying demands and supplies $\delta\,D$ within time $\delta\,T$ and with cost $\delta\,c(f)$.*

Proof. By rescaling time, we can assume without loss of generality that T and all transit times are integral. Let x be the static flow in the T-time-expanded network which corresponds to f. Consider a modified instance where all transit times of arcs are increased by a factor of δ. Then, the δ-condensed time-expanded network of the modified instance with time horizon $\delta\,T$ is identical to the T-time-expanded network of

24

the original instance, but with arc capacities multiplied by δ. In particular, $\delta\,x$ defines a feasible flow over time with time horizon $\delta\,T$ and cost $\delta\,c(f)$ satisfying demands and supplies $\delta\,D$ for the modified instance. Since transit times in the original instance are smaller, it can be seen as a relaxation of the modified instance. This yields the existence of f' and concludes the proof. $\qquad\square$

Lemma 4.8. *If T' is chosen to be at least $T(1 + \varepsilon + \varepsilon^2/2)\,(1 + \varepsilon)^2$, then there exists a static flow x' as described in step 4.*

Proof. Consider a quickest transshipment f^* on network $\mathcal{N}$ (with original transit times τ). The time horizon of f^* is T^* and its cost is at most C.

By Lemma 4.7 with $\delta = (1 + \varepsilon)^2$, there then exists a flow f that satisfies demands $D(1 + \varepsilon)^2$ at cost at most $C(1 + \varepsilon)^2$ and completes by time $T := T^*(1 + \varepsilon)^2$.

By Corollary 4.5, there exists a decomposition of $f = (f_P)_{P \in \mathcal{P}}$ into flows over time f_P on simple paths $P \in \mathcal{P}$ (see (8)).

Then, by Theorem 4.6, if transit times are rounded up to nearest multiple of $\Delta = \varepsilon^2 T/(2n)$, there exists a flow $\bar{f}$ satisfying demands $D(1 + \varepsilon)$ at cost $C(1 + \varepsilon)$ and completes by time $(1 + \varepsilon + \varepsilon^2/2)T = (1 + \varepsilon + \varepsilon^2/2)(1 + \varepsilon)^2 T^*$.

Thus, choosing $T' = (1 + \varepsilon + \varepsilon^2/2)(1 + \varepsilon)^2 T$ assures the existence of flow x' in step 4. $\qquad\square$

We remark that if we choose to allow additional storage, then we may simplify the algorithm by finding a flow satisfying demands D at cost C in step 4, and remove step 6 entirely. Then the flow f' can be simulated in the original network by holding flow at nodes.

It remains to show that the flow over time f computed in step 6 of the algorithm is feasible and satisfies the vector of demands and supplies D at cost at most C. The result on the demand and cost follows from the definition of f since f' satisfies demands $(1 + \varepsilon)\,D$ with cost at most $(1 + \varepsilon)\,C$. The feasibility of f follows by applying Theorem 4.6 to the flow over time f'. This yields the following main result of this section.

Theorem 4.9. *For an arbitrary $\varepsilon > 0$, a $(1 + \varepsilon)$-approximate solution to the quickest transshipment problem with bounded cost can be obtained from $O(\log(1/\varepsilon))$ static min-cost flow computations in a condensed time-expanded network with $O(n^2/\varepsilon^2)$ nodes and $O(mn/\varepsilon^2)$ arcs (without holdover arcs). In particular, this solution does not use intermediate node storage.*

For the case of the quickest transshipment problem without costs, the min-cost flow computations in the condensed time-expanded network can be replaced by max-flow computations.

4.6 Quickest Multicommodity Flows.

The result of Theorem 4.9 can be generalized to the quickest multicommodity flow problem with bounded cost. Figure 2 shows that the optimal solution to the quickest multicommodity flow problem may require the use of storage of flow at intermediate nodes. On the other hand, if storing at intermediate nodes is not allowed, then the optimal solution may contain non-simple flow paths. The analysis in (11) relies on the

25

fact that one can restrict to simple flow paths, since it uses (10). Indeed, it is shown in [18] that, unless P=NP, there is no FPTAS for the quickest multicommodity flow problem when intermediate node storage is prohibited and flow may only be sent on simple paths. If, however, intermediate node storage is allowed, then there exists an optimal solution that uses only simple flow paths: instead of flow traveling around a cycle, it can simply wait at the start node of the cycle. In this case, the approach described in Section 4.5 can be modified as follows.

A flow over time f which stores flow at intermediate nodes can no longer be decomposed into path flows over time as described in (8). In order to handle the setting with storage of flow at intermediate nodes in a path-based model, we introduce the following notation. A *path with delays* P^δ is given by a path P in $\mathcal{N}$ consisting of nodes $(v_0, v_1, \ldots, v_p)$, and a vector of non-negative delays $\delta = (\delta_1, \ldots, \delta_p)$. The value δ_i, $i = 1, \ldots, p$, specifies the amount of time that flow is stored at node v_i before it continues its journey towards node v_{i+1} in a flow over time on P^δ. Thus, flow entering P^δ at time θ enters arc $e = (v_\ell, v_{\ell+1})$, $\ell = 0, \ldots, p-1$, at time $\theta + \tau(P^\delta, e)$ with $\tau(P^\delta, e) := \sum_{j=1}^{\ell}(\tau_{(v_{j-1}, v_j)} + \delta_j)$.

Since in a given flow over time f with time horizon T every infinitesimal unit of flow describes a path with delays in $\mathcal{N}$, the flow over time f can be decomposed into (possibly infinitely many) flows over time f_{P^δ} on paths with delays P^δ. In this setting, (8) is replaced by

$$f_e(\theta) \;=\; \sum_{P^\delta \,:\, e \in P} f_{P^\delta}\big(\theta - \tau(P^\delta, e)\big) \tag{12}$$

for all $e \in A$. If f is given by a corresponding static flow x in a (condensed) time-expanded network with holdover arcs, then there exists a decomposition of f into flows over time on paths with delays in $\mathcal{N}$ whose number is bounded by the number of arcs in the (condensed) time-expanded network: Consider an arbitrary path decomposition of x and notice that any path in the (condensed) time-expanded network with holdover arcs yields a path with delays in $\mathcal{N}$.

Summarizing, a straightforward modification of the analysis in Section 4.5 (i.e., replacing (8) by (12)) yields the following result.

Theorem 4.10. *Consider an instance of the quickest multicommodity transshipment problem with bounded cost and intermediate node storage. For any $\varepsilon > 0$, a $(1 + \varepsilon)$-approximate solution can be found by $O(\log(1/\varepsilon))$ static multicommodity flow computations with bounded cost in a condensed time-expanded network with $O(n^2/\varepsilon^2)$ nodes and $O(mn/\varepsilon^2)$ arcs (including holdover arcs).*

5 Earliest Arrival Flows

In this section, we address the multiple source, earliest arrival flow problem: given a set of sources S with supplies $D_v > 0$, for $v \in S$, and a single sink t, send the supplies to the sink so that the amount of supplies arriving at the sink by time θ is the maximum possible, for all $\theta \geq 0$. We show how the result from Section 4 can be generalized to this problem.

We use the following notion of approximation for the problem of computing an earliest arrival flow. A flow over time is an α-approximate solution to this problem if, for all $d' \leq d := \sum_{v \in S} D_v$, the earliest point in time when d' units have arrived at the sink is within a factor of α of the earliest possible time. An algorithm which computes such a flow over time in polynomial time is called an α-approximation algorithm. In order to stress the property that the performance guarantee α is achieved *for all* $d' \leq d$, we say that such an algorithm has *universal* performance guarantee α. For the earliest arrival problem, we find a solution with universal performance guarantee $(1 + \varepsilon)$.

In Section 5.2 we show that a unit-interval discretization may not be sufficiently fine to achieve this guarantee, and describe how to determine a good initial discretization. This discretization may be too large, and it will be necessary to condense it. However, a uniformly crude discretization will not work, so in Section 5.3, we introduce a general framework for nonuniform condensed time-expanded networks and prove some useful properties. Finally, in Section 5.4 we derive a polynomial time algorithm that yields a univeral performance guarantee for the earliest arrival flow problem by using a nonuniform, condensed time-expanded network with intervals of geometrically increasing size.

5.1 Previous work

In the discrete time model, a universally maximal s-t-flow over time can be computed in the time-expanded network by using lexicographically maximal flows introduced by Minieka [28]. A *lexicographically maximal flow* is defined in a static network with multiple sources and/or sinks. There is a strict ordering on the sources and sinks, e.g., $\{\nu_1, \nu_2, \ldots, \nu_k\}$, where ν_i is used here to denote either a source or a sink. A lexicographically maximal flow is a flow that simultaneously maximizes the flow leaving each ordered subset of sources and sinks $\{\nu_1, \nu_2, \ldots, \nu_i\}$, $i = 1, \ldots k$. In the discrete time model, a universally maximal s-t-flow over time with time horizon T is a lexicographically maximal flow in the time-expanded graph with ordering of sources and sinks as

$$s_{T-1}, s_{T-2}, \ldots, s_1, s_0, t_{T-1}, t_{T-2}, \ldots, t_1, t_0 .$$

However, due to the exponential size of the time-expanded network, this insight does not lead to an efficient algorithm for the problem. As mentioned above, the algorithms of Wilkinson [35] and Minieka [28] are based on the Successive Shortest Path Algorithm. These also are not efficient algorithms for computing universally maximal dynamic flows since the Successive Shortest Path Algorithm requires an exponential number of iterations in the worst case; see, e.g., Zadeh [36].

While the result of Wilkinson and Minieka has originally been derived for the discrete time model, it also holds for the continuous time model. In this setting, the existence of universally maximal dynamic flows has been first observed by Philpott [32]. Fleischer and Tardos [11] show how the algorithms for the discrete time model mentioned above can be carried over to the continuous time setting.

In the continuous time model, an equivalent problem is the *universally quickest flow problem* which asks for a flow over time of value d such that the earliest point in time when d' units have arrived at the sink is simultaneously minimized for all $d' \leq d$ and

the earliest point in time when d' units have left the source is simultaneously maximized for all $d' \leq d$.

Hoppe and Tardos [23] compute a single-source single-sink universally maximal dynamic flow where the amount of flow is approximately optimal at any moment of time. Our Lemma 4.7 implies that this algorithm also achieves a universal guarantee.

An earliest arrival flow also exists for the case of multiple sources and a single sink [34]. In the discrete time model, such a flow over time can again be found by a lexicographically maximal flow computation in the time-expanded network. On the other hand, Fleischer [9] presents an instance with two sources and two sinks for which an earliest arrival flow does not exist.

Nonuniform time-expanded networks have been used previously to obtain exact algorithms for the quickest transshipment problem in the setting of zero transit times [9]. Partitioning time into intervals of geometrically increasing size has been used previously in conjunction with dynamic programming to derive approximation algorithms for problems in the area of machine scheduling [19, 4, 5, 1].

5.2   A Sufficiently Fine Discretization of Time

Unfortunately, a lexicographically maximal flow in a time-expanded network does not necessarily yield a universal performance guarantee for the quickest flow problem in the continuous-time model. Although we can interpret a static flow in a time-expanded network as a continuous flow over time (see proof of Lemma 4.1), 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, as the following example shows, it could be significant in the first time intervals.

Example.   The network is a single arc from source s to sink t with capacity 2 and transit time 0. There is a unit of supply at s and a unit of demand at t. A universally quickest flow sends flow from s to t at rate 2 during the interval $[0, \frac{1}{2})$, so that flow arrives at t at rate 2 in the interval $[0, \frac{1}{2})$. Averaging the flow over unit intervals yields a flow that arrives at t at rate 1 in the interval $[0, 1)$. Thus this flow has universal performance guarantee of at most 2. $\square$

The problem illustrated by this example can be resolved as follows. For an arbitrary $\varepsilon > 0$ with $1/\varepsilon$ integral, we discretize time into intervals of size ε. Then, averaging flow within each such interval delays flow that arrives at the sink after time 1 by at most a factor of $1 + \varepsilon$. It thus remains to take care of what happens until time 1.

Notice that flow arriving at the sink before time 1 can only use arcs with transit time 0. In particular, an earliest arrival flow until time 1 can be computed by restricting attention to the subnetwork consisting of these arcs. Hajek and Ogier [17] describe an algorithm that finds an earliest arrival flow in a network with a single sink and zero transit times using $O(n)$ maximum flow computations. Fleischer [9] gives an improved algorithm that solves the problem in the same asymptotic time as one maximum flow computation. Moreover, the analysis of this algorithm shows that the function describing the optimal rate of inflow into the sink is piecewise constant and has at most k

breakpoints $\theta_1, \ldots \theta_k$ where k is the number of sources. These breakpoints are independent of the discretization after time 1 and can be computed without this information in polynomial time. They can then be used to determine the appropriate discretization in $[0, 1)$. Thus, a sufficiently fine discretization of time guarantees that an earliest arrival flow until time 1 can be computed in the corresponding time-expanded network. In the example given above, a discretization of time into intervals of size $1/2$ is sufficient. Together with the observation stated above, this yields the following result.

Lemma 5.1. *Consider an instance of the earliest arrival problem with multiple sources with supply vector D and a single sink and let $\theta_1, \ldots, \theta_k$ be defined as above. Then, a $(1 + \varepsilon)$-approximate earliest arrival flow can be obtained from a lexicographically maximal flow computation in the time-expanded network $\mathcal{N}^T/\Delta$ where Δ is chosen such that $\theta_1, \ldots, \theta_k$ and ε are multiples of Δ.*

By discussion in [9] (specifically, Theorem 3.6 and Theorem 4.1), we can bound Δ from below by $(m \sum_e u_e)^{-k}$. Thus, the size of the resulting time-expanded network $\mathcal{N}^T/\Delta$ is pseudo-polynomial in the input size of the problem. Since time can be rescaled by a factor of $1/\Delta$, we can and will assume without loss of generality that $\Delta = 1$ and thus $\mathcal{N}^T/\Delta = \mathcal{N}^T$. In the next subsection we show how $\mathcal{N}^T$ can be turned into a network of polynomial size while only losing another factor of $1 + \varepsilon$ in the performance guarantee of the resulting polynomial time algorithm.

5.3 Nonuniform Condensed Time-Expanded Networks

As mentioned above, the size of the time-expanded network $\mathcal{N}^T$ is only pseudo-polynomial in the input size. In contrast to the situation for the quickest flow problem discussed in Section 4, using a uniformly rough discretization can lead to a much worse performance guarantee for the earliest arrival flow problem. Instead we will use a nonuniform discretization. In this section we describe a general framework for nonuniform time-expanded networks and establish, where possible, the appropriate generalizations of properties of uniform time-expanded networks.

Overloading the notation introduced in Section 4.1, for a sorted list $L = \langle \theta_0, \ldots, \theta_r \rangle$ with

$$0 \ = \ \theta_0 \ < \ \theta_1 \ < \ \cdots \ < \ \theta_{r-1} \ < \ \theta_r \ = \ T \ ,$$

the *L-time-expanded network* of $\mathcal{N} = (V, A)$, denoted by $\mathcal{N}^L$ is obtained by creating r copies of V, labeled V_0 through V_{r-1}, with the qth copy of node v denoted v_q, for $q = 0, \ldots, r - 1$. For every arc $e = (v, w) \in A$, and for every $q \geq 0$ with $\theta_q + \tau_e \leq \theta_{r-1}$, there is an arc e_q from v_q to $w_{q'_e}$ with

$$q'_e \ := \ \min\{q'' \mid \theta_q + \tau_e \leq \theta_{q''}\} \ . \tag{13}$$

When e is clear from context, we use q' instead of q'_e. The capacity of arc e_q is set to $u_e (\theta_{q+1} - \theta_q)$. In addition, there is a holdover arc from v_q to v_{q+1} with infinite capacity, for all $v \in V$ and $0 \leq q < r - 1$. Notice that this definition generalizes the definition of the T-time-expanded network $\mathcal{N}^T$; in particular, $\mathcal{N}^T = \mathcal{N}^L$ for $L = \langle 0, 1, \ldots, T \rangle$.

Whenever we consider a static flow in $\mathcal{N}^L$, we implicitly assume that t_{r-1} is its only sink; all flow arriving at t_q for $q < r - 1$ is sent to this sink on holdover arcs.

Lemma 5.2. *Let $L = \langle 0 = \theta_0, \ldots, \theta_r \rangle$ with $\theta_q - \theta_{q-1} \leq \theta_{q+1} - \theta_q$, for all $q = 1$ to $r - 1$. Then any static flow in $\mathcal{N}^L$ corresponds to a flow over time in $\mathcal{N}$ such that the amount of flow reaching t in $\mathcal{N}$ by time θ_{q+1}, $q = 0, \ldots, r - 1$, is the same as the amount of flow reaching node t_q in $\mathcal{N}^L$.*

Proof. Let x be a static flow in $\mathcal{N}^L$. We interpret x as a 'generalized' flow over time f' where, in contrast to 'standard' flows over time, transit times can now vary over time. Interpret the flow x_{e_q} on arc $e_q = (v_q, w_{q'})$ as flow f'_e sent at the constant rate $x_{e_q}/(\theta_{q+1} - \theta_q)$ into arc $e = (v, w)$ during the time interval $[\theta_q, \theta_{q+1})$ and arriving at node w during the time interval $[\theta_{q'}, \theta_{q'+1})$ at the constant rate $x_{e_q}/(\theta_{q'+1} - \theta_{q'})$. In particular, in the time interval $[\theta_q, \theta_{q+1})$, the transit time on arc e varies between $\theta_{q'} - \theta_q$ (flow entering the arc at time θ_q) and $\theta_{q'+1} - \theta_{q+1}$ (flow entering the arc immediately before time θ_{q+1}) in f'. Thus, since the sizes of the time intervals $[\theta_q, \theta_{q+1})$, $q = 0, \ldots, r - 1$, are nondecreasing and by choice of q', the transit time on arc e in f' is never smaller than τ_e.

By design, f' obeys flow rate capacity constraints and flow conservation. Moreover, for $q = 0, \ldots, r - 1$, the amount of flow reaching t in f' by time θ_{q+1} is the same as the amount of flow reaching node t_q in x. Finally, since transit times in f' are always lower bounded by the actual transit times τ_e of arcs $e \in A$ (by (13)), it can easily be interpreted as a regular flow over time f in $\mathcal{N}$ by introducing appropriate waiting times for every infinitesimal flow unit at the head of each arc. $\square$

Lemma 5.3. *Let $L = \langle 0 = \theta_0, \ldots, \theta_r \rangle$ with $\theta_q - \theta_{q-1} \leq \theta_{q+1} - \theta_q$, for all $q = 1$ to $r - 1$. Let f be any flow over time that completes by time θ_r in the network $\mathcal{N}$ modified so that all transit times are increased by $\Delta := \theta_r - \theta_{r-1}$. Then, f induces a feasible static flow in $\mathcal{N}^L$ of the same value.*

Proof. For all $e \in A$ and $0 \leq q \leq r - 1$, define x_{e_q} to be the total f-flow into e in interval $[\theta_q, \theta_{q+1})$. Since f is feasible, it obeys the capacity constraints; then by construction, so does x. By design, the value of x equals the value of f.

To establish flow conservation, consider the path $P = \langle w^0, w^1, \ldots, w^h = t \rangle$ travelled by an infinitesimal unit of flow in f. This infinitesimal unit of flow arrives at w_i at time φ_i and leaves w_i at some time $\xi_i \geq \varphi_i$. We argue that the corresponding unit of flow in x arrives at $w_{p_i}^i$ for some p_i satisfying $\theta_{p_i} \leq \varphi_i$. Thus it is available to be sent along P in $\mathcal{N}^L$ on arc $(w_{q_i}^i, w_{q_i'}^{i+1})$, where q_i is the maximum index ℓ satisfying $\theta_\ell \leq \xi_i$. This will show that x satisfies flow conservation.

Set $q := q_{i-1}$. By design of x, the infinitesimal unit of flow leaves w_q^{i-1} in $\mathcal{N}^L$ and thus arrives at $w_{q'}^i$, where q' is defined according to (13). By definition of q', we have that

$$
\begin{aligned}
\theta_{q'} - \theta_q &= \theta_{q'-1} - \theta_q + (\theta_{q'} - \theta_{q'-1}) \\
&< \tau_{(w^{i-1}, w^i)} + (\theta_r - \theta_{r-1}) = \tau_{(w^{i-1}, w^i)} + \Delta \ .
\end{aligned}
$$

This yields $\theta_{q'} < \theta_q + \tau_{(w^{i-1}, w^i)} + \Delta \leq \xi_{i-1} + \tau_{(w^{i-1}, w^i)} + \Delta = \varphi_i \leq \xi_i$. (The second inequality follows by definition of $q := q_{i-1} \leq \xi_{i-1}$ at the end of the previous paragraph; the subsequent equality follows by definition of φ_i.) This implies that $\theta_{q'} \leq \theta_{q_i}$, and thus the flow obeys conservation constraints. $\square$

5.4 An Approximation Scheme

To obtain a $(1 + \varepsilon)$-universal guarantee for the earliest arrival flow problem, we use a geometrically increasing time discretization. Let $p := \lceil 4n/\varepsilon^2 \rceil$ and define the list

$$
\begin{aligned}
L \ := \ \langle & 0, 1, 2, \ldots, 2p, \\
& 2(p+1), 2(p+2), \ldots, 4p, \\
& 4(p+1), 4(p+2), \ldots, 8p, \\
& \ldots \\
& 2^{\ell-1}(p+1), 2^{\ell-1}(p+2), \ldots, 2^{\ell}p \rangle \ .
\end{aligned}
$$

Here $\ell \in \mathbb{N}_0$ is chosen such that the resulting time-expanded network $\mathcal{N}^L$ is large enough to allow for a $(1 + O(\varepsilon))$-approximate time horizon. To be more precise, we choose the smallest ℓ such that $2\,T^* \le 2^{\ell}p$ where T^* is the time horizon of an optimal flow over time. Notice that the length of the list L is in $O(p \log T^*)$ and hence polynomially bounded in the input size and $1/\varepsilon$. It thus remains to be shown that a lexicographically maximal flow in the corresponding *geometrically-condensed time-expanded network* $\mathcal{N}^L$ yields a flow over time with universal performance guarantee $1 + O(\varepsilon)$.

Lemma 5.4. *A lexicographically maximal flow in $\mathcal{N}^L$ induces a flow over time in $\mathcal{N}$ with universal performance guarantee $1 + O(\varepsilon)$.*

Proof. Let $T \le T^*$ be the minimal time required to send $d' \le d$ units of flow to the sink. We consider the time-expanded network $\mathcal{N}^{L'}$ defined by the sublist $L' := \langle \theta_0, \ldots, \theta_{r'} \rangle$ of L with $r' := \min\{q \mid \theta_q \ge (1 + \varepsilon + \varepsilon^2/2)(1 + \varepsilon)\,T\}$. In other words, $\mathcal{N}^{L'}$ is obtained from $\mathcal{N}^L$ by removing all time layers V_q with $q \ge r'$ and all incident arcs. We show below that there is a flow in $\mathcal{N}^{L'}$ of value d'. Then, since in a lexicographically maximum flow in $\mathcal{N}^L$ at least d' units of flow reach $t_{r'-1}$, the corresponding flow over time in $\mathcal{N}$ sends at least d' to t by time $\theta_{r'}$ by Lemma 5.2. Hence this flow yields a universal performance guarantee of $1 + O(\varepsilon)$.

Let $\Delta' = \theta_{r'} - \theta_{r'-1}$. By choice of p and since we can assume that $\theta_{r'} \le 2T$, we have that

$$
\Delta' \ = \ \theta_{r'} - \theta_{r'-1} \ \le \ \theta_{r'}/p \ \le \ \varepsilon^2 T/(2n) \ . \tag{14}
$$

Since there is flow of value d' in $\mathcal{N}$ by time T, Lemma 4.7 implies that there is flow of value $d'(1 + \varepsilon)$ in $\mathcal{N}$ by time $T(1 + \varepsilon)$. Then, Theorem 4.6 together with (14) implies that there is flow of value d' that completes by time $T(1 + \varepsilon + \varepsilon^2/2)(1 + \varepsilon)$ in the network $\mathcal{N}$ with all transit times increased by Δ'. By Lemma 5.3, this implies that there is flow of value d' in $\mathcal{N}^{L'}$. $\qquad\square$

We have now established the main result of this section.

Theorem 5.5. *There exists a fully polynomial time approximation scheme for the problem of computing an earliest arrival flow in a network with multiple sources and a single sink.*

Acknowledgment

We thank Dan Stratila for his helpful comments.

References

[1] F. Afrati, E. Bampis, C. Chekuri, D. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein, and M. Sviridenko. Approximation schemes for minimizing average weighted completion time with release dates. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 32–43, New York City, NY, 1999.

[2] J. E. Aronson. A survey of dynamic network flows. *Annals of Operations Research*, 20:1–66, 1989.

[3] R. E. Burkard, K. Dlaska, and B. Klinz. The quickest flow problem. *ZOR — Methods and Models of Operations Research*, 37:31–58, 1993.

[4] S. Chakrabarti, C. Phillips, A. S. Schulz, D. B. Shmoys, C. Stein, and J. Wein. Improved scheduling algorithms for minsum criteria. In F. Meyer auf der Heide and B. Monien, editors, *Automata, Languages and Programming*, volume 1099 of *Lecture Notes in Computer Science*, pages 646–657. Springer, Berlin, 1996.

[5] F. A. Chudak and D. B. Shmoys. Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds. *Journal of Algorithms*, 30:323–343, 1999.

[6] L. Fleischer and M. Skutella. The quickest multicommodity flow problem. In W. J. Cook and A. S. Schulz, editors, *Integer Programming and Combinatorial Optimization*, volume 2337 of *Lecture Notes in Computer Science*, pages 36–53. Springer, Berlin, 2002.

[7] L. Fleischer and M. Skutella. Minimum cost flows over time without intermediate storage. In *Proceedings of the 14th Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 66–75, Baltimore, MD, 2003.

[8] L. K. Fleischer. Approximating fractional multicommodity flows independent of the number of commodities. *SIAM Journal on Discrete Mathematics*, 13:505–520, 2000.

[9] L. K. Fleischer. Faster algorithms for the quickest transshipment problem. *SIAM Journal on Optimization*, 12:18–35, 2001.

[10] L. K. Fleischer. Universally maximum flow with piece-wise constant capacity functions. *Networks*, 38:1–11, 2001.

[11] L. K. Fleischer and É. Tardos. Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, 23:71–80, 1998.

[12] L. R. Ford and D. R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6:419–433, 1958.

[13] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, 1962.

[14] D. Gale. Transient flows in networks. *Michigan Mathematical Journal*, 6:59–63, 1959.

[15] 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.

[16] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, Berlin, 1988.

[17] B. Hajek and R. G. Ogier. Optimal dynamic routing in communication networks with continuous traffic. *Networks*, 14:457–487, 1984.

[18] A. Hall, S. Hippler, and M. Skutella. Multicommodity flows over time: Efficient algorithms and complexity. In J. K. Lenstra and G. J. Woeginger, editors, *Automata, Languages and Programming*, Lecture Notes in Computer Science. Springer, Berlin, 2003. To appear.

[19] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of Operations Research*, 22:513–544, 1997.

[20] G. Handler and I. Zang. A dual algorithm for the constrained shortest path problem. *Networks*, 10:293–310, 1980.

[21] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17:36–42, 1992.

[22] B. Hoppe. *Efficient dynamic network flow algorithms*. PhD thesis, Cornell University, 1995.

[23] 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.

[24] B. Hoppe and É. Tardos. The quickest transshipment problem. *Mathematics of Operations Research*, 25:36–62, 2000.

[25] 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.

[26] D. H. Lorenz and D. Raz. A simple efficient approximation scheme for the restricted shortest path problem. *Operations Research Letters*, 28:213–219, 2001.

[27] N. Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4:414–424, 1979.

[28] E. Minieka. Maximal, lexicographic, and dynamic network flows. *Operations Research*, 21:517–527, 1973.

[29] R. G. Ogier. Minimum-delay routing in continuous-time dynamic networks with piecewise-constant capacities. *Networks*, 18:303–318, 1988.

[30] J. B. Orlin. Minimum convex cost dynamic network flows. *Mathematics of Operations Research*, 9:190–207, 1984.

[31] 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.

[32] A. B. Philpott. Continuous-time flows in networks. *Mathematics of Operations Research*, 15:640–661, 1990.

[33] 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.

[34] D. Richardson and É. Tardos, 2002. Personal communication.

[35] W. L. Wilkinson. An algorithm for universal maximal dynamic flows in a network. *Operations Research*, 19:1602–1612, 1971.

[36] N. Zadeh. A bad network problem for the simplex method and other minimum cost flow algorithms. *Mathematical Programming*, 5:255–266, 1973.