Minimum Cost Flows Over Time without Intermediate Storage*

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. Solving these problems raises issues that do not arise in standard network flows. One issue is the question of storage of flow at intermediate nodes. In most applications (such as, e. g., traffic routing, evacuation planning, telecommunications etc.), intermediate storage is limited, undesired, or prohibited.

The minimum cost flow over time problem is NP-hard. In this paper we 1) prove that the minimum cost flow over time never requires storage; 2) provide the first approximation scheme for minimum cost flows over time that does not require storage; 3) provide the first approximation scheme for minimum cost flows over time that meets hard cost constraints, while approximating only makespan.

Our approach is based on a condensed variant of time-expanded networks. It also yields fast approximation schemes with simple solutions for the quickest multicommodity flow problem.

Finally, using completely different techniques, we describe a very simple capacity scaling FPAS for the minimum cost flow over time problem when costs are proportional to transit times. The algorithm builds upon our observation about the structure of optimal solutions to this problem: they are universally quickest flows. Again, the FPAS does not use intermediate node storage. In contrast to the preceding algorithms that use a time-expanded network, this FPAS runs directly on the original network.

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 [3] 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] Since then, this model has proved to be very useful [1, 13]. 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 [10, 6], even for very simple series-parallel networks.

Problem Definitions. 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. Moreover, there is a set of terminals $S \subseteq V$ 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 $\sum_{v \in S} D_v = 0$.

A *flow over time* f on $\mathcal{N}$ with *time horizon* T (also called *makespan*) is given by Lebesgue-measurable functions $f_e : [0, T) \to \mathbb{R}^+$ where $f_e(\theta)$ determines the rate of flow (per time unit) 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 $f_e(\theta)$ entering arc $e = (v, w)$ at time θ arrives at w at time $\theta + \tau_e$. Thus, in order to obey the time horizon T, we require that $f_e(\theta) = 0$ for $\theta \in [T - \tau_e, T)$. In order to simplify notation, we sometimes use $f_e(\theta)$ for $\theta \notin [0, T)$, implicitly assuming that $f_e(\theta) = 0$ in this case.

With respect to flow conservation, there are two different models of flows over time. In the model with *intermediate storage of flow at nodes*, it is possible to hold inventory at a node before sending it onward. Thus, the flow conservation constraints are integrated over time to prohibit deficit at any node:

$$(1.1) \quad \int_0^\xi \left(\sum_{e \in \delta^+(v)} f_e(\theta) \; - \sum_{e \in \delta^-(v)} f_e(\theta - \tau_e) \right) d\theta \;\leq\; 0 \,,$$

*To appear in Proceedings of the 14th Annual ACM–SIAM Symposium on Discrete Algorithms, Baltimore, MD, January 2003.

†GSIA, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213, USA, Email: lkf@andrew.cmu.edu. Supported in part by IBM and by NSF through grant CCR-0049071.

‡Technische Universität Berlin, Institut für Mathematik, MA 6–1, Str. des 17. Juni 136, 10623 Berlin, Germany, Email: skutella@math.tu-berlin.de. Supported in part by the EU Thematic Networks APPOL I+II, Approximation and Online Algorithms, IST-1999-14084 and IST-2001-30012, and by the DFG Research Center 'Mathematics for key technologies: Modelling, simulation and optimization of real-world processes'.

[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.

for all $\xi \in [0, T)$, $v \in V \setminus S^+$. Here, $\delta^+(v)$ and $\delta^-(v)$ denote the set of arcs e leaving node v and entering node v, respectively. Moreover, we require that equality holds in (1.1) for $\xi = T$ and $v \in V \setminus S$, meaning that no flow should remain in the network after time T. In the model without intermediate storage of flow at nodes we additionally require that equality holds in (1.1) for all $\xi \in [0, T)$ and $v \in V \setminus S$.

A flow over time f satisfies the supplies and demands if by time T the net flow into each sink equals the demand at the sink and the net flow out of each source equals the supply at the source:

$$\int_0^T \left(\sum_{e\in\delta^+(v)} f_e(\theta) \; - \sum_{e\in\delta^-(v)} f_e(\theta - \tau_e) \right) d\theta \;=\; D_v$$

for all $v \in S$. As for the setting of static flows, we use the term *s-t-flow over time* for the case of a single source s and a single sink t. An s-t-flow over time satisfying the supply $D = D_s = -D_t$ has *value* $|f| = D$.

A flow over time f is *feasible* if it obeys the capacity constraints $f_e(\theta) \le u_e$, for all $\theta \in [0, T)$ and $e \in A$. Here, capacity u_e is interpreted as an upper bound on the rate of flow entering arc e, i. e., a capacity per unit time.

Given a vector of cost functions $(c_e)_{e\in A}$, the cost of a flow over time f is defined as

$$c(f) \;:=\; \sum_{e\in A} \int_0^T c_e(f_e(\theta))\, d\theta \;. \tag{1.2}$$

Given a source s, a sink t, a flow value D, and an integral time horizon T, the *(min-cost) flow-over-time problem* asks for a feasible s-t-flow over time of value D with time horizon T (and minimum cost). Similarly, for the case of a set of several terminals S with supplies and demands D_v, $v \in S$, the *(min-cost) transshipment-over-time problem* is to find a feasible flow over time with time horizon T, satisfying all supplies and demands (at minimum cost). The *quickest transshipment problem (with bounded cost)* asks for a solution with minimal makespan (and cost bounded by a given value C).

1.1 Results from the Literature

Flows over time. Many flow over time problems can be solved in an exponentially sized time-expanded network. The size of this graph depends linearly on the makespan 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.

Maximum Flows. Ford and Fulkerson [3] show how a path decomposition of a special static minimum cost flow in a network with transit times can be used to describe a maximum flow over time. This solution is periodic and does not require intermediate node storage.

Unlike standard network flows, the 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 [9, 7], and give a solution that does not require intermediate node storage. Their algorithm is not practical as it requires a submodular function minimization oracle for a subroutine.

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 $\theta = 0, \ldots, T$. Gale [4] observes that these flows exist; and Wilkinson [14] and Minieka [11] give equivalent pseudo-polynomial time algorithms to find them. Their 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 $\theta = 0, \ldots, T$ (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 [8, 7] 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 $0 \le \theta \le T$. An equivalent problem is the *universally quickest flow problem* which asks for a flow over time that sends D units of flow to a single sink such that the earliest point in time when d units have arrived at the sink is simultaneously minimized for all $d \le D$ and the earliest point in time when d units have left the source is simultaneously maximized for all $d \le D$.

Minimum Cost Flows. Orlin [12] describes a polynomial time algorithm to compute a 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. For the finite horizon problems that we treat in this paper, Klinz and Woeginger show via a reduction from partition that the minimum cost flow over time problem is NP-Hard [10]. In a recent paper [2], we describe a $(2 + \varepsilon)$-approximation algorithm for the quickest flow problem with bounded cost that does not require intermediate node storage. In the quickest flow problem, the task is to find a flow over time with minimal makespan T. We also give an FPAS for this problem that requires an exponential amount of additional storage. This approximation scheme approximates both cost and time.

Multicommodity Flows. The approximation results in [2] also apply to the quickest multicommodity flow problem with bounded cost. In the multicommodity setting, the FPAS relies on solving a linear program with $p = O(|A|^5 k^2/\varepsilon^2)$ variables. Lemma 4.2 of this paper implies that it is sufficient to obtain an approximately optimal solution to the LP, and thus it is possible to use an FPAS for concurrent multicommodity flow [5] instead of solving the LP exactly. The run time of the fastest known FPAS depends quadratically on p. Already in the setting without costs, multicommodity flows over time are NP-hard [6].

1.2 Our Contribution

We prove that the minimum convex cost transshipment over time never requires storage[2] and provide the first approximation scheme for minimum cost flows over time that does not require storage; this is also the first approximation scheme for minimum cost flows over time that meets hard cost constraints, while approximating only makespan.

Our algorithm makes use of a completely new and much simpler analysis of condensed time-expanded networks, which results in considerably reduced run times and simpler solutions, that is, solutions with fewer parts: In order to obtain $(1+\varepsilon)$-approximate solutions, we show that the number of time layers of the condensed time-expanded network can be chosen to be $O(|V|/\varepsilon^2)$ instead of the $O(|A|^4/\varepsilon^2)$ bound in [2]. As our algorithm employs standard minimum cost flow algorithms, it also yields a version that is the first strongly polynomial FPAS for the problem.

For the transshipment problem without costs, our FPAS is significantly faster and simpler than the exact algorithm of Hoppe and Tardos [9] when the number of sources and sinks is $\Omega(|V|^{1/3+\delta})$ for any $\delta > 0$.

Another consequence of our new approach is a significantly simpler FPAS for the multicommodity flow problem over time. This new FPAS is faster than the previous FPAS [2] by a factor of roughly $O(|A|^6k^4)$.

In Section 5, using very different techniques, we describe a new and simple capacity scaling FPAS for the minimum cost flow over time problem when costs are proportional to transit times. The FPAS does not use intermediate node storage, and runs in $\tilde{O}(|A|^2\epsilon^{-1}\log U)$ time, on a network with integer capacities bounded by U. This is a speed up of roughly $|A|^6$ over the general FPAS described in [2] and a significant reduction in the dependence on ϵ of the simpler general FPAS we provide here. In contrast to the preceding algorithms which require constructing a large time-expanded network, this FPAS runs directly on the original network $\mathcal{N}$. Our algorithm builds on a connection we establish between minimum cost flow over time problems when costs are proportional to transit times, and the universally maximal flow over time problem.

2 Preliminaries

2.1 Static Flows

A *static flow* x on $\mathcal{N}$ assigns every arc e a non-negative flow value x_e such that *flow conservation constraints* $\sum_{e\in\delta^+(v)} x_e - \sum_{e\in\delta^-(v)} x_e = 0$ for all $v \in V \setminus S$, are obeyed. The static flow x *satisfies the supplies and demands* if $\sum_{e\in\delta^+(v)} x_e - \sum_{e\in\delta^-(v)} x_e = D_v$ 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 the supply $D = D_s = -D_t$ has *value* $|x| = D$. Finally, a flow x is called *feasible* if it obeys the *capacity constraints* $x_e \le u_e$, for all $e \in A$. The cost of a static flow x is defined as $c(x) := \sum_{e\in A} c_e(x_e)$.

A path P in network $\mathcal{N}$ is a subgraph consisting of an ordered sequence of nodes $(v_0, v_1, \ldots, v_p) \subseteq V^{p+1}$ and arcs $(v_i, v_{i+1}) \in A$ between each consecutive pair of nodes. It is a well-known result from network flow theory that any static flow x in $\mathcal{N}$ can be decomposed into flows x_P on simple paths $P \in \mathcal{P}$, where $\mathcal{P}$ is the set of all possible simple flow paths (and flows on cycles[3]). That is, $x_e = \sum_{P\in\mathcal{P}\,:\,e\in P} x_P$ for all $e \in A$. The number $|\mathcal{P}|$ of simple paths can be bounded by the number of arcs m. Similarly, a flow over time f in $\mathcal{N}$ without intermediate storage can be decomposed into flows over time f_P on paths $P \in \mathcal{P}$. Here, $f_P(\theta)$ denotes the rate of flow entering path P at time θ and thus arriving at the end vertex of the path at time $\theta + \tau(P)$.

2.2 Time-Expanded Networks

Traditionally, flows over time are solved in a time-expanded network. Given a network $\mathcal{N} = (V, A)$ with integral transit times on the arcs and an integral time horizon T, the *T-time-expanded network* of $\mathcal{N}$, denoted $\mathcal{N}^T$ is obtained by creating T copies of V, labeled V_0 through V_{T-1}, with the θ^{th} copy of node v denoted v_θ, $\theta = 0, \ldots, T-1$. 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 $0 \le \theta < T - \tau_e$, there is an *transit arc* e_θ from v_θ to $w_{\theta+\tau_e}$ with the same capacity and cost as arc e. For each terminal $v \in S$ there is an additional infinite capacity *holdover arc* from v_θ to $v_{\theta+1}$, for all $v \in V$ and $0 \le \theta < T - 1$, 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. We treat the first copy v_0 of a source $v \in S^+$ as the corresponding source in $\mathcal{N}^T$; and treat the last copy v_{T-1} of a sink $v \in S^-$ as the corresponding sink in $\mathcal{N}^T$. In the model with intermediate storage of flow at nodes, we introduce holdover arcs for all nodes $v \in V$.

Any static flow in this time-expanded network corresponds to a flow over time of equal cost: 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

[2] Klinz and Woeginger [10] claim a special case of this result: the single-source, single-sink, minimum cost flow over time does not require storage.

[3] We will show in section 3 that the path decompositions we obtain contain only paths, no cycles.

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 $[\theta, \theta+1)$. Thus, we may solve any flow-over-time problem by solving the corresponding static flow problem in the time-expanded network.

One problem with this approach is that the size of $\mathcal{N}^T$ depends linearly on T, so that if T is not bounded by a polynomial in the input size, this is not a polynomial-time method of obtaining the required flow over time. However, if all arc lengths are a multiple of $\Delta > 0$ such that $\lceil T/\Delta \rceil$ is bounded by a polynomial in the input size, then instead of using the T-time-expanded network, we may rescale time and use a Δ-*condensed time-expanded network* that contains only $\lceil T/\Delta \rceil$ copies of V. In the Δ-condensed, time-expanded network, there are $\lceil T/\Delta \rceil$ copies of V. Copy V_θ corresponds to flow through V in the interval $[\theta\Delta, (\theta+1)\Delta)$. Thus, the capacity of transit arc e_θ is $u_e\Delta$. Other capacities and costs are the same as in the unit-interval, time-expanded network. We denote this condensed, time-expanded network by $\mathcal{N}^T/\Delta$.

LEMMA 2.1. ([2], LEMMA 4.1) *Suppose that all arc lengths are multiples of Δ and T/Δ is an integer. Then, any flow over time that completes by time T corresponds to a static flow of equal cost in $\mathcal{N}^T/\Delta$, and any flow in $\mathcal{N}^T/\Delta$ corresponds to a flow over time of equal cost that completes by time T.*

3 Min-Cost Flows without Storage

In this section we show that the ability to store flow at intermediate nodes does not help reduce the cost of a min-cost transshipment-over-time. This result also holds if the cost on an arc is an arbitrary convex function of the flow rate into the arc. As mentioned in Section 2.2, when transit times are integers, the min-cost transshipment-over-time problem with or without intermediate storage can be solved by solving the corresponding static flow problem in the time-expanded network $\mathcal{N}^T$.

THEOREM 3.1. *The cost of a minimum convex cost transshipment over time that does not use intermediate node storage is no more than the cost of a minimum convex cost transshipment over time using intermediate node storage.*

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.

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 vertices 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 $[\varphi+q, \varphi+q+1)$, $q > 0$ 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 $(i_{\theta-\tau_{ij}}, j_\theta)$

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

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 $x_{(i_\theta, j_{\theta+\tau_{ij}})} :=$

$x_{(i_\theta,j_{\theta+\tau_{ij}})} + \kappa$ and $x_{(i_{\theta-q},j_{\theta-q+\tau_{ij}})} := x_{(i_{\theta-q},j_{\theta-q+\tau_{ij}})} - \kappa$. If $(i_\theta, j_{\theta+\tau_{ij}}) \in P$ is blue, then $x_{(i_\theta,j_{\theta+\tau_{ij}})} := x_{(i_\theta,j_{\theta+\tau_{ij}})} - \kappa$ and $x_{(i_{\theta-q},j_{\theta-q+\tau_{ij}})} := x_{(i_{\theta-q},j_{\theta-q+\tau_{ij}})} + \kappa$. 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 such 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, 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 $|x(\delta(w_{\mu-q}))| > \kappa$, $|x(\delta(w_\mu))| < -\kappa$, or, since $\kappa \leq x(\delta(w_{\mu-q})) - x(\delta(w_\mu))$, $|x(\delta(w_\mu))|$ 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. □

Theorem 3.1 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 3.1. *For every instance of the minimum convex cost transshipment-over-time problem, there exists an optimal solution without intermediate 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. □

4 Approximation Schemes

In this section we present a fully polynomial time approximation scheme for the quickest transshipment problem with bounded (linear) cost which does not use storage at intermediate nodes. We also discuss a generalization of our approach to the multicommodity flow setting at the end of this section.

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.

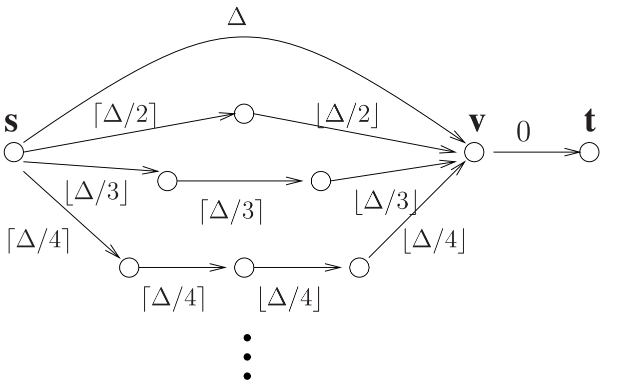

Figure 1: In this partially drawn unit capacity network, there are Δ paths from s to v. The i^{th} path contains i arcs, each with transit time roughly Δ/i.

In order to obtain provably good solutions in this way, one has to make sure that the following two conditions are fulfilled: (a) the value of an optimal solution to the instance with increased transit times (obtained in the condensed time-expanded network) approximates the value of an optimal solution in the original setting; (b) 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 a simple example to show that non-trivial problems have to be dealt with to address both (a) and (b).

Consider first a (sub)network consisting of four nodes $\{1, 2, 3, 4\}$ and three arcs $(1, 3)$, $(2, 3)$, $(3, 4)$. All capacities are 1. The transit times are $\tau_{(1,3)} = \Delta/2$, $\tau_{(2,3)} = \Delta$, and $\tau_{(3,4)} = 0$. A flow in the graph without rounded transit times can send $\Delta/2$ units of flow in interval $[0, \Delta/2)$ on each path $P_1 = 1 \to 3 \to 4$ and $P_2 = 2 \to 3 \to 4$. 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 $(3, 4)$.

Now consider the unit capacity (sub)network depicted in Figure 1. 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.

As pointed out in [2], condition (b) 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. If Δ' is large, then this requires a large amount of additional storage. Here, we present a new approach that works if storage of flow at nodes is not allowed.

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 (a). Second, we average the flow computed in the rounded network over sufficiently large intervals relative to Δ so that bottlenecks caused by problems of type (b) 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.

In the following, let T^* be the makespan of a quickest transshipment with cost bounded by C. We start with a description of the algorithm in Figure 2. The exact choice of the time horizon T' will be given in the analysis of the algorithm. (See Lemma 4.1.)

INPUT: network $\mathcal{N}$ with capacities, linear costs, and 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 $\Delta := \varepsilon^2\, T/n$;
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 $(1+\varepsilon)\, D$ at cost at most $(1+\varepsilon)\, C$;
5. interpret x' as flow over time f' with path decomposition $(f'_P)_{P \in \mathcal{P}}$;
6. set $f_P(\theta) := \frac{1}{1+\varepsilon} \frac{1}{\varepsilon\, T} \int_{\theta - \varepsilon\, T}^{\theta} f'_P(\xi)\, d\xi$ for all $P \in \mathcal{P}$ and $\theta \in [0, T' + \varepsilon\, T)$.

Figure 2: A fully polynomial time approximation scheme.

The flow over time f' lives in network $\mathcal{N}$ with transit times rounded up to multiples of $\Delta = \varepsilon^2\, T/n$. In contrast, the flow over time f is defined on $\mathcal{N}$ with original transit times. It uses 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, it follows from [2, Section 3] that there is a constant factor approximation algorithm for the quickest transshipment problem with bounded cost which yields a lower bound L and an upper bound U on T^* such that $U \in O(L)$. Using this, then 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. 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 $\varepsilon^2\, T/n$ where f'_P attains a positive constant value (notice that a 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.

We next discuss the choice of T' in the algorithm.

LEMMA 4.1. *If T' is chosen to be at least $T(1+\varepsilon+\varepsilon^2)\,(1+\varepsilon)^2$, 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_e. The time horizon of f^* is T^* and its cost is at most C. By Corollary 3.1, there exists a decomposition of $f^* = (f^*_P)_{P \in \mathcal{P}^*}$ into flows over time f^*_P on simple paths $P \in \mathcal{P}^*$.

Consider an arbitrary arc $e = (v,w) \in A$. The total flow into arc e at time θ in f^* is

$$f^*_e(\theta) \;=\; \sum_{P \in \mathcal{P}^* \,:\, e \in P} f^*_P\big(\theta - \tau(P,e)\big) \;\leq\; u_e\,. \tag{4.3}$$

Here, $\tau(P,e)$ denotes the length of the subpath of P which is obtained by removing arc e and all its successors. We obtain a 'smoothed' flow over time $(\hat{f}_P)_{P \in \mathcal{P}^*}$ with time horizon $(1+\varepsilon)\, T$ by defining

$$\hat{f}_P(\theta) \;:=\; \frac{1}{\varepsilon\, T} \int_{\theta - \varepsilon\, T}^{\theta} f^*_P(\xi)\, d\xi \tag{4.4}$$

for $\theta \in [0, (1+\varepsilon)\, T)$ and $P \in \mathcal{P}^*$. An illustrative example is given in Figure 3. 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 $\mathcal{N}$ when transit times are rounded up to multiples of $\varepsilon^2\, T/n$. We denote the rounded transit

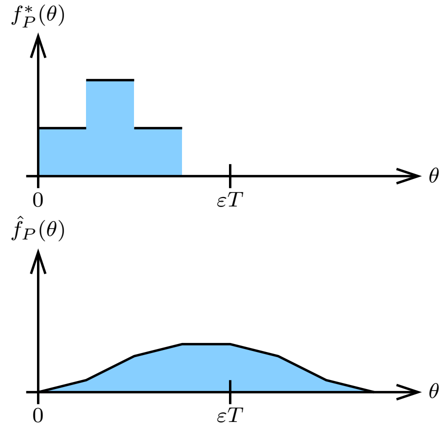

Figure 3: The 'smoothed' path flow over time $\hat{f}_P$ in comparison to the original flow over time f_P^* sent into path P.

time of arc $e \in A$ by $\tilde{\tau}_e$. Since every path $P \in \mathcal{P}^*$ is simple, it contains at most $n-1$ arcs; therefore,

$$\tau(P) \;\le\; \tilde{\tau}(P) \;\le\; \tau(P) + \varepsilon^2\, T \tag{4.5}$$

and $\tau(P,e) \le \tilde{\tau}(P,e) \le \tau(P,e) + \varepsilon^2\, T$, for all $e \in P$. Thus, in the setting with rounded transit times we get, for all $e \in A$ and $\theta \in [0, (1+\varepsilon)\, T + \varepsilon^2\, T)$,

$$\begin{aligned}
\hat{f}_e(\theta) &= \sum_{P\in\mathcal{P}^*:\, e\in P} \hat{f}_P(\theta - \tilde{\tau}(P,e))\\
&\overset{(4.4)}{=} \frac{1}{\varepsilon\, T} \sum_{P\in\mathcal{P}^*:\, e\in P} \int_{\theta-\tilde{\tau}(P,e)-\varepsilon\, T}^{\theta-\tilde{\tau}(P,e)} f_P^*(\xi)\, d\xi\\
&\le \frac{1}{\varepsilon\, T} \sum_{P\in\mathcal{P}^*:\, e\in P} \int_{\theta-\tau(P,e)-\varepsilon^2\, T-\varepsilon\, T}^{\theta-\tau(P,e)} f_P^*(\xi)\, d\xi \qquad (4.6)
\end{aligned}$$

(since $\tau(P,e) \le \tilde{\tau}(P,e) \le \tau(P,e) + \varepsilon^2\, T$)

$$\begin{aligned}
&= \frac{1}{\varepsilon\, T} \int_{\theta-\varepsilon^2\, T-\varepsilon\, T}^{\theta} \sum_{P\in\mathcal{P}^*:\, e\in P} f_P^*(\xi - \tau(P,e))\, d\xi\\
&\overset{(4.3)}{=} \frac{1}{\varepsilon\, T} \int_{\theta-\varepsilon^2\, T-\varepsilon\, T}^{\theta} f_e^*(\xi)\, d\xi\\
&\le \frac{\varepsilon^2\, T + \varepsilon\, T}{\varepsilon\, T}\, u_e \;=\; (1+\varepsilon)\, u_e\,. \qquad (4.7)
\end{aligned}$$

Thus, if we scale $\hat{f}$ by a factor of $1/(1+\varepsilon)$, we get a feasible flow over time for the setting with rounded transit times. The time horizon of this flow is $\hat{T} = (1+\varepsilon+\varepsilon^2)\, T$, its cost is $c(\hat{f})/(1+\varepsilon) \le C/(1+\varepsilon)$ and it satisfies demands $D/(1+\varepsilon)$.

In step 3, we require a flow to satisfy demands $D(1+\varepsilon)$. To obtain this, we use Lemma 4.2 below with $\delta = (1+\varepsilon)^2$, to see that it is sufficient to start with a time horizon of $T' := (1+\varepsilon)^2 \hat{T} = (1+\varepsilon+\varepsilon^2)(1+\varepsilon)^2 T$. □

We show that increasing demands and supplies by a factor of $1+\delta$ requires an increase in time and cost by at most $1+\delta$.

LEMMA 4.2. *Consider a flow over time f with time horizon T and cost $c(f)$, satisfying a vector of demands and supplies D. Then, there exists a flow over time f' satisfying demands and supplies $(1+\delta)\, D$ within time $(1+\delta)\, T$ and with cost $(1+\delta)\, c(f)$.*

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 $1+\delta$. Then, the $(1+\delta)$-condensed time-expanded network of the modified instance with time horizon $(1+\delta)\, T$ is identical to the T-time-expanded network of the original instance, but with arc capacities multiplied by a factor of $1+\delta$. In particular, $(1+\delta)\, x$ defines a feasible flow over time with time horizon $(1+\delta)\, T$ and cost $(1+\delta)\, c(f)$ satisfying demands and supplies $(1+\delta)\, D$ 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. □

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 from the same line of arguments given in the proof of Lemma 4.1 for the flow over time $\hat{f}$. This yields the following main result of this section.

THEOREM 4.1. *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.

Quickest Multicommodity Flows. The quickest multicommodity flow problem with costs is defined as follows. We are given a network $\mathcal{N} = (V, A)$ with capacities, transit times, and costs on the arcs. Moreover, there are k commodities $i = 1, \ldots, k$, each given by a source-sink pair $(s_i, t_i) \in V^2$ and a demand value D_i. We are looking for s_i-t_i-flows over time with time horizon T and value D_i, for $i = 1, \ldots, k$

that share the arc capacities: the sum of flow values over all commodities on an arc must never exceed the capacity of the arc. The sum of the costs of the k flows over time must not exceed a given budget C and the task is to minimize the common time horizon T.

In [2] we discuss an example which shows that an optimal solution to this problem must use intermediate storage of flow at 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 (4.6) relies on the fact that one can restrict to simple flow paths, since it uses (4.5). However, if intermediate storage is allowed, its easy to see that one can restrict to simple paths. In this case, we can generalize the approach given above to get the following theorem.

THEOREM 4.2. *Consider an instance of the quickest multicommodity flow problem with bounded cost and intermediate node storage. For any $\varepsilon > 0$, a $(1+\varepsilon)$-approximate flow over time with bounded cost 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 Length-Proportional Costs

In this section, we describe a simple and fast FPAS for the minimum cost flow over time problem when costs are proportional to transit times. This FPAS will also not use any intermediate storage. We begin by establishing a connection between this version of the minimum cost flow over time problem and an earliest-arrival, latest-departure flow.

5.1 Universally Quickest Flows

THEOREM 5.1. *Let D be the value of a maximum flow in $\mathcal{N}$ with time horizon T. A minimum, length-proportional cost flow over time of value D completing by time T that uses no intermediate node storage is also a universally quickest flow completing by time T.*

Proof. The cost of the flow over time is

$$\begin{aligned} c(f) &= \sum_{e\in A}\int_0^T c_e(f_e(\theta))\,d\theta \;=\; \int_0^T \sum_{e\in A}\tau_e\, f_e(\theta)\,d\theta \\ &= \int_0^T\int_0^\theta\left(\sum_{e\in\delta^+(s)} f_e(\varphi) - \sum_{e\in\delta^-(s)} f_e(\varphi-\tau_e)\right)d\varphi \\ &\quad - \int_0^\theta\left(\sum_{e\in\delta^-(t)} f_e(\varphi-\tau_e) - \sum_{e\in\delta^+(t)} f_e(\varphi)\right)d\varphi\,d\theta. \end{aligned}$$

The last equality follows since if there is no storage of flow then the time a unit of flow spends in transit is equal to the time it takes to travel from the source to the sink. The last expression is minimized when f is an earliest arrival, latest departure flow. □

Together with Theorem 3.1 this implies the equivalence between minimum length-proportional cost flows and universally quickest flows (i.e. earliest-arrival, latest-departure flows).

COROLLARY 5.1. *Let D be the value of a maximum flow in $\mathcal{N}$ with time horizon T. A minimum length-proportional cost flow over time of value D completing by time T is also a universally quickest flow completing by time T.*

In the discrete time model, a universally maximal flow over time can be computed in the time-expanded network by using lexicographically maximal flows introduced by Minieka [11]. 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. $\{\nu_1, \nu_2, \dots, \nu_k\}$, 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 $S_i = \{\nu_1, \nu_2, \dots, \nu_i\}$. A universally maximal flow over time with time horizon T is a lexicographically maximal flow in the time-expanded network with ordering of sources and sinks as $\{s_{T-1}, s_{T-2}, \dots, s_1, s_0, t_{T-1}, t_{T-2}, \dots, t_1, t_0\}$. However, due to the exponential size of the time-expanded network, this insight does not lead to an efficient algorithm for the problem.

Wilkinson and Minieka also describe algorithms that use only the original network $\mathcal{N}$. They are not polynomial, but only pseudo-polynomial, as they are based on the successive shortest path algorithm. The algorithm presented in Figure 4 is a slight modification of an interpretation by Hoppe and Tardos [8, 7] of algorithms by Wilkinson [14] and Minieka [11] for the universally maximal flow problem. Before discussing the algorithm, it is necessary to define some concepts and notation.

Chain Decompositions. Let $\gamma = \langle P, \omega\rangle$ be the static flow of value ω along path P. For a given $T \geq \tau(P)$, the static flow γ *induces* a flow over time with time horizon T, which sends flow at rate ω into path P during the time interval $[0, T-\tau(P))$ such that all flow reaches the end of P before time T. The value of this flow over time is $(T-\tau(P))\,\omega$.

Consider a path decomposition $\Gamma = \{\gamma_1, \dots, \gamma_k\}$ of a static flow x in $\mathcal{N}$, that is, $x = \sum_{i=1}^k \gamma_i$. If the transit times of all underlying paths in Γ are bounded by T, then Γ (and thus x) induces a *temporally repeated flow* $[\Gamma]^T$ which is the sum of all flows over time induced by $\gamma_i = \langle P_i, \omega_i\rangle$, for $i = 1, \dots, k$. The value of this flow over time is denoted by $|[\Gamma]^T|$ and equals $\sum_{i=1}^k (T-\tau(P_i))\omega_i$.

We also consider paths P in the bidirected network corresponding to $\mathcal{N}$ where, for each *forward arc* $e = (v, w) \in A$, there is also a *backward arc* $\bar{e} = (w, v)$ with transit time $\tau_{(w,v)} = -\tau_{(v,w)}$. For such a path P containing backward arcs, the corresponding generalized path flow $\gamma = \langle P, \omega \rangle$ in $\mathcal{N}$ assigns the non-positive value $-\omega$ to the corresponding forward arcs, that is, it pumps flow in the wrong direction through these forward arcs. A collection $\Gamma = \{\gamma_1, \ldots, \gamma_k\}$ of generalized flows on paths yields a feasible static flow $x = \sum_{i=1}^{k} \gamma_i$ if, for any arc $e \in A$, the sum of the flow values on this arc in $\gamma_1, \ldots, \gamma_k$ is non-negative; i.e. $\sum_{i:e\in\gamma_i} \omega_i - \sum_{i:\bar{e}\in\gamma_i} \omega_i \geq 0$.

Establishing the feasibility of the corresponding flow over time $[\Gamma]^T$ is considerably more complicated. First consider the case of only one generalized path flow $\Gamma = \{\langle P, v \rangle\}$ where P contains a backward arc (w, v) with $(v, w) \in A$. Then, flow sent by $[\Gamma]^T$ into arc (w, v) at time θ arrives in v at time $\theta + \tau_{(w,v)} = \theta - \tau_{(v,w)}$. Thus, the flow travels backwards in time. This is of course not feasible. However, if at time $\theta - \tau_{(v,w)}$ flow at the same or a higher rate was sent into arc (v, w), thus arriving in w at time θ, it can cancel the flow on (w, v). Thus, for a set of path flows Γ, the resulting flow over time $[\Gamma]^T$ is feasible if for any point θ in time and for any backward arc (w, v) the flow sent through (w, v) is canceled by flow sent on (v, w), arriving in w at time θ. Such a flow over time $[\Gamma]^T$ is called *chain-decomposable*; in particular, any temporally repeated flow is chain-decomposable. For an elaborate discussion of this topic we refer to [9, 7].

Given a feasible static flow x in $\mathcal{N}$, define $\mathcal{N}_x$ to be the residual graph of x. For nodes $s,\ y \in V$, define $d_x(s, y)$ to be the shortest path distance with respect to τ from s to y in $\mathcal{N}_x$. (If there is no path from s to y in $\mathcal{N}_x$, then $d_x(s, y) = \infty$.)

The Successive Shortest Path Algorithm. The algorithms described by Wilkinson and Minieka are variants of the well-known successive shortest path algorithm. A restatement of their algorithm appears in Figure 4.

```
MCFT(T, D):

Γ ← ∅, D' ← D
x ← zero flow
while d_x(s,t) < T and D' > 0 {
      P ← shortest (s,t)-path in N_x
      v ← min{ residual capacity of P, D'/(T−τ(P)) }
      augment x by v along P
      Γ ← Γ + {⟨v,P⟩}
      D' ← D' − v(T − τ(P))
}
return Γ
```

Figure 4: Algorithm for universally maximal flow.

THEOREM 5.2. (WILKINSON [14] AND MINIEKA [11]) *Let D be the value of a maximum flow in $\mathcal{N}$ with time horizon T. Let Γ be the set of chain flows returned by* $\mathsf{MCFT}(T, D)$. *Then, $[\Gamma]^T$ is a universally quickest flow over time in $\mathcal{N}$ completing by time T.*

Theorem 5.2 and Corollary 5.1 imply that when D is the value of a maximum flow in $\mathcal{N}$ with time horizon T, then $\mathsf{MCFT}(T, D)$ returns a set of chain flows Γ such that $[\Gamma]^T$ is a minimum cost flow over time for cost vector $c = \tau$. The following theorem states a stronger result: when $c = \tau$, $[\Gamma]^T$ is a minimum cost flow over time for *all* feasible D.

THEOREM 5.3. *For cost vector $c = \tau$ and set of chain flows Γ returned by* $\mathsf{MCFT}(T, D)$, *$[\Gamma]^T$ is a minimum cost flow over time in $\mathcal{N}$ of value D.*

Proof Sketch. We prove this by interpreting $[\Gamma]^T$ in the time-expanded network and producing dual variables π that demonstrate the optimality of $[\Gamma^T]$ by satisfying complementary slackness conditions in the time-expanded network.

We start by determining the flow in $[\Gamma]^T$ on arc (y, z) at time θ, i. e. $f^*_{(y,z)}(\theta)$. In the time-expanded network, this corresponds to the flow on arc $(y_\theta, z_{\theta+\tau_{(y,z)}})$. Let

$$\Gamma' \ := \ \{\gamma \in \Gamma \mid d_{x^\gamma}(s, y) \leq \theta \text{ and } d_{x^\gamma}(y, t) \leq T - \theta\} \ .$$

Here, x^γ denotes the static flow x in $\mathsf{MCFT}(T, D)$ before the iteration in which the path flow γ is added to Γ. Let $x' := \sum_{\gamma\in\Gamma'} \gamma$. Then $f^*_{(y,z)}(\theta) = x'_{(y,z)}$.

We determine the dual variable $\pi(y_\theta)$ of node y_θ in the time-expanded network by

$$\pi(y_\theta) \ := \ \begin{cases} -d_{x'}(s, y) & \text{if } d_{x'}(s, y) \leq \theta, \\ -\infty & \text{otherwise.} \end{cases}$$

The proof concludes by showing that the vector π and the flow corresponding to $[\Gamma]^T$ in the time-expanded network satisfy standard complementary slackness conditions for minimum cost flows. The details are technical and are contained in the full paper. □

5.2 A Capacity-Scaling FPAS

The successive shortest path algorithm described in the previous section requires an exponential number of iterations in the worst case; see, e. g., Zadeh [15]. For the universally maximal flow problem, Hoppe and Tardos [8, 7] present a fully polynomial time approximation scheme based on successive shortest paths in a capacity scaling framework. For every $\varepsilon > 0$, their algorithm efficiently computes a flow over time whose value is within a factor of $(1 - \varepsilon)$ of the universally maximal dynamic flow over any time interval $[0, \theta]$, $\theta = 0, \ldots, T$, and the same performance guarantee holds for the departure schedule. Their result, however, does not imply that the same algorithm is an FPAS for the minimum length-proportional-cost flow over time problem, since they approximate flow value, not flow cost. The Hoppe-Tardos algorithm is presented in Figure 5 as $\mathsf{Universal}(T, \varepsilon)$.

$\mathsf{Universal}(T, \varepsilon)$:

```
Γ ← ∅; Δ ← 1; ũ ← u; x ← zero flow
while (∃(s,t)-path in N_{ũ,x} of length ≤ T) {
    ρ ← 0
    while (ρ < mΔ/ε) and
     (∃(s,t)-path in N_{ũ,x} of length ≤ T) {
        P ← shortest (s,t)-path in N_{ũ,x}
        v ← residual capacity of P
        augment x by v along P
        Γ ← Γ + {⟨v,P⟩}
        ρ ← ρ + v
    }
    Δ ← 2Δ
    ∀ yz ∈ E : ũ_{yz} ← ũ_{yz} − (ũ^x_{yz} mod Δ)
}
return Γ
```

Figure 5: FPAS for universally maximal flow.

We prove that a modification yields an FPAS for the flow over time problem with length-proportional costs.

THEOREM 5.4. (HOPPE AND TARDOS [8]) $[\Gamma]^T$ *is a feasible flow over time.*

Our algorithm $\mathsf{ApproxMCFT}(T, D, \varepsilon)$ starts by invoking $\mathsf{Universal}(T, \varepsilon)$ modified so that augmentation is allowed on paths of value Δ as long as $\rho < 4m\Delta/\varepsilon$ (instead of $\rho < m\Delta/\varepsilon$). This change does not affect the proof and thus the correctness of the corresponding version of Theorem 5.4. Denote the resulting set of chain flows by Γ. $\mathsf{ApproxMCFT}(T, D, \varepsilon)$ then increases T to $T(1+\varepsilon)$ and modifies Γ to Γ' by reducing the value of the longest chain flows in Γ to zero, one-by-one, until $|[\Gamma']^{T(1+\varepsilon)}| = D$.

THEOREM 5.5. $\mathsf{ApproxMCFT}(T, \varepsilon)$ *computes a flow over time of value D and makespan $T(1+\varepsilon)$ with cost at most the cost of the min-cost flow over time of value D and makespan T.* $\mathsf{ApproxMCFT}(T, \varepsilon)$ *runs in time $O(\varepsilon^{-1} m(m + n\log n)\log U)$, on a network with integer capacities bounded by U.*

Proof Sketch. The essential idea of the proof is to show that the flow lost by rounding down residual capacities in one iteration can be compensated by increasing the time horizon for flow paths added to Γ in the previous iteration from T to $T(1+\varepsilon)$. Since the successive shortest path algorithm adds paths in increasing order of cost, the cost of a "lost" flow path is at least the cost of the flow paths in the previous iteration, and thus the cost of the new flow is at most the cost of the original. The full proof is technical and is contained in the full paper.

The run time is dominated by the run time of $\mathsf{Universal}(T, \varepsilon)$ for which each path calculation takes $O(m + n\log n)$ time, and there are at most $4m/\varepsilon$ paths per iteration. Since Δ is doubled each iteration, the number of iterations is bounded by $O(\log U)$. □

References

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

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

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

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

[5] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *39th Annual IEEE Symposium on Foundations of Computer Science*, pages 300–309, 1998.

[6] A. Hall and M. Skutella. Personal communication, 2002.

[7] B. Hoppe. *Efficient Dynamic Network Flow Algorithms*. PhD thesis, Cornell University, June 1995. Department of Computer Science Technical Report TR95-1524.

[8] B. Hoppe and É. Tardos. Polynomial time algorithms for some evacuation problems. In *Proc. of 5th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 433–441, 1994.

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

[10] 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*, number 920 in Lecture Notes in Computer Science, pages 329–343. Springer-Verlag, 1995.

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

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

[13] 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, *Handbooks in Operations Research and Management Science: Networks*. Elsevier Science Publishers B. V., 1995.

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

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