Optimal Rounding of Instantaneous Fractional Flows Over Time

by
Lisa K. Fleischer
James B. Orlin

OR 340-99 August 1999

Optimal Rounding of Instantaneous Fractional Flows Over Time

Lisa K. Fleischer[*] James B. Orlin[†]

August 5, 1999

Abstract

A transshipment problem with demands that exceed network capacity can be solved by sending flow in several waves. How can this be done in the minimum number, T, of waves, and at minimum cost, if costs are piece-wise linear convex functions of the flow? In this paper, we show that this problem can be solved using $\min\{m, \log T, \frac{\log(m\Gamma U)}{1+\log(m\Gamma U)-\log(mU)}\}$ maximum flow computations and one minimum (convex) cost flow computation. Here m is the number of arcs, Γ is the maximum supply or demand, and U is the maximum capacity. When there is only one sink, this problem can be solved in the same asymptotic time as one minimum (convex) cost flow computation. This improves upon the recent algorithm in [5] which solves the quickest transshipment problem (the above mentioned problem without costs) on k terminals using $k \log T$ maximum flow computations and k minimum cost flow computations. Our solutions start with a stationary fractional flow, as described in [5], and use rounding to transform this into an integral flow. The rounding procedure takes $O(n)$ time.

[*]email: lisa@ieor.columbia.edu. Department of Industrial Engineering and Operations Research, Columbia University.

[†]email: jorlin@mit.edu. Sloan School of Management, Massachusetts Institute of Technology. Supported by ONR through grant N00014-96-1-0051.

1 Introduction

Network flow theory deals with several fundamental problems including the shortest path problem, the maximum flow problem, the assignment problem, the minimum cost flow problem, and the multicommodity flow problem. A significant body of literature is devoted to these five problems. (See [1], for example.) In the 1960s, Ford and Fulkerson [8] introduced network flows over time to include time in the network model. Since then, network flows over time have been widely used in a variety of applications to transportation, financial planning, manufacturing, and logistics. See, for example, the surveys by Aronson [4] and Powell, et al. [18]. Despite the significance of flows over times to a wide range of problems, the class of network flows over time has not received nearly as much attention as the more classical network flow problems. Much of the work on problems of flow over time reduces the problem to a classical flow problem by using an exponentially sized, time-expanded graph [4, 18].

In this paper, we discuss a special case of flows over time: we assume all transit times are zero. This special case has been considered in [2, 5, 6, 12, 15, 22, 14], among others. Flows over time with zero transit times capture some time-related issues: they can be used to model instances when network capacities restrict the quantity of flow that can be sent at any one time. Solving these problems efficiently may help in finding a more efficient exact or approximate algorithm to solve harder problems involving flows over time with transit times or multicommodity demands.

1.1 The Model

A network $\mathcal{N} = (V, E, u)$ has vertex set V of cardinality n, and arc set E of cardinality m. Arc e has capacity u_e and $U := \max_e u_e$. Node i has demand γ_i, with $\Gamma := \max_i |\gamma_i|$. Nodes with non-zero demands are called *terminals.* Terminals with negative demand are called *sources*; those with positive demand are called *sinks.* The sum of supplies over all terminals equals zero. We use k to denote the number of terminals.

A *(static) transshipment problem* is defined on an arbitrary network with edge capacity vector u and with node demand vector γ. The objective is to find a flow $f : E \rightarrow \mathbf{R}_{\geq 0}$ satisfying all demands. We define $f_{ji} = -f_{ij}$ for $ij \in E$. In other words we seek f such that $f_e \leq u_e$ for all edges e, and $\sum_j f_{ij} = \gamma_i$ for all vertices i.

A *transshipment over time* is a time-dependent flow $f(t)$ through a with non-zero supply and demand vector γ associated with V, and a time bound T. Below we give a discrete-time model for transshipment over time. However, our algorithm solves both the discrete time and continuous time problems, since, as noted in [5, 7], an optimal solution to the discrete time problem can be easily transformed into an optimal solution for the continuous-time problem. A transshipment over time obeys edge capacity constraints $f(t) \leq u$ for all $t \in \{1, 2, \dots, T\}$, flow conservation constraints $\sum_{t=1}^{r} \sum_j f_{ij}(t) \leq \gamma_i$ for all $r \in \{1, 2, \dots, T\}$ and all $i \in V$, and zeroes all supplies and demands by time T: $\sum_{t=1}^{T} \sum_{j \in V} f_{ij}(t) = \gamma_i$ for all $i \in V$. The *quickest transshipment problem* is a transshipment over time that zeroes all supplies and demands in the minimum possible time. The quickest transshipment defines the minimum time T needed for the existence of a feasible transshipment. Solving a quickest transshipment problem with fixed supplies is useful for clearing a network after a communication breakdown.

To further generalize this model, we can include costs. Let c_e be a cost function associated with edge e. In the problem discussed in this paper, c_e is a function of the flow on the edge at a particular moment of time, but c_e is not a function of time. The cost of a transshipment f which completes by time T is $\sum_{t=1}^{T} \sum_{e \in E} c_e(f_e(t))$. A *minimum cost transshipment over time* is a quickest transshipment that has minimum cost. In this paper, we present an algorithm to solve this problem that first finds the minimum time necessary for feasibility. Given this time bound, the algorithm then finds a minimum cost solution. The converse problem, finding a quickest, minimum cost transshipment, can be solved by first determining the minimum cost solution by assuming infinite capacities, and then searching for the minimum time bound that has a solution with this cost.

In this paper, we consider convex cost functions that are piecewise linear between successive integers. Since we want an integral solution, the assumption that the cost function is piecewise linear between integers is natural. It is useful to make this assumption, since now there always exists an integer solution that matches the cost of the cheapest fractional solution.

A flow over time is called *stationary* if it is the same in each time period. For each of the above problems, we can define a corresponding stationary version of the problem where in addition to the above requirements, we insist the flow must be stationary. Fleischer [5] shows that every feasible transshipment over time problem has a stationary solution. The stationary problem can be solved by multiplying all arc capacities by T and solving the resulting static transshipment, which is a maximum flow problem. Since all original supplies, demands, and capacities are integers, the solution is an integral flow. The stationary flow is then obtained by dividing this flow by T. This flow has the property that the rate of flow entering a terminal node i is $\frac{\gamma_i}{T}$. A feasible stationary flow that completes in the minimum possible time can then be found using either discrete Newton's method [20] or binary search to find the minimum feasible time T. The following theorem is a result from [5].

Theorem 1.1 *A quickest stationary flow can be found in* $\min\{\frac{\log(m\Gamma U)}{1+\log(m\Gamma U)-\log(mU)}, \log T, m\}$ *maximum flow computations.*

The maximum flow algorithm with the lowest complexity with respect to m and n is by Goldberg and Rao, and runs in $O(m\min\{m^{1/2}, n^{2/3}\}\log \frac{n^2}{m}\log U)$ time [10].

1.2 Our Contribution

Many of the applications of flows over time need integral solutions: when flows are of big objects, like airplanes, or train engines, the demands are often small, so fractional solutions are not very useful. By rounding carefully, we show how a stationary flow f can be transformed into an integral flow over time maintaining the same flow balance constraints, and such that this new flow g satisfies $c(g) \leq c(f)$. The rounding technique is simple and operates in two stages on the support graph of edges with fractional flow. First cycles in this graph are canceled, and then the edges in each tree are rounded in linear time, starting from a leaf and proceeding in a depth or breadth first manner. We think of the interval of time $[0,T)$ as a cycle on a clock, and define a *circular interval* as an interval modulo T. If the stationary flow in arc e has value f_e per time unit, then the the integral solution we find has the property that the flow in arc e has value $\lfloor f_e \rfloor$ in a circular interval, and value $\lceil f_e \rceil$ for the remainder of the time.

Our rounding technique takes only linear time. Since finding a quickest stationary flow takes more time, this implies that a quickest transshipment can be found in the same asymptotic time as finding the minimum feasible time T. This is the fastest known algorithm to solve this problem, and improves upon [5] by almost a factor of k. If there is just one sink, then it is known [5] that the minimum feasible time T can be found in strongly polynomial time using one call of the parametric maximum flow algorithm of Gallo, Grigoriadis, and Tarjan [9]. Thus, our rounding technique implies an improvement over the previous best algorithm by a factor of k [5] for this special case as well. The Gallo, et al. algorithm is based on the Goldberg and Tarjan [11] maximum flow algorithm and runs in the same asymptotic time. In addition, since there always exists a minimum (convex) cost transshipment over time that is stationary (Section 2), our rounding technique yields a fast algorithm for computing an integral minimum (convex) cost transshipment over time.

The model discussed here solves the problem when the demands and the supplies are fixed at the beginning of the problem, and there is no external change to these values. However, everyday usage often involves continuous streams of traffic. In Section 4, we show that all of the algorithms presented here allow for constant streams of external flow into or out of any node in the network.

The model as presented in this paper can also handle finite buffer capacity at the nodes of the network. Suppose storage capacity at node i is a_i. Then we have the additional constraint (in the discrete-time model) that $\sum_{t=1}^{r}\sum_j f_{ji}(t) \leq a_i + \gamma_i$, for all $r \in \{1,\ldots,T\}$ and all $i \in V$. A stationary flow never increases the absolute value of supply or demand at any node, and our rounding technique maintains this property. Together with Theorem 2.1 this implies that our solution solves the problem for all $a \geq 0$.

1.3 Related Work

The fastest previous algorithm for solving integral, quickest transshipment problem with zero transit times uses $O(k \log \Gamma)$ maximum flow computations and k minimum cost flow computations [5]. Previously, Hoppe and Tardos [13] describe the only known polynomial time algorithm to solve the quickest transshipment problem with general transit times. However, their algorithm repeatedly calls the ellipsoid method as a subroutine, so it is theoretically slow, and also not practical. Their work improves on the theoretical complexity of previous approaches however. Traditional approaches to solving the transshipment over time consider the discrete time model and make use of the time expanded network — a network containing T copies of the original network [4, 18]. For fine discretizations, this network is very large and not practical to work with.

A slightly different type of convex cost flow over time problem is discussed by Orlin [16]. He looks at infinite horizon convex cost flows over time and describes an algorithm that finds the minimum average convex cost circulation that repeats indefinitely in a network with general transit times.

Our paper and the above-mentioned results all look at discrete time problems. Flows over time have also been considered in the continuous time setting [2, 3, 17, 19]. Most of the work in this area has examined networks with time-varying edge capacities, storage capacities, or costs. The focus of this research is on proving the existence of optimal solutions for classes of time-varying functions, and proving the convergence of algorithms that eventually find solutions. These algorithms are not polynomial and implementations do not seem able to handle problems with more than a few nodes. For the case in which capacity functions are constant, Fleischer and Tardos [7] extend the polynomial, discrete-time transshipment over time algorithm in [13] to work in the continuous-time setting.

There are some additional continuous time problems that have polynomial time algorithms. A *universally quickest transshipment* is a quickest transshipment that simultaneously minimizes the amount of excess left in the network at every moment of time. An optimal solution may require fractional flow sent over fractional intervals of time. There is a two-source, two-sink example for which a universally quickest transshipment does not exist [5]. Hajek and Ogier [12] describe an algorithm that solves the universally quickest transshipment problem in networks with multiple sources, a single sink, and zero transit times. Their algorithm uses $O(n)$ maximum flow computations. Fleischer [5] describes how this problem can be solved in $O(mn \log(n^2/m))$ time — using the parametric maximum flow algorithm of Gallo, et al., in conjunction with maximum flows. This algorithm extends easily to the minimum (convex) cost problem by substituting minimum (convex) cost flow computations for maximum flow computations.

Another continuous time problem that has a polynomial time algorithm is the universally quickest flow problem with piecewise constant capacity functions. This is the universally quickest transshipment problem with just one source and sink. However, now capacities are allowed to vary over time. For this problem, Ogier [15] describes an algorithm that uses nl maximum flow computations in a graph on nl nodes and $(m+n)l$ arcs. Fleischer [6] shows how this can be improved to run in the same asymptotic time as one preflow-push maximum flow computation in a graph with nl nodes and $(n+m)l$ arcs. A preflow-push maximum flow algorithm runs in $O(mn\log(n^2/m))$ time [11] on a graph with n nodes and m arcs.

2 Minimum cost flows over time

As stated in Section 1.1, any feasible transshipment has a stationary solution. In this section, we show that there is always a minimum cost solution that is stationary.

Theorem 2.1 *There is an optimal stationary solution to any feasible minimum (convex) cost transshipment over time, even with non-zero buffer capacity.*

Proof: Let f be any minimum convex cost transshipment over time. Let f^T be defined by $f^T_e := \sum_{t=1}^{T} f_e(t)$, the total flow on edge e. By convexity of the cost function, the cost of this flow is at least as large as the cost of the flow $g = f^T/T$, which is a feasible, stationary transshipment over time. Note that this argument is independent of $\sum_{t=1}^{r} \sum_j f_{ji}(t)$, the amount of buffer capacity used at node i at time r. ∎

A stationary, minimum (convex) cost transshipment over time can be obtained by computing a minimum cost static transshipment in the network with capacities multiplied by T and with the cost function $\bar{c}$ defined by $\bar{c}(f) = c(f/T) \times T$. Note that $\bar{c}$ is convex when c is, and that $\bar{c}$ is minimized at f if and only if c is minimized at f/T. Thus the resulting flow f, when divided by T, gives a feasible flow over time whose cost equals $\bar{c}(f)$ and is minimum among all feasible transshipments over time.

3 Rounding a stationary flow

In this section, we explain how to transform a stationary transshipment into an integral transshipment over time that has the same cost. We do this by considering the graph G' consisting of just the arcs with fractional flow. The algorithm proceeds in two stages: It first cancels cycles in this graph. Then, one arc at a time, it determines an interval in which the flow on the arc will be rounded up to the nearest integer. For the remaining time, the flow on this arc is rounded down to the nearest integer. This is done in a way to ensure that the total flow sent along the arc in the interval $[0, T)$ remains the same. This then implies that the cost of sending the flow remains the same.

3.1 Canceling fractional flow cycles

If the graph consisting of the edges with fractional flow contains cycles, then we can cancel these cycles by sending flow around the cycle until the flow on at least one edge in the cycle becomes integral. Iterating, we eventually obtain a stationary flow such that the graph consisting of edges with fractional flow is acyclic. Such a solution is a basic feasible solution for the stationary problem.

Let f denote the optimal stationary flow. Let $\lfloor f \rfloor$ denote f rounded down and let $\lceil f \rceil$ denote f rounded up. Consider the stationary transshipment problem with the additional constraints $\lfloor f \rfloor \leq f \leq \lceil f \rceil$. Since we assume that costs are linear between successive integers, this modified problem is a minimum cost flow problem, even if the original problem has convex costs. We can convert f to a basic feasible flow for this problem by canceling fractional flow cycles. We can cancel a cycle by sending flow in either direction. Optimality conditions for minimum cost flow imply that if a cycle exists, the cost of the cycle (obtained by summing the costs on all edges of the cycle) must be 0. Thus we can cancel flow in either direction and maintain the cost of the initial solution. It takes $O(n)$ time to find a (undirected) cycle in G' using depth first search, and at most m cycles are cancelled since each cancellation removes at least one edge from the set of edges with fractional flow. Thus all cycles are cancelled in $O(mn)$ time.

For the minimum cost transshipment over time problem, this step is not a bottleneck, since computing a minimum cost flow takes more time. However, for the quickest transshipment, the maximum flow algorithm of Goldberg and Rao [10] has a run time that does not dominate this one. Hence, we note that it is possible to cancel cycles in $O(m \log n)$ time, using the dynamic tree data structure of Sleator and Tarjan [21]. The details of this procedure are described in [21].

3.2 Rounding fractional flow

In this section, we assume that G', the graph of fractional flows, is a forest. The rounding algorithm considers each tree in this forest independently, roots each tree at an arbitrary leaf, and rounds the flow on the edges in a tree starting from the root and spreading throughout the tree in a breadth first manner. For each arc, the rounding procedure determines an interval in which the flow on the arc will be rounded up to the nearest integer. For the remaining time, the flow on this arc is rounded down to the nearest integer. The algorithm does this while ensuring that the total flow sent along the arc in the interval $[0, T)$ remains the same. This then implies that the cost of sending the flow remains the same.

We assume a tree F of fractional flows is stored in a simple data structure so that the tree has a root, which is assigned to be a node with degree one. Thus the root node has a single child. The arcs are oriented according to the direction that the fractional flow travels down the arc. Without loss of generality, we assume all arcs in F are directed away from the root: Each arc with fractional flow α directed toward the root is replaced by two arcs, one with unit flow directed towards the root, and one with fractional flow $1 - \alpha$ directed away from the root. Only the new arc with fractional flow is kept in the tree.

For each arc (i, j) of the tree, we define the *predecessor* of j to be i and denote this by $\text{pred}(j) = i$. Suppose that nodes 2, 3, 4, and 5 are the children of node 1. We refer to node 2 as the *left child* of node 1, and refer to node 3 as the *right sibling* of node 2. Similarly node 4 (respectively, node 5) is the right sibling of node 3 (respectively, node 4).

Recall that we are considering all addition and intervals to be modulo T, so that if $a > b$, then $[a, b)$ denotes the circular interval $[a, T) \cup [0, b)$.

Let $x(u, v)$ denote the fractional part of the flow in arc (u, v). Let $d(v)$ denote the fractional part of the rate of increase of supply at node v. That is, $d(v)$ is the fractional part of $\frac{\gamma_v}{T}$. Rounding the flows entering and leaving v implicitly has the effect of rounding the rate of increase of supply at v. We make this explicit to provide a cleaner explanation, by introducing a new node v' with arc (v, v'), and set $x(v, v') = d(v)$. The transformed problem has fractional rates of increase of supply d' defined by $d'(v) = 0$ and $d'(v') = d(v)$. Rounding up the flow on arc (v, v') in the transformed problem corresponds to rounding up the rate of increase of supply at node v in the original problem. With this transformation, all internal nodes v of F appear as non-terminal nodes. Thus for all internal nodes of F, the fractional part of the flow into v equals the fractional part of the sum of flows out of v.

Let $a(u, v)$ denote the "start time" for the round up of arc (u, v). It is the first period in the cyclic interval in which flow in (u, v) is rounded up. Let $b(u, v)$ denote the start time for the round down of arc (u, v). Given $a(u, v)$, we define $b(u, v) := a(u, v) + T \times x(u, v) \mod T$. The flow on arc (u, v) is rounded up in the circular interval $[a(u, v), b(u, v))$ and rounded down in the circular interval $[b(u, v), a(u, v))$.

We determine $a(u, v)$ as follows, and discuss a concrete example below and in Figure 1. If u is a node in F with predecessor node w, and v is the left child of u, then we set $a(u, v) = a(w, u)$. If

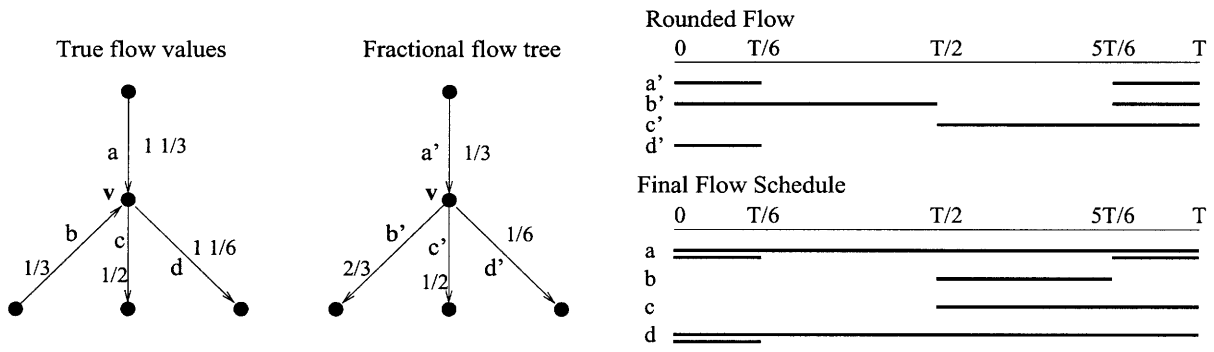

Figure 1: An example of how flow is rounded on a subtree and the discussion in the proof of Theorem 3.1. This subtree has the interval $[5T/6, T/6)$ passed to it from higher in the tree. The rounded flow (top set of intervals) covers the whole interval $[0, T)$ at least once, and the key interval $[5T/6, T/6)$ exactly one more time.

w is a node with predecessor u, and v is the right sibling of node w, then we set $a(u, v) = b(u, w)$. Assuming that the stationary flow comes from a basic feasible solution to a static flow problem with integer data, then $x(u, v) \times T$ is integral, and hence the rounded intervals have integer length.

For example, consider the flow in Figure 1. At every unit of time, fractional flow of value 1/3 is entering node v and flows of 2/3, 1/2, and 1/6 are leaving node v. Suppose we have previously rounded the flow in the incoming arc up to the nearest integer in the interval $[5T/6, T/6)$. Then we round the outgoing flows as follows: Fractional flow of 2/3 is rounded to one from $t = 5T/6$ to $t = T$ and from $t = 0$ to $t = T/2$, and is rounded down to 0 for the remaining interval $[T/2, 5T/6)$. Fractional flow of 1/2 is rounded up to one in the interval $[T/2, T)$, and is rounded down to zero in the remaining interval $[0, T/2)$. Fractional flow of 1/6 is rounded up to one in the interval $[0, T/6)$. Since the sum of the fractional flows leaving v is one more than the amount of fractional flow entering v, there must be an additional unit of integral flow entering v at every time step. This additional unit of flow entering v is used to balance the rounded flows now leaving v.

Each arc (u, v) gets rounded up $x(u, v)$ of the time and rounded down $1 - x(u, v)$ of the time, thus leading to a net balance of flow out of node u. Also, the upper and lower bounds on flows are automatically satisfied. So the key issue is whether there is balance in each time unit.

Theorem 3.1 *The rounding algorithm transforms a basic, feasible stationary transshipment into a feasible integral transshipment over time in $O(n)$ time.*

Proof: Let v be a non-terminal node of F with r children $\{u_1, \ldots, u_r\}$ and with $\text{pred}(v) = w$. At every unit of time, the stationary flow f satisfies the following: the fractional part of flow into v

equals the fractional part of the sum of flows leaving v. More precisely,

$$\sum_{i=1}^{r} x(v,u_i) \times T = x(w,v) \times T + \kappa T, \quad \kappa \in \mathbf{Z}^+$$

This means that the amount of integral flow arriving at node v before the rounding is κT more than the sum of integral flows leaving node v. Thus at any moment of time, we can round up the flow on at least κ of the fractional edges leaving node v and supply the flow on these edges with the extra integer flow arriving at v. In fact, for our rounding scheme to maintain flow conservation without using node storage, our solution needs to round up the flow up on κ arcs leaving v at all times, and on an additional arc leaving v for the interval of time that we rounded up the flow on the arc entering v. This is exactly what our rounding scheme accomplishes. Starting at the moment of time when we round up the flow on the arc entering v, we schedule rounded up flow contiguously, tracing the $[0,T)$ interval κ times and the $[a(w,v), b(w,v))$ interval $\kappa+1$ times by starting at $a(w,v)$, scheduling unit flow of total value $\sum_{i=1}^{r} T \times x(v,u_i)$ and ending at $b(w,v)$. (Note that $a(w,v) + \sum_{i=1}^{r} T \times x(v,u_i) = b(w,v) \bmod T$.)

If v is a source or sink, then v is a leaf (or the root) of F, by construction, and the only rounding we do is on one arc entering (or leaving) v. In this case, the rounding amounts to rounding the rate of increase of supply at v, and we needn't worry about balance at every time step.

The rounding procedure spends a constant amount of time considering each arc in each fractional tree. Hence the total time to perform the rounding is bounded by the number of nodes in the graph. ∎

4 Incoming and outgoing traffic

While the solving a transshipment problem with fixed supplies and demands is useful for clearing a network after a communication breakdown, everyday usage more often involves continuous streams of traffic. In this section, we show that the algorithms presented in this paper, like the algorithm in [5, 12], allow for constant streams of flow into or out of any node in the network.

The algorithm described in the preceding sections first finds a fractional flow that is constant over time, and then rounds the flow. To find the fractional flow, we sum up capacities per unit time over the length of our time horizon. We treat these constant rate external flows in the same manner, multiplying the value of the rate of incoming flow by T, and treating these as additional supplies and demands. Any solution to the resulting transshipment over time problem will have constant rates of flow from these sources and sinks, of additional value equal to the rate of external flow to these nodes. Since we assume all rates of flow are integral, this only contributes integral flows to the resulting flow over time, and hence is not affected by the rounding.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, Englewood Cliffs, NJ, 1993.

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

[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. Fleischer. Faster algorithms for the quickest transshipment problem with zero transit times. In *Proceedings of the Ninth Annual ACM/SIAM Symposium on Discrete Algorithms*, pages 147–156, 1998. Submitted to SIAM Journal on Optimization.

[6] L. Fleischer. Universally maximum flow with piecewise constant capacities. In G. Cornuejols, R. E. Burkard, and G. J. Woeginger, editors, *Integer Programming and Combinatorial Optimization: 7th International IPCO Conference*, number 1610 in Lecture Notes in Computer Science, pages 151–165, Graz, Austria, June 1999. Springer-Verlag.

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

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

[9] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, 18(1):30–55, 1989.

[10] A. V. Goldberg and S. Rao. Beyond the flow decomposition barrier. *Journal of the ACM*, 1999.

[11] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM*, 35:921–940, 1988.

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

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

[14] F. H. Moss and A. Segall. An optimal control approach to dynamic routing in networks. *IEEE Transactions on Automatic Control*, 27(2):329–339, 1982.

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

[16] J. B. Orlin. Minimum convex cost dynamic network flows. *Mathematics of Operations Research*, 9(2):190–207, 1984.

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

[18] W. B. Powell, P. Jaillet, and A. Odoni. Stochastic and dynamic networks and routing. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Handbooks in Operations Research and Management Science: Networks.* Elsevier Science Publishers B. V., 1995.

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

[20] T. Radzik. Approximate generalized circulation. Technical Report 93-2, Cornell Computational Optimization Project, Cornell University, 1993.

[21] D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26:362–391, 1983.

[22] G. I. Stassinopoulos and P. Konstantopoulos. Optimal congestion control in single destination networks. *IEEE transactions on communications*, 33(8):792–800, 1985.