Approximately Optimal Control of Fluid Networks

Lisa Fleischer* Jay Sethuraman†

Abstract

We give an approximation algorithm for the optimal control problem in fluid networks. Such problems arise as fluid relaxations of multiclass queueing networks, and are used to find approximate solutions to complex job shop scheduling problems. In a network with linear flow costs and linear, per-unit-time holding costs, our algorithm finds a drainage of the network, that for given constants $\varepsilon > 0$ and $\delta > 0$ has total cost $(1+\varepsilon)\mathsf{OPT} + \delta$, where OPT is the cost of the minimum cost drainage. The complexity of our algorithm is polynomial in the size of the input network, $\frac{1}{\varepsilon}$, and $\log \frac{1}{\delta}$. The fluid relaxation is a continuous problem. While the problem is known to have a piecewise constant solution, it is not known to have a polynomially-sized solution. We introduce a natural discretization of polynomial size and prove that this discretization produces a solution with low cost. This is the first polynomial time algorithm with a provable approximation guarantee for fluid relaxations.

1 Introduction

1.1 Problem description and formulation.

Motivated by the optimal control of multiclass queueing networks, we consider a class of continuous-time multicommodity flow problems in a directed network. Specifically, we are given a directed network $\mathcal{N} = (V \cup \{s\}, A)$, with commodities $k = 1, \ldots, K$, and a *sink* s; all capacities and costs are non-negative and commodity-dependent. For commodity k, node v has storage capacity $a_k(v)$, per-unit-time linear holding cost $h_k(v)$, and initial supply of commodity k of $d_k^\circ(v)$; edge e has flow-rate capacity $\mu_k(e)$, and linear flow cost $c_k(e)$. The flow-rate capacity is an upper bound of the flow-rate of commodity k on edge e if e is fully devoted to commodity k. If the use of edge e is divided among several commodities, then the flow-rate capacity for commodity k is $\mu_k(e)$ multiplied by the fraction of edge e alloted to commodity k. This can be represented by the following constraint,

$$\sum_{k \in K} \frac{f_k(e,t)}{\mu_k(e)} \leq 1,$$

where $f_k(e,t)$ is the flow-rate of commodity k on e at time t.

The multiflow problem with holding costs (MHC): We seek a flow (over time) that eventually drains all supplies to the sink s, obeys all the capacity constraints, while minimizing total flow and holding costs.[1] For this problem, it is possible that the optimal solution has exponential complexity: the number of changes in the flow pattern may be exponential in the network size. Our main result is an efficient algorithm for finding a near-optimal feasible flow: given constants $\varepsilon > 0$ and $\delta > 0$, we find a solution with total cost at most $(1+\varepsilon)\mathsf{OPT} + \delta$, where OPT is the cost of the minimum cost drainage. The complexity of our algorithm is polynomial in the size of the input network, $\frac{1}{\varepsilon}$, and $\log \frac{1}{\delta}$.

We consider two versions of this problem, and give the same guarantee for both. The *free flow* version, in which flow of commodity k is allowed to travel on any set of paths to reach the sink s; and the *fixed paths* version, where flow of commodity k must travel along a pre-specified path (or set of paths), and the problem is to determine when to continue flow along each arc in the path.

The problem of finding the optimal flow rates $f(\cdot,\cdot)$ for the free-flow version may be formulated as a continuous linear programming problem as described below. We discuss modifications necessary to handle the fixed-paths version in Section 4.2.

*Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA 15213, USA, Email: lkf@andrew.cmu.edu. Supported in part through NSF CAREER Award CCR-0049071 and NSF Award EIA-0049084.

†Department of Industrial Engineering and Operations Research, Columbia University, New York, NY 10027, USA, Email: jay@ieor.columbia.edu. Supported in part through NSF CAREER Award DMI-0093981 and IBM Faculty Partnership Award.

[1] While the problem is defined with only one sink, this is without loss of generality: for any $v \in V$ with $h_k(v) = 0$ we create an arc from v to s with infinite capacity, zero cost, (for commodity k) and impose an infinitesimal holding cost on v.

Minimize

$$\sum_{k\in K}[\sum_{e\in A} c_k(e)\int_0^\infty f_k(e,t)\,dt + \sum_{v\in V} h_k(v)\int_0^\infty d_k(v,t)dt]$$

subject to

$$\forall\, v\in V, t\in R_+,$$

$$d_k(v,t) = d_k^\circ(v) - \int_0^t [\sum_{e\in\delta^+(v)} f_k(e,\theta) - \sum_{e\in\delta^-(v)} f_k(e,\theta)]d\theta$$

$$\forall e\in A, t\in R_+, \qquad \sum_{k\in K}\frac{f_k(e,t)}{\mu_k(e)} \le 1$$

$$\forall v\in V, t\in R_+, k\in K, \quad 0\le d_k(v,t)\le a_k(v)$$

$$\forall e\in A, t\in R_+, k\in K, \quad f_k(e,t)\ge 0$$

In this formulation, $d_k(v,t)$ represents the storage of commodity k in node v at time t. The first constraint conserves flow for each commodity-node pair at each point in time; the second constraint restricts the total amount of work an edge can perform at any moment of time; and the final constraint enforces the storage capacity for each commodity-node pair at each time.

Continuous linear programs were introduced by Bellman [7, 8], who studied a linear optimal control problem in production planning. In spite of a tremendous amount of effort, general continuous linear programs remain difficult to solve [2]. Our interest in these problems is due to their ability to model a variety of dynamic resource allocation problems, described next; fortunately, the problems of interest in the applications have a special structure [2, 29], which we exploit to provide efficient solutions.

1.2 Motivation.

The production planning problem faced by a manufacturer owning a network of flexible machines can be described as follows. The manufacturer produces K products, and a priori estimates of the demand for each product is available. Each product is produced by processing raw material through a fixed sequence of machines ("stages"), requiring varying amounts of processing time at each of the machines in this sequence. Holding costs are used at each stage for each product to capture the opportunity cost of the resources invested. The objective is to produce the required quantities of the various products at minimum cost.

If all of the data are known with certainty, this is a simply a job shop scheduling problem with the holding cost objective, which is already notoriously difficult to solve exactly. Moreover, an optimal schedule is usually not robust to changes in problem data. This is an important limitation because, in practice, several additional difficulties arise: the processing time for each product at each stage may be random; the demand estimates may need to change because of additional orders or cancellations for certain products. These additional features can be modeled using stochastic processes, leading to the notion of a multiclass queueing network.

Multiclass queueing networks serve as useful models for problems in which several types of *activities* compete for a limited number of shared *resources* [13, 15, 20]. Examples include shared computer systems, manufacturing systems that produce different types of products, and telecommunication systems where heterogeneous traffic types (email, file transfers, video etc.) share common resources (buses in a local area network, routers in gateways). The optimal control problem in a multiclass queueing network is to find an optimal allocation of the available resources to activities *over time*. Recognizing the importance and the inherent intractability of this problem, the research community has focused its attention, for the most part, on developing tractable approximations [4, 11, 12, 21, 22, 25, 27, 33]. Two promising classes of approximations have emerged as a result: *Brownian* approximations and *Fluid* approximations. Both of these approximations arise as formal limits of multiclass queueing networks under (different) time and space scalings. Brownian models typically make use of the mean and variance of the associated stochastic processes in deriving a simpler control problem; unfortunately, except for problems that are essentially one dimensional, this control problem is itself intractable [20]. Fluid relaxations, the subject of this paper, ignore the variance of the associated stochastic processes, and depend only on their mean.

Fluid relaxations are deterministic, continuous approximations to stochastic, discrete networks. Essentially, we replace *discrete* jobs moving *stochastically* through a network by a *continuous*, *deterministic* fluid flow. In addition, we allow a resource to be "shared" among multiple activities simultaneously. Any optimal control problem in a multiclass queueing network can be addressed using a three-step approach: (a) formulate the appropriate fluid relaxation of the problem; (b) solve the fluid relaxation; and (c) use the optimal solution to the fluid relaxation to derive an implementable solution for the original control problem. (This is similar in spirit to deriving reasonable solutions to integer programs based on solving their linear programming relaxations.) In fact, the most successful methods for controlling multiclass queueing networks rely on the BIGSTEP approach [22], which results in *discrete-review* policies. In such a policy, the state of the queueing network is reviewed at discrete points in time. At each review point, a processing plan is formulated for the next review period based on the work present in the system. The computation of this process-

ing plan is essentially a fluid relaxation (of the sort described earlier) in which the initial supplies are the observed workload. This plan is then translated to an implementable plan in the actual system, at the end of which the system is reviewed again. The implementation question is also non-trivial because the jobs are discrete, processing times are variable, etc. The success of this approach depends on the efficiency of solving the fluid relaxation and the effectiveness of the "translation" scheme.

Given an optimal (or near-optimal) solution to the fluid relaxation, effective translation schemes have been designed for various problem classes. Recent applications of this approach include near-optimal schedules for deterministic job shop problems with the makespan and holding cost objectives [9, 10], asymptotically optimal schedules for stochastic job shops with the makespan objective [14], and asymptotically optimal schedules for multiclass queueing networks [6, 26]. All of these results rely on the solution to associated fluid relaxation(s). While the fluid relaxation for the makespan objective is solvable in closed form, the case of linear holding costs is significantly more difficult. In this paper, we shall focus on the problem of solving this fluid relaxation efficiently. For this and related problems, we provide the first efficient algorithm with a *provable* performance guarantee.

1.3 Previous work and related problems.

Fluid relaxations belong to a specially structured class of continuous linear programs called *state constrained separated continuous linear programs* (SCSCLP). In the absence of upper bounds on storage, these are called *separated continuous linear programs* (SCLP). The flow-rate functions on the edges are the "control" variables, and the storage at the nodes are the "state" variables; the term "separated" refers to the absence of state feedback. SCLPs were first introduced by Anderson [1] as a continuous model for job shop scheduling. Anderson, Nash, and Perold [3] characterized the extreme point solutions to SCLP. In addition, for problems with linear data, they showed the existence of an optimal solution in which the flow-rate functions are piecewise constant (hence, piecewise linear node-storages) with a finite number of pieces. The complexity of SCLP is still unresolved; in fact, it is not known if the *size* of the optimal solution is polynomially bounded by the input size.

In a series of papers [29, 30, 31, 32], Pullan carried out an extensive study of SCLPs and variants; he proposed an elegant dual for this problem, established strong duality, and designed a class of convergent algorithms, based on time-discretization. Pullan's algorithm starts with a guess of the breakpoints in the optimal solution. With respect to this fixed set of breakpoints, the problem can be solved as a linear program. To compute a lower bound, another linear program with twice as many breakpoints is constructed, with a slightly modified cost function; the cost function is modified in such a way that every feasible solution to its dual can be used to construct a feasible solution to the dual of the original continuous linear program with identical cost. Thus, by solving these two (ordinary) linear programs, one can estimate the duality gap. If the gap is not small enough, the number of breakpoints is doubled, with a new breakpoint added at the midpoints of the original breakpoints. As one can see, a naive implementation of this algorithm becomes impractical soon; to overcome this difficulty, variants have been developed in which redundant breakpoints are identified and removed every once in a while [28], leading to the so-called adaptive discretization algorithms. Luo and Bertsimas [24] introduced SCSCLP, established strong duality, and proposed a convergent class of algorithms for this problem. Their algorithm is also based on time discretization, removes redundant breakpoints, but solves quadratic programs in intermediate steps. All of these algorithms guarantee convergence, but provide neither a bound on the number of iterations needed, nor a bound on the number of breakpoints in the solution computed.

In the special case when all holding costs are equal, the problem is solved by a flow that minimizes the total supply left in the network at every moment in time. Optimal solutions for this problem (called a *universally quickest transshipment*) along with polynomial time algorithms to compute it are described in [19, 16]. A more complicated problem that is not known to have a polynomial sized solution is the problem of minimizing the total time flow takes to reach the sink t from a specified source s when it takes flow time to travel from the tail of an edge to the head of an edge. This is the universally quickest flow problem with transit times. For this problem, Hoppe and Tardos describe a fully polynomial approximation scheme [23]. When in addition there are multiple sources, a fully polynomial approximation scheme is described in [17].

One key difference between universally quickest flows (with uniform holding costs and with or without travel times) and MHC (with general holding costs) is that an optimal solution to MHC may require sending flow on non-simple paths, while optimal solutions to universally quickest flows never require this.

The MHC problem on a line – a tandem network – for the special case when holding costs are nondecreasing as they approach the sink s is solvable in polynomial time [5].

1.4 Our Contribution.

Our main contribution is the first provably efficient algorithm for approximately solving MHC: our algorithm works for both the free-flow and the fixed-paths versions. Given constants $\varepsilon > 0$ and $\delta > 0$, we find a solution with total cost at most $(1+\varepsilon)\mathsf{OPT} + \delta$, where OPT is the cost of the minimum cost drainage. The complexity of our algorithm is

polynomial in the size of the input network, $\frac{1}{\epsilon}$, and $\log \frac{1}{\delta}$.

Our algorithm also uses time discretization, but, in contrast to previous approaches for MHC and SCLP, our algorithm works with a *fixed* time partition. A *fixed* time partition is used previously in the approximation scheme to minimize total time the flow spends in the network when there are transit times and multiple sources [17]. We prove that the optimal instantaneous holding cost function is a convex, decreasing function, and use this to devise strong lower bounds for the problem based on the time partition. We use a time expanded network with side constraints, with network copies representing geometrically increasing units of time. Our algorithm finds a flow with constant flow rates within each time interval in the partition. This is in contrast to prior discretization-based algorithms [29, 24] which adaptively refine the discretization, and are unable to bound the number of breakpoints in the computed solution. Our approximation scheme provides a systematic way to control the *solution complexity*: if a solution with a small number of breakpoints is desired, our scheme could be adapted by suitably choosing ϵ and δ.

In addition to providing the desired solution, our algorithm also provides a bound on the sub-optimality of the given solution. In particular, our algorithm may be used in an adaptive setting: given a solution produced by our algorithm, the contribution towards improving the approximation guarantee of individual breakpoints can be assessed, and then removed if deemed small enough. Alternatively, the algorithm can start with a coarse discretization and then the returned solution and bound will suggest which intervals would be best to refine in order to improve the value of the solution.

This is especially significant because the number of pieces in an optimal solution may not be polynomially bounded in the input size; moreover, solutions with frequently changing controls may be unusable in practice.

2 Preliminaries

Input form and size. Our network has $n = |V|$ vertices and $m = |E|$ arcs. While the control problem in fluid networks is defined for arbitrary input, we assume that we are handling numerical input specified as the ratio of two integers, the maximum of which is bounded by U. Thus the size of the input to the problem can be expressed as a polynomial in terms of n, m, and $\log U$.

Without loss of generality, we assume that the capacity function u is integral. This can be done by multiplying capacities and demands by the least common multiple of capacity denominators, and dividing the costs by the identical number. The solution to the resulting problem has the same cost as the original, and can be transformed into a solution to the original problem simply by dividing the flow rate at each moment of time by the same scaling factor.

Notation. We use $f(t)$ to denote control f at time t. We use $f(e)$ to denote the K-component vector of functions of time that describe the control of each commodity on arc e. We use $f(e,t)$ to denote the vector of specific commodity flow values on e at time t. An optimal control is denoted f^*.

Control f and initial storage d° induce a vector of vertex storage functions, denoted d_f. We use $d_f(t)$ to denote d_f vector evaluated at time t. We use $d_f(v)$ to denote the storage function at v. We use $d_f(v,t)$ to denote the storage at v at time t. When f is clear from context, we may use d instead of d_f. The storage function vector of an optimal control f^* is denoted d^*.

We abbreviate the objective function $\sum_{k\in K}[\ \sum_{e\in A} c_k(e) \int_0^\infty f_k(e,t)\ dt\ +\ \sum_{v\in V} h_k(v) \int_0^\infty d_k(v,t)\ dt\ ]$ as $\int_0^T c^\mathsf{T} f(t) + h^\mathsf{T} d(t)\ dt$, for an appropriate upper bound T, and refer to the instantaneous value at t as $c^\mathsf{T} f(t) + h^\mathsf{T} d(t)$.

3 Structure and Use of the Discretization

A key tool in our algorithm is a *non-uniform* time expanded network. Section 3.1 describes the structure and properties of this network. Section 3.2 describes some structure of the optimal solution. Section 3.3 combines the content of these two previous sections to develop a new lower bound for the optimal control problem that we use to prove approximate optimality of our algorithm.

3.1 Time-expanded networks.

We can compute a feasible, but not, in general, optimal control by using a uniform time-expanded network. A *time-expanded* network of $\mathcal{N} = (V, A)$ with time horizon T is denoted $\mathcal{N}^T$ and contains a copy of $\mathcal{N}$ for every time interval in $[0, T)$ of the form $[\theta, \theta + 1)$ for $\theta = 0, 1, \ldots, T-1$. The copy for interval $[\theta, \theta + 1)$ is denoted V_θ. The copies of vertex v and arc e in V_θ are denoted v_θ and e_θ, respectively. The flow capacity restrictions on $e \in A$ are interpreted as flow capacity restrictions for e_θ for each $\theta = 0, \ldots, T-1$. In addition, if storage is permitted at v, then there is a *holdover arc* from v_θ to $v_{\theta+1}$ of capacity $a_k(v)$ for each commodity $k = 1, \ldots, K$, for all $\theta = 0, \ldots, T-1$. Finally, there are holdover arcs $(s_\theta, s_{\theta+1})$ of infinite capacity for all $\theta = 0, \ldots, T-1$.

A trivial upper bound on the amount of time required by the optimal flow, if finite, to empty the network is simply $\sum_{k\in K} \sum_{v\in V} d_k^\circ(v)$, since at worst, the network drains flow at a rate equal to the minimum capacity, which is at least one if the problem is feasible. Thus, for the rest of the paper, we assume $T = \sum_{k\in K} \sum_{v\in V} d_k^\circ(v) \le n|U|^n$. A flow in the time-expanded network $\mathcal{N}^T$ corresponds to the control f obtained by interpreting the flow on arc e_θ as the flow rate on e in the interval $[\theta, \theta + 1)$ and interpreting the flow on arc $(v_\theta, v_{\theta+1})$ as the storage level at v at time $\theta + 1$. Since the obtained flow rates are constant on unit intervals, this

completely specifies f. Similarly, any control f corresponds to a flow x in $\mathcal{N}^T$: x is obtained by averaging f on unit intervals.

We will use variants of $\mathcal{N}^T$ to obtain upper and lower bounds on the cost of an optimal control. To motivate the structure and costs associated with these variants, we begin with some intuition for why $\mathcal{N}^T$, even when based on a very fine discretization, will not typically yield an optimal solution: A solution computed using $\mathcal{N}^T$ has the property that it is constant over the time intervals in the discretization. If the optimal control is fitted to the discretization, it would be necessary to average the flow over each interval. While averaging will maintain feasibility and flow costs, it does not maintain holding costs: consider a buffer with holding cost 1 and one unit of flow, and an arc leaving the buffer with capacity ten. If the flow is sent at maximum capacity from the start, then the holding cost is $\int_0^{1/10}(1-10x)dx = 1/20$. If the flow is kept in the buffer as long as possible and sent at maximum capacity at the end of the unit interval, the holding cost is $9/10 + \int_0^{1/10}(1-10x)dx = 19/20$. The average of either of these flows is the flow that sends flow at rate 1/10 of capacity throughout the unit interval, and this has holding cost $\int_0^1(1-x)dx = 1/2$. There are symmetric cost disparities for the case of flow that is entering the buffer.

Since $\mathcal{N}^T$ is computing a flow that is constant over intervals, we assign holding costs to arcs entering nodes and leaving nodes in V_θ to capture the resulting costs. Each vertex v_θ is associated with its own copy of holdover arcs entering and leaving v_θ. The cost on the entering arc captures the holding cost of flow that starts the interval at v, and the cost on the leaving arc captures the holding cost of flow that ends the interval at v. Thus flow that stays at v in the interval incurs both costs. Since flow is sent at a constant rate out of and into v, the holding cost for flow at v in the unit interval is the product of the holding cost at v, times the length of the interval represented by V_θ, in this case 1, and the average of the interval's initial and final storage levels at v. Thus the cost on the entering arc should the product of $1/2$ the holding cost at v, and the cost on the leaving arc should be the same.

We implement this as follows: The *time-expanded network with costs* modifies a time expanded network $\mathcal{N}^T$ by creating a new vertex v'_θ for each vertex v_θ in $\mathcal{N}^T$. The arc set of $\mathcal{N}^T$ is modified by replacing each holdover arc $(v_\theta, v_{\theta+1})$ with two arcs $(v_\theta, v'_{\theta+1})$ and $(v'_{\theta+1}, v_{\theta+1})$. The new arcs each have the capacity of the old arc, and cost $h_k(v)/2$ for commodity k. For each vertex $v \in V$, the arc (v'_0, v_0) is introduced with capacity $d_k^\circ(v)$ and cost $h_k(v)/2$ for commodity k, $k = 1, \ldots, K$. Arc e_θ has cost $c_k(e)$ for commodity k. For N^T, denote this modified network with costs as $\mathcal{N}_c^T$. Note that, aside from the first vertex v'_0, the set of added vertices are unnecessary for accurate computation. We add them for the sake of clarity.

THEOREM 3.1. *A flow x in $\mathcal{N}_c^T$ that sends, for all $v \in V$, $k \in K$, $d_k^\circ(v)$ units of flow from v'_0 to s_T corresponds to a control f in $\mathcal{N}$ with the same cost.*

Proof. Given x, let f be the piecewise constant flow obtained by interpreting $x_k(e_\theta)$ as the flow rate of commodity k on e in $[\theta, \theta+1)$ for all $k \in \{0, \ldots, K\}$, $e \in A$. Since f is constant on unit intervals, the rate of drainage from $v \in V$ in $[\theta, \theta+1)$ is constant on this interval. Thus the holding cost at v in this interval is $\sum_{k\in K} \frac{1}{2}h_k(v)|d_k(v,\theta) - d_k(v,\theta+1)| + h_k(v)\min\{d_k(v,\theta), d_k(v,\theta+1)\}$. For $0 \le \theta \le T-2$, this is captured by x as the cost of the flow of commodity k on (v'_θ, v_θ) plus the cost of the flow of commodity k on $(v_\theta, v'_{\theta+1})$. For $\theta = T-1$, this is the cost of flow of commodity k on (v'_{T-1}, v_{T-1}), since $d_k(v,T) = 0$ for all $k \in K$. The flow cost on this interval is simply the sum of the flow costs on arcs in V_θ. □

COROLLARY 3.1. *If f^* sends flow at a rate that is constant on unit intervals, then a minimum cost flow in $\mathcal{N}_c^T$ yields a minimum cost control.*

Unfortunately, we cannot use Corollary 3.1 to obtain an optimal control f^* in general since there is no guarantee that f^* is constant on unit intervals. If f^* sends a lot of flow at the beginning of an interval, and very little at the end, then its holding cost will be significantly lower than the holding cost of its average over the interval. We describe how to obtain a low cost approximation to f^* in Section 4.

However, even if f^* is constant over unit intervals, the algorithm implied by computing a minimum cost flow in $\mathcal{N}_c^T$ is pseudopolynomial: its complexity depends polynomially on $|U|$, and hence is exponential in the size of the input parameter $\log|U|$. Thus, to obtain a polynomial algorithm, it is necessary to work with smaller networks.

Instead of using V_θ to represent one unit of time, we can instead use V_θ to represent a time interval of length Δ. Then, the capacity of commodity $k \in K$ on each arc in V_θ is multiplied by Δ, and the cost on arcs entering and leaving V_θ are also multiplied by Δ. (That is, cost of commodity $k \in K$ on (v'_θ, v_θ) and $(v_\theta, v'_{\theta+1})$ is multiplied by Δ for all $v \in V$, $k \in K$.) Condensed time expanded networks are introduced in [17]. Our version differs from this previous version in that it allows condensing time over arbitrary intervals, not just intervals of uniform or increasing size. Flow in such a "condensed" time-expanded network corresponds to a control by dividing the flow on arc e_θ by Δ: If V_θ corresponds to the interval $[a, a+\Delta)$ then the control sends flow onto e_θ at rate $x(e_\theta)/\Delta$ for this entire interval. The storage level of commodity k at v at time $a+\alpha\Delta$ for $\alpha \in [0,1]$ is $(1-\alpha)\,x_k(v'_\theta, v_\theta) + \alpha\, x_k(v_\theta, v'_{\theta+1})$. The effect on the corresponding control of condensing the interval $[a, a+\Delta)$ in the time-expanded network to just one copy of $\mathcal{N}$ is to average the control over a longer interval –

an interval of length Δ. Thus, the coarser the time-expanded network, the higher the cost of the minimum cost flow and the corresponding control.

Given a set $\mathcal{I}$ of disjoint intervals that completely cover $[0,T)$, we denote the corresponding time-expanded network as $\mathcal{N}^T_{c,\mathcal{I}}$. The proof of the following theorem is similar to the proof of Theorem 3.1.

THEOREM 3.2. *A flow x in $\mathcal{N}^T_{c,\mathcal{I}}$ that sends, for all $v \in V$, $k \in K$, $d^\circ_k(v)$ units of flow from v'_0 to s_T corresponds to a control f in $\mathcal{N}$ with the same cost.*

3.2 Structure of an Optimal Solution.

In this section, we describe the structure of an optimal solution and show that the optimal instantaneous holding cost function is convex and decreasing. This is used crucially is establishing lower bounds for the fluid relaxation.

Anderson, Nash, and Perold [3] characterized the extreme point solutions to a class of continuous linear programs that include fluid relaxations. In particular, they proved the following (proof omitted), but do not give any bound on the number of breakpoints of f^*.

LEMMA 3.1. ([3] THEOREM 4) *For any instance of MHC, there always exists a piecewise constant f^*.*

COROLLARY 3.2. *$c^\mathsf{T} f^*(t)$ is a piecewise constant function of t.*

It is easy to see that an optimal solution may send flow on non-simple paths. In particular, it may be better to send excess supplies to a vertex with cheap holding costs while waiting for sufficient capacity to the sink. However, as the following lemma implies, the the total holding cost accrued in a unit interval decreases with time.

LEMMA 3.2. *$h^\mathsf{T} d^*(t)$ is a convex, decreasing function of t.*

Proof. If $h^\mathsf{T} d^*$ is not convex, then there is a lower tangent l to $h^\mathsf{T} d^*$ with discontinuous intersection with $h^\mathsf{T} d^*$. Let $0 < t_1 < t_3 < t_2$ be such that $h^\mathsf{T} d^*(t_1)$ and $h^\mathsf{T} d^*(t_2)$ are on l, $h^\mathsf{T} d^*(t_3)$ is not on l, and for all $t_1 \le t \le t_2$, $h^\mathsf{T} d^*(t)$ is on or above l. Modify f on the interval $[t_1, t_2)$ by replacing $f(e,t)$ with the average flow rate $\frac{1}{t_2-t_1}\int_{t_1}^{t_2} f(e,t)\,dt$ for all $e \in A$ and all $t \in [t_1, t_2)$. Call the new control $\bar{f}$. Since f obeys capacity constraints, so does $\bar{f}$. Note that $d_{\bar{f}}(t_1) = d^*(t_1)$ and $d_{\bar{f}}(t_2) = d^*(t_2)$ but that for $t \in (t_1, t_2)$, $d_{\bar{f}}$ changes linearly from $d^*(t_1)$ to $d^*(t_2)$; i.e. $d_{\bar{f}}(t) = \frac{t_2-t}{t_2-t_1} d^*(t_1) + \frac{t-t_1}{t_2-t_1} d^*(t_2)$. Since d^* is nonnegative, so is $d_{\bar{f}}$. By choice of t_1 and t_2, the total holding cost over $[t_1, t_2)$ is strictly less with $d_{\bar{f}}$. Since in addition $\int_{t_1}^{t_2} c^\mathsf{T} f^*(t)\,dt = \int_{t_1}^{t_2} c^\mathsf{T} \bar{f}(t)\,dt$, this contradicts the optimality of f^*. Hence $h^\mathsf{T} d^*$ is convex.

Since $h^\mathsf{T} d^*(0) = h^\mathsf{T} d^\circ > 0$, $h^\mathsf{T} d^*(T) = 0$, and $h^\mathsf{T} d^*$ is convex, $h^\mathsf{T} d^*$ is also decreasing. □

Notice that the above proof extends to show that $h^\mathsf{T} d^*(t)$ is convex decreasing even when f^* is restricted to send flow of commodity k along a prespecified path.

This proof extends trivially to the case of a control computed via a minimum cost flow in $\mathcal{N}^T_{c,\mathcal{I}}$. We summarize this in the following corollary.

COROLLARY 3.3. *The piecewise constant control f obtained from a minimum cost flow in $\mathcal{N}^T_{c,\mathcal{I}}$ yields a storage function vector $d(t)$ so that $h^\mathsf{T} d(t)$ is a convex, decreasing function of t.*

3.3 A Strong Lower Bound.

Theorem 3.1 describes how to obtain upper bounds on the cost of a minimum cost control. To obtain a lower bound, we combine ideas of sections 3.1 and 3.2.

LEMMA 3.3. *For any interval partition $\mathcal{I}$ with corresponding breakpoints $0 = b_0 < b_1 < \cdots < b_r = T$,*
$\langle a \rangle$ the cost of the control obtained by setting the flow rate in an interval of $\mathcal{I}$ to be the average of f^* over the interval is $\int_0^T c^\mathsf{T} f^*(t)\,dt + \frac{1}{2} h^\mathsf{T} d^\circ + \sum_{\theta=1}^r (b_\theta - b_{\theta-1})\, h^\mathsf{T} d^*(b_\theta))$;
$\langle b \rangle$ $\int_0^T c^\mathsf{T} f^*(t)\,dt + \sum_{\theta=1}^r (b_\theta - b_{\theta-1})\, h^\mathsf{T} d^*(b_\theta))$ is a lower bound on the cost of an optimal control f^*.

Proof. To show $\langle b \rangle$, it suffices to show that $\mathbf{h} := \sum_{\theta=1}^r (b_\theta - b_{\theta-1})\, h^\mathsf{T} d^*(b_\theta))$ is a lower bound on the holding cost of the optimal control. Note that $\mathbf{h}$ is the integral of the decreasing step function $l(t) := h^\mathsf{T} d^*(b_\theta)$ for all $t \in (b_{\theta-1}, b_\theta]$, for all $\theta = 1, \ldots r$. By Lemma 3.2, $h^\mathsf{T} d^*(t)$ is convex and decreasing function of t, hence $l(t) \le h^\mathsf{T} d^*(t)$ for all $t \in (0,T]$ and thus $\int_0^T l(t) d(t)\,dt \le \int_0^T h^\mathsf{T} d^*(t)\,dt$.

For $\langle a \rangle$, the cost of f^* averaged over intervals in $\mathcal{I}$ is the sum of the flow costs and the holding costs. The sum of the averaged flow costs is independent of $\mathcal{I}$ and equal to $\int_0^T c^\mathsf{T} f^*(t)\,dt$. The sum of the resulting holding costs is

$$\begin{aligned} &\sum_{\theta=1}^r \frac{1}{2}(b_\theta - b_{\theta-1})\,[h^\mathsf{T} d^*(b_\theta) + h^\mathsf{T} d^*(b_{\theta-1})] \\ &\quad = \frac{1}{2}\sum_{\theta=1}^r (b_\theta - b_{\theta-1})\, h^\mathsf{T} d^*(b_\theta) + \\ &\quad\quad \frac{1}{2}\sum_{\theta=1}^r (b_\theta - b_{\theta-1})\, h^\mathsf{T} d^*(b_{\theta-1}) \qquad (3.1) \\ &\quad = \frac{1}{2} h^\mathsf{T} d^\circ + \sum_{\theta=1}^r (b_\theta - b_{\theta-1})\, h^\mathsf{T} d^*(b_\theta), \end{aligned}$$

where the last equality follows from $h^\mathsf{T} d^*(b_r) = h^\mathsf{T} d^*(T) = 0$. □

Without knowing f^*, we cannot compute the lower bound described in Lemma 3.3. The following lemma gives a computable lower bound.

LEMMA 3.4. *If x is a minimum cost flow in $\mathcal{N}^T_{c,\mathcal{I}}$ for interval partition $\mathcal{I}$ with corresponding breakpoints $0 = b_0 < b_1 < \cdots < b_r = T$, f is the corresponding control, and d is the corresponding vector of storage functions, then $\int_0^T c^\mathsf{T} f(t)\,dt + \sum_{\theta=1}^r (b_\theta - b_{\theta-1})\, h^\mathsf{T} d(b_\theta) \le \int_0^T c^\mathsf{T} f^*(t) + h^\mathsf{T} d^*(t) dt$.*

Proof. Since x yields a minimum cost piecewise-constant control with breakpoints in $B = \{b_0, b_1, \ldots, b_r\}$ it minimizes the integral of the piecewise linear cost curve of the corresponding control with breakpoints in B. The integral breaks down into sum of the area under two curves: $c^\mathsf{T} f$ and $h^\mathsf{T} d$. Using (3.1) with d^* replaced by d, we have that the area under $h^\mathsf{T} d$ is $\frac{1}{2} h^\mathsf{T} d^\circ + \sum_{\theta=1}^r (b_\theta - b_{\theta-1})\, h^\mathsf{T} d(b_\theta)$. Since the first term in this expression is a constant independent from x, we have that x minimizes $\int_0^T c^\mathsf{T} f(t)\,dt + \sum_{\theta=1}^r (b_\theta - b_{\theta-1})\, h^\mathsf{T} d(b_\theta)$, subject to f being piecewise constant with breakpoints in B. Since this is at most the lower bound in Lemma 3.3 $\langle b \rangle$, this is at most $\int_0^T c^\mathsf{T} f^*(t) + h^\mathsf{T} d^*(t)\,dt$. □

4 An Approximation Scheme for Minimum Cost Control

We first describe the approximation scheme for MHC with free flow. In section 4.2, we show how to modify this in the setting of both simple and nonsimple fixed flow paths.

4.1 Free flow controls.

Our approximation scheme for MHC uses a time expanded network with network copies representing geometrically increasing units of time. A similar idea, but with a more complicated network to handle transit times, was introduced in [17] for approximating universally quickest flows with transit times.

The discretization uses $\frac{1}{2\varepsilon}(\lceil \log \frac{T h^\mathsf{T} d^\circ}{\delta\varepsilon} \rceil)$ copies of $\mathcal{N}$. These copies are partitioned into $q := \lceil \log \frac{T h^\mathsf{T} d^\circ}{\delta\varepsilon} \rceil$ sets of cardinality $\frac{1}{2\varepsilon}$ each. Denote these sets by $N_0, N_1, \ldots, N_{q-1}$. N_0 is the set of intervals of size $\frac{2\delta}{h^\mathsf{T} d^\circ}$ covering interval $[0, \frac{\delta}{\varepsilon h^\mathsf{T} d^\circ})$. N_1 is the set of intervals of size $\frac{2\delta}{h^\mathsf{T} d^\circ}$ covering interval $[\frac{\delta}{\varepsilon h^\mathsf{T} d^\circ}, \frac{2\delta}{\varepsilon h^\mathsf{T} d^\circ})$. For $1 < i < q-1$, N_i is the set of intervals of size $\frac{2^i\delta}{h^\mathsf{T} d^\circ}$ covering interval $[\frac{2^{i-1}\delta}{\varepsilon h^\mathsf{T} d^\circ}, \frac{2^i\delta}{\varepsilon h^\mathsf{T} d^\circ})$. N_{q-1} is the set of intervals of size $\frac{T\varepsilon}{2}$ covering interval $[\frac{T}{2}, T)$. Let $\mathcal{I}'$ be the set of all these intervals, and let B' be the set corresponding to the endpoints of these intervals.

THEOREM 4.1. *The control that corresponds to the minimum cost flow x in the time-expanded network based on intervals $\mathcal{I}'$ has cost at most $(1+\varepsilon)\mathsf{OPT} + \delta$.*

Proof. We compare the cost of the control $\bar{f}$ obtained by averaging f^* over each interval in $\mathcal{I}'$ to the lower bound implied by $\bar{f}$ as described in Lemma 3.3$\langle b \rangle$. Let $\bar{d}$ be the supplies induced by d° and $\bar{f}$. This lower bound is the sum of $\int_0^T c^\mathsf{T} f^*(t)\,dt$ and the integral of the decreasing step function $l(t) := h^\mathsf{T} d^*(b_\theta)$ for all $t \in (b_{\theta-1}, b_\theta]$, for all $\theta = 1, \ldots r$. We show that

$$\int_0^T h^\mathsf{T} \bar{d}(t)\,dt \;\le\; \delta + (1+\varepsilon) \int_0^T l(t)\,dt. \tag{4.2}$$

Since $\bar{f}$ corresponds to a flow in the discretized time expanded network, the control f corresponding to x has cost at most the cost of $\bar{f}$. Combined with the fact that $\int_0^T c^\mathsf{T} f^*(t)\,dt = \int_0^T c^\mathsf{T} \bar{f}(t)\,dt$ and Lemma 3.3, this observation and (4.2) imply the theorem.

Since $h^\mathsf{T} \bar{d}(t)$ and $l(t)$ are decreasing functions on $(0, T]$, we can evaluate their integrals by considering the area under each curve in horizontal strips. Note that $h^\mathsf{T} \bar{d}(t) = l(t)$ for all $t \in B'$.

Consider first the horizontal strip from $h^\mathsf{T} \bar{d}(\frac{\delta}{\varepsilon h^\mathsf{T} d^\circ})$ to $h^\mathsf{T} d^\circ$ as depicted in Figure 1. The area of the difference $h^\mathsf{T} \bar{d}(t) - l(t)$ in this strip can be broken down to the sum of areas of $h^\mathsf{T} \bar{d}(t) - l(t)$ over each interval of size $\frac{2\delta}{h^\mathsf{T} d^\circ}$. Since $h^\mathsf{T} \bar{d}$ is convex, decreasing, and equals the decreasing step function l at the end points, this difference is the sum of areas of triangles each with base $\frac{2\delta}{h^\mathsf{T} d^\circ}$, and total height bounded by $h^\mathsf{T} d^\circ$. Thus the difference in the areas in this topmost strip is at most δ.

Now consider any horizontal strip defined by the interval $[\, h^\mathsf{T} \bar{d}(T/2^{j-1}),\ h^\mathsf{T} \bar{d}(T/2^j)]$ for $j = 0, \ldots, q-1$. We will show that the area under curve $h^\mathsf{T} \bar{d}(t)$ that intersects this strip is at most $1+\varepsilon$ times the area under curve $l(t)$ that intersects this strip. Since this is true for all j; and summed over all j, these strips cover the interval $[0, h^\mathsf{T} \bar{d}(\frac{2\delta}{c^\mathsf{T} d^\circ})]$, this implies inequality (4.2).

First note that $l(t)$ and $h^\mathsf{T} \bar{d}(t)$ meet at both $t = T/2^j$ and $t = T/2^{j-1}$. Thus, both areas include the area of the strip to the left of $t = \frac{T}{2^j}$: this is the area of the rectangle with height $H_j := h^\mathsf{T} d(T/2^j) - h^\mathsf{T} d(T/2^{j-1})$ and width $\frac{T}{2^j}$. Both areas include no area to the right of $t = \frac{T}{2^{j-1}}$. Consider now the area in the strip along the horizontal axis from $\frac{T}{2^j}$ to $\frac{T}{2^{j-1}}$. In this interval, time is discretized into intervals of size $\frac{T\varepsilon}{2^j}$. Since $l(t)$ and $h^\mathsf{T} \bar{d}$ agree at all endpoints of these intervals, the area between the $h^\mathsf{T} \bar{d}$ and $l(t)$ in this strip is the area of the triangle with height equal to the height of the strip and base equal to the size of the discretized interval. Thus this area is $H_j \times \frac{T\varepsilon}{2^{j+1}}$. With our previous observations on the area to the left and right in this strip, this implies that in this strip, the ratio of the area under $h^\mathsf{T} \bar{d}(t)$ to the ratio under $l(t)$ is at most $(1+\varepsilon)$. □

Remarks. 1. While Theorem 4.1 yields a firm guarantee on the quality of the solution obtained, Lemma 3.4 may be used to obtain a specific guarantee for each particular instance. The specific guarantee may show that the actual approximation is of better quality than Theorem 4.1 promises.

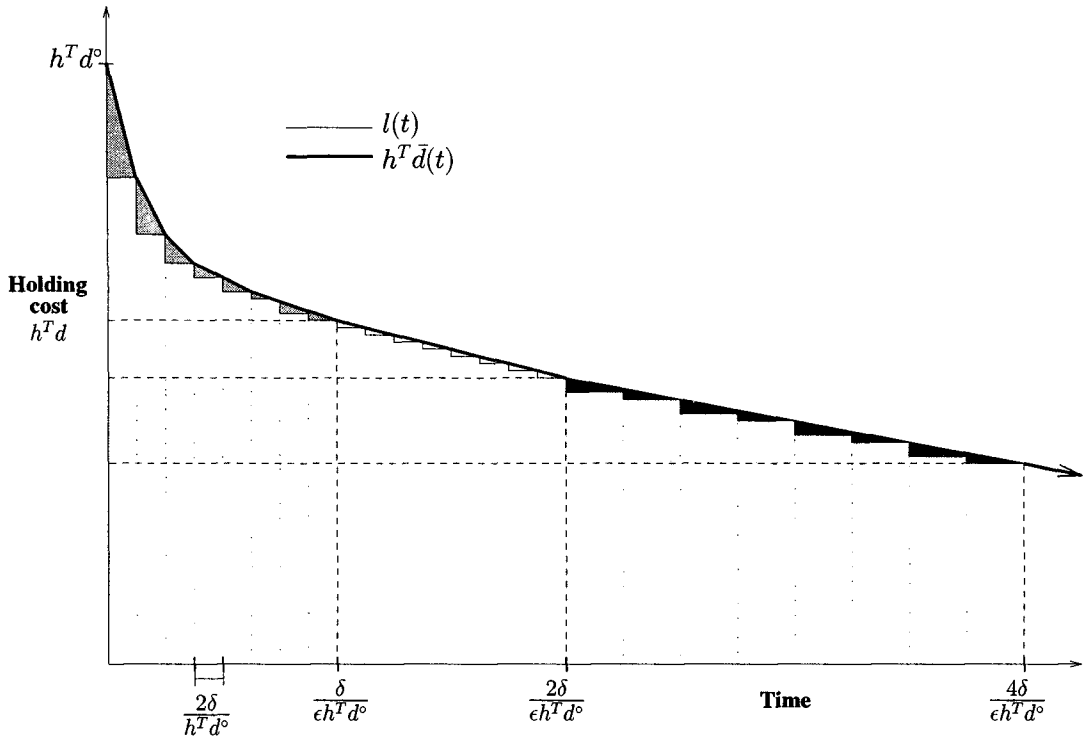

Figure 1: The medium shaded region corresponds to the area of $h^\mathsf{T}\bar{d}(t) - l(t)$ between points $h^\mathsf{T}d^\circ$ and $h^\mathsf{T}\bar{d}(\frac{\delta}{\epsilon h^\mathsf{T}d^\circ})$ on the vertical axis. The lightly shaded region is the strip for $j = q - 2$. The dark shaded region corresponds to the area of $h^\mathsf{T}\bar{d}(t) - l(t)$ between points $h^\mathsf{T}\bar{d}(T/2^{q-3})$ and $h^\mathsf{T}\bar{d}(T/2^{q-2})$ on the vertical axis.

Thus, Lemma 3.4 in conjunction with Theorem 3.2 can be used in an iterative manner to find a good discretization for any specific instance: starting with a very coarse discretization, one could iteratively refine only those intervals with large difference between the upper and lower bounds, while leaving large areas of the discretization at a coarse level.

2. In practice, it is desirable to have a control with few breakpoints. Thus, after computing the approximate flow, we can use Lemma 3.4 to remove breakpoints that are not necessary for the approximation guarantee.

3. Theorem 4.1 also holds in the setting of convex flow costs c, as averaging c over an interval only reduces total costs.

4.2 Fixed flow paths.

In this section we show how to modify the approach described in the previous sections to handle versions of the problem where the flow path for a commodity is fixed a priori.

Simple paths. If the supply originating at vertex v must follow a fixed path to the sink, we can incorporate this into the discretization by treating the supply from this sink as a single commodity. In the case when the path is simple, we can force it to follow the path by changing the capacity of arcs not on this path to 0 for this commodity. The resulting problem is a multicommodity flow problem on a polynomially sized network, which can be solved in polynomial time via linear programming.

Nonsimple paths. In the case when the path is not simple, we handle the path specification more carefully. In this case, it is not sufficient to restrict the flow of the commodity to arcs on the path, since the flow could then "skip" the cycle, or travel the cycle more times than specified. Instead, we could explicitly list the paths in the time-expanded network that the flow could follow. There are an exponential number of such paths, however, so we cannot afford to list them all explicitly. We argue here that the resulting, path-based linear program can be solved in polynomial time by keeping only an implicit representation of the paths.

We start by describing the path-based linear program corresponding to the time-expanded network with intervals $\mathcal{I}$ corresponding to breakpoint set B. Let $\mathcal{P}_k$ be the set of permissible paths for commodity k. For a vector, such as c, defined on the arcs in the time expanded network, we let $c(P) := \sum_{e_\theta \in P} c(e_\theta)$.

minimize

$$\sum_{P \in \mathcal{P}_k} c(P)x(P)$$

subject to

$$\begin{aligned} \sum_{P \in \mathcal{P}_k} x(P) &\geq d_k, \quad \forall\, k \in K \\ \sum_{k \in K} \sum_{P \in \mathcal{P}_k : e_\theta \in P} x(P)/\mu_k(e) &\leq 1, \\ & \forall\, e \in A,\ \forall\, \theta \in B \end{aligned}$$

This LP has an exponential number of variables. The column pricing problem is, given vectors $w \in \mathbb{R}^{|B| \times A}$, find for each commodity k, the permissible path $P \in \mathcal{P}_k$ minimizing

$$c(P) + \sum_{e_\theta \in P} \frac{w_{e_\theta}}{\mu_e}.$$

We can define the distance of edge e for commodity k as $c(e) + w_{e_\theta}/\mu_e$, reducing the pricing problem to a restricted shortest path problem. This shortest path problem can be solved exactly by a simple labeling algorithm even if the permissible path for commodity k is non-simple. Fix a commodity k; suppose its associated path visits a node v l times. Then the label for each copy v_θ of v in the time expanded network will be an l tuple $(b_1, b_2, \ldots, b_l)$, with b_i representing the shortest path from the source to v_θ with i visits to v (including the last). The entry b_i for node v_θ depends only b_i for node $v_{\theta-1}$ and the label of its predecessor in this path, and so can be computed efficiently. This labeling scheme can be used to identify the shortest path $P \in \mathcal{P}_k$, solving the pricing problem. This implies, via the ellipsoid algorithm [18], that we can solve the LP in polynomial time.

In practice, we would embed the polynomial time, approximate restricted shortest path subroutine within a column-generation framework for solving these linear programs.

4.3 Heuristic improvement.

In addition to the modification suggested at the end of section 4, we suggest a modification here that will improve the number of discretizations needed in the case that there are infinite capacity arcs. In particular, we show how to improve the estimate of the cost computed in the first moments of time in such a case. This is not covered in general by Corollary 3.1, since one simple usefulness of infinite capacity arcs is to allow an arbitrary amount of flow to be transported instantaneously from one node to another. Any flow using infinite capacity arcs in such a manner will not be constant over any non-zero interval of time in which they are used. This is particularly important in the first interval of time. To capture the usage of infinite capacity arcs at time 0, we modify $\mathcal{N}_c^T$ by adding the infinite capacity arcs of $\mathcal{N}$ to the vertex set $V_0' := \{v_0' \mid v \in V \cup \{s\}\}$. That is, for each arc $e \in A$ that has infinite capacity, we include a copy e_0' in V_0' with infinite capacity and 0 cost. This modified network now allows for instantaneous shipment of flow along infinite capacity arcs at the start of an otherwise piecewise constant control f.

References

[1] E. J. Anderson. *A continuous model for job-shop scheduling.* PhD thesis, University of Cambridge, 1978.

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

[3] E. J. Anderson, P. Nash, and A. F. Perold. Some properties of a class of continuous linear programs. *SIAM J. Control and Optimization*, 21:758–765, 1983.

[4] F. Avram, D. Bertsimas, and M. Ricard. Fluid models of sequencing problems in open queueing networks: an optimal control approach. In F. P. Kelly and R. J. Williams, editors, *Stochastic Networks*, volume 71 of *Proceedings of the International Mathematics Association*, pages 199–234. Springer-Verlag, New York, 1995.

[5] F. Avram, D. Bertsimas, and J. Sethuraman. Optimal control of fluid tandem networks. *Manuscript in preparation*, 2002.

[6] N. Bauerle. Asymptotic optimality of tracking policies in stochastic networks. *Annals of Applied Probability*, 10(4):1065–1083, 2000.

[7] R. Bellman. Bottleneck problem and dynamic programming. *Proc. Nat. Acad. Sci.*, 39:947–951, 1953.

[8] R. Bellman. *Dynamic Programming.* Princeton University Press, New Jersey, 1957.

[9] D. Bertsimas, D. Gamarnik, and J. Sethuraman. From fluid relaxations to practical algorithms for high multiplicity job shop scheduling: the holding cost objective. *Operations Research*, accepted for publication, 2002.

[10] D. Bertsimas and J. Sethuraman. From fluid relaxations to practical algorithms for job shop scheduling: the makespan objective. *Mathematical Programming*, 92(1):61–102, 2002.

[11] H. Chen and A. Mandelbaum. Discrete flow networks: Bottleneck analysis and fluid approximations. *Mathematics of Operations Research*, 16(2):408–446, 1991.

[12] H. Chen and A. Mandelbaum. Hierarchical modeling of stochastic networks, part i: fluid models. In D. D. Yao, editor, *Stochastic Modeling and Analysis of Manufacturing Systems*, pages 47–105, New York, NY, 1994. Springer-Verlag.

[13] H. Chen and D. D. Yao. *Fundamentals of Queueing Networks: Performance, Asymptotics, and Optimization.* Springer-Verlag, New York, 2001.

[14] J. G. Dai and G. Weiss. A fluid heuristic for minimizing makespan in job-shops. *Operations Research*, to appear, 2002.

[15] J. Filipiak. *Modelling and Control of Dynamic Flows in Communication Networks.* Springer Verlag, Berlin, 1988.

[16] L. Fleischer. Faster algorithms for the quickest transshipment problem. *SIAM J. on Optimization*, 12(1):18–35, 2001.

[17] L. Fleischer and M. Skutella. The quickest multicommodity flow problem. In *9th International Integer Programming and Combinatorial Optimization Conference*, pages 36–53, 2002.

[18] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.

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

[20] J. M. Harrison. *Brownian motion and stochastic flow systems.* John Wiley & Sons, 1985.

[21] J. M. Harrison. Brownian models of queueing networks with heterogenous customer populations. In W. Fleming and P. L. Lions, editors, *Stochastic Differential Systems, Stochastic Control Theory and Applications*, Proceedings of the International Mathematics Association, pages 147–186. Springer-Verlag, 1988.

[22] J. M. Harrison. The bigstep approach to flow management in stochastic processing networks. In F. P. Kelly, S. Zachary, and I. Ziedins, editors, *Stochastic Networks: Theory and Applications*, pages 57–90. Oxford University Press, 1996.

[23] 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.

[24] X. Luo and D. Bertsimas. A new algorithm for state-constrained separated continuous linear programs. *SIAM Journal on control and optimization*, 37(1):177–210, 1999.

[25] C. Maglaras. *Dynamic Control of Stochastic Processing Networks: A Fluid Model Approach.* PhD thesis, Stanford University, August 1998.

[26] C. Maglaras. Discrete-review policies for scheduling stochastic networks: Trajectory tracking and fluid-scale asymptotic optimality. *Annals of Applied Probability*, 10(3):897–929, 2000.

[27] S. P. Meyn. Stability and optimization of queueing networks and their fluid models. In G. G. Yin and Q. Zhang, editors, *Mathematics of Stochastic Manufacturing Systems*, volume 33 of *Lectures in Applied Mathematics*, pages 175–200. American Mathematical Society, 1997.

[28] A. B. Philpott and M. Craddock. An adaptive discretization algorithm for a class of continuous network programs. *Networks*, 26:1–11, 1995.

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

[30] M. C. Pullan. On the solution of a class of continuous linear programs. *SIAM Journal on Control and Optimization*, 32:1289–1296, 1994.

[31] M. C. Pullan. Forms of optimal solutions for separated continuous linear programs. *SIAM Journal on Control and Optimization*, 33(6):1952–1977, November 1995.

[32] M. C. Pullan. A duality theory for separated continuous linear programs. *SIAM Journal on Control and Optimization*, 34(3):931–965, May 1996.

[33] J. Sethuraman. *Scheduling Multiclass Queueing Networks and Job Shops using Fluid and Semidefinite Relaxations.* PhD thesis, Massachusetts Institute of Technology, September 1999.