FROM FLUID RELAXATIONS TO PRACTICAL ALGORITHMS FOR HIGH-MULTIPLICITY JOB-SHOP SCHEDULING: THE HOLDING COST OBJECTIVE

DIMITRIS BERTSIMAS

Sloan School of Management and Operations Research Center, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, dbertsim@mit.edu

DAVID GAMARNIK

IBM T. J. Watson Research Center, Yorktown Heights, New York 10598, gamarnik@watson.ibm.com

JAY SETHURAMAN

Department of Industrial Engineering and Operations Research, Columbia University, New York, New York 10027, jay@ieor.columbia.edu

We design an algorithm for the high-multiplicity job-shop scheduling problem with the objective of minimizing the total holding cost by appropriately rounding an optimal solution to a fluid relaxation in which we replace discrete jobs with the flow of a continuous fluid. The algorithm solves the fluid relaxation optimally and then aims to keep the schedule in the discrete network close to the schedule given by the fluid relaxation. If the number of jobs from each type grow linearly with N, then the algorithm is within an additive factor $O(N)$ from the optimal (which scales as $O(N^2)$); thus, it is asymptotically optimal. We report computational results on benchmark instances chosen from the OR library comparing the performance of the proposed algorithm and several commonly used heuristic methods. These results suggest that for problems of moderate to high multiplicity, the proposed algorithm outperforms these methods, and for very high multiplicity the overperformance is dramatic. For problems of low to moderate multiplicity, however, the relative errors of the heuristic methods are comparable to those of the proposed algorithm, and the best of these methods performs better overall than the proposed method.

Received December 1999; revisions received July 2000, September 2001; accepted September 2002.
Subject classifications: Production/scheduling, deterministic: approximation algorithms for deterministic job shops. Queues, optimization: asymptotically optimal solutions to queueing networks.
Area of review: Manufacturing, Service, and Supply Chain Operations.

1. INTRODUCTION

In this paper, we consider the high-multiplicity job-shop scheduling problem with the objective of minimizing holding costs defined as follows. We have a set of I job types on J machines. Job type i consists of J_i stages (also referred to as "tasks"), each of which must be completed on a particular machine. The pair (i, k) represents the kth stage of the ith job and has processing time $p_{i,k}$. Suppose that we have n_i jobs of type i. We are given nonnegative weights $w_{i,k}$ for type i jobs at stage k; our objective is to find a schedule that minimizes

$$\int_{t=0}^{\infty} \sum_{i=1}^{I} \sum_{k=1}^{J_i} w_{i,k} n_{i,k}(t)\, dt,$$

where $n_{i,k}(t)$ is the number of type i jobs in stage k at time t. Note that when $w_{i,k} = w_i$ (weight is independent from the stage k), the contribution of any individual job a of type i to the objective function is exactly $w_i DC(a)$, where $DC(a)$ is the completion time of job a at the last stage J_i. Thus, our objective generalizes a well-studied special case—minimize the sum of weighted completion times of all jobs.

In this paper, we consider a fluid relaxation of the problem, in which we replace discrete jobs with the flow of a continuous fluid. The idea of creating a feasible schedule by rounding a solution to a fluid relaxation for the job-shop scheduling problem, but for the makespan objective, was first introduced by Bertsimas and Gamarnik (1999) (see also Dai and Weiss 2002). The algorithm by Bertsimas and Gamarnik (1999) produces a schedule with makespan $C_{\max} + O(\sqrt{C_{\max}})$, where $C_{\max}$ is the lower bound provided by the fluid relaxation. Bertsimas and Sethuraman (2002) propose a more dynamic way to round the fluid relaxation that leads to a schedule with makespan at most $C_{\max} + (I+2)P_{\max}J_{\max}$, where I is the number of distinct job types, $J_{\max}$ is the maximum number of stages of any job type, and $P_{\max}$ is the maximum processing time over all tasks. In the present paper, we extend the technique of Bertsimas and Sethuraman (2002) to accomodate the objective of minimizing holding costs. Computational results on a class of benchmark instances indicate that the proposed algorithm is *practical*, in the sense that it is fast, easy to implement, and provides good quality solutions for high-multiplicity job-shop problems.

The motivation for considering the fluid relaxation under the holding-cost objective, comes from optimal control of multiclass queueing networks, which are stochastic and dynamic generalizations of job shops. In recent years there has been considerable progress in solving the fluid relaxation in multiclass queueing networks. Focusing on objective functions that minimize holding costs, Avram et al. (1995) use the Pontryagin maximum principle and find an optimal solution to the fluid relaxation explicitly. However, the description of the optimal fluid solution, while insightful for the original problem, involves the enumeration of an exponential number of cases. Luo and Bertsimas (1999), building upon the work of Pullan (1993), use the theory of continuous linear programming to propose a convergent numerical algorithm for the problem that is able to efficiently solve problems involving hundreds of machines and job types; we use this algorithm in solving the fluid relaxations throughout this work.

Contributions

1. We describe an efficient algorithm called the fluid-synchronization algorithm under the holding-cost objective (FSA-HC) to round an optimal fluid solution such that the resulting schedule is asymptotically optimal; the specific asymptotics we consider is a sequence of job-shop problems in which the number of type i jobs initially present is $\alpha_i N$, with α_i nonnegative constants and $N \to \infty$. We show that rounding an optimal fluid solution appropriately results in a schedule that incurs $O(N)$ extra cost compared to the optimal cost of the fluid job shop. We also show that the optimal fluid cost is $O(N^2)$, and the difference between the optimal fluid cost and the optimal cost of the original problem is at most $O(N)$. This implies that the scheduling algorithm we construct is asymptotically optimal. Specifically, the relative error between the cost of the FSA-HC $Z_{\mathrm{D}}(N)$ and the cost of an optimal schedule $Z_{\mathrm{JS}}(N)$ is

$$\frac{Z_{\mathrm{D}}(N)-Z_{\mathrm{JS}}(N)}{Z_{\mathrm{JS}}(N)} \leqslant O\left(\frac{1}{N}\right).$$

2. We report computational results on the performance of the FSA-HC to a subset of the 82 benchmark problems in the OR library (http://mscmga.ms.ic.ac.uk/info.html). We also compare the performance of the proposed algorithm and several commonly used heuristic methods. These results suggest that for problems of moderate to high multiplicity, the proposed algorithm outperforms these methods, and for very high multiplicity the performance gain is dramatic. For problems of low to moderate multiplicity, however, the relative errors of the heuristic methods are comparable to those of the proposed algorithm, and the best of these methods performs better overall than the proposed method. Given that (a) it is common manufacturing practice in job shop environments to solve problems with high multiplicity, and (b) the algorithm is simple to implement and fast, the FSA-HC should be considered a candidate for practical application.

Related Work

The job-shop scheduling problem with the makespan objective has been widely studied. For a review of this literature, see Hall (1997), Karger et al. (1997), Hall et al. (1997), Bertsimas and Sethuraman (2002), and the references therein.

In contrast, the job-shop scheduling problem with the weighted completion time objective has received little attention in the discrete optimization literature. However, fluid relaxations with holding-cost objective have been studied extensively in the queueing literature. We provide a brief overview of this literature, which will also serve to place our results in perspective.

Fluid relaxations have been the subject of intensive research during the last decade. An important breakthrough was achieved by Dai (1995) and Rybko and Stolyar (1992), who established that stability of multiclass queueing networks is implied by stability of their deterministic fluid counterparts. Motivated by the success of these ideas in analyzing stability, there has been a growing literature in finding near-optimal scheduling policies using fluid relaxations. Papers that address this issue include the ones by Avram et al. (1995), Atkins and Chen (1995), Chen and Yao (1993), Eng et al. (1996), and Meyn (1997a, b). All of these papers formulate the fluid relaxation, find a fluid solution (optimal or otherwise), and then heuristically interpret the fluid solution to derive a discrete policy; except for Meyn (1997b), none of these papers presents any performance analysis of the derived discrete policy (except in fairly restricted settings). Meyn (1997b) discusses a policy-iteration algorithm and demonstrates its (quicker) convergence to optimality when initiated with an optimal fluid solution. Although this establishes an intimate connection between the large-state behavior of a multiclass queueing network and its fluid model, this property does not seem to be directly usable in designing a good policy.

Maglaras (2000), building on the BIGSTEP approach of Harrison (1996), proposed a class of policies based on solving fluid relaxations repeatedly. For any policy in this class, the nominal length of a review period is computed; based on the queue lengths at the beginning of each review period, the length of the nominal review period, and the planned "safety-stocks" for each class, a fluid-type problem is solved. A solution to this fluid-type problem is then used to derive a processing plan for that review period. The next review of the system is conducted as soon as this processing plan is completed, which could be different from the next nominal review time instant because of the stochastic nature of the processing times. This is an example of a "discrete-review" policy. Maglaras (2000) proves the stability of a fairly broad range of discrete-review policies and establishes their *fluid-scale asymptotic optimality*. An important distinction is that Maglaras considers a steady-state problem, but proves performance guarantees of a transient nature. Our work, in contrast, considers a transient problem to begin with. Thus, when restricted to transient problems, it may be possible to use results of Maglaras (2000) to obtain asymptotically optimal schedules. (The scheme presented in Maglaras 2000 appears to use the fact that the effective arrival rate of each class is strictly positive, and so has to be modified suitably to address the job-shop problem without arrivals.) However, our approach has two distinct advantages: First, we provide an explicit rate of convergence to optimality; second, we solve the fluid relaxation *once*, and do not re-solve it at intermediate points in time.

An interesting recent development is the work of Queyranne and Sviridenko (1999), in which they consider approximation algorithms for shop-scheduling problems with a minimal sum of job completion times objective. Their main result is the following: If there exists a polynomial-time algorithm for a class of multiprocessor job-shop problems that guarantees a makespan no larger than ϵ times the trivial lower bound (the so-called *congestion-dilation* bound), then they describe a polynomial-time algorithm for minimizing the weighted completion time that is within a factor of 8ϵ of the optimum. (Their algorithm works for a generalization in which release dates are also given.) Their algorithm involves use of the approximation scheme for the makespan objective. Note that the polynomial-time approximation schemes for the makespan objective do not always satisfy the hypothesis of their statement: The work of Queyranne and Sviridenko (1999) requires a makespan guarantee that is within a factor of ϵ of a *lower bound*, not the optimal makespan itself. In fact, recent results of Hoogeveen et al. (1998) show that the job-shop problem with the objective of minimizing weighted completion time does not have a polynomial-time approximation scheme unless $P = NP$.

We conclude our description of related work by outlining the similarities and differences with the earlier work of Bertsimas and Sethuraman (2002). In both papers, we solve a fluid relaxation of the underlying scheduling problem and use the solution to compute *nominal* start times for all the tasks. We can view the nominal start time of a particular task as its ideal start time. The nominal start times of the collection of tasks available for processing at a particular machine determine the task to be processed next; this is computed using the fluid-synchronization algorithm. The fluid-synchronization algorithm used here is similar to the one used in Bertsimas and Sethuraman (2002), which considers the makespan objective but differs from it in one crucial aspect: For the makespan objective, scheduling a job to start *earlier* than its nominal start time causes no difficulty, but for the holding-cost objective, it is *critical* that no task be scheduled *prior* to its nominal start time. While this seems like an innocent change in retrospect, this observation is crucial in proving Theorem 1, a key ingredient in the proof of our main theorem (Theorem 6). A second crucial ingredient is dealing with nonintegral queue lengths. For the makespan objective, the fluid solution is trivial, has a constant control, and so can be dealt with relatively easily. For the holding-cost objective, the fluid solution typically has multiple pieces, with a (different) constant control in each piece. It is often the case that the values of the queue lengths at the intermediate points are fractional, which is allowed in the fluid relaxation but disallowed in our output schedule. Thus, another (technical) difficulty in dealing with such fluid solutions is that prior to applying a rounding heuristic, it is necessary to construct a fluid solution with integer breakpoints (see §3.3).

In addition to these technical differences, our paper makes a conceptual contribution. Typically, algorithms that solve (or approximately solve) makespan objectives, and algorithms that solve (or approximately solve) weighted completion-time objectives share very little in common. Thus, it is rare that an algorithm that addresses the makespan objective is useful at all (in whatever form) in solving the weighted completion-time version of the problem. Our paper provides precisely such an algorithm, which yields asymptotically optimal schedules for high-multiplicity job-shop problems.

Structure of the Paper

In §2, we formulate the problem and define our notation. In §3, we introduce the fluid relaxation. In §4, we describe an algorithm to discretize an optimal fluid solution for the holding-cost objective and show that it provides an asymptotically optimal schedule. In §5, we present computational results on a variety of job-shop instances from the OR library. Section 6 contains some concluding remarks.

2. PROBLEM FORMULATION AND NOTATION

In the job-shop scheduling problem there are J machines $\sigma_1, \sigma_2, \ldots, \sigma_J$ which process I different types of jobs. Each job type is specified by the sequence of machines to be processed on, and the processing time on each machine. In particular, jobs of type $i, i = 1, 2, \ldots, I$ are processed on machines $\sigma^i_1, \sigma^i_2, \ldots, \sigma^i_{J_i}$ in that order, where $1 \leqslant J_i \leqslant J_{\max}$. The time to process a type i job on machine σ^i_k is denoted by $p_{i,k}$. Throughout, we assume that $p_{i,k}$ are integers. We put $P_{\max} = \max_{i,k} p_{i,k}$ and $\sigma_{\max} = \max_j |\sigma_j|$.

The jobs of type i that have been processed on machines $\sigma^i_1, \ldots, \sigma^i_{k-1}$, but not on machine σ^i_k, are queued at machine σ^i_k and are called "type i jobs in stage k" or "class (i, k)" jobs. We will also think of each machine σ_j as a collection of all type and stage pairs that it processes. Namely, for each $j = 1, 2, \ldots, J$,

$$\sigma_j = \left\{(i, k) : \sigma_j = \sigma^i_k, 1 \leqslant i \leqslant I, 1 \leqslant k \leqslant J\right\}.$$

There are n_i jobs for each type i initially present at their corresponding first stage. Let $w_{i,k}$ be nonnegative integer holding-cost rates associated with (i, k) jobs. Let $n_{i,k}(t)$ be the number of (i, k) jobs at machine σ^i_k at time t. Our objective is to find a scheduling policy that minimizes

$$\int_{t=0}^{\infty} \sum_{i=1}^{I} \sum_{k=1}^{J_i} w_{i,k} n_{i,k}(t)\, dt.$$

We impose the following restrictions on the schedule.

1. The schedule must be nonpreemptive. That is, once a machine begins processing a stage of a job, it must complete that stage before doing anything else.

2. Each machine may work on at most one task at any given time.

3. For $k > 1$, stage k of a job can begin only after the completion of its $(k-1)$st stage.

As mentioned earlier, we consider a sequence of job-shop problems for which the number of initial jobs of type i is $\alpha_i \cdot N$. Specifically, the sequence of job-shop problems we consider is indexed by N, which varies while all other quantities remain the same. In this paper, we construct a scheduling algorithm which is asymptotically optimal, as we let $N \to \infty$ and treat α_i as nonnegative constants independent of N.

3. THE FLUID JOB-SHOP SCHEDULING PROBLEM

3.1. Problem Formulation and Properties

In this section, we describe a continuous relaxation of the job-shop scheduling problem. In a fluid job shop, there are J machines $\sigma_1, \sigma_2, \ldots, \sigma_J$ and I job types. Each job type is specified by the sequence of machines σ_k^i, $k = 1, 2, \ldots, J_i$ on which it has to be processed; the processing time of a type i job on machine σ_k^i is a positive real number $p_{i,k}$. For convenience, we let $\mu_{i,k} = 1/p_{i,k}$; we can think of $\mu_{i,k}$ as the rate at which machine σ_k^i processes (i,k) jobs. We refer to type i jobs which have been processed on machines $\sigma_1^i, \sigma_2^i, \ldots, \sigma_{k-1}^i$, but not on machine σ_k^i, as jobs of class (i,k) or (i,k) jobs. We let $x_{i,k}(t)$ be the number of jobs of class (i,k) at time t. The number of type i jobs initially present, $x_{i,1}(0)$, is also denoted by x_i and can take arbitrary nonnegative values; we assume that $x_{i,k}(0) = 0$ for $k > 1$. In contrast to the discrete problem, the number of (i,k) jobs at time t can assume arbitrary nonnegative real values; for that reason, we think of this as the fluid level of class (i,k) at time t. Let $T_{i,k}(t)$ be the total amount of time machine σ_k^i works on class (i,k) jobs in the interval $[0,t)$. We first present all of the constraints:

$$x_{i,1}(t) = x_i - \mu_{i,1} T_{i,1}(t), \quad i = 1, 2, \ldots, I, t \geqslant 0, \tag{1}$$

$$x_{i,k}(t) = \mu_{i,k-1} T_{i,k-1}(t) - \mu_{i,k} T_{i,k}(t),$$
$$k = 2, \ldots, J_i, i = 1, 2, \ldots, I, t \geqslant 0, \tag{2}$$

$$0 \leqslant \sum_{(i,k) \in \sigma_j} (T_{i,k}(t_2) - T_{i,k}(t_1))$$
$$\leqslant t_2 - t_1, \ \forall t_2 > t_1, \ t_1, t_2 \geqslant 0, \ j = 1, 2, \ldots, J, \tag{3}$$

$$x_{i,k}(t) \geqslant 0, \quad T_{i,k}(t) \geqslant 0. \tag{4}$$

Constraints (1) and (2) capture the dynamics of the system. These equations merely state that the fluid level of class (i,k) at time t is the initial fluid level plus the amount of fluid that has arrived from class $(i, k-1)$ by time t minus the amount of class (i,k) fluid that has been processed by machine σ_k^i by time t. Constraints (4) reflect the fact that the fluid level of class (i,k) and the amount of time allocated by machine σ_k^i to class (i,k) are nonnegative. Constraint (3) is the capacity constraint for each machine—the total amount of time devoted to processing by machine j in an interval $[t_1, t_2)$ cannot exceed the length of the interval $t_2 - t_1$. Our objective function for the fluid job shop is

$$\int_0^\infty \sum_{i=1}^{I} \sum_{k=1}^{J_i} w_{i,k} x_{i,k}(t)\, dt.$$

The problem of whether a polynomial-time algorithm exists for the fluid-control problem is still open. However, based on several structural properties for this class of problems (see Anderson and Nash 1987), Luo and Bertsimas (1999), based on earlier work by Pullan (1993), propose provably convergent discretization-based methods that are able to quickly solve large-scale instances in practice. The algorithm of Luo and Bertsimas (1999) is used in our computational study.

A key property of the fluid job-shop problem that we shall make use of extensively is stated as Proposition 1; its proof can be found in Anderson and Nash (1987).

Proposition 1. *There exists an optimal solution for the fluid job-shop scheduling problem such that $x(t)$ is piecewise linear with a finite number of pieces.*

Note that by Proposition 1 there is always an optimal fluid solution such that $T_{i,k}(t)$ is piecewise linear and has a finite number of pieces. For this solution, we define

$$u_{i,k}(t) = \frac{dT_{i,k}(t)}{dt}. \tag{5}$$

Because $T_{i,k}(t)$ is piecewise linear, Equation (5) does not determine $u_{i,k}(t)$ at the (finitely many) breakpoints; at each of these breakpoints, we set

$$u_{i,k}(t) = u_{i,k}(t^+).$$

Clearly, $u_{i,k}(t)$ can be interpreted as the instantaneous fraction of effort allocated to class (i,k) jobs by machine σ_k^i at time t. We shall find it convenient to work with $u_{i,k}(t)$ instead of $T_{i,k}(t)$. The refore, Proposition 1 guarantees the existence of an optimal fluid solution with piecewise constant control. This property enables us to use repeatedly the machinery developed in Bertsimas and Sethuraman (2002) for the makespan objective to obtain asymptotically optimal schedules.

3.2. A Lower Bound

Let $Z_{\mathrm{F}}(N)$ denote the cost of an optimal fluid solution when the number of initial jobs of type i is $\alpha_i \cdot N$; similarly, let $Z_{\mathrm{JS}}(N)$ denote the cost of the optimal solution to the corresponding discrete job-shop problem.

We now establish a useful relationship between $Z_{\mathrm{F}}(N)$ and $Z_{\mathrm{JS}}(N)$. Ideally we would like to establish that $Z_{\mathrm{F}}(N)$ is a lower bound on the cost of an optimal job-shop schedule for the discrete network. While we have been unable to establish this result in general, we can prove the following theorem.

Theorem 1. (a) $Z_{\mathrm{F}}(N) = CN^2$.
(b) $Z_{\mathrm{JS}}(N) \geqslant Z_{\mathrm{F}}(N) - O(N)$.

Proof. (a) This follows immediately from the formulation of the fluid relaxation. More formally, suppose we have a solution to the fluid relaxation for $N=1$ (i.e., $n_i=\alpha_i$). This solution consists of the "allocation" variables $T^1_{i,k}(t)$, with the corresponding "queue-length" variables $x^1_{i,k}(t)$. We can use these to find a solution to the fluid relaxation when $n_i=\alpha_i\cdot N$ as follows. We set

$$T^N_{i,k}(N\cdot t)=NT^1_{i,k}(t),$$
$$x^N_{i,k}(N\cdot t)=Nx^1_{i,k}(t).$$

(b) To prove this part, we "fluidize" the optimal solution to the job-shop problem. Consider any feasible schedule to the discrete job-shop problem. We can "convert" this schedule into a schedule for the fluid network by processing the job "continuously." This is illustrated in Figure 1.

The feasibility of this schedule is immediate from the feasibility of the schedule for the discrete network. The extra cost incurred is

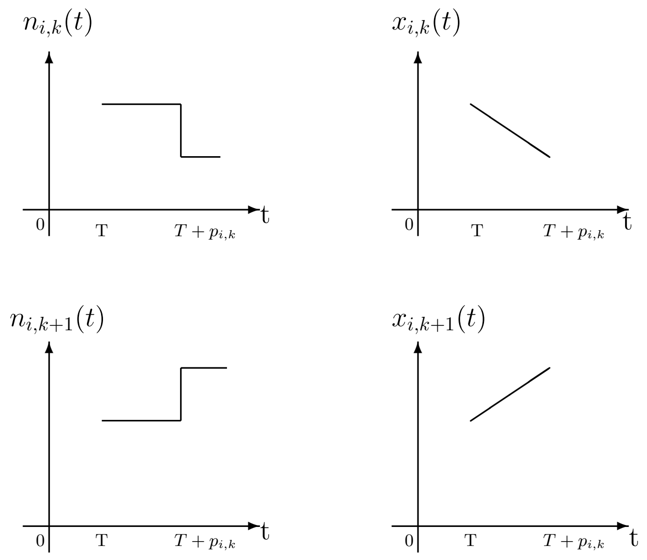
$$\sum_{i=1}^{I}\sum_{k=1}^{J_i}p_{i,k}\frac{(w_{i,k+1}-w_{i,k})}{2}\alpha_i N. \quad \square$$

For the special case in which all of the weights for a particular job type are equal (i.e., $w_{i,k}$ is independent of k), Z_{F} is indeed a lower bound for Z_{JS}. In the rest of this paper we drop the "N" and use Z_{F} and Z_{JS} instead; we emphasize, however, that both Z_{F} and Z_{JS} depend on N.

3.3. A Suboptimal Fluid Solution with Integral Queue Lengths at the Breakpoints

Proposition 1 establishes that a fluid optimal solution is characterized by piecewise-linear controls; i.e., the control is constant between successive breakpoints. Our overall strategy is to construct a near-optimal algorithm to the job-shop scheduling problem by rounding the optimal fluid solution. In this section we show that, starting with the optimal fluid solution, we can construct a piecewise-linear fluid solution with cost close to the optimal fluid cost and such that the queue lengths at the breakpoints are integral. This is a property that is needed in the construction of our main rounding algorithm.

Figure 1. "Fluidizing" a discrete job-shop schedule.



Lemma 2. *Consider a feasible fluid solution that applies constant controls in the time interval* $[0,L]$. *At time* 0 *the fluid level of class* (i,k) *is* $x_{i,k}$ *and at time* L *it is* $(y_{i,k})$. *Let* c *be the average holding cost associated with this solution during the interval* $[0,L]$. *For any integer* $M\geqslant J_{\max}$, *we can construct a new feasible fluid solution which applies constant controls in the time interval* $[0,\hat L]$, *where* $\hat L = L+2\sigma_{\max}P_{\max}J_{\max}$ *that has the following properties*:

1. *The solution starts from the configuration* $(\lfloor x_{i,1}\rfloor+M, \lfloor x_{i,2}\rfloor,\ldots,\lfloor x_{i,J_i}\rfloor)$ *and ends in configuration* $(\lfloor y_{i,1}\rfloor+M-J_{\max},\lfloor y_{i,2}\rfloor,\ldots,\lfloor y_{i,J_i}\rfloor)$ *at time* $\hat L$.
2. *The average holding cost of this solution* $\hat c$ *during the time interval* $[0,\hat L]$ *satisfies*

$$\hat c = c+\sum_{i,k}(x_{i,k}+y_{i,k})\sigma_{\max}P_{\max}J_{\max} + 2IM(2\sigma_{\max}P_{\max}J_{\max}+L). \tag{6}$$

Proof. Denote by $u_{i,k}$ the service effort that the original fluid solution allocates to the class (i,k). It then follows that

$$\mu_{i,k}u_{i,k}L=\sum_{l=1}^{k}(x_{i,l}-y_{i,l})\geqslant 0. \tag{7}$$

The new linear fluid solution will be defined over the time interval $[0,\hat L]$, where

$$\hat L = L+2\sigma_{\max}P_{\max}J_{\max}. \tag{8}$$

Specifically, we allocate effort $v_{i,k}$ to the class (i,k) over the interval $[0,\hat L]$, where

$$v_{i,k}=\frac{\sum_{l=1}^{k}(\lfloor x_{i,l}\rfloor-\lfloor y_{i,l}\rfloor)+J_{\max}}{\mu_{i,k}\hat L}.$$

We now show that $v_{i,k}$ define a feasible linear fluid solution with the associated cost satisfying (6). We first show that $v_{i,k}\geqslant 0$. Note that

$$\mu_{i,k}v_{i,k}\hat L=\sum_{l=1}^{k}(x_{i,l}-y_{i,l}) +\sum_{l=1}^{k}\big(\lfloor x_{i,l}\rfloor - x_{i,l}+y_{i,l}-\lfloor y_{i,l}\rfloor\big)+J_{\max}.$$

From the definition of $J_{\max}$ and applying the inequality part of (7) we obtain $v_{i,k}\geqslant 0$.

Now fix an arbitrary station σ_j. We have

$$\begin{aligned}\sum_{(i,k)\in\sigma_j} v_{i,k} &= \sum_{(i,k)\in\sigma_j}\frac{\sum_{l=1}^{k}(x_{i,l}-y_{i,l})}{\mu_{i,k}L}\cdot\frac{L}{\hat L}\\ &\quad+\sum_{(i,k)\in\sigma_j}\frac{\sum_{l=1}^{k}(\lfloor x_{i,l}\rfloor-x_{i,l}+y_{i,l}-\lfloor y_{i,l}\rfloor)+J_{\max}}{\mu_{i,k}\hat L}\\ &\leqslant \sum_{(i,k)\in\sigma_j}u_{i,k}\frac{L}{\hat L}+\sum_{(i,k)\in\sigma_j}\frac{k+J_{\max}}{\mu_{i,k}\hat L},\end{aligned}$$

where the inequality follows from the equality part of (7). However, by feasibility of the original solution, we have

$$\sum_{(i,k)\in\sigma_j} u_{i,k} \leqslant 1.$$

Also,

$$\sum_{(i,k)\in\sigma_j} \frac{k+J_{\max}}{\mu_{i,k}\hat{L}} \leqslant P_{\max}\sigma_{\max}\frac{2J_{\max}}{\hat{L}}.$$

From the definition of $\hat{L}$ in (8) it follows that

$$\sum_{(i,k)\in\sigma_j} v_{i,k} \leqslant 1.$$

We now show that the solution ends with queue lengths in class (i,k) equal to $\lfloor y_{i,k}\rfloor$. In fact, because the queue length of the class $(i,1)$ at time 0 is $\lfloor x_{i,k}\rfloor + M$ by assumption, then the queue length at time $\hat{L}$ in class $(i,1)$ is

$$\lfloor x_{i,1}\rfloor + M - \mu_{i,1}v_{i,1}\hat{L} = \lfloor x_{i,1}\rfloor + M - (\lfloor x_{i,1}\rfloor - \lfloor y_{i,1}\rfloor + J_{\max})$$
$$= \lfloor y_{i,1}\rfloor + M - J_{\max}.$$

Similarly, for $k = 2, 3, \ldots, J_i$, the queue length of the class (i,k) at the end time $\hat{L}$ is

$$\lfloor x_{i,k}\rfloor + \mu_{i,k-1}v_{i,k-1}\hat{L} - \mu_{i,k}v_{i,k}\hat{L}$$
$$= \lfloor x_{i,k}\rfloor + \sum_{l=1}^{k-1}(\lfloor x_{i,l}\rfloor - \lfloor y_{i,l}\rfloor)$$
$$+ J_{\max} - \sum_{l=1}^{k}(\lfloor x_{i,l}\rfloor - \lfloor y_{i,l}\rfloor) - J_{\max} = \lfloor y_{i,k}\rfloor.$$

To finish the proof, we analyze the cost of the constructed solution. Note that for each class (i,k), $k \geqslant 2$, the corresponding cost is the area of the trapezoid with height $\hat{L}$ and the lengths $\lfloor x_{i,k}\rfloor$, $\lfloor y_{i,k}\rfloor$. The area is then equal to

$$\tfrac{1}{2}(\lfloor x_{i,k}\rfloor + \lfloor y_{i,k}\rfloor)\hat{L}$$
$$\leqslant \tfrac{1}{2}(x_{i,k} + y_{i,k} + 2)(L + 2\sigma_{\max}P_{\max}J_{\max})$$
$$= \tfrac{1}{2}(x_{i,k} + y_{i,k})L + (x_{i,k} + y_{i,k})\sigma_{\max}P_{\max}J_{\max}$$
$$+ 2\sigma_{\max}P_{\max}J_{\max} + L.$$

Similarly, for the classes $(i,1)$, $i = 1, 2, \ldots, I$, the corresponding cost is

$$\tfrac{1}{2}(x_{i,1} + y_{i,1})L + (x_{i,1} + y_{i,1})\sigma_{\max}P_{\max}J_{\max}$$
$$+ (2M - J_{\max})(2\sigma_{\max}P_{\max}J_{\max} + L).$$

However, for all i, k, $\frac{1}{2}(x_{i,k} + y_{i,k})L$ is the cost of the original solution corresponding to the class (i,k). We conclude that the total cost of the constructed solution $\hat{c}$ satisfies

$$\hat{c} \leqslant c + \sum_{i,k}(x_{i,k} + y_{i,k})\sigma_{\max}P_{\max}J_{\max}$$
$$+ IJ_{\max}(2\sigma_{\max}P_{\max}J_{\max} + L)$$
$$+ I(2M - J_{\max})(2\sigma_{\max}P_{\max}J_{\max} + L)$$
$$= c + \sum_{i,k}(x_{i,k} + y_{i,k})\sigma_{\max}P_{\max}J_{\max}$$
$$+ 2IM(2\sigma_{\max}P_{\max}J_{\max} + L). \quad \square$$

This completes the proof of Lemma 2.

PROPOSITION 2. *Consider a feasible fluid solution that has piecewise-constant controls and has initial queue lengths $\alpha_i N$ for class i jobs. Suppose that the number of pieces is R and the queue length of the class (i,k) at the end of the rth piece is $Nx^r_{i,k}$. We can construct a new fluid solution with R pieces such that the initial queue lengths are $n_i = \lfloor N\alpha_i\rfloor + RJ_{\max}$, the queue length of the class (i,k) at the end of the rth piece is $\lfloor x^r_{i,1}\rfloor + (R-r)J_{\max}$ for $k=1$ and $\lfloor x^r_{i,k}\rfloor$ for $k>1$, and the cost of this solution $\hat{c}$ satisfies*

$$\hat{c} \leqslant c + O(N).$$

PROOF. We apply Lemma 2 to each individual piece $r = 1, 2, \ldots, R$ of the original fluid solution. Note that the values $x^r_{i,k}$ (queue lengths of the original fluid solution scaled by $1/N$) depend on α_i, but do not depend on N. Then, the difference between the costs c and $\hat{c}$ in Lemma 2 depends linearly on N. This completes the proof. $\square$

Note that by definition $x^R_{i,k} = 0$ for all classes (i,k). Thus, the new fluid solution will also have $\lfloor x^R_{i,k}\rfloor = 0$; i.e., all jobs will be processed in the new fluid solution.

4. THE FLUID-SYNCHRONIZATION ALGORITHM FOR THE HOLDING-COST OBJECTIVE

In this section, we describe the fluid-synchronization algorithm under the holding-cost objective (FSA-HC), which discretizes an optimal fluid solution. The algorithm is based on a repeated application of a variation of the fluid-synchronization algorithm (FSA) (called the revised fluid-synchronization algorithm (RFSA)) introduced by Bertsimas and Sethuraman in (2002). We describe the RFSA in detail in §4.1 and prove certain properties. Specifically, we show that for each piece of the optimal piecewise-linear fluid solution, the extra cost incurred by implementing the RFSA compared to the cost incurred by the fluid solution is $O(N)$. Our overall scheduling algorithm is then based on applying the RFSA for each individual piece and showing that the extra cost compared to the fluid cost is $R \cdot O(N) = O(N)$, where R is the number of pieces in the fluid solution. Because the cost of the fluid solution is $O(N^2)$, this would imply that the extra cost is of lower order. The rest of the section is organized as follows. We introduce the RFSA in §4.1. In §4.2, we introduce the FSA-HC, and in §4.3 we analyze its performance.

4.1. The Revised Fluid-Synchronization Algorithm

The RFSA is a variant of the FSA developed for the makespan objective in Bertsimas and Sethuraman (2002). The FSA applies to any feasible fluid solution in which jobs are serviced at constant rate. However, there is one important difficulty in using the FSA directly. For the holding-cost objective, processing a job "too soon" may be just as bad as processing a job "too late." For example, consider the nth (i,k) job and suppose $w_{i,k} \ll w_{i,k+1}$. The operations of the FSA are governed by the discrete start time $DS_{i,k}(n)$ and the nominal start time $NS_{i,k}(n)$ the nth (i,k) job (formal definitions are given below). Under the FSA, if $DS_{i,k}(n) \ll NS_{i,k}(n)$, then this job is processed sooner than necessary at stage k, thereby reaching stage $(k+1)$ substantially earlier and, therefore, accumulating holding costs at a much higher rate. This is in sharp contrast to the makespan objective, where there is no incentive for a machine to idle. We overcome this difficulty by modifying our definition of when a job becomes *available*. This variant of FSA is what we call the *revised fluid-synchronization algorithm* (RFSA). To introduce it, we adopt certain definitions from Bertsimas and Sethuraman (2002).

Definitions. Note that machine σ_j requires a certain processing time to process jobs that eventually come to it, which is

$$C_j = \sum_{(i,k)\in\sigma_j} p_{i,k} n_i.$$

The quantity C_j is called the congestion of machine σ_j. We denote the maximum congestion by

$$C_{\max} \equiv \max_{j=1,\ldots,J} C_j. \tag{9}$$

In addition, for machine σ_j, we let

$$U_j = \sum_{(i,k)\in\sigma_j} p_{i,k}$$

and

$$P_j = \max_{(i,k)\in\sigma_j} p_{i,k}. \tag{10}$$

Namely, U_j is the workload of machine σ_j when only one job per type is present, and P_j is the maximum processing time at σ_j. Finally, let

$$U_{\max} = \max_{1\leqslant j\leqslant J} U_j \tag{11}$$

and

$$P_{\max} = \max_{1\leqslant j\leqslant J} P_j. \tag{12}$$

We also introduce:

Discrete start time ($DS_{i,k}(n)$). This is the start time of the nth (i,k) job in the discrete network, i.e., the time at which the nth (i,k) job is scheduled for processing in the (discrete) job shop, under the RFSA defined below.

Discrete completion time ($DC_{i,k}(n)$). This is the completion time of the nth (i,k) job in the discrete network. In particular,

$$DC_{i,k}(n) = DS_{i,k}(n) + p_{i,k}. \tag{13}$$

Fluid start time ($FS_{i,k}(n)$). This is the start time of the nth (i,k) job in the fluid relaxation (for the makespan objective), and is given by

$$FS_{i,k}(1) = 0, \tag{14}$$

$$FS_{i,k}(n) = FS_{i,k}(n-1) + \frac{C_{\max}}{n_i}, \qquad n > 1. \tag{15}$$

Fluid completion time ($FC_{i,k}(n)$). This is the completion time of the nth (i, k) job in the fluid relaxation (for the makespan objective), and is given by

$$FC_{i,k}(n) = FS_{i,k}(n) + \frac{C_{\max}}{n_i}. \tag{16}$$

Nominal start time ($NS_{i,k}(n)$). The nominal start time of the nth (i,k) job is defined as follows:

$$NS_{i,1}(n) = FS_{i,1}(n), \tag{17}$$

$$NS_{i,k}(1) = DS_{i,k-1}(1) + p_{i,k-1}, \qquad k > 1, \tag{18}$$

$$NS_{i,k}(n) = \max\left\{NS_{i,k}(n-1) + \frac{C_{\max}}{n_i}, DS_{i,k-1}(n) + p_{i,k-1}\right\}, \qquad n, k > 1. \tag{19}$$

Nominal completion time ($NC_{i,k}(n)$). The nominal completion time of the nth (i, k) job is defined as follows:

$$NC_{i,k}(n) = NS_{i,k}(n) + \frac{C_{\max}}{n_i}. \tag{20}$$

As a convention, we define $DS_{i,0}(n) = DC_{i,0}(n) = 0$, for all i, n. Similarly, we define $p_{i,0} = 0$ for all i, n.

Each job in the discrete network is assigned a *status* at each of its stages, which is one of *not available, available, in progress*, or *departed*. The status of the nth (i,k) job at time t is

- *Not available*, if $0 \leqslant t < \max\{DC_{i,k-1}(n), NS_{i,k}(n)\}$.
- *Available*, if $\max\{DC_{i,k-1}(n), NS_{i,k}(n)\} \leqslant t < DS_{i,k}(n)$.
- *In progress*, if $DS_{i,k}(n) \leqslant t < DC_{i,k}(n)$.
- *Departed*, if $t \geqslant DC_{i,k}(n)$.

Description of the RFSA. Scheduling decisions in the discrete network are made at well-defined *scheduling epochs*. Scheduling epochs for machine σ_j are instants of time at which either some job completes service at σ_j and there is at least one *available* job at σ_j, or some job *becomes* available at an idle machine σ_j. Suppose machine σ_j has a scheduling epoch at time t. Among all the *available* jobs at machine σ_j, the RFSA schedules the one with the *smallest nominal start time*. This scheduling decision, in turn, determines the nominal start time of this job at its next stage. The key difference between the FSA and the RFSA is thus in the definition of *available* jobs: Under the FSA, job n of class (i,k) is declared as available at time $DC_{i,k-1}(n)$, while under the RFSA it is declared available at $\max\{DC_{i,k-1}(n), NS_{i,k}(n)\}$. In other words, under the RFSA, *no job is scheduled to start prior to its nominal start time*. As in the case of the FSA, it is easy to see inductively that the RFSA is well defined.

Elementary Results for the RFSA. The following theorems relate the fluid and discrete completion times of a job when the discrete schedule is computed using the RFSA.

Theorem 3. *Let $DC_{i,k}(n)$ be the completion time of the nth (i,k) job in the discrete schedule computed by the RFSA, and let $FC_{i,k}(n)$ be its completion time in the fluid relaxation. Then,*

$$DC_{i,k}(n) \leqslant FC_{i,k}(n) + \sum_{l=1}^{k}(2P_{\sigma_l^i} + U_{\sigma_l^i}) \tag{21}$$

and

$$DC_{i,k}(n) \geqslant FC_{i,k}(n-1). \tag{22}$$

Proof. Equation (21) was proved in Bertsimas and Sethuraman (2002) under the FSA. A careful examination of the proof in Bertsimas and Sethuraman (2002) reveals that the same argument holds for the RFSA as well.

Equation (22) follows by the definition of the RFSA as follows:

$$\begin{aligned} DS_{i,k}(n) &\geqslant NS_{i,k}(n)\\ &\geqslant NS_{i,k}(n-1) + \frac{C_{\max}}{n_i}\\ &\geqslant FS_{i,k}(n-1) + \frac{C_{\max}}{n_i}\\ &= FC_{i,k}(n-1). \end{aligned}$$

Thus, $DC_{i,k}(n) \geqslant DS_{i,k}(n) \geqslant FC_{i,k}(n-1)$. □

4.2. Algorithm FSA-HC

In this section, we provide a complete description of algorithm FSA-HC. Its main idea is as follows.

Suppose the optimal fluid solution has R pieces. Following Lemma 2 and Proposition 2, we first construct a modified fluid solution with R pieces which has integral queue lengths at the breakpoints and has a cost which exceeds the optimal cost by at most $O(N)$. Note that the initial queue lengths of the modified solution are assumed to be $\lfloor n_i \rfloor + RJ_{\max}$, $i = 1, 2, \ldots, I$, if the original initial queue length is n_i. This means that we introduce for each class i additional $RJ_{\max}$ virtual jobs.

Let T^i denote the time at which piece i ends for this modified fluid solution (also the time at which piece $(i+1)$ begins), and let $T^0 = 0$ be the time origin. Thus, piece i starts at time T^{i-1} and ends at time T^i, for $1 \leqslant i \leqslant R$. We discretize each piece separately using the RFSA described earlier in this section. Specifically, for each piece $r = 1, 2, \ldots, R$ we formulate a makespan scheduling problem on a suitably defined input and apply the RFSA. In this way we obtain times $\widehat{T}^0 = 0, \widehat{T}^1, \widehat{T}^2, \ldots, \widehat{T}^R$ such that the vector of queue lengths at $\widehat{T}^i$ in the discrete network is exactly the same as the vector of queue lengths at T^i in the modified solution to the fluid relaxation. We then evaluate and compare the cost of each piece and show that the discretization error accumulated over all the R pieces is asymptotically negligible compared to the total fluid cost.

The following definitions will be needed in a formal description of the FSA-HC.

- **Length of piece r.** $L^r = T^r - T^{r-1}$.
- **Fluid queue length.** $x_{i,k}^r$ denotes the queue length of (i,k) jobs in the modified solution to the fluid relaxation (according to Proposition 2). Specifically, if the queue lengths of the optimal fluid solutions at the breakpoints are $NX_{i,k}^r$, then

$$\begin{aligned} x_{i,1}^r &= \lfloor NX_{i,1}^r \rfloor + (R-r)J_{\max},\\ x_{i,k}^r &= \lfloor NX_{i,k}^r \rfloor, \qquad k = 2, \ldots, J_i. \end{aligned} \tag{23}$$

Recall from Theorem 1 that the optimal fluid solution depends linearly on N and, as a result, the values $X_{i,k}^r$ depend only on α_i. Thus, the queue lengths $x_{i,k}^r$ depend linearly on N.

- **Number of jobs processed in piece r.** $y_{i,k}^r$ denotes the number of (i,k) jobs processed by the modified fluid solution in piece r; clearly, $y_{i,k}^r = \mu_{i,k} u_{i,k}^r L^r$, where $u_{i,k}^r$ is the constant control on (i,k) jobs for piece r.

We need an additional definition before we can describe the FSA-HC. In the makespan objective, the fluid solution is constant, and all of the jobs required to be processed are in their corresponding first stages. The latter property is true for the first piece in the holding-cost objective, but may be violated for the subsequent pieces. Moreover, in the makespan objective, the fluid solution starts with a number of class (i,k) jobs and drives them to zero within a single piece in the solution. Hence, we need to enhance our definition of "job types." This naturally leads to the definition of auxiliary variables discussed next.

We define class $(i,k,l;r)$ jobs that represent those type i jobs that move from stage k to stage l during the rth piece of the fluid relaxation. Let z_{ikl}^r be the number of such jobs. For convenience, we define z_{ikk}^r to be the number of type i jobs that remain at stage k during piece r. We also define class $(i,k,E;r)$ jobs that represent those type i jobs that start at stage k, but depart from the network during the rth piece of the fluid relaxation. Let z_{ikE}^r be the number of such jobs. We next illustrate the computation of z_{ikl}^r in an example, to motivate a formal algorithm to compute these quantities that follow next.

Consider the following example (see Figure 2): There are four machines and two types of jobs. Type 1 jobs require service at machines 1, 2, 3, and 4 in that order; Type 2 jobs require service at machines 4, 3, 2, and 1 in that order. The processing requirements and the holding-cost rates at the various stages for each job type are shown in Table 1. Suppose we have 250 jobs of Type 1 and 500 jobs of Type 2 initially. The fluid solution shown in Table 2, while not optimal, has objective function value close to the optimal fluid cost. Moreover, the vector of queue lengths at the end points of each piece is integral. The auxiliary variables associated with this fluid solution are shown in Table 3. In

Figure 2. A four-station network.

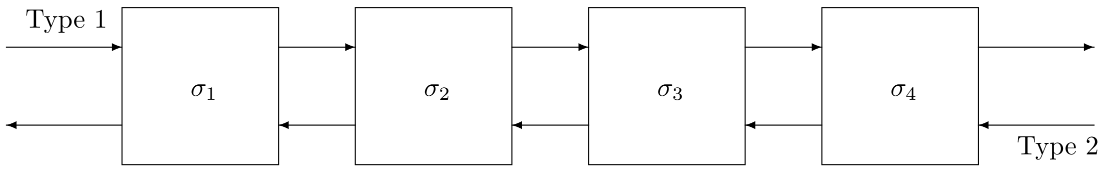


Table 3, the entry E refers to the external environment: This just indicates that the corresponding jobs leave the network.

The auxiliary variables define the requirements for each piece and capture exactly the dynamics of the rth piece of the fluid solution. The algorithm to compute the auxiliary variables proceeds as follows. It will be useful to define some quantities in describing the algorithm. The *outflow* of class (i, k) in piece r, outflow$(i, k; r)$, is the number of type i jobs that were in stage k at T^{r-1}, but ended up in stage $k' > k$ at T^r. (Recall that if a job is waiting at stage k, it has undergone processing up to its $(k-1)$st stage.) By definition, outflow$(i, k; r)$ is at most $x_{i,k}^{r-1}$, the number of (i, k) jobs at T^{r-1}. Also, outflow$(i, k; r)$ cannot exceed the total number of jobs whose kth stage is processed in piece r. From these two observations, we have

$$\text{outflow}(i, k; r) = \min\left\{\sum_{p=1}^{k}(x_{i,p}^{r-1} - x_{i,p}^{r}),\, x_{i,k}^{r-1}\right\}.$$

Similarly, let inflow$(i, k; r)$ be the number of $(i, k; r)$ jobs that were in some stage $k' < k$ at time T^{r-1}, but ended up in stage k at time T^r. Again, by definition, we have

$$\text{inflow}(i, k; r) - \text{outflow}(i, k; r) = x_{i,k}^{r} - x_{i,k}^{r-1}.$$

Computing the auxiliary variables z_{ikl}^r (for $l \leqslant k$) now reduces to the problem of allocating the outflow$(i, k; r)$ to the stages $l \geqslant k$ appropriately. We do this one stage at a time, starting from $l = k$: In this case, z_{ikl}^r is just the number of jobs that "stayed" at stage k during $[T^{r-1}, T^r)$, which is exactly $x_{i,k}^{r-1} - \text{outflow}(i, k; r)$. For $l > k$, clearly, z_{ikl}^r cannot exceed outflow$(i, k; r)$ or inflow$(i, l; r)$. For $l = k + 1$, we set

$$z_{ikl}^r = \min\{\text{outflow}(i, k; r), \text{inflow}(i, l; r)\},$$

and subtract z_{ikl}^r from both outflow$(i, k; r)$ and inflow$(i, l; r)$. The latter step is to account for the outflow of z_{ikl}^r jobs into stage k, and the inflow of z_{ikl}^r jobs into stage l. Thus, the modified definition of outflow$(i, k; r)$ reflects the remaining amount of jobs that need to flow out of stage k, which is then used in computing the z_{ikl}^r for $l = k + 2$, etc.

Table 1. Holding costs and processing times.

	Type 1	Type 2
Holding costs	(4, 1, 2, 1)	(4, 1, 2, 1)
Processing times	(1, 8, 4, 2)	(2, 4, 1, 8)

A formal description of the algorithm to compute the auxiliary variables z_{ikl}^r using the vector of fluid queue lengths at time T^{r-1} and T^r is shown in Figure 3.

The discretization algorithm for the holding-cost objective can thus be described as follows:

Fluid Synchronization Algorithm for Holding Costs (FSA-HC)

Step 1. Solve the fluid-control problem, and obtain the optimal fluid relaxation. This can be accomplished by applying the algorithm of Luo and Bertsimas (1999). The optimal fluid relaxation has R pieces, and breakpoints $0, T^1, \ldots, T^R$ and the corresponding lengths of the pieces are $L^r = T^{r+1} - T^r$.

Step 2. Following Proposition 2 and starting with the optimal fluid solution, construct a new piecewise-linear fluid solution with R pieces, such that the queue lengths at the breakpoints are integral.

Step 3. For $r = 1, 2, \ldots, R$:

(a) Define new class $(i, k, l; r)$ jobs defined to be jobs of type i that move from stage k to stage l during the rth piece, and new class (ikE, r) jobs defined to be jobs of type i that start at stage k but depart from the network during the rth piece.

(b) Compute the number z_{ikl}^r and z_{ikE}^r of such jobs by applying the algorithm shown in Figure 3.

(c) Apply the RFSA on the new network with z_{ikl}^r $(i, k, l; r)$ jobs and z_{ikE}^r (ikE, r) jobs. The new breakpoints will now be $0, \widehat{T}^1, \ldots, \widehat{T}^R$ and the corresponding lengths of the pieces will be $\widehat{L}^r = \widehat{T}^{r+1} - \widehat{T}^r$.

In essence, we view each piece r of the modified fluid solution as a job-shop scheduling problem with a makespan objective, but with new classes $(i, k, l; r)$, $(i, k, E; r)$. L^r plays the role of $C_{\max}$, job types are indexed by $(i, k, l; r)$ and $(i, k, E; r)$, and the auxiliary variables z_{ikl}^r play the role of the n_i.

We note that the jobs in the discretized solution may no longer be processed in FCFS order. To see this, consider

Table 2. A near-optimal fluid solution.

Type 1	Type 2
(250, 0, 0, 0)	(500, 0, 0, 0)
(0, 218, 0, 32)	(375, 125, 0, 0)
(0, 136, 0, 114)	(0, 406, 0, 0)
(0, 104, 0, 0)	(0, 370, 0, 0)
(0, 0, 0, 0)	(0, 250, 0, 0)
(0, 0, 0, 0)	(0, 0, 0, 0)

Table 3. Auxiliary variables z^r_{ikl} for the fluid solution of Table 2.

	Type	Origin Stage	Destination Stage	Number of Jobs
Piece 1	1	1	2	218
	1	1	4	32
	2	1	2	125
Piece 2	1	2	4	82
	2	1	2	375
	2	2	E	94
Piece 3	1	2	E	32
	1	4	E	114
	2	2	E	36
Piece 4	1	2	E	104
	2	2	E	120
Piece 5	2	2	E	250

a solution in which $z^r_{ikl} \gg z^r_{ik'l}$ for some k, k' such that $k < k' < l$. In this case, in our interpretation of the fluid solution, type $(i, k, l; r)$ jobs are processed at a much faster rate than the type $(i, k', l; r)$ jobs, and so it is possible for some job at stage k to reach the destination l prior to some job at stage k'.

4.3. Analysis of the FSA-HC

In this section, we calculate the cost of the discrete schedule the FSA-HC produces and compare it to that of the fluid relaxation. Our analysis proceeds on a job-by-job basis. The outline of the analysis is as follows.

1. We focus on $(i, k, l; r)$ jobs in the rth piece, and we evaluate the cost of these jobs in the discrete network and in the fluid relaxation.
2. We find an expression for an upper bound on the difference between the cost accumulated in rth piece and the corresponding cost of the fluid solution in the same piece.
3. We show that the total error, summed over all pieces, is asymptotically negligible compared to the cost of the fluid solution.

Figure 3. Computing the values of the auxiliary variables z^r_{ikl}.

```
For $i = 1, 2, \ldots, I$ :
  For $k = 1, 2, \ldots, J_i$ :
        $\text{outflow}(i, k; r) = \min\left\{\sum_{p=1}^{k}(x^{r-1}_{i,p} - x^r_{i,p}),\, x^{r-1}_{i,k}\right\}.$
        $\text{inflow}(i, k; r) = x^r_{i,k} - x^{r-1}_{i,k} + \text{outflow}(i, k; r).$
      $\text{outflow}(i, E; r) = 0;\ \text{inflow}(i, E; r) = \sum_{p=1}^{J_i}(x^{r-1}_{i,p} - x^r_{i,p}).$
  For $k = 1, 2, \ldots, J_i$ :
    $z^r_{ikk} = x^{r-1}_{i,k} - \text{outflow}(i, k; r).$
    For $l = k+1, k+2, \ldots, J_i, E$:
        $z^r_{ikl} = \min\{\text{outflow}(i, k; r), \text{inflow}(i, l; r)\}.$
          $\text{outflow}(i, k; r) := \text{outflow}(i, k; r) - z^r_{ikl};$
          $\text{inflow}(i, l; r) := \text{inflow}(i, l; r) - z^r_{ikl};$
```

Lemma 4. *The cost of the rth piece of the fluid relaxation is equal to*

$$C^r_f = \sum_{i=1}^{I}\sum_{k=1}^{J_i}\sum_{l=k}^{J_i}\left(\frac{w_{i,k} z^r_{ikl} L^r}{2} + \frac{w_{i,l} z^r_{ikl} L^r}{2}\right). \tag{24}$$

Proof. To evaluate the cost of the rth piece in the fluid network, we observe that

- The inventory level of $(i, k, l; r)$ jobs at stage k *decreases linearly* from z^r_{ikl} to zero.
- The inventory level of $(i, k, l; r)$ jobs at stage l *increases linearly* from zero to z^r_{ikl}.
- All of the intermediate stages (if any) have zero inventory level for $(i, k, l; r)$ jobs.

Thus, in the rth piece of the fluid solution, the cost incurred by $(i, k, l; r)$ jobs in stage k is

$$\frac{w_{i,k} z^r_{ikl} L^r}{2},$$

and the cost incurred by (i, k, l, r) jobs in stage l is

$$\frac{w_{i,l} z^r_{ikl} L^r}{2}.$$

Observing that jobs of type $(i, k, l; r)$ do not incur cost at any other stage, we see that the cost of type $(i, k, l; r)$ jobs is

$$\left(\frac{w_{i,k} z^r_{ikl} L^r}{2} + \frac{w_{i,l} z^r_{ikl} L^r}{2}\right). \tag{25}$$

Summing Equation (25) over all possible job types, we obtain Equation (24). □

We now evaluate the cost of type i jobs in the discretized solution corresponding to piece r. For convenience, we shift the origin so that $T^{r-1} = 0$, and so $T^r = L^r$.

Lemma 5. *The cost of the rth piece in the discrete network is at most*

$$C^r_d = \sum_{i=1}^{I}\sum_{k=1}^{J_i}\sum_{l=k}^{J_i} C^r_d(i, k, l), \tag{26}$$

where

$$\begin{aligned} &C^r_d(i,k,l)\\ &\quad = w_{i,k} L^r\left(\frac{z^r_{ikl}+1}{2}\right) + w_{i,k} z^r_{ikl}(2P_{\max}+U_{\max})\\ &\qquad + w_{i,l} L^r\left(\frac{z^r_{ikl}+1}{2}\right) + w_{i,l} z^r_{ikl} J_{\max}(2P_{\max}+U_{\max})\\ &\qquad + \sum_{p=k+1}^{l-1} w_{i,p}\big(L^r + z^r_{ikl}(p-k+1)(2P_{\max}+U_{\max})\big). \end{aligned} \tag{27}$$

Proof. The cost of the rth piece in the discrete network can be computed as follows. We focus on jobs of type $(i, k, l; r)$, such that $z^r_{ikl} > 0$. Otherwise, the cost contribution of the type $(i, k, l; r)$ is zero. For convenience, we renumber these jobs, if necessary, so that the jobs of type

$(i, k, l; r)$ are numbered $1, 2, \ldots, z^r_{ikl}$. For $k \leqslant p \leqslant l$, recall that $DC_{ikl,p}(n)$ is the completion time of the nth type $(i, k, l; r)$ job at stage p. (We suppress r from $DC_{ikl,p}(n)$ to simplify the already-congested notation.) Clearly, the cost of type $(i, k, l; r)$ jobs is given by

$$\sum_{n=1}^{z^r_{ikl}} w_{i,k} DC_{ikl,k}(n) + \sum_{n=1}^{z^r_{ikl}} w_{i,l}(\hat{L}_r - DC_{ikl,l-1}(n)) + \sum_{n=1}^{z^r_{ikl}} \sum_{p=k+1}^{l-1} w_{i,p}(DC_{ikl,p}(n) - DC_{ikl,p-1}(n)). \tag{28}$$

We next evaluate each of the three terms in Equation (28) separately. First, consider the last term in Equation (28). Using Equation (21) for type $(i, k, l; r)$ jobs, we conclude that

$$DC_{ikl,p}(n) \leqslant FC_{ikl,p}(n) + (p-k+1)(2P_{\max} + U_{\max}), \tag{29}$$

for $k \leqslant p \leqslant l$. Also, by definition, for any p such that $k \leqslant p \leqslant l$,

$$FC_{ikl,p}(n) = n\frac{L^r}{z^r_{ikl}}. \tag{30}$$

Combining Equations (29) and (30), we obtain

$$DC_{ikl,p}(n) \leqslant n\frac{L^r}{z^r_{ikl}} + (p-k+1)(2P_{\max} + U_{\max}), \quad k \leqslant p \leqslant l. \tag{31}$$

From Equation (22), we obtain

$$DC_{ikl,p}(n) \geqslant (n-1)\frac{L^r}{z^r_{ikl}}. \tag{32}$$

Using Equations (31) and (32), we obtain

$$\begin{aligned}
&\sum_{n=1}^{z^r_{ikl}} \sum_{p=k+1}^{l-1} w_{i,p}\big(DC_{ikl,p}(n) - DC_{ikl,p-1}(n)\big)\\
&\quad\leqslant \sum_{n=1}^{z^r_{ikl}} \sum_{p=k+1}^{l-1} w_{i,p}\left(n\frac{L^r}{z^r_{ikl}} - (n-1)\frac{L^r}{z^r_{ikl}} + (p-k+1)(2P_{\max}+U_{\max})\right)\\
&\quad= \sum_{n=1}^{z^r_{ikl}} \sum_{p=k+1}^{l-1} w_{i,p}\left(\frac{L^r}{z^r_{ikl}} + (p-k+1)(2P_{\max}+U_{\max})\right)\\
&\quad= \sum_{p=k+1}^{l-1} w_{i,p}\big(L^r + z^r_{ikl}(p-k+1)(2P_{\max}+U_{\max})\big).
\end{aligned}$$

We next consider the second term of Equation (28). From Theorem 9 in Bertsimas and Sethuraman (2002), we know that the discretization of the rth piece finishes at time $\hat{L}^r$, such that

$$\hat{L}^r \leqslant L^r + J_{\max}(2P_{\max} + U_{\max}). \tag{33}$$

Using Equations (33) and (32), we obtain

$$\begin{aligned}
&\sum_{n=1}^{z^r_{ikl}} w_{i,l}(\hat{L}_r - DC_{ikl,l-1}(n))\\
&\quad\leqslant \sum_{n=1}^{z^r_{ikl}} w_{i,l}\left(L_r + J_{\max}(2P_{\max}+U_{\max}) - \frac{(n-1)L^r}{z^r_{ikl}}\right)\\
&\quad= w_{i,l}z^r_{ikl}L^r + w_{i,l}z^r_{ikl}J_{\max}(2P_{\max}+U_{\max}) - w_{i,l}\frac{L^r}{z^r_{ikl}}\sum_{n=1}^{z^r_{ikl}}(n-1)\\
&\quad= w_{i,l}L^r\left(\frac{z^r_{ikl}+1}{2}\right) + w_{i,l}z^r_{ikl}J_{\max}(2P_{\max}+U_{\max}).
\end{aligned} \tag{34}$$

Finally, we consider the first term of Equation (28). Using Equation (31), we obtain

$$\begin{aligned}
&\sum_{n=1}^{z^r_{ikl}} w_{i,k}DC_{ikl,k}(n)\\
&\quad\leqslant \sum_{n=1}^{z^r_{ikl}} w_{i,k}\left(\frac{nL^r}{z^r_{ikl}} + (2P_{\max}+U_{\max})\right)\\
&\quad= w_{i,k}L^r\left(\frac{z^r_{ikl}+1}{2}\right) + w_{i,k}z^r_{ikl}(2P_{\max}+U_{\max}).
\end{aligned} \tag{35}$$

The cost of type $(i, k, l; r)$ jobs in the rth piece in the discrete network is obtained by adding Equations (33)–(35), which yields Equation (27). $\square$

We are now ready to prove that the FSA-HC yields an asymptotically optimal schedule.

Theorem 6. *Consider a job-shop scheduling problem with I job types and J machines $\sigma_1, \sigma_2, \ldots, \sigma_J$. Given initially $\alpha_i N$ jobs of type $i = 1, 2, \ldots, I$, the* FSA-HC *produces a schedule with cost $Z_{\mathrm{D}}(N)$ such that*

$$Z_{\mathrm{D}}(N) \leqslant Z_{\mathrm{F}}(N) + O(N). \tag{36}$$

In particular,

$$\frac{Z_{\mathrm{D}}(N)}{Z_{\mathrm{JS}}(N)} \leqslant 1 + O\left(\frac{1}{N}\right), \tag{37}$$

and thus

$$\frac{Z_{\mathrm{D}}(N)}{Z_{\mathrm{JS}}(N)} \to 1, \tag{38}$$

as

$N \to \infty$.

Proof. Let $Z_{\mathrm{F}}(N)$ be the cost of the optimal fluid solution. Let $Z'_{\mathrm{F}}(N)$ be the cost of the modified fluid solution after applying the construction of Proposition 2.

From Equations (24) and (26), we have

$$C_d^r - C_f^r \leqslant \sum_{i=1}^{I}\sum_{k=1}^{J_i}\sum_{l=k}^{J_i}\left\{\frac{w_{i,k}L^r}{2} + w_{i,k}z_{ikl}^r(2P_{\max}+U_{\max}) + \frac{w_{i,l}L^r}{2} + w_{i,l}z_{ikl}^r J_{\max}(2P_{\max}+U_{\max}) + \sum_{p=k+1}^{l-1} w_{i,p}\big(L^r + z_{ikl}^r(p-k+1) \cdot (2P_{\max}+U_{\max})\big)\right\}.$$

From the proof of part (a) of Theorem 1 and from Proposition 2, the terms z_{ikl}^r and L^r all vary linearly with N. Thus,

$$C_d^r - C_f^r \leqslant AN,$$

for some (large enough) constant A. Thus,

$$Z_{\mathrm{D}}(N) - Z'_{\mathrm{F}}(N) = \sum_{r=1}^{R}(C_d^r - C_f^r) \leqslant ARN,$$

which establishes $Z_{\mathrm{D}}(N) \leqslant Z'_{\mathrm{F}}(N) + O(N)$, because R is also a constant. From Proposition 2 we have $Z'_{\mathrm{F}}(N) \leqslant Z_{\mathrm{F}}(N) + O(N)$, and thus $Z_{\mathrm{D}}(N) \leqslant Z_{\mathrm{F}}(N) + O(N)$.

From Theorem 1(b), we have $Z_{\mathrm{F}}(N) \leqslant Z_{\mathrm{JS}}(N) + O(N)$. Thus,

$$\frac{Z_{\mathrm{D}}(N)}{Z_{\mathrm{JS}}(N)} \leqslant \frac{Z_{\mathrm{F}}(N)+O(N)}{Z_{\mathrm{F}}(N)-O(N)} \leqslant \frac{CN^2+O(N)}{CN^2-O(N)} = 1 + O\left(\frac{1}{N}\right),$$

from which (38) follows. □

Remark. We note that any algorithm that uses the fluid relaxation will incur $O(N)$ error in the worst case. For example, consider a single machine with N jobs, with $w_i = 1$. The cost of an optimal discrete schedule is $N(N+1)/2$, but the optimal fluid cost is $N^2/2$.

5. COMPUTATIONAL RESULTS

In this section, we report computational results for the objective of minimizing weighted completion times. This is the special case of the holding-cost objective in which the weights are all 1; i.e., $w_{i,k} = 1$ for all i, k. For our computational study, we chose a subset of 20 instances from the OR library (http://mscmga.ms.ic.ac.uk/info.html); the results shown on these instances are representative of the results obtained for our algorithm in general. The results reported in Table 4 are for these 20 benchmarks. The number of machines ranged from 5 to 20, and the number of job types ranged from 5 to 50. All execution times were under two minutes on a SUN workstation. The first set of results aim to illustrate experimentally the effect of asymptotic optimality. The second set of results compares the proposed algorithm and several commonly used heuristic methods.

Asymptotic Optimality

For each benchmark, we assume that each job type has N jobs in their first stage, and we report results for $N = 1$, $N = 2$, $N = 5$, $N = 10$, $N = 100$, and $N = 500$. The lower bound based on the fluid relaxation, $Z_{\mathrm{F}}(1)$, is shown in the second column, and is valid for $N = 1$; the lower bound for $N = n$ is $n^2 Z_{\mathrm{F}}(1)$. The subsequent columns report the

Table 4. Performance of the FSA-HC on job-shop instances in the OR-library.

	Z_{F}	$\frac{Z_{\mathrm{D}}-N^2Z_{\mathrm{F}}}{N^2Z_{\mathrm{F}}}$					
Benchmark	$(N=1)$	$N=1$	$N=2$	$N=5$	$N=10$	$N=100$	$N=500$
abz5	4,154.54	1.731	1.663	1.302	0.876	0.087	0.014
abz6	3,116.64	1.689	1.437	1.101	0.823	0.093	0.011
ft06	109.06	2.111	1.813	1.763	1.106	0.147	0.025
ft10	2,740.45	2.117	1.987	1.671	1.037	0.436	0.022
ft20	9,493.73	1.989	1.700	1.481	1.002	0.210	0.019
la01	2,837.45	1.965	1.573	1.320	1.129	0.313	0.016
la02	2,802.26	1.270	1.113	0.912	0.614	0.128	0.023
la03	2,471.49	1.961	1.672	1.475	1.131	0.254	0.014
la04	2,473.30	2.114	1.842	1.386	1.141	0.195	0.014
la05	2,501.91	1.320	1.219	1.214	1.067	0.411	0.016
la06	5,732.63	2.630	2.315	2.059	1.254	0.193	0.008
la10	5,998.61	1.767	1.645	1.323	1.006	0.255	0.011
la11	10,000.16	2.749	2.119	1.346	1.043	0.197	0.009
la13	9,715.28	2.643	2.216	1.414	1.095	0.351	0.011
la15	10,097.26	2.891	2.148	1.730	1.533	0.471	0.021
la17	2,983.00	2.653	2.351	1.985	1.438	0.336	0.021
la19	3,072.54	2.717	2.185	1.754	1.324	0.372	0.019
orb01	3,013.75	2.018	1.811	1.439	1.007	0.221	0.007
orb03	2,831.91	2.005	1.837	1.601	1.105	0.119	0.016
orb05	2,719.82	1.882	1.473	1.338	0.903	0.143	0.009

value of the *relative error*,

$$\frac{Z_D(N)-Z_F(N)}{Z_F(N)}=\frac{Z_D(N)-N^2Z_F(1)}{N^2Z_F(1)}.$$

(The results in terms of a more familiar measure—the ratio of the cost of the heuristic schedule and the optimal schedule—are obtained by adding one to the relative error.) From the results reported in Table 4, we observe that the relative error does converge to zero as N increases, as predicted by Theorem 6. The relative error is of the order of 100% for $N=10$, 40% for $N=100$, and 1% for $N=500$. Compared with the asymptotics for the makespan objective reported in Bertsimas and Sethuraman (2002) for the same problems, we observe that for the makespan objective the corresponding errors are about 10% for $N=10$, 1% for $N=100$, 0.05% for $N=500$; i.e., we need perhaps an order of magnitude of more jobs in the system to obtain the same accuracy. The relative error is $O(1/N)$, but the hidden constant is much higher for the holding-cost objective compared to makespan. This is not too surprising, as the number of pieces R will enter in the constant. Note also that the relative error reported here is the error incurred by the algorithm FSA-HC with respect to the fluid lower bound; the performance of the FSA-HC compared to the true value will be at least as good, usually better.

Comparison with Other Heuristic Methods

We next present results comparing the relative errors of the algorithm FSA-HC with those of several simple heuristic methods. The results are reported for exactly the same benchmarks. Each heuristic method is employed in a nonpreemptive manner and essentially identifies the task to be executed next by a machine whenever that machine needs to make a scheduling decision. (In the queueing literature, these are also referred to as "priority rules" or "priority policies.")

We tested eight common heuristic methods, listed below.

(a) Shortest Task Time (STT): schedule the *task* with the smallest processing time.

(b) Longest Task Time (LTT): schedule the *task* with the largest processing time.

(c) Shortest Processing Time (SPT): schedule the *task* with the smallest total processing time. In other words, a type i job receives priority over a type i' job if

$$\sum_{k=1}^{J_i} p_{i,k} < \sum_{k=1}^{J_{i'}} p_{i',k}.$$

(d) Longest Processing Time (LPT): schedule the *task* with the largest total processing time.

(e) Shortest Remaining Processing Time (SRPT): schedule the *task* with the smallest remaining job processing time. In other words, class (i,l) job receives priority over (i',l') job if

$$\sum_{k=l}^{J_i} p_{i,k} < \sum_{k=l'}^{J_{i'}} p_{i',k}.$$

(f) Longest Remaining Processing Time (LRPT): schedule the *task* with the largest remaining total processing time.

(g) Last Buffer First Serve (LBFS): schedule the *task* with the smallest remaining number of subsequent tasks.

(h) First Buffer First Serve (FBFS): schedule the *task* with the largest remaining number of subsequent tasks.

We chose these heuristic methods primarily because they are easy to implement and have roughly the same implemenation complexity as our algorithm (FSA-HC). (Strictly speaking, we need to solve the fluid relaxation once to implement FSA-HC; the complexity of solving the fluid relaxation exactly is still open. However, the fluid relaxation does not have to be re-solved for various values of N, so we ignore this cost.)

Tables 5 through 8 display the relative errors of FSA-HC and those of the eight heuristic methods for various values of N. For $N=500$, the fluid-based algorithm emerges as a clear winner; but for smaller values of N, there is no clear winner. For $N=1$, FSA-HC yields the best result for the six benchmarks abz5, abz6, ft10, orb01, orb03, and orb05; among the remaining benchmarks, SRPT gives the best results for 8, LBFS for 3, SPT for 2, and STT for 1. For the next-higher value of $N=10$, the SRPT heuristic emerges as a clear winner in all but three benchmarks (la02, la03, and la05). Even for these benchmarks, SRPT performs very competitively. While the quality of the schedule found by FSA-HC is not poor, it is still far from the performance of many of these heuristic methods. One reason for this behavior is that the number of jobs in the system is not large enough to offset the "idleness" introduced in the system as a result of discretizing each piece separately. For $N=100$, the fluid-based algorithm begins to perform competitively with the best heuristic for many benchmarks; still, the performance of SRPT is impressive here. As mentioned earlier, for $N=500$, the fluid-based algorithm outperforms all of the tested heuristic methods by a substantial margin. In this case (and hence for larger values of N as well), the artificial idleness introduced by discretizing each piece separately, and the costs associated with it, are negligible in relation to the optimal fluid cost itself. As expected, the relative error of FSA-HC decreases with N and appears to converge to zero reasonably quickly.

It is also interesting to observe that the performance of the eight heuristic methods stabilizes fairly quickly. The best heuristic method among the chosen eight methods for any given benchmark is the same, whether $N=100$ or $N=500$. Specifically, SRPT is the best heuristic method for the benchmarks abz5, ft20, la01, la04, la10, la13, la19, and orb01; SPT is the best heuristic method for the benchmarks abz6, ft06, la03, la06, la11, la15, and orb05; STT is the best heuristic method for ft10, la02, and orb03; and LBFS is the best heuristic method for the benchmarks la05 and la17. This trend continues for larger values of N, and more interestingly, the relative errors for larger N are virtually unchanged. The relative errors of the best heuristic method for each benchmark for $N=5{,}000$ and

Table 5. Comparison of the FSA-HC with simple dispatch rules ($N = 1$).

	Z_F	$\frac{Z_H - N^2 Z_F}{N^2 Z_F}$								
Benchmark	($N = 1$)	FSA-HC	STT	LTT	SPT	LPT	SRPT	LRPT	LBFS	FBFS
abz5	4,154.54	1.731	2.017	2.121	1.981	2.048	2.285	2.183	1.939	2.161
abz6	3,116.64	1.689	1.880	2.176	1.957	1.982	1.982	1.896	1.890	1.896
ft06	109.06	2.111	2.118	1.861	1.962	2.191	1.962	2.328	0.999	2.338
ft10	2,740.45	2.117	2.206	3.193	2.315	2.722	2.336	2.854	2.485	3.057
ft20	9,493.73	1.989	0.724	1.668	0.708	1.241	0.749	1.716	0.862	1.735
la01	2,837.45	1.965	1.130	1.356	1.128	1.321	1.092	1.305	1.108	1.354
la02	2,802.26	1.270	0.927	1.532	0.945	1.462	0.992	1.487	1.022	1.460
la03	2,471.49	1.961	1.079	1.742	0.947	1.265	0.941	1.261	1.283	1.185
la04	2,473.30	2.114	1.301	1.672	1.410	1.302	1.267	1.735	1.378	1.472
la05	2,501.91	1.320	0.986	1.067	0.936	1.058	0.918	1.204	0.917	1.212
la06	5,732.63	2.630	0.838	1.250	0.792	1.015	0.700	1.191	0.943	1.239
la10	5,998.61	1.767	0.755	1.228	0.743	1.064	0.711	1.237	0.817	1.155
la11	10,000.16	2.749	0.685	1.143	0.634	0.933	0.612	1.313	0.746	1.224
la13	9,715.28	2.643	0.647	0.984	0.606	0.959	0.570	1.229	0.603	1.239
la15	10,097.26	2.891	0.802	1.118	0.692	0.988	0.725	1.252	0.785	1.309
la17	2,983.00	2.653	1.712	1.700	1.588	1.679	1.595	1.772	1.540	1.794
la19	3,072.54	2.717	1.841	2.094	1.820	1.864	1.739	2.046	1.874	2.171
orb01	3,013.75	2.018	2.623	2.615	2.454	2.169	2.338	3.005	2.609	2.943
orb03	2,831.91	2.005	2.526	3.369	2.403	2.843	2.159	2.679	2.738	3.863
orb05	2,719.82	1.882	2.065	2.290	2.096	2.502	2.299	2.737	2.184	2.360

$N = 10,000$ are displayed in Table 9. We suspect that the reason for convergence of relative errors to limiting nonzero values lies in the fluid limit corresponding to a particular heuristic under consideration. Specifically, we conjecture that the limiting relative error is the ratio of the cost corresponding to the fluid limit of the heuristic scheduling rule divided by the optimal cost over all fluid limits (which is asymptotically achieved by our proposed FSA-HC policy). Proving this conjecture falls outside the scope of this paper.

Overall, it appears that even though these heuristic methods perform well for small to moderate values of N, they do not scale well. In particular, on these benchmarks they do not appear to yield asymptotically optimal schedules, as the relative error appears to stabilize at a value that is strictly positive.

These computations highlight the strengths and weaknesses of the FSA-HC that we summarize below:

1. For problems of low to moderate multiplicity ($N = 1 - 100$), the relative errors of the heuristic methods

Table 6. Comparison of the FSA-HC with simple dispatch rules ($N = 10$).

	Z_F	$\frac{Z_H - N^2 Z_F}{N^2 Z_F}$								
Benchmark	($N = 1$)	FSA-HC	STT	LTT	SPT	LPT	SRPT	LRPT	LBFS	FBFS
abz5	4,154.54	0.876	0.471	0.701	0.439	0.459	0.308	1.017	0.314	1.066
abz6	3,116.64	0.823	0.468	1.056	0.430	0.697	0.383	1.048	0.473	1.173
ft06	109.06	1.106	0.522	1.191	0.382	0.650	0.350	1.022	0.512	1.370
ft10	2,740.45	1.037	0.326	1.452	0.510	0.978	0.470	1.998	0.532	1.974
ft20	9,493.73	1.002	0.173	1.251	0.163	0.493	0.129	1.428	0.289	1.529
la01	2,837.45	1.129	0.412	0.795	0.266	0.613	0.246	0.879	0.314	0.815
la02	2,802.26	0.614	0.223	0.680	0.238	0.450	0.286	0.976	0.316	0.855
la03	2,471.49	1.131	0.281	1.038	0.278	0.514	0.284	0.729	0.423	0.865
la04	2,473.30	1.141	0.338	1.044	0.348	0.701	0.329	1.070	0.473	0.851
la05	2,501.91	1.067	0.309	0.696	0.194	0.429	0.211	0.876	0.201	0.913
la06	5,732.63	1.254	0.273	0.723	0.229	0.510	0.213	0.965	0.520	0.934
la10	5,998.61	1.006	0.242	0.719	0.176	0.541	0.106	0.984	0.300	0.890
la11	10,000.16	1.043	0.305	0.839	0.169	0.548	0.157	1.079	0.350	0.999
la13	9,715.28	1.095	0.281	0.686	0.178	0.478	0.122	1.047	0.252	1.067
la15	10,097.26	1.533	0.376	0.781	0.196	0.554	0.227	0.998	0.330	1.147
la17	2,983.00	1.438	0.365	1.011	0.438	0.534	0.294	1.042	0.252	1.102
la19	3,072.54	1.324	0.321	1.178	0.371	0.483	0.306	0.965	0.319	1.122
orb01	3,013.75	1.007	0.736	1.186	0.587	0.750	0.515	1.927	0.581	1.745
orb03	2,831.91	1.105	0.609	1.500	0.617	0.793	0.534	1.334	0.758	2.619
orb05	2,719.82	0.903	0.558	0.802	0.487	0.570	0.395	1.043	0.582	0.926

Table 7. Comparison of the FSA-HC with simple dispatch rules ($N = 100$).

	Z_F	$\frac{Z_H - N^2 Z_F}{N^2 Z_F}$								
Benchmark	($N = 1$)	FSA-HC	STT	LTT	SPT	LPT	SRPT	LRPT	LBFS	FBFS
abz5	4,154.54	0.087	0.290	0.497	0.158	0.158	0.075	0.989	0.094	0.946
abz6	3,116.64	0.093	0.315	1.037	0.189	0.386	0.228	0.979	0.225	1.082
ft06	109.06	0.147	0.378	0.990	0.111	0.458	0.121	0.904	0.286	1.307
ft10	2,740.45	0.436	0.175	1.309	0.352	0.644	0.252	1.818	0.309	1.892
ft20	9,493.73	0.210	0.140	1.200	0.110	0.429	0.072	1.410	0.233	1.497
la01	2,837.45	0.313	0.357	0.711	0.163	0.467	0.118	0.840	0.247	0.773
la02	2,802.26	0.128	0.148	0.575	0.154	0.336	0.212	0.934	0.236	0.805
la03	2,471.49	0.254	0.228	0.850	0.216	0.413	0.222	0.704	0.322	0.814
la04	2,473.30	0.195	0.290	0.976	0.280	0.581	0.260	1.011	0.332	0.783
la05	2,501.91	0.411	0.236	0.627	0.127	0.361	0.133	0.848	0.126	0.882
la06	5,732.63	0.193	0.225	0.683	0.168	0.443	0.190	0.942	0.472	0.911
la10	5,998.61	0.255	0.197	0.674	0.116	0.422	0.046	0.961	0.241	0.867
la11	10,000.16	0.197	0.271	0.812	0.116	0.489	0.120	1.057	0.314	0.979
la13	9,715.28	0.351	0.245	0.688	0.129	0.425	0.085	1.032	0.248	1.050
la15	10,097.26	0.471	0.339	0.750	0.156	0.479	0.199	0.972	0.346	1.135
la17	2,983.00	0.336	0.246	0.731	0.260	0.311	0.181	0.957	0.140	1.063
la19	3,072.54	0.372	0.195	1.021	0.247	0.328	0.157	0.916	0.200	1.048
orb01	3,013.75	0.221	0.538	1.084	0.410	0.447	0.315	1.825	0.395	1.560
orb03	2,831.91	0.119	0.410	1.153	0.424	0.573	0.411	1.226	0.556	2.471
orb05	2,719.82	0.143	0.429	0.742	0.243	0.409	0.303	0.861	0.379	0.768

are comparable to those of the proposed algorithm, and the best of the heuristic methods performs overall better than the proposed method.

2. For problems of moderate to high multiplicity ($N = 100 - 500$), the proposed algorithm outperforms the other methods.

3. For problems of very high multiplicity ($N = 5{,}000; 10{,}000$), the overperformance becomes more dramatic, as the relative error of the best of the heuristic methods stabilizes to a strictly positive value, whereas the proposed algorithm is asymptotically optimal (i.e., the relative error tends to zero).

Overall, for relatively large job-shop problems, the relative error of the schedule computed by FSA-HC is quite small. Given the high-quality solutions the algorithm finds, and given that the running time of the algorithm is linear in the number of jobs present, the FSA-HC represents, in our opinion, a simple and practical method for solving job-shop scheduling problems of moderate to high multiplicity.

Table 8. Comparison of the FSA-HC with simple dispatch rules ($N = 500$).

	Z_F	$\frac{Z_H - N^2 Z_F}{N^2 Z_F}$								
Benchmark	($N = 1$)	FSA-HC	STT	LTT	SPT	LPT	SRPT	LRPT	LBFS	FBFS
abz5	4,154.54	0.014	0.285	0.489	0.124	0.120	0.040	0.988	0.083	0.938
abz6	3,116.64	0.011	0.304	1.030	0.178	0.364	0.225	0.975	0.199	1.075
ft06	1,09.06	0.025	0.365	0.969	0.089	0.441	0.112	0.894	0.272	1.301
ft10	2,740.45	0.022	0.170	1.282	0.335	0.607	0.239	1.800	0.301	1.880
ft20	9,493.73	0.019	0.137	1.195	0.106	0.423	0.068	1.407	0.229	1.494
la01	2,837.45	0.016	0.352	0.705	0.156	0.454	0.109	0.836	0.244	0.769
la02	2,802.26	0.023	0.143	0.565	0.147	0.326	0.205	0.930	0.230	0.801
la03	2,471.49	0.014	0.223	0.851	0.210	0.406	0.214	0.701	0.313	0.809
la04	2,473.30	0.014	0.286	0.970	0.275	0.568	0.215	1.008	0.324	0.778
la05	2,501.91	0.016	0.230	0.651	0.123	0.355	0.126	0.846	0.120	0.879
la06	5,732.63	0.008	0.222	0.678	0.163	0.437	0.188	0.940	0.468	0.909
la10	5,998.61	0.011	0.196	0.671	0.111	0.414	0.041	0.959	0.245	0.864
la11	10,000.16	0.009	0.267	0.809	0.111	0.484	0.115	1.055	0.323	0.977
la13	9,715.28	0.011	0.241	0.684	0.124	0.420	0.081	1.032	0.205	1.049
la15	10,097.26	0.021	0.336	0.749	0.155	0.475	0.194	0.970	0.363	1.134
la17	2,983.00	0.021	0.236	0.718	0.244	0.292	0.177	0.948	0.132	1.059
la19	3,072.54	0.019	0.187	0.960	0.234	0.318	0.169	0.910	0.202	1.040
orb01	3,013.75	0.007	0.510	1.069	0.394	0.425	0.301	1.814	0.372	1.546
orb03	2,831.91	0.016	0.398	1.133	0.411	0.545	0.447	1.216	0.537	2.454
orb05	2,719.82	0.009	0.420	0.732	0.227	0.388	0.326	0.844	0.359	0.755

Table 9. The relative error of the best heuristic methods for large N.

Benchmark	Best Heuristic Method	RE ($N = 5{,}000$)	RE ($N = 10{,}000$)
abz5	SRPT	0.040	0.039
abz6	SPT	0.177	0.177
ft06	SPT	0.084	0.084
ft10	STT	0.169	0.169
ft20	SRPT	0.069	0.069
la01	SRPT	0.106	0.106
la02	STT	0.142	0.141
la03	SPT	0.209	0.209
la04	SRPT	0.213	0.213
la05	LBFS	0.119	0.119
la06	SPT	0.162	0.162
la10	SRPT	0.040	0.040
la11	SPT	0.110	0.110
la13	SRPT	0.080	0.080
la15	SPT	0.155	0.155
la17	LBFS	0.132	0.132
la19	SRPT	0.166	0.166
orb01	SRPT	0.298	0.298
orb03	STT	0.395	0.395
orb05	SPT	0.224	0.224

6. CONCLUSIONS

The major insights from our analysis are:

1. Given that the fluid relaxation ignores all the combinatorial details of the problem, our results imply that as the number of jobs increases, the combinatorial structure of the problem is increasingly less important, and, as a result, a fluid approximation of the problem that only takes into account the dynamic character of the problem becomes increasingly exact.

2. The FSA-HC is attractive from a practical perspective. First, it is simple to implement and it is fast. Second, its performance on the 20 problems in the OR-library shows that it leads to high-quality solutions for problems of moderate to high multiplicity, outperforming commonly used heuristic rules. Given that high-multiplicity problems are often solved in practice, especially in a manufacturing environment, the FSA-HC should be an attractive algorithm in such settings.

ACKNOWLEDGMENTS

The authors thank the reviewers and the associate editor for helpful comments. This research was partially supported by NSF grant DMI-9610486 and by the MIT-Singapore alliance.

REFERENCES

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

Atkins, D., H. Chen. 1995. Performance evaluation of scheduling control of queueing networks: Fluid model heuristics. *Queueing Systems Appl.* **21** 391–413.

Avram, F., D. Bertsimas, M. Ricard. 1995. Fluid models of sequencing problems in open queueing networks: An optimal control approach. F. P. Kelly, R. J. Williams, eds. *Stochastic Networks. Proc. Internat. Math. Assoc.,* Vol. 71. Springer-Verlag, New York, 199–234.

Bertsimas, D., D. Gamarnik. 1999. Asymptotically optimal algorithms for job shop scheduling and packet routing. *J. Algorithms* **33**(2) 296–318.

——, J. Sethuraman. 2002. From fluid relaxations to practical algorithms for job shop scheduling: The makespan objective. *Math. Programming* **92**(1) 61–102.

Chen, H., D. Yao. 1993. Dynamic scheduling of a multiclass fluid network. *Oper. Res.* **41**(6) 1104–1115.

Dai, J. G. 1995. On positive Harris recurrence of multiclass queueing networks: A unified approach via fluid limit models. *Ann. Appl. Probab.* **5** 49–77.

——, G. Weiss. 2002. A fluid heuristic for minimizing makespan in job shops. *Oper. Res.* **50**(4) 692–707.

Eng, D., J. Humphrey, S. P. Meyn. 1996. Fluid network models: Linear programs for control and performance bounds. *13th World Congress Intern. Fed. Automatic Control*. San Francisco, CA.

Hall, L. 1997. Approximation algorithms for scheduling. D. Hochbaum, ed. *Approximation Algorithms for $\mathcal{NP}$-Hard Problems*. PWS Publishing Company, Boston, MA.

——, A. S. Schulz, D. B. Shmoys, J. Wein. 1997. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Math. Oper. Res.* **22**(3) 513–544.

Harrison, J. M. 1996. The bigstep approach to flow management in stochastic processing networks. F. P. Kelly, S. Zachary, I. Ziedins, eds. *Stochastic Networks: Theory and Applications*. Clarendon Press, Oxford, U.K., 57–90.

Hoogeveen, H., P. Schuurman, G. Woeginger. 1998. Non-approximability results for scheduling problems with minsum criteria. R. E. Bixby, E. A. Boyd, R. Z. Rios-Mercado, eds. *Integer Programming and Combinatorial Optimization (IPCO-VI Proceedings)*. Lecture Notes in Computer Science, Vol. 1412. Springer-Verlag, New York, 353–366.

Karger, D., C. Stein, J. Wein. 1999. Scheduling algorithms. M. J. Atallah, ed. *Algorithms and Theory of Computation Handbook*. CRC Press, Boca Raton, FL.

Luo, X., D. Bertsimas. 1999. A new algorithm for state-constrained separated continuous linear programs. *SIAM J. Control Optim.* **37**(1) 177–210.

Maglaras, C. 2000. Discrete-review policies for scheduling stochastic networks: Trajectory tracking and fluid-scale asymptotic optimality. *Ann. Appl. Probab.* **10**(3) 897–929.

Meyn, S. P. 1997a. The policy improvement algorithm for Markov decision processes with general state space. *IEEE Trans. Automatic Control* **42**(12) 1663–1680.

——. 1997b. Stability and optimization of queueing networks and their fluid models. G. G. Yin, Q. Zhang, eds. *Mathematics of Stochastic Manufacturing Systems. Lectures in Appl. Math.,* Vol. 33. American Mathematical Society, Providence, RI, 175–200.

Pullan, M. C. 1993. An algorithm for a class of continuous linear programs. *SIAM J. Control Optim.* **31**(6) 1558–1577.

Queyranne, M., M. Sviridenko. 1999. Approximation algorithms for shop scheduling problems with minsum criteria. Technical report, Faculty of Commerce, University of British Columbia, Vancouver, British Columbia, Canada.

Rybko, A. N., A. L. Stolyar. 1992. Ergodicity of stochastic processes describing the operations of open queueing networks. *Problems Inform. Transmission* **28** 199–220.