

Multicommodity Flows Over Time: Efficient Algorithms and Complexity*

Alex Hall^{1,**}, Steffen Hippler², and Martin Skutella^{3,***}

¹ Computer Engineering and Networks Laboratory (TIK)
Gloriastrasse 35, ETH Zentrum, 8092 Zurich, Switzerland
hall@tik.ee.ethz.ch

² Institut für Mathematik, Technische Universität Berlin
Straße des 17. Juni 136, 10623 Berlin, Germany
hippler@math.tu-berlin.de

³ Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany
skutella@mpi-sb.mpg.de

Abstract. Flow variation over time is an important feature in network flow problems arising in various applications such as road or air traffic control, production systems, communication networks (e.g., the Internet), and financial flows. The common characteristic are networks with capacities and transit times on the arcs which specify the amount of time it takes for flow to travel through a particular arc. Moreover, in contrast to static flow problems, flow values on arcs may change with time in these networks.

While the ‘maximum s - t -flow over time’ problem can be solved efficiently and ‘min-cost flows over time’ are known to be NP-hard, the complexity of (fractional) ‘multicommodity flows over time’ has been open for many years. We prove that this problem is NP-hard, even for series-parallel networks, and present new and efficient algorithms under certain assumptions on the transit times or on the network topology. As a result, we can draw a complete picture of the complexity landscape for flow over time problems.

Keywords: network flow, routing, flow over time, dynamic flow, complexity, efficient algorithm

1 Introduction

A crucial characteristic of network flows occurring in real-world applications is flow variation over time. This characteristic is not captured by classical ‘static’ network flow models known from the literature. Moreover, apart from the effect that flow values on arcs may change over time, there is a second temporal dimension in many applications: Usually, flow does not travel instantaneously through a network but requires a certain amount of time to travel through each arc. Thus, not only the amount of flow to be transmitted but also the time needed for the transmission plays an essential role. Various interesting examples can be found in the survey articles of Aronson [1] and Powell, Jaillet, and Odoni [15].

The Model. Ford and Fulkerson [7, 8] introduce the notion of *flows over time* (also called ‘dynamic flows’) which comprises both temporal features mentioned above. They consider networks (directed

* An extended abstract of this work appeared in the conference proceedings [10].

** Supported by the joint Berlin/Zurich graduate program Combinatorics, Geometry, and Computation (CGC), financed by ETH Zurich and the German Science Foundation (DFG)

*** Supported in part by the EU Thematic Networks APPOL I & II, Approximation and Online Algorithms, IST-1999-14084, IST-2001-30012.

graphs) $G = (V, E)$ with *capacities* u_e and *transit times* τ_e on the arcs $e \in E$. The transit time τ_e 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 time⁴. In this setting, the capacity u_e 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. 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 in Section 2.

Results from the Literature. Ford and Fulkerson [7, 8] observe that a flow-over-time problem in a given network with transit times on the arcs can be transformed into an equivalent static flow problem in the corresponding *time-expanded network*. The time-expanded network contains one copy of the node set of the underlying network for each discrete time step θ (building a *time layer*). Moreover, for each arc e with transit time τ_e in the given network, there is a copy between each pair of time layers of distance τ_e in the time-expanded network. Thus, a discrete flow over time in the given network can be interpreted as a static flow in the corresponding time-expanded network. Since this interrelation works in both directions, the concept of time-expanded networks allows to solve a variety of flow over time problems by applying algorithmic techniques developed for static network flows; see, e.g., [6]. Notice, however, that one has to pay for this simplification of the considered flow problem in terms of an enormous increase in the size of the network. In particular, the size of the time-expanded network is only pseudo-polynomial in the input size and thus does not directly lead to efficient algorithms for computing flows over time.

Ford and Fulkerson [7, 8] give an efficient algorithm for the problem of sending the maximum possible amount of flow from one source s to one sink t within a given time horizon T . The problem can be solved by essentially one ‘static’ 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 s - t -paths. The corresponding flow over time starts to send flow on each path at time zero, and repeats each so long as there is enough time left in the T time units for the flow along the path to arrive at the sink. A flow over time featuring this structure is called *temporally repeated*.

A problem closely related to the one considered by Ford and Fulkerson is the *quickest s - t -flow problem*. Here, instead of fixing the time horizon T and asking for a flow over time of maximal value, the value of the flow (demand) is fixed and T is to be minimized. This problem can be solved in polynomial time by incorporating the algorithm of Ford and Fulkerson into a binary search framework. Using Megiddo’s method of parametric search [14], Burkard, Dlaska, and Klinz [2] present a faster algorithm which solves the quickest s - t -flow problem in strongly polynomial time.

Hoppe and Tardos [12, 11] study the *quickest transshipment problem* which asks for a flow over time satisfying given supplies and demands at the nodes of a network within minimum time. Surprisingly, this problem turns out to be much harder than the special case with a single source and sink. Hoppe and Tardos give a polynomial time algorithm for the problem, but this algorithm relies on submodular function minimization and is thus much less efficient than for example the algorithm of Ford and Fulkerson for maximum s - t -flows over time.

⁴ In fact, the discrete flow model considered by Ford and Fulkerson is slightly different from the model we consider in this paper. However, Fleischer and Tardos [6] point out that the two models are essentially equivalent; see also [4].

Even more surprising, Klinz and Woeginger [13] show that the problem of computing a minimum cost s - t -flow over time with prespecified value and time horizon is NP-hard. On the other hand, this problem can be solved in pseudo-polynomial time by a static min-cost flow computation in the time-expanded network. Klinz and Woeginger also point out that the class of temporally repeated flows does in general not contain a min-cost s - t -flow over time. In fact, it is even strongly NP-hard to compute a temporally repeated solution of minimum cost [13].

Fleischer and Skutella [4, 5] introduce a ‘condensed’ variant of time-expanded networks which is based on a rougher discretization of time and therefore leads to networks whose size is polynomially bounded in the input size. This approach yields fully polynomial time approximation schemes (FPTASes) for various variants of the weakly NP-hard quickest flow problem with costs. The best known result for the strongly NP-hard problem of computing a quickest temporally repeated flow of minimum cost is a $(2 + \epsilon)$ -approximation algorithm [4], which is based on a length-bounded static flow computation.

Contribution of this Paper. The results in [4, 5] also hold for the more general setting with multiple commodities. Multicommodity flows model the transportation of several distinct types of flow through a single network. The resulting problems are typically much harder than their single-commodity counterparts. For example, the only known polynomial-time algorithms for static multicommodity flow computations require general linear programming techniques (e.g., the ellipsoid method or interior point methods). The complexity of the multicommodity flow over time problem (without costs) has so far been open. Hoppe [11] poses the problem of developing a polynomial time algorithm to solve fractional multicommodity flows over time. In Section 5, we prove that such an algorithm does not exist, unless $P=NP$. In fact, the multicommodity flow over time problem is NP-hard, even when restricted to series-parallel networks or to the case of only two commodities.

Flows over time raise issues that do not arise in standard network flows. One issue is storage of flow 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, storage is unnecessary, even in the NP-hard setting with costs [5]. However, for the quickest multicommodity flow problem, there exist instances where the time horizon of an optimal solution increases by a factor of $4/3$ when storage of flow at intermediate nodes is prohibited [4]. In Section 6 we prove that the multicommodity flow over time problem with simple flow paths and without storage of flow at intermediate nodes is strongly NP-hard. Without the latter restriction the problem can be solved in pseudo-polynomial time as a static multicommodity flow problem in the time-expanded network. The best known result for the strongly NP-hard variant with simple flow paths and no intermediate storage is a 2-approximation algorithm for the quickest multicommodity flow problem [4]. An overview of the complexity landscape of flows over time is given in Table 1.

Motivated by the results on the hardness of multicommodity flows over time in Sections 5 and 6, we study special conditions on the transit times of arcs and on the network topology under which multicommodity flows over time can be computed in polynomial time. In Section 3 we consider arbitrary network topologies with transit times on the arcs satisfying the following condition: All paths (in the corresponding bidirected network) between every fixed pair of nodes have the same transit time. This condition is, for example, obviously satisfied for the important but non-trivial case of tree networks. We show that, under this assumption, many flow over time problems can be solved as static flow problems in a polynomial-size variant of time-expanded networks with $O(n)$

Table 1. The complexity landscape of flows over time in comparison to the corresponding static flow problems. The third column ‘transshipment’ refers to single-commodity flows with several source and sink nodes. The NP-hardness results marked with a ‘*’ are proved in this paper. The weak NP-hardness results even hold for series-parallel networks. On the other hand, we prove that these problems can efficiently be solved in tree networks and arbitrary networks with ‘uniform path-lengths’. The ‘pseudo-poly’ entries follow since the respective problems can be solved as static flow problems in the time-expanded network. The quoted approximation results hold for the corresponding quickest flow problems.

	s - t -flow	transshipment	min-cost flow	multicommodity flow
(static) flow	poly	poly ($\simeq s$ - t -flow)	poly	poly (\simeq LP)
flow over time with storage	poly [7] (\simeq min-cost flow)	poly [12] (\simeq subm. func.)	pseudo-poly NP-hard [13] FPTAS [5]	pseudo-poly NP-hard* FPTAS [5]
flow over time without storage			FPTAS [5]	strongly NP-hard* 2-approx. [4]

time layers ($n := |V|$). We believe that this result is also of interest for flow over time problems, like the quickest transshipment problem, which are known to be solvable in polynomial time for arbitrary transit times. While the algorithm of Hoppe and Tardos [12, 11] relies on submodular function minimization, we can solve the special case of the problem as a static s - t -flow problem in a network with $O(n^2)$ nodes and $O(nm)$ arcs ($m := |E|$). The presented approach works for both settings, with and without storage of flow at intermediate nodes.

Finally, in Section 4 we consider networks with arbitrary transit times where every node has at most one outgoing arc. In particular, there is a unique source-sink path for every commodity in such networks. Under the assumption that storage of flow at intermediate nodes is allowed, we present a simple greedy algorithm for the quickest multicommodity flow problem: Whenever there is a conflict between several commodities using the same arc, the algorithm gives top priority to the commodity which is furthest from its sink node. We prove that this simple strategy yields an optimal solution in polynomial time. The proof uses a generalized notion of ‘earliest arrival flows’.

2 Preliminaries

We are considering network flow problems in a network (directed graph) $G = (V, E)$ with $n := |V|$ nodes and $m := |E|$ arcs. For an arc $e = (v, w)$ we write $\text{head}(e) := w$ and $\text{tail}(e) := v$. For a node $v \in V$, the terms $\delta^+(v)$ and $\delta^-(v)$ denote the set of arcs leaving node v ($\text{tail}(e) = v$) and entering node v ($\text{head}(e) = v$), respectively. Each arc $e \in E$ has associated with it a positive capacity u_e and a nonnegative transit time $\tau_e \in \mathbb{R}^+$. Moreover, in the setting with costs, each arc e has associated a non-negative cost coefficient c_e which determines the per unit cost for sending flow through the arc.

There is a set of commodities $K = \{1, \dots, k\}$, where every commodity $i \in K$ is defined by a set of terminals $S_i \subseteq V$ which can be partitioned into a subset of sources S_i^+ and sinks S_i^- . Every source node $v \in S_i^+$ has a supply $D_{v,i} \geq 0$ and every sink $v \in S_i^-$ has a demand $D_{v,i} \leq 0$ such that $\sum_{v \in S_i} D_{v,i} = 0$. In the special case of only one source $s_i \in V$ and one sink $t_i \in V$ we let $d_i := D_{s_i,i} = -D_{t_i,i}$ and refer to d_i as the *demand* of commodity i .

Static Flows. A static (multicommodity) flow x in G assigns every arc-commodity pair e, i a non-negative flow value $x_{e,i}$ such that *flow conservation* holds:

$$\sum_{e \in \delta^-(v)} x_{e,i} - \sum_{e \in \delta^+(v)} x_{e,i} = 0 \quad \text{for all } v \in V \setminus S_i \text{ and } i \in K. \quad (1)$$

The static flow x satisfies the supplies and demands if

$$\sum_{e \in \delta^+(v)} x_{e,i} - \sum_{e \in \delta^-(v)} x_{e,i} = D_{v,i} \quad \text{for all } v \in S_i \text{ and } i \in K.$$

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 $e \in E$. The cost of a static flow x is defined as $c(x) := \sum_{e \in E} c_e x_e$.

Flows Over Time. A (multicommodity) flow over time f in G with time horizon T is given by Lebesgue-measurable functions $f_{e,i} : [0, T) \rightarrow \mathbb{R}^+$ where $f_{e,i}(\theta)$ is the rate of flow (per time unit) of commodity i entering arc e at time θ . 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.

The flow $f_{e,i}(\theta)$ of commodity i entering arc e at time θ arrives at $\text{head}(e)$ at time $\theta + \tau_e$. All arcs must be empty from time T on, i.e., $f_{e,i}(\theta) = 0$ for $\theta \geq T - \tau_e$. To generalize the notion of *flow conservation* to flows over time, we integrate the flow conservation constraints (1) over time. Depending on the model, *storage of flow at intermediate nodes* might be allowed. That is, flow entering a node can be held back for some time before it is sent onward. To rule out deficit at any node, we require that, for all $i \in K, \theta \in [0, T)$,

$$\int_0^\theta \left(\sum_{e \in \delta^+(v)} f_{e,i}(\xi) - \sum_{e \in \delta^-(v)} f_{e,i}(\xi - \tau_e) \right) d\xi \leq \begin{cases} 0 & \text{for } v \in V \setminus S_i^+, \\ D_{v,i} & \text{for } v \in S_i^+. \end{cases} \quad (2)$$

Moreover, we require that equality holds in (2) for $i \in K, \theta = T$, and $v \in V \setminus S_i$, meaning that no flow should remain in the network after time T . In the model without storage of flow at intermediate nodes, we additionally require that equality holds in (2) for all $i \in K, \theta \in [0, T)$, 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 $D_{v,i}$:

$$\int_0^T \left(\sum_{e \in \delta^+(v)} f_{e,i}(\xi) - \sum_{e \in \delta^-(v)} f_{e,i}(\xi - \tau_e) \right) d\xi = D_{v,i} \quad \text{for all } i \in K, v \in S_i. \quad (3)$$

In the setting of flows over time, the capacity u_e is an upper bound on the *rate* of flow entering arc e at any moment of time. Thus, a flow over time f is feasible if $f_e(\theta) \leq u_e$ for all $\theta \in [0, T)$ and $e \in E$. Here, $f_e(\theta) := \sum_{i \in K} f_{e,i}(\theta)$ is the total rate at which flow is entering arc e at time θ . The cost of a flow over time f is defined as $c(f) := \sum_{e \in E} c_e \int_0^T f_e(\theta) d\theta$.

Problem Definition. Given a network G with capacities and transit times on the arcs, a set of commodities with supplies and demands at their terminals, and a time horizon T , the *multicommodity flow over time problem* asks for a feasible flow over time with time horizon T , satisfying all supplies

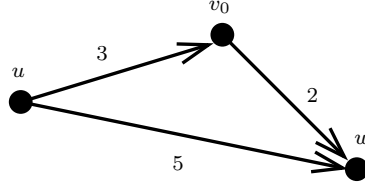


Fig. 1. Example of a network with uniform path-lengths. The numbers at the arcs indicate transit times.

and demands. In the setting with costs, the *min-cost (multicommodity) flow over time problem* asks for such a flow over time with minimum cost. Another interesting objective for flows over time is to minimize the time horizon: The *quickest (multicommodity) flow problem (with costs)* is to find a flow over time in G that satisfies all supplies and demands within minimal time T (and whose cost is bounded by a given *budget* C). Finally, in the *maximum (multicommodity) flow over time problem (with costs)* we are given a time horizon T and instead of having supplies and demands at the terminals, the goal is to maximize the total amount of flow being sent from sources to sinks (under the condition that the costs are bounded by a given *budget* C). Of course, flow of each commodity can only be sent from its sources to its sinks.

3 Networks with Uniform Path-Lengths

In this section we present a polynomial-time algorithm for the min-cost multicommodity flow over time problem in a special class of networks which, in particular, comprises trees. Even on trees the multicommodity flow over time problem is far from being trivial. For example, it follows by a straightforward reduction from the *wavelength routing problem* [3] that finding an *integral* multicommodity flow over time is NP-hard in binary trees.

For a given network $G = (V, E)$ with transit times on the arcs, we let $\overleftrightarrow{G} = (V, \overleftrightarrow{E})$ denote the corresponding bidirected network with $\overleftrightarrow{E} := E \cup \overleftarrow{E}$ and $\overleftarrow{E} := \{(v, u) \mid (u, v) \in E\}$. Moreover, the ‘transit time’ of a backward arc $(v, u) \in \overleftarrow{E}$ is set to $\tau_{(v,u)} := -\tau_{(u,v)}$. In the following we assume that G is connected and that the transit time of every directed cycle in \overleftrightarrow{G} is zero. The latter requirement is equivalent to the condition that, for all $u, v \in V$, the transit time of any two u - v -paths in \overleftrightarrow{G} is equal. Therefore we refer to this class of networks with transit times as *networks with uniform path-lengths*. An example is given in Figure 1.

Let $v_0 \in V$ be an arbitrary but fixed node. For $v \in V$ let τ_v denote the transit time of a v - v_0 -path in \overleftrightarrow{G} . In the network depicted in Figure 1, these values are $\tau_u = 3$, $\tau_{v_0} = 0$, and $\tau_w = -2$. For a given time horizon T , let

$$\mathcal{T} := \{\tau_v, T + \tau_v \mid v \in V\} .$$

For the network in Figure 1 and $T = 7$ we get $\mathcal{T} = \{3, 10, 0, 7, -2, 5\}$. Notice that τ_v is the earliest point in time at which flow emerging from node v could possibly arrive at v_0 . Similarly, $T + \tau_v$ is the latest point in time at which flow can be sent from v_0 to v such that it still arrives in time, i.e., before time T . Hence, \mathcal{T} contains all ‘essential’ points in time at which decisions have to be made at node v_0 . More precisely, we show below (Lemma 1) that it is sufficient to change the outflow rate out of arcs arriving at v_0 and the inflow rate into arcs leaving v_0 at these points in time only. The

same property holds for all other nodes $v \in V$ and incident arcs when \mathcal{T} is replaced by

$$\mathcal{T} - \tau_v := \{\theta - \tau_v \mid \theta \in \mathcal{T}\} .$$

Since $|\mathcal{T}| \leq 2|V|$, this insight constitutes the backbone of the results presented in this section.

We introduce some additional notation: Sort the elements in \mathcal{T} such that $\theta_1 < \theta_2 < \dots < \theta_q$. Moreover, let $\theta_0 := -\infty$ and $\theta_{q+1} := \infty$, and define a corresponding partition of the time axis $[-\infty, \infty)$ by time intervals $I_j := [\theta_j, \theta_{j+1})$, $j = 0, \dots, q$. In Figure 1 with $T = 7$ we get $q = 6$ and the time intervals $I_0 = [-\infty, -2)$, $I_1 = [-2, 0)$, $I_2 = [0, 3)$, $I_3 = [3, 5)$, $I_4 = [5, 7)$, $I_5 = [7, 10)$, and $I_6 = [10, \infty)$.

Lemma 1. *If there exists a flow over time f with time horizon T satisfying all supplies and demands, then there exists a corresponding solution \bar{f} with $c(\bar{f}) = c(f)$ and the following additional property: For every arc $e = (u, v) \in E$ and every time interval $I_j - \tau_u := [\theta_j - \tau_u, \theta_{j+1} - \tau_u)$, the flow rate $\bar{f}_{e,i}(\theta)$ is constant for $\theta \in I_j - \tau_u$, for every commodity $i \in K$.*

It follows from the property of \bar{f} stated in Lemma 1 that also the inflow rate $\bar{f}_{e,i}(\theta - \tau_e)$ into node v on arc $e = (u, v)$ at time θ is constant for $\theta \in I_j - \tau_v$. By definition $\tau_u = \tau_v + \tau_e$ such that $\theta \in I_j - \tau_v$ if and only if $\theta - \tau_e \in I_j - \tau_u$.

Proof. For $e = (u, v) \in E$ and $i \in K$, define

$$\bar{f}_{e,i}(\theta) := \frac{1}{|I_j|} \int_{I_j - \tau_u} f_{e,i}(\xi) d\xi \quad \text{for } \theta \in I_j - \tau_u, j = 0, \dots, q.$$

Here, $|I_j| := \theta_{j+1} - \theta_j$ denotes the length of the time intervals I_j and $I_j - \tau_u$. Hence, \bar{f} arises from f by averaging flow on arcs $e = (u, v)$ within time intervals $I_j - \tau_u$, $j = 0, \dots, q$.

By definition of \mathcal{T} , each time interval $I_j - \tau_u$ is either contained in $[0, T - \tau_e)$ or disjoint from $[0, T - \tau_e)$, for all $e = (u, v) \in E$. This is clear since $\mathcal{T} - \tau_u$ contains both 0 and $T - \tau_e$. (For instance, for $T = 7$ and node u in Figure 1, the intervals $I_j - \tau_u$, $j = 0, \dots, q$, are $[-\infty, -5)$, $[-5, -3)$, $[-3, 0)$, $[0, 2)$, $[2, 4)$, $[4, 7)$, and $[7, \infty)$.) In particular, for all $e \in E$ and $i \in K$, we get $\bar{f}_{e,i}(\theta) = 0$ for $\theta \notin [0, T - \tau_e)$ since this property certainly holds for the given solution f .

Moreover, it follows from the definition of \bar{f} that no flow is rerouted compared to f . Thus, \bar{f} satisfies all supplies and demands (see (3)) and its cost is equal to the cost of f . It is easy to see that \bar{f} satisfies the capacity constraints since f does. It therefore remains to show that \bar{f} satisfies the flow conservation constraints (2).

Consider a commodity $i \in K$ and a node $v \in V$. Notice that, by definition of \bar{f} , flow of commodity i leaving node v on some outgoing arc is averaged within each time interval $I_j - \tau_v$. Moreover, the same observation applies to flow entering node v on some incoming arc $e = (u, v)$ since by definition $\tau_u = \tau_v + \tau_e$ and thus $I_j - \tau_v - \tau_e = I_j - \tau_u$. Hence, the overall flow balance in node v is averaged during time intervals $I_j - \tau_v$. As a consequence, \bar{f} fulfills flow conservation since f does. In the remainder of the proof, we give a more precise but less intuitive argument for the latter conclusion.

Consider again a commodity $i \in K$ and a node $v \in V \setminus S_i^+$ (the following argument easily extends to nodes $v \in S_i^+$). By definition of \bar{f} , the left hand side of (2) is equal for f and \bar{f} if $\theta := \theta_j - \tau_v$ for some j . To see this, note that $\theta_j - \tau_v - \tau_e = \theta_j - \tau_u =: \theta'$, for all arcs $e = (u, v) \in \delta^-(v)$

and thus θ' is on an interval boundary for u as well. Let $\delta_{v,i,j}$ denote the (non-positive) left hand side of (2) for \bar{f} and $\theta := \theta_j - \tau_v$. Then, for $\theta_j - \tau_v < \theta < \theta_{j+1} - \tau_v$, the left hand side of (2) for \bar{f} is a convex combination of $\delta_{v,i,j}$ and $\delta_{v,i,j+1}$ and therefore non-positive as well. This concludes the proof. \square

It follows from the last part of the proof of Lemma 1 that storage of flow at intermediate nodes does not occur in \bar{f} if it does not occur in f .

Corollary 1. *Lemma 1 also holds if storage of flow at intermediate nodes is prohibited.*

The min-cost multicommodity flow over time problem can now be formulated as a linear program of polynomial size. For $e \in E$, $i \in K$, and $j = 1, \dots, q-1$, the variable $x_{e,i,j}$ denotes the amount of flow of commodity i that is sent into arc $e = (u, v)$ during the time interval $I_j - \tau_u$ at constant rate $x_{e,i,j}/|I_j|$.

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e \sum_{j=1}^{q-1} x_{e,i,j} \\ \text{s.t.} \quad & \sum_{j=1}^{q-1} \left(\sum_{e \in \delta^+(v)} x_{e,i,j} - \sum_{e \in \delta^-(v)} x_{e,i,j} \right) = D_{v,i} \quad \text{for all } i \in K, v \in S_i, \end{aligned} \quad (4)$$

$$\sum_{j=1}^p \left(\sum_{e \in \delta^+(v)} x_{e,i,j} - \sum_{e \in \delta^-(v)} x_{e,i,j} \right) \leq 0 \quad \text{for all } i \in K, v \in V \setminus S_i^+, \text{ and } p = 1, \dots, q-1, \quad (5)$$

$$\sum_{j=1}^p \left(\sum_{e \in \delta^+(v)} x_{e,i,j} - \sum_{e \in \delta^-(v)} x_{e,i,j} \right) \leq D_{v,i} \quad \text{for all } i \in K, v \in S_i^+, \text{ and } p = 1, \dots, q-1, \quad (6)$$

$$\sum_{i \in K} x_{e,i,j} \leq |I_j| u_e \quad \text{for all } e \in E, j = 1, \dots, q-1, \quad (7)$$

$$x_{e,i,j} = 0 \quad \text{for all } e = (u, v) \in E, i \in K, \text{ and } I_j - \tau_u \not\subseteq [0, T - \tau_e], \quad (8)$$

$$x_{e,i,j} \geq 0 \quad \text{for all } e \in E, i \in K, \text{ and } j = 1, \dots, q-1.$$

Constraints (4) correspond to (3) and enforce the satisfaction of all supplies and demands. Constraints (5) and (6) are a reformulation of the flow conservation constraints (2). In particular, replacing “ \leq ” by “ $=$ ” in (5) yields a formulation for the model where storage of flow at intermediate nodes is prohibited. Constraints (7) correspond to the capacity constraints. Finally, constraints (8) ensure that flow can only occur within the time interval $[0, T]$.

Since linear programs can be solved efficiently (e.g., by interior point methods), we get the following main result of this section.

Theorem 1. *The min-cost multicommodity flow over time problem (with or without storage of flow at intermediate nodes) in networks with uniform path-lengths can be solved in polynomial time.*

While this result relies on general linear programming techniques, we can give more efficient algorithms for the special case of a single commodity. These algorithms are based on the insight

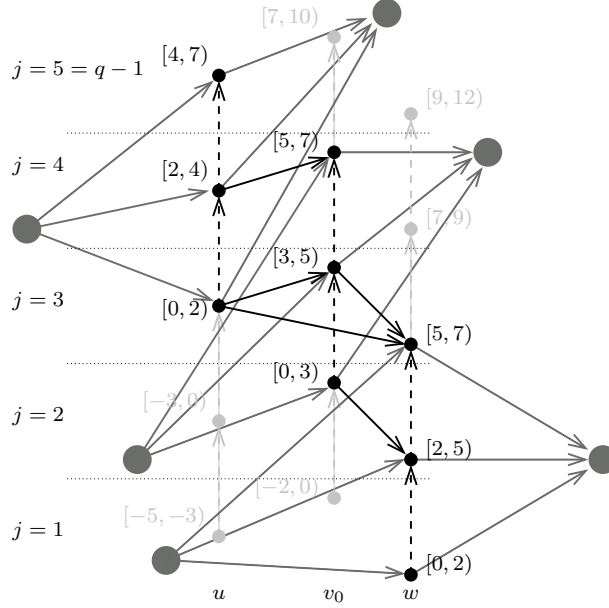


Fig. 2. The network G' for the underlying network G given in Figure 1 and $T = 7$. We assume that every node in G is both a source and a sink for some commodities. The six thick nodes are the corresponding super-sources and super-sinks. The dashed arcs are holdover arcs which are only present if storage of flow at intermediate nodes is allowed. The nodes are labeled with the corresponding intervals $I_j - \tau_u, I_j - \tau_{v_0}$, and $I_j - \tau_w$, respectively. Nodes are drawn in light gray if their interval is not a subset of $[0, T)$

that the linear program stated above can be interpreted as a classical network flow problem in a network $G' = (V', E')$. An illustrative example of the following construction of G' is given in Figure 2.

For every node $v \in V$ and every $j = 1, \dots, q - 1$ with $I_j - \tau_v \subseteq [0, T)$, there is a copy v_j of v in V' . Moreover, if storage of flow at intermediate nodes is allowed, there is an infinite capacity ‘holdover arc’ between successive copies of v , i.e., $(v_j, v_{j+1}) \in E'$. For every arc $e = (u, v) \in E$ and every $j = 1, \dots, q - 1$ with $I_j - \tau_u \subseteq [0, T - \tau_e)$, there is a copy $e_j = (u_j, v_j)$ of e in E' ; the capacity of e_j is $|I_j|u_e$, its cost is c_e . Finally, for every commodity $i \in K$ and every source node $v \in S_i^+$ (sink node $v \in S_i^-$), we add a super-source (super-sink) v^i to V' and corresponding infinite capacity arcs (v^i, v_j) (respectively (v_j, v^i)) to E' , which connect v^i to all copies $v_j \in V'$ of v . The supply (demand) of super-source (super-sink) v^i is set to $D_{v,i}$. All other nodes are intermediate nodes.

Observation 1. A feasible static flow in G' corresponds to a feasible solution to the linear program of equal cost and vice versa. Here, the linear programming variable $x_{e,i,j}$ represents the flow value of commodity i on arc e_j in G' . The flow on the arcs incident to super-sources and super-sinks are defined accordingly.

This observation leads to the following strengthened version of Theorem 1.

Theorem 2. A min-cost (multicommodity) flow over time problem in a network $G = (V, E)$ with uniform path-lengths can be solved by a static min-cost (multicommodity) flow computation in a network G' with $O(n^2)$ nodes and $O(nm)$ arcs.

We finally turn to the quickest multicommodity flow problem (with bounded cost) in networks with uniform path-lengths. As a result of Theorem 2, this problem can be solved within a binary search framework by a series of static flow computations. In the following we discuss an algorithm with considerably improved running time. In particular, for the case of a single commodity, this algorithm can be implemented to run in strongly polynomial time.

First notice that the construction of the network G' only depends on the sorting of the elements in $\mathcal{T} = \{\tau_v, T + \tau_v \mid v \in V\}$ but not on the exact choice of T (only the arc capacities in G' depend on the exact choice of T). Since there are at most $O(n^2)$ choices of T leading to different sortings of the elements of \mathcal{T} (the sorting only changes when $\tau_v = \tau_u + T$ for some pair $u, v \in V$), the right sorting can be found by binary search in $O(\log n)$ steps. What remains is a parametric network flow problem in G' where the capacities of arcs linearly depend on the exact choice of T .

Theorem 3. *The quickest (multicommodity) flow problem in a network $G = (V, E)$ with uniform path-lengths can be solved by $O(\log n)$ static (multicommodity) flow computations and one parametric static flow computation in networks with $O(n^2)$ nodes and $O(nm)$ arcs.*

4 Networks with Out-Degree at most One

In this section we discuss a combinatorial greedy algorithm for the quickest multicommodity flow problem in networks whose nodes have either out- or in-degree at most one. This class contains paths, cycles, in- and out-trees and also combinations such as a cycle with one or more of its nodes being roots of in- respectively out-trees. In the following we assume that every commodity $i \in K$ has exactly one source node s_i and one sink node t_i . An important feature of the networks under consideration is that there exists a unique s_i - t_i -path P_i for every commodity i . In particular, it is useless to consider costs in this setting since every feasible solution has the same cost.

The basic notion of our greedy algorithm is to schedule the flow according to priorities that are assigned to the individual commodities in such a way that the higher a commodity's priority, the longer (with respect to the number of arcs) its remaining flow path lying ahead. Intuitively, in a traffic network this approach corresponds to giving the right of way to those road users that are furthest from their destinations. Since this approach introduces waiting times at intermediate nodes, we do not restrict storage of flow at intermediate nodes in this section.

Restricting to the Case of Out-Degree at most One. We may henceforth assume without loss of generality that the network G fulfills $|\delta^+(v)| \leq 1$ for all $v \in V$. We can do so, for the setting where G has in-degree at most one, i.e., $|\delta^-(v)| \leq 1$ for all $v \in V$, is symmetric hereunto in the following way. Consider an arbitrary instance of the problem where G has in-degree at most one. Evidently, the network $\overleftarrow{G} = (V, \overleftarrow{E})$, with $e = (v, w) \in \overleftarrow{E}$ if and only if $(w, v) \in E$, has out-degree at most one. We set the transit time of arc $(v, w) \in \overleftarrow{E}$ to $\tau_{(v,w)} := \tau_{(w,v)}$. Now, consider an arbitrary feasible T -horizon flow f for the canonically reversed problem instance in \overleftarrow{G} . Then, if we let the flow f run backwards in time, starting at T and ending at 0, we get the 'blueprint' of a feasible T -horizon flow for the original problem in G . In particular, this argument can be applied to any optimal solution of the reversed instance and yields an optimal solution to the original instance. We may therefore assume from now on that there is at most one arc emanating from any node.

Earliest Arrival Flows. If we consider the situation with only a single commodity, a flow over time with time horizon T is called an *earliest arrival flow* if the amount of flow arriving at the sink is maximized until any given moment $\theta \in [0, T)$ in time [9]. For the setting with multiple commodities, we use an intuitive extension of the notion of earliest arrival flows.

Let $f_{v,i}(\theta)$ be the inflow-rate of commodity i into node v at time θ . If $v = s_i$ or if v is not on the unique s_i - t_i -path P_i , then $f_{v,i}(\theta) = 0$ for all $\theta \in [0, T)$. Otherwise,

$$f_{v,i}(\theta) := f_{e,i}(\theta - \tau_e) \quad \text{for all } \theta \in [0, T),$$

with $e = (u, v)$ and u being the predecessor of v on path P_i . A feasible T -horizon multicommodity flow f is called an *earliest arrival multicommodity flow* if the total amount of flow arriving at any node $v \in V$ until any time $\theta \in [0, T)$, i.e., the expression

$$\sum_{i \in K} \int_0^\theta f_{v,i}(\xi) d\xi,$$

is maximal among all feasible solutions. Notice that earliest arrival multicommodity flows do not exist for general instances. In the current setting, however, they do. This will become clear in the proof of correctness of our algorithm. The following Lemma 2 states that earliest arrival flows are optimal.

Lemma 2. *Suppose we are given an instance of the quickest multicommodity flow problem in a network whose nodes have out-degree at most one. Let f be an earliest arrival multicommodity flow having time horizon T . Then, f is a quickest flow.*

Proof. Assume that f is not a quickest flow. Let f^* denote an optimal flow with time horizon $T^* < T$. In the current setting, we have unique source-sink paths P_i for all commodities $i \in K$. The total amount of flow to be sent over any arc in the network is thus uniquely determined. As f is not optimal, there exists a node $v \in V$ such that

$$\sum_{i \in K} \int_0^{T^*} f_{v,i}(\xi) d\xi = \sum_{i \in \tilde{K}} \int_0^{T^*} f_{v,i}(\xi) d\xi < \sum_{i \in \tilde{K}} d_i = \sum_{i \in \tilde{K}} \int_0^{T^*} f_{v,i}^*(\xi) d\xi = \sum_{i \in K} \int_0^{T^*} f_{v,i}^*(\xi) d\xi,$$

where $\tilde{K} := \{i \in K \mid v \in P_i \setminus \{s_i\}\}$. This contradicts the earliest arrival property of f . \square

Evidently, the earliest arrival property is sufficient but not necessary for a quickest flow f .

The Greedy Property. Due to the uniqueness of the source-sink paths, we can tell all possible ‘conflicts’ between commodities *a priori*, i.e., before sending any flow through the network. We may specify a right-of-way rule for the commodities that solely depends on the paths the commodities are using. In order to do so, we assign a node-dependent priority $p : V \times K \rightarrow \mathbb{N}$ to each node $v \in V$ and to all commodities $i \in K$. The assignment is given by

$$p_i(v) := |\{v, \dots, t_i\}| - 1 \quad \text{for all } i \in K \text{ and } v \in P_i,$$

where v, \dots, t_i denotes the sequence of nodes along P_i that follow node v . For all nodes $v \notin P_i$, we set $p_i(v) := 0$. The assignment implies that, whenever $p_j(v) > p_i(v)$ for $i, j \in K$ and $v \in V$, the flow of commodity j has to travel to a sink t_j that is ‘further away’ than the sink t_i of commodity i .

For simplicity of presentation, we from now on assume that no two commodities share the same sink. This can be done without loss of generality, for every instance where some commodities $i, j \in K$ have a common sink $t \in V$ may be transformed into an equivalent instance with distinct sink nodes for the two commodities: simply ‘split’ the node t into two sink nodes t_i and t_j , connected by an infinite capacity arc with transit time 0. As a consequence, at each node $v \in V$, the priority values $p_i(v) > 0$, $i \in K$, are pairwise different.

The Greedy Algorithm. We denote by T an upper bound on the optimal time horizon. This bound is not necessary when implementing the following greedy algorithm, it is only introduced for simplicity of presentation.

For the greedy algorithm we adopt the notion of *preflows* from classical network flow theory. We define the excess of commodity $i \in K$ in node $v \in V \setminus \{s_i, t_i\}$ to be

$$ex_{v,i}(\theta) := \int_0^\theta (f_{v,i}(\xi) - f_{e,i}(\xi)) d\xi \quad \text{for } \theta \in [0, T],$$

where e is the unique arc with $\text{tail}(e) = v$. For the source of i we define the excess to be

$$ex_{s_i,i}(\theta) := d_i - \int_0^\theta f_{e,i}(\xi) d\xi \quad \text{for } \theta \in [0, T],$$

where e is the unique arc with $\text{tail}(e) = s_i$.

A *preflow over time* obeys the capacity constraints and the flow conservation property (2), but the demands are not necessarily satisfied, i.e., (3) does in general not hold, and nodes other than the sinks might have positive excess at time T . In other words, $ex_{v,i}(T) > 0$ might hold for some $i \in K$ and $v \in V \setminus \{t_i\}$. Moreover, we require that $ex_{s_i,i}(T) \geq 0$ for all $i \in K$, i.e., the flow of commodity i leaving the source s_i must be bounded by d_i .

The greedy algorithm starts with the trivial preflow f , where $f_{e,i}(\theta) = 0$ for all $e \in E$, $i \in K$, and $\theta \in [0, T)$. It modifies this preflow by iteratively pushing flow of one commodity through one arc such that it arrives at the head of this arc as early as possible (i.e., in an earliest arrival fashion). For a given preflow over time f , let $r_e(\theta) := u_e - f_e(\theta)$ denote the residual capacity of arc $e \in E$ at time $\theta \in [0, T)$.

In every iteration, the greedy algorithm chooses a new node-commodity pair $(v, i) \in V \times K$ with highest priority $p_i(v) > 0$ such that flow of commodity i has already been pushed into v in an earlier iteration but (v, i) has not been considered yet. Let e be the unique arc leaving v (i.e., e is the next arc on the path P_i after v). Then, for increasing $\theta \in [0, T)$, the flow rate $f_{e,i}(\theta)$ is computed as follows: If currently $ex_{v,i}(\theta) > 0$ (i.e., there is excess in node v), then $f_{e,i}(\theta) := r_e(\theta)$; otherwise, $f_{e,i}(\theta) := \min\{f_{v,i}(\theta), r_e(\theta)\}$. Notice, that the excess function $ex_{v,i}$ and the residual capacity function r_e must permanently be updated. We argue below that the resulting function $f_{e,i}$ is a step function with at most $4k$ breakpoints, for every $e \in E$ and $i \in K$.

Theorem 4. *The greedy algorithm yields a quickest multicommodity flow and runs in time $O(nk^2)$.*

Proof. We prove that the greedy algorithm yields an earliest arrival (and therefore an optimal) flow by induction on the number of iterations.

Induction hypothesis:

At the end of an iteration, the preflow f has the ‘earliest arrival property’, restricted to all pairs (v, i) , for which commodity i has already arrived at node $v \in P_i$.

Let $K_v^f \subseteq K$ be the subset of commodities which have already arrived at node v in f . Then, the restricted earliest arrival property reads:

$$\text{for all } v \in V \text{ and } \theta \in [0, T), \quad \sum_{i \in K_v^f} \int_0^\theta f_{v,i}(\xi) d\xi \quad \text{is maximal among all preflows } f'. \quad (9)$$

This property obviously holds in the beginning, before the first iteration. If Property (9) holds after every iteration, f is an earliest arrival flow in the end.

Induction Step: By construction of the greedy algorithm, in every iteration flow of some commodity is pushed ‘as early as possible’ through an arc to the next node. Therefore the restricted earliest arrival property still holds after this iteration. This concludes the proof of optimality.

Running Time: The greedy algorithm has $O(nk)$ iterations. We show that each iteration takes time $O(k)$. Given a preflow f , we call $f_e(\theta)$, $\theta \in [0, T)$, the *flow profile* of arc $e \in E$. We show below that f_e is a step function after each iteration of the greedy algorithm. Let $B(f_e)$ denote the number of *breakpoints* of f_e . Now, given an arbitrary arc $e = (v, w) \in E$, we sort the commodities in descending manner according to the priorities in node v , i.e., we have $K = \{i_1, \dots, i_k\}$ with $p_{i_j}(v) \geq p_{i_{j+1}}(v)$, for $j = 1, \dots, k - 1$.

Claim. For any $j \in \{1, \dots, k\}$, in order to ship the entire flow of commodities i_1, \dots, i_j over arc e , the algorithm computes a flow profile f_e which is a *step-function* with no more than $2j$ breakpoints, i.e., $B(f_e) \leq 2j$.

Given this claim and considering the commodities in reverse order, we can easily count the number of breakpoints of f_{e,i_j} : The flow profile f_e resulting after sending commodities i_1, \dots, i_j over e has no more than $2j$ breakpoints. On the other hand, we also know that the flow profile of commodities i_1, \dots, i_{j-1} has at most $2j - 2$ breakpoints. It thus follows that the flow rate of i_j on e changes at most $4j - 2$ times and can thus be computed in $O(k)$ time.

It remains to prove the claim. The underlying (trivial) notion of the proof is that the sum of two step functions f, f' again yields a step function f'' , and, in the worst case, the numbers $B(f), B(f')$ of breakpoints of f, f' add up to the number $B(f'')$ of breakpoints of f'' . Assume by contradiction that there is an arc e and a number j for which the claim is false. We choose j minimally and take the first arc e on path P_{i_j} for which the claim does not hold.

We first argue that $j > 1$: For the first commodity $i_1 \in K$ to be sent over arc $e \in E$, it is apparent that the entire demand d_{i_1} is shipped within a single constant flow-rate time interval. To see this, note that in case P_{i_1} does not start with e , commodity i_1 has the highest priority in all previous nodes on P_{i_1} as well. This yields a flow profile f_e with two breakpoints. We thus have $B(f_e) = 2$.

It is also easy to see that e is not the first arc on path P_{i_j} : By minimality of j , we have $B(f_e) \leq 2j - 2$ before commodity i_j is sent over arc e . By construction of the greedy algorithm, pushing commodity i_j from source s_{i_j} through arc e can add at most 2 new breakpoints to f_e : The first one

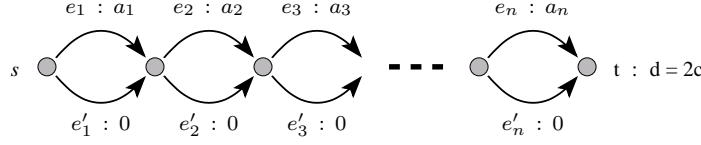


Fig. 3. Reduction from PARTITION: All arcs have unit capacity. The arc transit times are given in the picture. For the upper arcs e_j , $j = 1, \dots, n$, they are taken from the PARTITION instance: a_1, \dots, a_n . The transit times of the lower arcs e'_j are 0. At the sink node t , a demand of $d = 2c$ is present.

when we start to send flow of commodity i_j through e ; the second one when we stop to send flow of commodity i_j .

We can assume by choice of j and e that commodities i_1, \dots, i_j arrive on several arcs e_1, \dots, e_q at $\text{tail}(e)$ such that $B(f_{e_1}) + \dots + B(f_{e_q}) \leq 2j$. That is, the inflow profile of commodities i_1, \dots, i_j in node $\text{tail}(e)$ has at most $2j$ breakpoints. If this inflow profile fits into arc e , i.e., if its maximum is bounded by the capacity u_e , it is equal to the flow profile f_e and we are done.

Otherwise, consider a time interval $[\theta, \theta']$ such that the inflow profile exceeds u_e during this time interval. When flow is pushed through arc e , the *peaks* above u_e in the inflow profile are truncated and the arc e is saturated until time θ' . Then, at time θ' , there is positive excess in node $\text{tail}(e)$ which is removed by keeping the arc e saturated up to some moment in time $\theta'' > \theta'$ when the flow profile f_e finally returns to the inflow profile again. Notice that this truncation/saturation operation does not increase the number of breakpoints. This concludes the proof of the claim and the proof of the theorem. \square

5 Weak NP-Hardness Results

In this section we prove NP-hardness for the multicommodity flow over time problem. As in the last section, we consider instances with one source s_i , one sink t_i , and demand d_i for every commodity $i \in K$. In Section 5.1, we show that the multicommodity flow over time problem with or without storage at intermediate nodes is weakly NP-hard. Then, in Section 5.2 we refine the proof and show that this hardness result already holds for the special case of only two commodities.

5.1 NP-Hardness with and without Intermediate Storage

Theorem 5. *The multicommodity flow over time problem with or without storage at intermediate nodes is NP-hard on series-parallel networks. The same holds for the maximum multicommodity flow over time problem.*

In the following discussion we concentrate on the multicommodity flow over time problem. At the end of the proof we very briefly note how the maximum multicommodity flow over time problem can be handled.

We give a reduction from the well-known NP-hard PARTITION problem: Given n integer numbers $a_1, \dots, a_n \in \mathbb{N}$ with $\sum_{i=1}^n a_i = 2L$ for some $L \in \mathbb{N}$, the task is to decide whether there is a subset $B \subseteq \{1, \dots, n\}$ such that $\sum_{i \in B} a_i = L$. Given an instance of PARTITION, we construct a *chord* as shown in Figure 3 and introduce the first commodity with source s , sink t , and

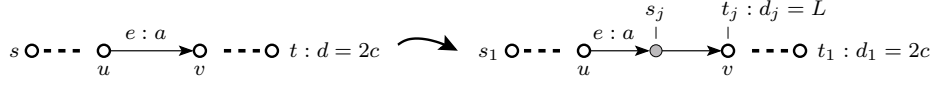


Fig. 4. To let only c units of flow pass through e in the given time horizon $T = L + c$, a commodity j is added with demand $d_j = L$. The new arc (s_j, v) has capacity 1 and length 0.

demand $d = 2c$. Here, $c < 1$ is some positive constant. Further commodities will be added later. The time horizon is set to $T := L + c$. This constitutes the main building block of the reduction, which is similar to other well known reductions of PARTITION to flow problems; see, e.g., [13].

Let \mathcal{P} denote the set of all 2^n paths from s to t . For $P \in \mathcal{P}$, let $\tau(P) := \sum_{e \in P} \tau_e$ be the length of path P . For a given flow over time f (of the first commodity) and a path $P \in \mathcal{P}$, let x_P denote the total amount of flow of the first commodity routed along P , that is, $x_P = \int_0^T f_P(\theta) d\theta$. Furthermore, we define $x_e := \sum_{P \in \mathcal{P}: e \in P} x_P$ to be the amount of flow of the first commodity routed through arc e .

Lemma 3. *It is NP-hard to decide whether a flow amount of $2c$ can be routed from s to t within time $T = L + c$, such that $x_e \leq c$ holds for all arcs e .*

Proof. We will show that such a routing exists if and only if there is a solution to the PARTITION instance, i.e., there is $B \subseteq \{1, \dots, n\}$ such that $\sum_{i \in B} a_i = L$. First the easy direction: If such a subset B exists, route c units of flow along the path P_B consisting of arcs $\{e_i \mid i \in B\} \cup \{e'_i \mid i \notin B\}$. The remaining c units are routed along the complementary path $P_{\bar{B}}$ with $\bar{B} := \{1, \dots, n\} \setminus B$. It is clear that $x_e = c$ for all $e \in E$ and $\tau(P_B) = \tau(P_{\bar{B}}) = L$. Therefore, $2c$ units can be routed in time T .

Assume that $2c$ units can be routed in time $T = L + c$ such that $x_e \leq c$ for all $e \in E$. It is easy to see that $x_e = c$ must hold for all arcs e . Consider the following sum of weighted path lengths

$$\begin{aligned} \sum_{P \in \mathcal{P}} x_P \tau(P) &= \sum_{P \in \mathcal{P}} x_P \sum_{i \in \{1, \dots, n\}: e_i \in P} a_i = \sum_{i=1}^n a_i \sum_{P \in \mathcal{P}: e_i \in P} x_P \\ &= \sum_{i=1}^n a_i x_{e_i} = c \sum_{i=1}^n a_i = 2cL. \end{aligned}$$

In particular, the weighted average path length is equal to L . No flow can be routed on a path of length strictly larger than L since $c < 1$ and all a_i are integer. Thus, sending a positive amount of flow on a path of length strictly smaller than L would also imply that the weighted average path length is strictly smaller than L . Hence, every path which carries a positive amount of flow must have length exactly L and therefore induces a subset B with $\sum_{i \in B} a_i = L$. \square

Our aim is now to enforce the bound $x_e \leq c$ for all arcs e by introducing one further commodity j per arc. We split every arc $e = (u, v)$ into two consecutive arcs (u, s_j) and (s_j, v) with unit capacity; see Figure 4. The transit time of (s_j, v) is zero and the transit time of (u, s_j) equals the transit time of the original arc e . Notice that this modification has no impact on feasible flows over time for the first commodity. Now we introduce the additional commodity j with source s_j and sink $t_j = v$. The demand of this new commodity is set to L . Therefore, at most c additional units of

flow of the first commodity can be sent through arc (s_j, v) within time $T = L + c$. Note that flow does not need to be stored at intermediate nodes, thus the reduction works for both models, with and without storage. This concludes the proof of the first part of Theorem 5.

Maximum Multicommodity Flow Over Time. The presented construction cannot be used directly to obtain the NP-hardness for the maximum multicommodity flow over time problem. In the gadget depicted in Figure 4, we need that exactly L units of flow of commodity j are sent. But in a solution for the maximum multicommodity flow setting, where the demands are not given explicitly, it might be advantageous to send more than L units. To enforce this constraint nevertheless, one can enhance the gadget by adding another commodity which basically blocks an arc that j has to take for c units of time. One can then show that every optimal flow can be transformed into a flow in which exactly L units of j are sent. We will not go into details and only note that the construction is similar to the one presented in the proof of Theorem 6 below, except that by adding a new commodity one can ensure that the graph remains series-parallel. Also note that, again similar to the following proof, it is necessary to simulate storage at certain nodes, in case that this is not allowed.

5.2 NP-hardness for the Special Case of Two Commodities

Next we present a gadget with which the bound $x_e \leq c$ in Lemma 3 can be achieved by adding only one new commodity. This *blocking gadget* itself might be interesting in other contexts as well. Modifying it so that an arbitrary amount of flow can be chosen for each arc, offers the possibility to, e.g., reduce the *Length Bounded Static Flow* problem to the multicommodity flow over time problem. The former problem is known to be NP-hard already for one commodity. The presentation and proofs though are fitted for the special case introduced above and yield the following theorem.

Theorem 6. *The multicommodity flow over time problem with or without storage at intermediate nodes is already NP-hard for the case of only two commodities. The same holds for the maximum multicommodity flow over time problem.*

In the following discussion we will concentrate on the multicommodity flow over time problem with storage and then describe how the problem without storage and the maximum multicommodity flow over time problem can be handled.

Blocking Gadget. The mentioned, more complex gadget to enforce the bound $x_e \leq c$ is depicted in Figure 5. Note that only one commodity needs to be added for all its occurrences, i.e., all such gadgets share the same source and sink vertices s_2 and t_2 , respectively.

In the proof of the lemma below we will show that it is optimal to send c units of flow of the first commodity along arcs e_f, e_d, e_e within the time interval $[0, c)$ (notice that we have increased the demand of the first commodity by c for each of the $2n$ gadgets). This blocks arc e_d such that it is not possible to send more than L units from s_2 to t_2 via e_b, e_c, e_d . We show that it is optimal to send exactly L units, thus as desired e_b is blocked by commodity 2 for all but c units in time interval $[0, L + c)$. In contrast to the simple gadget in Figure 4, flow might need to be stored (at node w).

Let G be the original (Figure 3) and G_m be the modified network, in which all arcs are replaced by an instance of the blocking gadget (as mentioned above, all these instances share the same source and sink vertices s_2 and t_2 , respectively). Let E_b denote the set of all new arcs of type e_b ,

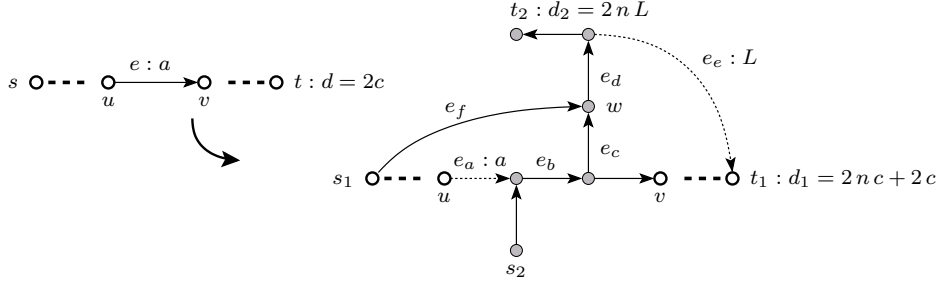


Fig. 5. This picture sketches the structure with which each arc is replaced. All arcs have capacity 1. The transit times are 0, except for e_a whose transit time is taken from the corresponding arc in the original network, and e_e whose transit time is set to L . These arcs are highlighted. The demands added to the pair (s_1, t_1) —which corresponds to (s, t) in Figure 3—and (s_2, t_2) are motivated in the proof of Lemma 4.

see Figure 5. Note that $|E_b| = 2n$. Analogously to x_e , let $x_{e,i}$ denote the total amount of flow of commodity i routed through arc e .

Lemma 4. *Let $T = L + c$ be the time horizon. An s - t -flow over time in G which satisfies demand $d = 2c$ in time horizon T , obeying the constraints $x_e \leq c$ for all arcs e of G , exists if and only if there is a 2-commodity flow over time in G_m which satisfies demands $d_1 = 2nc + 2c$ and $d_2 = 2nL$ in time T .*

Proof. We start with the more difficult direction: From a flow over time in G_m which satisfies demands d_1 and d_2 in time T , we derive an s - t -flow over time in G with time horizon T which satisfies demand d and does not violate the arc constraints $x_e \leq c$ for any arc e . Consider a single gadget. First of all note that we may assume that $x_{e_c,1} = 0$. Flow of the first commodity along e_c might as well be routed directly on e_f . Similarly, $x_{e_b,2} = x_{e_c,2} = x_{e_d,2}$ may be assumed. Flow of the second commodity coming from e_b not directly continuing along arc e_c would unnecessarily block other arcs in E_b , to which it could be sent directly from s_2 . To simplify notation we denote by $x_2 := x_{e_b,2}$ the total amount of flow of commodity 2 through this gadget.

Since all capacities are 1 and the time horizon T is equal to $L + c$, we know that the inequalities

$$x_{e_d,1} + x_2 = x_{e_d,1} + x_{e_d,2} \leq L + c \quad (10)$$

and

$$x_{e_b,1} + x_2 = x_{e_b,1} + x_{e_b,2} \leq L + c \quad (11)$$

hold. Furthermore, from $\tau(e_e) = L$ we get that $x_{e_d,1} \leq c$. To achieve equality, c units of flow of the first commodity have to be sent in time $[0, c)$ along arcs e_f, e_d, e_e . We argue that this must be the case if demands d_1 and d_2 are satisfied. Let us assume by contradiction that $x_{e_d,1} = c - c_1 < c$. Since no other e_d type arc can carry more than c units of flow of the first commodity and since $d_1 = 2nc + 2c$, at least $2c + c_1$ units of the first commodity must be sent along the chord, via e_b type arcs to t_1 . Note that G_m contains $2n$ copies of the gadget.

To be more precise, let us consider two gadgets corresponding to some pair of parallel arcs e and e' in the chord G . Let $x_{e_b,1}, x_2$ and $x'_{e_b,1}, x'_2$, respectively, denote flow amounts as defined

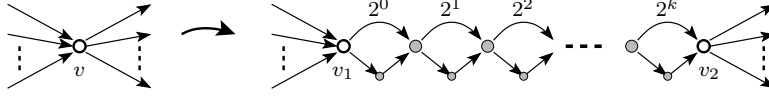


Fig. 6. Replace a node by a chord. The new arcs have infinite capacities. As depicted, the transit times of all upper arcs are powers of 2. All lower arcs have transit time 0. The extra nodes for the lower arcs are added only to avoid multi graphs. The value of k is chosen sufficiently large.

above. Clearly the mentioned $2c + c_1$ units of flow have to pass through this pair completely and therefore $x_{e_b,1} + x'_{e_b,1} \geq 2c + c_1$. With (11) we get $x_2 + x'_2 \leq 2L + 2c - (x_{e_b,1} + x'_{e_b,1}) \leq 2L - c_1$. Since this holds for all n pairs of gadgets, demand $d_2 = 2nL$ cannot be satisfied. This contradicts our assumptions and thus $x_{e_d,1} = c$ holds for all gadgets.

With (10) we now get that x_2 must equal L for all gadgets; otherwise demand d_2 cannot be met. Note that these L units of flow might (partially) need to be stored at w because e_d is saturated in $[0, c)$. This is the only node in the gadget where the possibility to store flow is required.

Applying inequality (11) again, we derive that $x_{e_b,1} \leq c$ for all gadgets. Thus to construct a flow over time in G that satisfies demand $d = 2c$ within time horizon T and obeys the arc constraints, we simply route the $2c$ units of the first commodity which are routed along arcs in E_b correspondingly in G .

The easier direction remains: Given an s - t -flow over time in G of value $d = 2c$ and time horizon T , adhering to $x_e \leq c$ for all arcs e , we directly transfer it to flow of the first commodity in G_m in the canonical way. Additionally, $2nc$ units of the first commodity are routed along e_d type arcs within time interval $[0, c)$. Thus, demand $d_1 = 2nc + 2c$ is met.

To meet demand $d_2 = 2nL$, exactly L units of the second commodity are sent through each gadget. This is possible since $x_{e_d,1} = x_{e_b,1} = c$ and equality in (10) and (11) can be achieved by possibly storing flow at w . In other words, both e_d and e_b only carry c units of the first commodity such that, in addition, L units of the second commodity can be sent. This altogether gives a flow over time in G_m satisfying demands d_1 and d_2 in time T . \square

Lemma 4 and Lemma 3 together yield that the multicommodity flow over time problem with storage at intermediate nodes is already NP-hard for the case of only two commodities, proving part of Theorem 6. We now describe how storage at nodes can be simulated for the case where storage is not allowed and give the idea of how to adapt the proof to the maximum multicommodity flow over time problem.

Simulating Intermediate Storage. If all transit times and the time horizon are multiples of a constant (in our case c), we can scale everything to integer values. Then, with Lemma 2 in [4] we know that it suffices to consider the model with discrete (integral) time-steps. Each node at which we wish to have storage (all “ w -type” nodes; see proof of Lemma 4) is replaced by the gadget in Figure 6.

We choose k big enough, e.g., $k := \lceil \log(T) \rceil$. It is clear that flow entering v_1 at a certain time-step can be postponed to continue from v_2 at (and perhaps split to several) arbitrary later time-steps.

Maximum Multicommodity Flow Over Time. To obtain the NP-hardness result for the maximum multicommodity flow over time problem, Lemma 3 can be applied as is. It is possible to reformulate Lemma 4 for the maximum flow case. One can prove that a maximum flow of value $2nL + 2nc + 2c$

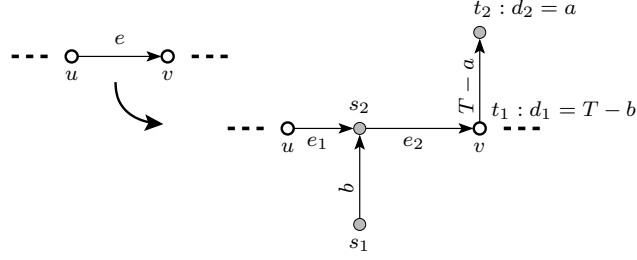


Fig. 7. Goal of the *traffic light gadget*: Given time horizon T , block arc e during time intervals $[0, a)$ and $[b, T)$, where $a \leq b$. All arcs have capacity 1 and transit time 0, except (s_1, s_2) which has transit time b , and (t_1, t_2) which has transit time $T - a$.

in time $T = L + c$ can be modified such that exactly the demands d_1 and d_2 are met. The proof is omitted; it consists of simple exchange arguments.

6 Strong NP-Hardness Results

In this section we prove strong NP-hardness for the multicommodity flow over time problem without storage of flow at intermediate nodes when only simple flow paths are allowed; see Section 6.1. The reason for the latter restriction is to forbid simulation of storage by adding loops to a path. Note that in the case *with* storage this is no true restriction because every loop can be deleted from a path, storing the corresponding flow for the amount of time it would have spent traveling along the loop. Then, in Section 6.2 we prove that, under the above restriction, the quickest multicommodity flow problem does not have an FPTAS, unless $P=NP$.

6.1 Strong NP-Hardness for the Case with Simple Flow Paths and no Intermediate Storage

The NP-hardness proof uses a reduction from the well known 3-PARTITION problem. In this problem, a set of $3n$ items $B = \{1, \dots, 3n\}$ with associated sizes $b_1, \dots, b_{3n} \in \mathbb{N}$, and a number $L \in \mathbb{N}$ are given, with $L/4 < b_i < L/2$, for each i , and $\sum_{i=1}^{3n} b_i = nL$. It is a strongly NP-hard problem to decide whether B can be partitioned into n disjoint sets I_0, \dots, I_{n-1} such that $\sum_{i \in I_j} b_i = L$, for $j = 0, \dots, n - 1$. Note that due to the bounds on the item sizes b_i , all sets I_j must have cardinality 3.

Before presenting the reduction, we introduce a useful little gadget. In the following, all arcs have capacity 1 and transit time 0, if not marked otherwise.

The Traffic Light Gadget. Figure 7 shows the *traffic light gadget* which can be used to replace a single arc e . Any flow over time that satisfies demands d_1 and d_2 in the given time horizon T , completely blocks arc e_2 during the time intervals $[0, a)$ and $[b, T)$ with flow of commodity 2 and 1, respectively. These intervals would so to speak be the “red phases”. As a result, only during the “green phase” $[a, b)$ flow of other commodities can pass through the arc. In the construction, arc e is split into two arcs so flow of commodities 1 and 2 has to travel directly via e_2 to t_1 and t_2 , respectively.

transit times add up to L and defining the path such that it uses these three and no other e_i -type arcs in row j ; note that the arc after $v_{j,3n+1}$ has transit time 1. The selection of arcs $e_{i_1}, e_{i_2}, e_{i_3}$ for row j is taken directly from the set $I_j = \{i_1, i_2, i_3\}$ of the given partitioning. Since this partitioning is a solution to the 3-PARTITION instance, each e_i is only touched once in the entire path and therefore the path is simple, as required.

We now come to the harder direction and prove that a flow over time which satisfies demand $d_1 = 1$ within time horizon $T = nL + n + 1$ and passes all traffic light gadgets during their “green phases”, directly yields a feasible solution I_0, \dots, I_{n-1} to the underlying instance of 3-PARTITION.

Assume we are given such a flow over time which satisfies demand d_1 within time horizon T . Note that flow of commodity 1 can only pass through the traffic light gadgets $l_{j,i}$ in the given intervals. If an amount of δ passes through a gadget outside its “green phase”, the time horizon increases to at least $T + \delta$.

We choose any path P in the set \mathcal{P}_1 of all paths from s_1 to t_1 , with $x_P > 0$ and start by proving that P traverses the grid row by row. Consider node $v_{j,i}$, $i \leq 3n$, in row j : either the flow goes directly to node $v_{j,i+1}$, staying in row j , or it continues to arc e_i . To see via which traffic light gadget the flow leaves the head node of e_i in the latter case, we need to know at what time it entered $v_{j,i}$ (which yields the time when it leaves e_i). To this end, we consider the last traffic light gadget $l_{j,i'}$, $i' < i$, before $v_{j,i}$ on P and distinguish two cases:

- i) $i' = 0$: In this case, flow along P can enter $v_{j,i}$ only within time interval $[jL + j, jL + j + 1)$.
- ii) $i' > 0$: In this case, flow along P enters $v_{j,i}$ within $[jL + j + b_{i'}, (j+1)L + j + 1)$.

In both cases, flow along P arrives at $v_{j,i}$ and thus at e_i within the time interval $[jL + j, (j+1)L + j + 1)$. On arc e_i , the flow is delayed by b_i and thus arrives at the head of e_i before $(j+1)L + (j+1) + b_i$. This is the lower bound of the interval of $l_{j+1,i}$. Since flow cannot be stored, it therefore cannot leave via any $l_{j',i}$ with $j' > j$. Similarly flow on P cannot leave via any $l_{j',i}$ with $j' < j$. Thus, if P contains $v_{j,i}$ and e_i , it must continue to $v_{j,i+1}$, staying in row j . Consequently, the only point where P can leave row j is to the right of $v_{j,3n+1}$, where it can only go to row $j+1$.

Now that we know that P traverses the grid row by row, we prove that it touches exactly 3 arcs $e_{i_1}, e_{i_2}, e_{i_3}$, with $b_{i_1} + b_{i_2} + b_{i_3} = L$ in each fixed row $j \in \{0, \dots, n-1\}$. The length of the time-span a unit of flow can spend in this row is solely determined by the traffic light gadgets before and after the row, namely $l_{j,0}$ and $l_{j+1,0}$. It is easy to check that the “red phases” of all other traffic light gadgets in row j do not restrict this time-span (notice that the last arc of the row has transit time 1).

From the “green phase” $l_{j,0}$ and $l_{j+1,0}$ it follows that the length of the time-span is in the interval $(L, L+2)$ (subtract the lower and upper bound of $l_{j+1,0}$ and $l_{j,0}$, correspondingly). Since all transit times are integer and flow cannot be stored at nodes, the time-span must have length $L+1$. Of this, 1 unit of time is spent in the last arc of the row. All e_i , $i = 1, \dots, 3n$, have integer transit times, and all other arcs have transit time 0. Thus the transit times of the e_i -type arcs touched in row j , denoted by $E_j := \{e_{i_1}, e_{i_2}, \dots\}$, must sum up to L .

Since $L/4 < b_i < L/2$ holds for all $i \in \{1, \dots, n\}$, the cardinality of E_j must be 3. Thus, we can set $I_j := \{i_1, i_2, i_3\}$, if $E_j := \{e_{i_1}, e_{i_2}, e_{i_3}\}$, for $j \in \{0, \dots, n-1\}$. As a consequence of P being a simple path the sets I_j are disjoint and therefore form a solution to the underlying instance of 3-PARTITION. This concludes the proof of the first part of the theorem.

Maximum Multicommodity Flow. To carry over the hardness result to the maximum flow case, one mainly has to take a closer look at the traffic light gadget; see Figure 7. It is easy to prove with an exchange argument that any maximum flow over time with time horizon T can be transformed into a flow that locally, in each traffic light gadget, satisfies the demands d_1 and d_2 and consequently there are again “green” and “red phases” where flow of the main commodity can, respectively cannot pass through the gadget. The remaining argumentation in the above proof remains unchanged and yields the following result. If there is a maximum flow over time which sends 1 unit of flow of the main commodity from s_1 to t_1 in time $T = n \cdot L + n + 1$, then a solution to the 3-PARTITION instance exists. This concludes the proof. \square

6.2 No FPTAS for the Case with Simple Flow Paths and no Intermediate Storage

Unfortunately, Theorem 7 does not immediately yield the non-existence of an FPTAS (under the assumption $P \neq NP$) for the quickest multicommodity flow problem, since the objective function values are not necessarily integral. To obtain such a result anyhow, we prove the existence of a certain gap in the optimal values which the objective function can take.

Theorem 8. *Unless $P=NP$, there is no FPTAS for the quickest multicommodity flow problem with simple flow paths and without storage of flow at intermediate nodes.*

Proof. We consider the reduction presented above. Given a no-instance of 3-PARTITION, we show that the time horizon of a quickest flow is at least $T + \delta$, where $T = nL + n + 1$ and $\delta \in \Omega(1/n^2)$. From the proof of Theorem 7 we know that a yes-instance leads to a time horizon of T . This relative gap of $\delta/T \in \Omega(1/(n^3 L))$ and the fact that 3-PARTITION is strongly NP-hard, give the desired result. (An FPTAS for quickest flow could solve 3-PARTITION in pseudo-polynomial time.)

If we are given a no-instance of 3-PARTITION, every path $P \in \mathcal{P}_1$ in the corresponding quickest flow instance violates at least one traffic light gadget. Otherwise, we could satisfy the demand $d_1 = 1$ in time T by sending it completely along that path, thereby obtaining a solution to the 3-PARTITION instance. To see this, note that all paths have integer lengths and all intervals of the traffic light gadgets have integer bounds.

If every path violates at least one gadget and there are $O(n^2)$ such gadgets, it follows that $\delta \in \Omega(1/n^2)$; in the worst case, d_1 is distributed evenly among the gadgets. This concludes the proof of the theorem. \square

Acknowledgements. We are much indebted to Lisa Fleischer, Ekkehard Köhler, Katharina Langkau, and Jim Orlin for helpful discussions on the topic of this paper.

References

1. J. E. Aronson. A survey of dynamic network flows. *Annals of Operations Research*, 20:1–66, 1989.
2. R. E. Burkard, K. Dlaska, and B. Klinz. The quickest flow problem. *ZOR — Methods and Models of Operations Research*, 37:31–58, 1993.
3. T. Erlebach and K. Jansen. Call scheduling in trees, rings and meshes. In *Proceedings of the 30th Hawaii International Conference on System Sciences*, pages 221–222. IEEE Computer Society Press, 1997.

4. 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.
5. 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.
6. L. K. Fleischer and É. Tardos. Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, 23:71–80, 1998.
7. L. R. Ford and D. R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6:419–433, 1958.
8. L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, 1962.
9. D. Gale. Transient flows in networks. *Michigan Mathematical Journal*, 6:59–63, 1959.
10. A. Hall, S. Hippler, and M. Skutella. Multicommodity flows over time: Efficient algorithms and complexity. In J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger, editors, *Automata, Languages and Programming*, volume 2719 of *Lecture Notes in Computer Science*, pages 397–409. Springer, Berlin, 2003.
11. B. Hoppe. *Efficient dynamic network flow algorithms*. PhD thesis, Cornell University, 1995.
12. B. Hoppe and É. Tardos. The quickest transshipment problem. *Mathematics of Operations Research*, 25:36–62, 2000.
13. 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.
14. N. Megiddo. Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, 4:414–424, 1979.
15. 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.