Efficient Continuous-Time Dynamic Network Flow Algorithms *

L. Fleischer† É. Tardos‡

August 1996

Abstract

We extend the discrete-time dynamic flow algorithms presented in [5, 19, 13, 9, 10, 8] to solve the analogous continuous-time dynamic flow problems. These problems include finding maximum dynamic flows, quickest flows, universally maximum dynamic flows, lexicographically maximum dynamic flows, dynamic transshipments, and quickest transshipments in networks with capacities and transit times on the edges.

1 Introduction

Ford and Fulkerson introduced the maximal dynamic flow problem in [5]. The problem is defined on a *dynamic network* $\mathcal{N} = (V, E, u, \tau, \{s, t\})$: a set of nodes V, a set of directed edges E with non-negative, integral capacities u and transit times τ, and a subset S of nodes called terminals—here, a source s and a sink t. Given this and a time horizon T, the objective is to find a dynamic flow that sends as much flow as possible from the source to the sink in time T. Time, in the Ford-Fulkerson model, is measured in discrete steps, so that if one unit of flow leaves node i at time θ on arc (i, j), one unit of flow arrives at node j at time $\theta + \tau_{ij}$, where τ_{ij} is the transit time of arc (i, j). The model also allows storage at any node in the network.

The research on dynamic flow problems has taken two approaches. One approach models time in discrete time steps. The other approach models time continuously. Research of the first type typically uses the time-expanded network, either explicitly in the algorithms, or implicitly in the proofs, to produce theoretically or practically efficient algorithms. Research using the second

*Cornell University, School of Operations Research and Industrial Engineering, Technical Report TR1166

†email: lisaf@orie.cornell.edu. School of Operations Research and Industrial Engineering, Cornell University. Supported in part by an AAUW Selected Professions Engineering Dissertation Fellowship and by ONR through grant N00014-96-1-0050.

‡Computer Science Department and School of Operations Research and Industrial Engineering, Cornell University. Research supported in part by an NSF PYI award, by NSF through grant DMS 9505155, and by ONR through grant N00014-96-1-0050.

approach has considered networks with time-varying capacities and costs, and has focussed on proving the existence of optimal solutions while further generalizing the model. In this paper, we try to relate the two approaches by extending some of the polynomial time algorithms that work in the discrete-time model to produce optimal continuous dynamic flows.

A *time-expanded network* is a directed graph that contains one copy of every node in the dynamic network for every time step, i.e., its vertex set is $\{j(\theta) : j \in V, \theta \in \{0, 1, \ldots, T\}\}$; and, for every arc (j, k) in the dynamic network, it contains arcs $(j(\theta), k(\theta + \tau_{jk}))$, $\theta \in \{0, \ldots, T - \tau_{jk}\}$. The time-expanded network is useful because the transit times are implicit in the network, and thus all problems that we can solve in traditional networks, we can solve in this one. For example, the maximum dynamic flow problem in time T becomes, in this graph, a maximum flow problem from source $s(0)$ to sink $t(T)$. Unfortunately, this graph may be very large, and is thus not practical to work with for large discretizations.

There are algorithms for problems in the discrete-time model that work with the dynamic network directly. Ford and Fulkerson [5] describe an algorithm to solve the maximum dynamic flow problem with one minimum cost circulation computation. Wilkinson [19] and Minieka [13] both describe a simple, but exponential time algorithm to solve the universally maximum flow problem. In their surveys, Aronson [4] and Powell et al. [16] summarize much of the progress made since then. Recently, Hoppe and Tardos [9, 10, 8] have described several polynomial time algorithms for discrete dynamic network problems including approximate universally maximum dynamic flows, lexicographically maximum flows, and dynamic transshipment.

There has also been much research on continuous network flows, most of it for very general networks. Anderson, Nash, and Philpott [2] consider the problem of finding a maximum flow in a network with zero transit times, but with capacities and node storage that vary over time, and develop a duality theory for continuous network flows. In their book, Anderson and Nash present this result as well as other work on continuous linear programs [1]. Since then, there has been much more work on continuous dynamic flow problems in networks with time-varying edge capacities, storage capacities, or costs. For example, see [3, 15, 14, 18, 17]. One focus of this research is to extend the class of these time-varying functions for which optimal solutions exist. These general continuous flow problems appear to be very hard to solve: the algorithms developed so far fall short of being efficient, either theoretically or practically, and the implementations do not seem able to handle problems with more than a few nodes.

Hajek and Ogier [7] provide the first polynomial time algorithm to solve a continuous dynamic network flow problem. They consider a network with zero transit times, constant upper bounds

on the flow rates, and infinite node storage, and prove a polynomial time algorithm to empty the network of excess supply in minimum time.

In this paper, we extend the polynomial algorithms developed for the discrete-time model to a continuous-time model more general than the Hajek and Ogier model, but more specialized than most of the models discussed in the continuous flow literature. We assume that transit times are integral and constant over time. Edge capacities, denoted u_e, are now upper bounds on the rates of flow, and are constant and integral per unit time. Like Ford and Fulkerson, we allow storage at the nodes. However, the optimal solutions found by the algorithms presented here will not use any storage. Since our solutions are thus unaffected by upper bounds on storage capacities, we assume, for simplicity, that storage capacities are infinite. One artifact of this discrete time model is that an edge with zero transit time and positive capacity can transmit a positive quantity of flow in zero time. This is counterintuitive, and is corrected in the continuous time model.

In Section 2, we introduce notation and vocabulary for discussing dynamic flows. In Sections 3-7, we prove continuous versions of the above-mentioned algorithms for maximum dynamic flow, quickest flow, universally maximum flow, lexicographically maximum flow, dynamic transshipment, and quickest transshipment.

2 Definitions and Notation

A *static flow* is a function f on the arcs of a graph that obeys *capacity constraints* $0 \leq f \leq u$ and *flow conservation constraints* $\sum_j f_{ij} = 0$, $\forall i \in V \backslash S$. A *static circulation* is a static flow that must also satisfy conservation constraints at the terminals. The *residual network* of a static flow f subject to capacities u is the same network, with capacities redefined as the *residual capacities* $u^f = u - f$.

A *discrete dynamic flow* is a function g that assigns a flow to each arc at each time step. It must also obey capacity constraints $0 \leq g(\theta) \leq u$ for all time steps θ. Since we allow storage at the nodes, the flow conservation constraints are summed over time to prohibit deficit at any node: $\sum_{r=0}^{\theta} \sum_j g_{ij}(r) + g_{ji}(r - \tau_{ji}) \geq 0$, $\forall \theta \in \{0, \ldots, T\}$, $\forall i \in V \backslash \{s, t\}$.

A *continuous dynamic flow* is a function x that defines the rate of flow (per unit time) entering each arc at each moment of time. The capacity constraints are now flow rate constraints, and the flow conservation constraints are the same as for the discrete dynamic flow, with the sum over time replaced by an integral.

All of the discrete-time algorithms are based on the concept of chain flows. A *chain flow* $\gamma = (v, P)$

is a static flow of value $|\gamma| = v$ along path P that starts and ends in the terminal set S of a dynamic network. The length of the chain flow, $\tau(\gamma)$, equals the length of P—the sum of the transit times of the edges in P. Given time horizon T, any chain flow $\gamma = (v, P)$ with $\tau(\gamma) \leq T$ induces a dynamic flow $[\gamma]^T$ by sending v units of flow along P from time 0 until time $T - \tau(\gamma)$. In the discrete setting, this flow has value $v(T-\tau(\gamma)+1)$, while in the continuous setting, it has value $v(T-\tau(\gamma))$. This difference reflects a deficiency of the discrete model—the continuous model captures what we expect from a flow over time.

A *chain decomposition* of a static flow f is a set of chain flows, $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_r\}$, that satisfy $\sum_i \gamma_i = f$. We call Γ a *standard chain decomposition* of f if all chain flows in Γ use edges in the same direction that f does. If Γ is a standard chain decomposition of a feasible flow f and all the chain flows in Γ have length less than or equal to T, then Γ induces a feasible dynamic flow obtained by summing the dynamic flows induced by each of the chain flows. This is called a *standard chain decomposable flow* and is denoted by $[\Gamma]^T$.

2.1 Some Notes on Feasibility and Continuity

If Γ is a non-standard chain decomposition of a feasible flow f, more care must be taken to show that the dynamic flow induced by the chain flows in Γ is feasible. The flow conservation constraints are easily checked. The problem is the capacity constraints: a chain flow that cancels the flow of another chain flow on an edge e might induce a dynamic flow that arrives at e before the flow it is supposed to cancel. Hoppe and Tardos [10, 8] introduce the use of non-standard chain decompositions and show that the dynamic flows induced by the ones used in their algorithms are feasible. In particular, any chain flow that cancels flow at edge e will induce a dynamic flow that arrives at e later than the flow it cancels, and leaves e earlier than the flow it cancels. These feasible, chain-flow-induced dynamic flows are called *chain decomposable flows.*

Given any feasible discrete dynamic flow, we can transform this into a feasible continuous dynamic flow by defining the flow rate x on arc (i, j) in time interval $[\theta, \theta+1)$, $\theta \in \mathcal{Z}$ to be equal to $|f_{ij}(\theta)|$. This transformation has the nice property that the amount of flow arriving at j via arc (i, j) in the unit interval beginning at time θ in the continuous flow will equal the amount of flow that arrives at j via arc (i, j) at time step θ in the discrete flow. All transformations of discrete dynamic flows into continuous dynamic flows mentioned in this paper will be of this natural form. We prove here that these natural transformations produce optimal continuous dynamic flows.

If our time horizon is integral, than the natural transformation of chain decomposable flows are also feasible. If the time horizon T is not integral, then we can create a continuous T-horizon flow

by naturally transforming a discrete $\lceil T \rceil$-horizon chain-decomposable flow into a continuous chain decomposable flow, and stop sending flow along each chain flow γ at time $T - \tau(\gamma)$ instead of time $\lceil T \rceil - (\gamma)$. We call such a change the *natural extension* of the original discrete chain decomposable flow. If the discrete chain decomposable flow is feasible, the continuous chain decomposable flow that results from the natural transformation of this flow is also feasible. Since all transit times are integral, all chain flows have integral length, and so the chain flows used with time bound $\lceil T \rceil$ can also be used with time bound T. Thus reducing the time bound does not affect the time a chain flow starts using an edge. It does reduce the time that the chain flow uses the edge, however. But since the dynamic flow induced by a chain flow is shortened by the same amount for each chain flow, no gap is created between a flow and a cancelling flow. Hence the new flow is also feasible.

All the dynamic flows discussed in this paper are based on chain decomposable flows, and all the extensions of the discrete algorithms will use natural extensions of these flows. We have just seen that the resulting flows must be feasible, hence the proofs in this paper are restricted to establishing optimality.

3 The Maximum Dynamic Flow Problem

Ford and Fulkerson introduced the maximal dynamic flow problem in [5]. The object is to send as much flow from source to sink by a given time bound. They showed how to solve this discrete-time problem by computing a minimum cost flow problem in a related network.

In the continuous-time version, we are looking to find flow rate x to maximize the amount of flow entering t. We formulate this maximum continuous dynamic flow problem below.

$$\begin{array}{lll} \text{maximize} & v = \int_0^T \sum_{k \in V} [x_{kt}(\theta - \tau_{kt}) - x_{tk}(\theta)] \, \mathrm{d}\theta & \\ \text{subject to} & \int_0^\theta \sum_{k \in V} [x_{kj}(\theta - \tau_{kj}) - x_{jk}(\theta)] \, \mathrm{d}\theta \geq 0 & j \in V,\ \theta \in [0, T] \\ & 0 \leq x_{jk}(\theta) \leq u_{jk} & j, k \in V,\ \theta \in [0, T] \end{array}$$

In this section, we will show that the natural transformation of a $T-1$-horizon, maximum discrete dynamic flow yields a T-horizon, maximum continuous dynamic flow, for T integral; and that we can use the Ford-Fulkerson algorithm to find a maximum dynamic flow for any real $T > 0$. Below, we define a generalized cut, a la Anderson, Nash and Philpott [2], with capacity that is an upper bound on the value of any feasible continuous dynamic flow. We will later show that this bound is met by the natural transformation of the discrete dynamic flow generated by the Ford-Fulkerson

algorithm.

If we let $j(\theta)$ represent node j at time θ, then a *generalized cut* C in the network $\mathcal{N}$ is defined by a set of *cut points* $\{\alpha_j\}, 1 \leq j \leq n$ in $[0,T]$, so that $j(\theta) \in C, \forall \theta \geq \alpha_j$, with $\alpha_s = 0$ and $\alpha_t = T$. This definition is actually more restrictive than the cut defined in [1], but we will show that there is a continuous dynamic flow that saturates such a cut. The *capacity* of generalized cut C is the sum over all edges of the amount of flow that can cross an edge while the end points are on different sides of the cut. Formally, this is

$$\sum_{jk \in E} \int_{\alpha_j}^{\alpha_k - \tau_{jk}} u_{jk} \, \mathrm{d}\theta = \sum_{jk \in E} \max\{0, \alpha_k - \tau_{jk} - \alpha_j\} u_{jk}. \tag{1}$$

Theorem (Anderson, Nash, and Philpott [2]): If x is a feasible, continuous dynamic flow and C is a generalized cut, then the value of the flow x is bounded from above by the capacity of C.

Proof: Let $y_j(\theta)$ be the amount stored in node j at time θ.

$$y_j(\theta) = \int_0^{\theta} \sum_{k \in V} [x_{kj}(\theta - \tau_{kj}) - x_{jk}(\theta)] \, \mathrm{d}\theta, \; j \in V - \{s,t\}$$

Since we require storage to be nonnegative at all times, we have that

$$\begin{aligned} \int_{\alpha_j}^{T} \sum_{k \in V} [x_{kj}(\theta - \tau_{kj}) - x_{jk}(\theta)] \, \mathrm{d}\theta &= y_j(T) - y_j(\alpha_j) \\ &\leq y_j(T), \; j \in V - \{s,t\}. \end{aligned} \tag{2}$$

We can rewrite v as the amount of flow leaving the source by time T, minus the amount stored in the network at time T:

$$v = \int_0^{T} \sum_{k \in V} [x_{sk}(\theta) - x_{ks}(\theta - \tau_{ks})] \, \mathrm{d}\theta - \sum_{j \in V - \{s,t\}} y_j(T). \tag{3}$$

Using (2), (3), together with the fact that $\alpha_s = 0$, and $\alpha_t = T$, we get the following inequality, which leads to the result.

$$\begin{aligned} v &\leq \sum_{j \in V} \int_{\alpha_j}^{T} \sum_{k \in V} [x_{jk}(\theta) - x_{kj}(\theta - \tau_{kj})] \, \mathrm{d}\theta \\ &= \sum_{j \in V} \sum_{k \in V} \int_{\alpha_j}^{T} [x_{jk}(\theta) - x_{kj}(\theta - \tau_{kj})] \, \mathrm{d}\theta \\ &= \sum_{j \in V} \sum_{k \in V} \int_{\alpha_j}^{\alpha_k - \tau_{jk}} [x_{jk}(\theta) - x_{kj}(\theta - \tau_{kj})] \, \mathrm{d}\theta \\ &\leq \sum_{jk \in E} \int_{\alpha_j}^{\alpha_k - \tau_{jk}} u_{jk} \, \mathrm{d}\theta \end{aligned}$$

■

To compute the maximum discrete-time dynamic flow in time $T-1$, Ford and Fulkerson add an infinite capacity arc from the sink to the source with transit time $-T$. Next, a static, minimum cost circulation f is computed in this network, with transit times as costs. If $|f| = 0$, then there is no source-sink path with transit time $\leq T-1$, so the maximum dynamic flow is also 0. Thus we assume in the remainder of this section that $|f| > 0$. The resulting flow in the original network is decomposed into chain flows γ_i such that $f = \sum_i \gamma_i$. This static flow is transformed into a dynamic flow of value $T|f| - \sum_{jk \in E} \tau_{jk} f_{jk}$ by sending as much flow as possible along each path at each time step. Ford and Fulkerson show that this is a maximum discrete-time dynamic flow by exhibiting a saturated cut in the time-expanded graph. Let α_j be the distance from the source to node j in the residual graph of f. Then the saturated cut C in the time-expanded graph is

$$C = \{j(\theta) | \theta \geq \alpha_j\}. \tag{4}$$

This cut has capacity

$$\sum_{jk \in E} \max\{0, \alpha_k - \tau_{jk} - \alpha_j\} u_{jk}. \tag{5}$$

Using complementary slackness of minimum cost flows [5] which states $\max\{0, \alpha_k - \tau_{jk} - \alpha_j\}(u_{jk} - f_{jk}) = 0$ for every edge, we can rewrite (5) as

$$\sum_{jk \in E} (\alpha_k - \alpha_j) f_{jk} - \sum_{jk \in E} \tau_{jk} f_{jk}.$$

Since $\alpha_s = 0$ and $\alpha_t = T$ (recall $|f| > 0$), we see the cut has capacity equal to $T|f| - \sum_{jk \in E} \tau_{jk} f_{jk}$.

Theorem 3.1 *The continuous maximum dynamic flow in time T in a dynamic network has value $T|f| - \sum_{jk \in E} \tau_{jk} f_{jk}$, where f is a a minimum cost circulation in the network with an additional sink-to-source arc with cost $-T$ and infinite capacity.*

Proof: Compute a minimum cost circulation in the extended network described in the theorem Decompose the flow into paths and send as much flow as possible along each path γ_i from time 0 until time $T - \tau(\gamma_i)$. The value of this flow is $\sum_i \int_0^{T-\tau(\gamma_i)} |\gamma_i| \mathrm{d}\theta$, which, using $|f| = \sum_i |\gamma_i|$, equals $T|f| - \sum_{jk \in E} \tau_{jk} f_{jk}$. This is the capacity of the generalized cut defined by the α_j's over time T. Since this argument holds for all real $T \geq 0$, the theorem is proved. ■

4 The Quickest Flow Problem

The flip problem of the maximum dynamic flow problem, the *quickest flow problem* asks for a flow of integral value v that completes in minimal time. In the discrete setting, this is solved easily by

binary search on the time bound T. Note that in this model, it may be possible to send more flow in the optimal time than is necessary to deplete the supply. The following theorem implies that binary search may also be used to solve the quickest continuous flow problem (QCFP).

Theorem 4.1 *Given a dynamic network with integral supply, the minimum time necessary to solve QCFP may be expressed as a rational number with denominator bounded by the size of a minimum cut in the network.*

Proof: First note that the maximum continuous dynamic flow is a continuous function of T. Thus, unlike the quickest discrete-time flow, the value of the maximum continuous dynamic flow in the optimal time of the quickest continuous flow is exactly equal to the supply. Fix T, and the maximum continuous flow x has value $|x| = T|f| - \sum_{jk \in E} \tau_{jk} f_{jk}$, for some static flow f. Given f, and the supply quantity v, we could then find that $T = (v + \sum_{jk \in E} \tau_{jk} f_{jk})/|f|$. By assumption, v and the transit times are integral, and thus, by well-known facts about min-cost flows, f is integral. Hence the optimal T for the quickest continuous flow is rational with denominator bounded by $|f|$, which is bounded by the minimum cut in the network. ■

5 Universally Maximum Dynamic Flows

A maximum dynamic flow in time T is an *earliest arrival flow* if it is also a maximum dynamic flow for every time $0 \leq \theta \leq T$. A *latest departure flow* is a flow with time horizon T that maximizes the amount of flow leaving the source in every interval $[\theta, T]$, $0 \leq \theta \leq T$. A *universally maximum dynamic flow* is an earliest arrival and latest departure flow. Such flows exist in both the discrete and continuous time models [6, 15]. Wilkinson [19] and Minieka [13] give algorithms to compute a universally maximum discrete dynamic flow based on Ford and Fulkerson's shortest augmenting path algorithm. In this section, we show that, as with the maximum dynamic flow problem, we can use this discrete-time algorithm to produce a universally maximum continuous dynamic flow. (Our discussion of the Wilkinson and Minieka algorithm follows the presentation of Hoppe [8]).

Let Γ be the set of chain flows induced by paths found by the shortest augmenting paths algorithm, and let $\Gamma_\theta = \{\gamma_i \in \Gamma : \tau(\gamma_i) < \theta\}$.

Theorem 5.1 $[\Gamma]^T$ *is a universally maximum continuous dynamic flow.*

Proof: The sum of the chain flows in any Γ_θ is a minimum cost flow, and it can be transformed to a minimum cost circulation by adding an arc from sink to source with length $-\theta$. By Theorem 3.1,

the value of the flow entering the sink for any interval $(0, \theta)$ is maximum for any θ, and hence the flow is an earliest arrival flow.

$[\Gamma]^T$ is also a latest departure flow, and hence a universally maximum flow. Consider the dynamic flow induced by chain flow $\gamma_i \in \Gamma$. For every $|\gamma_i|$ units of flow entering the sink at time θ, there are $|\gamma_i|$ units of flow leaving the source at time $T - \theta$. Summing over all chain flows in Γ, we see that the departure schedule is symmetric to the arrival schedule, and hence the dynamic flow is also a latest departure flow. ■

5.1 A "Snapshot" Algorithm

Unfortunately, the algorithm of Wilkinson and Minieka is not a polynomial time algorithm, since the shortest augmenting paths algorithm might use an exponential number of paths [20]. Hoppe and Tardos [9, 8] give two polynomial time algorithms related to this problem. The first is a "snapshot" algorithm that computes the value of the T-horizon, universally maximum discrete dynamic flow on a specified edge at a specified time step, denoted $f^*_{yz}(\theta, T)$. We extend this algorithm to compute the rate of flow entering a specified edge in a specified time interval $(\theta - 1, \theta]$, $\theta \in Z$.

Hoppe and Tardos observe that in the discrete time model, $f^*_{yz}(\theta, T)$ could be described by a subset of the set of chain flows used in the T-horizon, universally maximum discrete dynamic flow algorithm; that the sum of flows in this subset also describes a minimum cost flow; and that $f^*_{yz}(\theta, T)$ depends only on this minimum cost flow, and not on the specific chain decomposition. They then give a polynomial time algorithm to compute this minimum cost flow.

To extend this algorithm to work in the continuous setting, we first note that since all arcs in the original network have integral lengths, a minimum cost circulation in the network with return arc of length $-\theta \in \mathcal{Z}$ is also a minimum cost circulation in the same network with return arc of length $-(\theta - \epsilon)$, for any $\epsilon \in (0, 1)$.

Let x be the rate of flow determined by the universally maximum continuous dynamic flow algorithm discussed above. First consider the case when T is integral. Then, integral transit times imply that x changes only at integral time intervals. Thus $f^*_{yz}(\theta, T)$ computed by the Hoppe-Tardos algorithm can be interpreted, in the continuous-time model, as the rate of flow entering an edge over any time interval of form $(\theta, \theta + 1]$, $\theta \in Z$, $\theta \leq T - 1$.

If T is not integral, x can change only at times of the form θ and $\theta + \mathrm{fr}(T)$ for $\theta \in \mathcal{Z}$, where $\mathrm{fr}(T)$ is the fractional part of T. The flow entering an arc (y, z) changes either when the dynamic flow induced by a chain flow starts passing through y or stops passing through y. Integral transit times

imply that the former happens only at times θ, and the latter only at times $\theta + \text{fr}(T)$. Thus the flow rate is constant over any interval of form $(\theta, \theta + \text{fr}(T))$ or $(\theta + \text{fr}(T), \theta + 1)$. The chain flows that send flow through arc (y, z) during the interval $(\theta, \theta + \text{fr}(T))$ but not in the interval $(\theta + \text{fr}(T), \theta + 1)$, are the flows that have distance $\lfloor T \rfloor - \theta$ to travel from y to t. That is, we can compute the flow rate over interval $(\theta, \theta + \text{fr}(T))$ by computing $f^*_{yz}(\theta, \lceil T \rceil)$. The flows that entering arc (y, z) beyond time $\theta + \text{fr}(T)$, have distance less than $\lfloor T \rfloor - \theta$ to travel to t, and thus we can compute the flow rate over interval $(\theta + \text{fr}(T), \theta + 1)$ by computing $f^*_{yz}(\theta, \lfloor T \rfloor)$. This proves the following extension to Hoppe's result [8].

Theorem 5.2 *Let x be the universally maximum continuous dynamic flow as computed via the shortest augmenting paths algorithm. The rate of flow entering any edge yz in time interval $(\theta, \theta + \text{fr}(T))$ or time interval $(\theta + \text{fr}(T), \theta + 1)$ for $\theta \in \mathcal{Z}$, is constant and can be computed in $O(\log(nU)\text{MCF})$ time, where U is the largest flow rate in the dynamic network.*

5.2 Approximate Universally Maximum Dynamic Flow

Hoppe and Tardos [9, 8] describe an approximation algorithm that returns a discrete-time flow of value within $(1 - \epsilon)$ of the universally maximum discrete dynamic flow over any time interval $[0, \theta]$, $\theta \in \{0, 1, \ldots, T\}$. The algorithm combines capacity scaling with the shortest augmenting paths algorithm. The initial phase uses the dynamic network as given and mimics the shortest augmenting paths algorithm until the static flow value exceeds $\frac{m}{\epsilon}$, at which point residual capacities are rounded down to the nearest even number. With each augmentation, the flow along the path is added to a set of chain flows Γ that will induce the final dynamic flow. Each successive phase uses the residual network passed by the previous phase with capacities evenly divisible by Δ, augments flow along shortest paths until the static flow value of the new augmentations exceeds $\frac{\Delta m}{\epsilon}$, adds the augmenting chain flows to Γ, and then rounds down residual capacities so that they are evenly divisible by 2Δ. This continues until there is no augmenting path of length less than or equal to T.

Remark: This algorithm has a special property that we will use in the proof of Theorem 5.3. Consider the change to Γ if we increase T by one. Notice that no chain flows will leave Γ, and, all those that enter Γ have length equal to $T + 1$. This is a property of the shortest augmenting paths algorithm that is maintained here.

Theorem 5.3 *The continuous transformation of the Hoppe-Tardos $(1-\epsilon)$-approximate universally maximum discrete dynamic flow is a $(1-\epsilon)$- approximate universally maximum continuous dynamic flow.*

Proof: Since the natural transformation of the universally maximum discrete dynamic flow is a universally maximum continuous dynamic flow, the natural transformation of the approximate discrete dynamic flow yields a continuous dynamic flow of value within $(1-\epsilon)$ of the universally maximum continuous dynamic flow over any time interval $[0,\theta]$, $\theta \in \{0,1,\ldots,\lfloor T \rfloor\}$. To see that this flow is within $(1-\epsilon)$ of the universally maximum dynamic flow for any time interval $[0,\theta], 0 \leq \theta \leq T$, we again consider two cases: T integral, and T real.

If T is integral, then we observed in Section 5.1 that the flow rate is constant over any interval $(\theta,\theta+1]$, $\theta \in \mathcal{Z}$. Let u and v be the respective values of the approximate and optimal dynamic flows at time $\theta \in \mathcal{Z}$. Note that $u \geq (1-\epsilon)v$. At time $\theta+1$, these values will be $u+c$ and $v+d$, with $u+c \geq (1-\epsilon)(v+d)$. Since both approximate and optimal flows have constant rates of flow in the interval $(\theta,\theta+1]$, a weighted average of these two inequalities shows that the value of the approximate flow at time $\theta+\delta$, $0<\delta<1$ is still within $1-\epsilon$ of the optimal.

We now argue that the validity of the algorithm for integral T implies validity for all T. Our previous remark implies that the set of chain flows used in the optimal and approximate flows in time T are the same as the set of chain flows used in the corresponding $\lceil T \rceil$-horizon flows. Thus the amount of flow reaching the sink by time $\theta \leq T$ in the T-horizon flows is the same as the amount of flow reaching the sink by the same time in the $\lceil T \rceil$-horizon flows, and thus the value of the approximate T-horizon flow at time $\theta \leq T$ is within $1-\epsilon$ of the optimal, for any $T \geq 0$. ■

6 Lexicographically Maximum Dynamic Flows

A *lexicographically maximum dynamic flow* is a feasible dynamic flow that, given an ordering of the terminals $S = \{s_0, s_1, \ldots, s_{k-1}\}$, and a time bound T, maximizes the amount of flow leaving each terminal in the given order. The set of terminals may contain both sources and sinks: maximizing the amount of flow leaving a terminal is equivalent to minimizing the amount of flow entering the terminal. Minieka [13] and Megiddo [12] both noted that this objective is also equivalent to simultaneously maximizing the amount of flow leaving every high priority subset $S_i := \{s_0, s_1, \ldots, s_{i-1}\}$, $0 \leq i \leq k$.

If T is integral, then the natural transformation of a lexicographically maximum discrete dynamic flow in time $T-1$ is a lexicographically maximum continuous dynamic flow: the flow leaving every high priority subset S_i, $0 \leq i \leq k$ is a maximum continuous dynamic flow by the observation of Minieka and Megiddo, and Theorem 3.1.

Hoppe and Tardos [10, 8] describe an algorithm that computes a lexicographically maximum dy-

namic flow in the discrete-time setting via k minimum cost flows. They iteratively compute a discrete dynamic flow maximizing the amount of flow leaving every high priority subset of terminals S_i, by computing a minimum cost circulation in an extended network. This circulation is decomposed into chain flows, which are added to the set of chain flows that induce the final dynamic flow.

We can extend this algorithm to produce a continuous dynamic flow via the natural extension of the discrete flows induced by chain flows. We must argue that the resulting flow maximizes the amount leaving every high priority subset S_i, $0 \leq i \leq k$ when T is not integral. First note that the algorithm will return the same chain flows for time horizons T and $\lceil T \rceil$. Next note that if T is a rational number with denominator bounded by d, then the algorithm will return the same chain flows for time horizon T as it would on the network modified by multiplying T and all transit times by d. Since the dynamic flow induced in the latter case is a lexicographically maximum continuous dynamic flow, the same is true in the former: the only difference in the two cases is the time scale. By taking limits of the lexicographically maximum continuous dynamic flows for rational time horizons, we see that the same algorithm will return a lexicographically maximum continuous dynamic flow for all real $T \geq 0$.

7 The Dynamic Transshipment Problem

With the *dynamic transshipment problem*, we have a dynamic network, an integral vector v of supplies at the terminals, and a time bound T. The objective is to find a feasible dynamic flow that zeroes all supplies within time T. Hoppe and Tardos [10, 8] solve the discrete-time version of this problem by reducing it, in polynomial time, to the lexicographically maximum dynamic flow problem. Their reduction is somewhat complicated, and relies on an oracle to test feasibility. The goal of this section is to show that it is still valid for continuous dynamic flows.

Hajek and Ogier [7] describe an algorithm that computes a quickest continuous dynamic transshipment in the special case that all transit times are zero and there is only one sink. While their algorithm does not appear to extend to handle non-zero transit times, or more than one sink (without restricting the number of sources), it is much simpler than the more general algorithms mentioned here, running in time $O(|V|^4)$.

7.1 Dynamic Transshipment Feasibility

For the static transshipment problems, Ford and Fulkerson provide a criterion for feasibility: a supply vector v is feasible if and only if, for every subset of terminals $A \subseteq S$, the maximum flow out of A is at least $v(A)$. For the discrete dynamic transshipment problem, Klinz [11] observed the corresponding relation. For a fixed time T and a subset A of terminals, define $o(A)$ to be the value of the maximum discrete dynamic flow that can be sent from sources in A to sinks in $S - A$ in time T. We can compute $o(A)$ via one maximum dynamic flow by adding a super source connected to the sources in A by infinite capacity zero transit time arcs and a super sink similarly connected to the sinks in $S - A$. For feasibility of the dynamic transshipment problem we must have $v(A) \leq o(A)$ for every $A \subseteq S$. Klinz noted that by considering the Ford-Fulkerson criterion in the time expanded graph, this condition was also sufficient for feasibility. In the continuous model, there is no time-expanded graph to consider; nevertheless, the following theorem shows that the analogous condition is sufficient for feasibility of the continuous dynamic transshipment problem. Let $\bar{o}$ be the maximum continuous dynamic flow function for fixed T.

Theorem 7.1 *The continuous dynamic transshipment problem is feasible if and only if* $v(A) \leq \bar{o}(A)$ *for every subset* $A \subseteq S$.

Proof: First consider T integral. The discussion in Section 3 implies that $v(A) \leq \bar{o}(A)$ for all $A \subseteq S$ if and only if $v(A) \leq o(A)$ for all $A \subseteq S$ in discrete time $T - 1$. This second condition implies that the discrete time problem is feasible in time $T - 1$, which, by transforming the feasible discrete flow to a continuous flow, implies that the continuous problem is feasible in time T.

By Theorem 4.1, $v(A) = \bar{o}(A)$ only if T is a rational number with denominator bounded by the maximum size of a static flow in the network, $|f|$. If T is such a number, then we can consider the same network with transit times and time bound multiplied by a constant $d \leq |f|$, and hence the above argument applies. If T is not such a number, since $\bar{o}$ is a continuous and nondecreasing function of T, we can reduce T to the closest such number while maintaining $v(A) \leq \bar{o}(A)$ for all $A \subseteq S$. The new problem is then feasible, so the original problem must be also. ■

The above theorem implies that a simple exhaustive search, testing all subsets $A \subseteq S$ will prove feasibility or find a violated set. Hoppe and Tardos [10, 8] describe a strongy-polynomial time algorithm to test feasibility that relies on submodular function minimization.

7.2 Dynamic Transshipment Algorithm

A subset $A \subseteq S$ satisfing $v(A) = o(A)$ is called a *tight* subset. If the terminals can be ordered $\{s_0, s_1, \ldots, s_{k-1}\}$ so that each subset S_i is tight, then a lexicographically maximum flow is also a dynamic transshipment. Hoppe and Tardos [10, 8] noted this and used this fact to motivate their algorithm to find a dynamic transshipment. They successively add terminals to the network with parametrized capacities or transit times to create a chain of nested tight subsets of length equal to the number of terminals, and then find a lexicographically maximum flow.

Since both the lexicographically maximum dynamic flow algorithm and the criteria for feasibility hold for continuous dynamic flows, the natural transformation of the discrete dynamic transshipment will provide a continuous dynamic transshipment, if T is integral. If T is a rational number with denominator d, then we can solve the dynamic transshipment problem in the network with T and transit times multiplied by d. In doing so, we may introduce arcs with capacities and transit times that are not evenly divisible by d, and hence will be fractional when we translate our solution back to the original network, but these fractions will be integer multiples of $1/d$.

Finally, we can bound the size of denominator d: if T is irrational, or has large denominator, then the argument in the following section shows that either the problem is infeasible, or we can reduce T and the problem remains feasible.

7.3 The Quickest Transshipment

Given a dynamic network, and a supply vector v, the *quickest transshipment* is a feasible dynamic transshipment that satisfies all supplies in the minimum time T. Since the dynamic transshipment problem is feasible as long as $v(A) \leq \bar{o}(A)$ for all $A \subseteq S$, and $\bar{o}(A)$ for fixed A is a continuous and nondecreasing function of T, at the minimum time T there must be some A that is tight. Theorem 4.1 then implies that T must be a rational number with bounded denominator and thus, using the dynamic transshipment algorithm, we can solve the quickest transshipment problem via binary search for the minimal T.

Acknowledgements

We would like to thank Bruce Hoppe for sharing his insights on and enthusiasm for dynamic flows with us.

References

[1] E. J. Anderson and P. Nash. *Linear Programming in Infinite-Dimensional Spaces.* John Wiley & Sons, 1987.

[2] E. J. Anderson, P. Nash, and A. B. Philpott. A class of continuous network flow problems. *Mathematics of Operations Research*, 7:501–14, 1982.

[3] E. J. Anderson and A. B. Philpott. A continuous-time network simplex algorithm. *Networks*, 19:395–425, 1989.

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

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

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

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

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

[9] 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.

[10] B. Hoppe and É. Tardos. The quickest transshipment problem. In *Proc. of 6th Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 512–521, 1995.

[11] B. Klinz. Personal communication.

[12] N. Megiddo. Optimal flows in networks with multiple sources and sinks. *Mathematical Programming*, 7:97–107, 1979.

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

[14] A. Orda and R. Rom. On continuous network flows. *Operations Research Letters*, 17:27–36, 1995.

[15] A. B. Philpott. Continuous-time flows in networks. *Mathematics of Operations Research*, 15(4):640–661, November 1990.

[16] 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.

[17] M. C. Pullan. A study of general dynamic network programs with arc time-delays. *SIAM Journal on Optimization*. To appear.

[18] M. C. Pullan. An algorithm for a class of continuous linear programs. *SIAM Journal fo Control and Optimization*, 31(6):1558–1577, November 1993.

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

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