From Fluid Relaxations to Practical Algorithms for Job Shop Scheduling: the Holding Cost Objective

Dimitris Bertsimas * David Gamarnik † Jay Sethuraman ‡

December 1999

Abstract

We design an algorithm for the 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 in N. We report computational results based on benchmark instances chosen from the OR library that suggest that the algorithm is a practical and attractive alternative for solving job shop scheduling problems of even moderate multiplicity.

*Boeing Professor of Operations Research, Sloan School of Management and Operations Research Center, Massachusetts Institute of Technology, E53-363, Cambridge, MA 02139. The research of the author was partially supported by NSF grant DMI-9610486 and by the MIT-Singapore alliance.

†T.J. Watson Research Center, IBM, Yorktown Heights, NY 10598.

‡Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA 02139.

1 Introduction

In this paper, we consider the 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 k^{th} stage of the i^{th} job, and has processing time $p_{i,k}$. Suppose that we have n_i jobs of type i. We are given non-negative 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. We note that our objective generalizes several well-studied special cases, including minimizing weighted completion time of jobs ($w_{i,k} = w_i$).

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 [4]. The algorithm by Bertsimas and Gamarnik [4] 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 [5] 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 [5] to accomodate the objective of minimizing holding costs.

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, Bertsimas, and Ricard [3] 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 [14], building upon the work of Pullan [18], use the theory of continuous linear programming to propose a convergent numerical algorithm for the problem that is able to solve efficiently 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 series of job shop problems in which the number of type i jobs initially present is $\alpha_i N$, with α_i 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
$$\frac{Z_{\mathrm{D}}(N) - Z_{\mathrm{JS}}(N)}{Z_{\mathrm{JS}}(N)} \leq \mathrm{O}\left(\frac{1}{N}\right).$$

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). These results indicate that the $FSA-HC$ is attractive from a practical perspective. First, it is simple to implement and it is fast. Second, its performance on the benchmark problems shows that it leads to very high quality solutions even for problems of moderate multiplicity. Given that especially in a manufacturing environment, jobs **do** have multiplicity, 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 [9], Karger, Stein and Wein [13], Hall, Schulz, Shmoys and Wein [10], Bertsimas and Sethuraman [5] 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 [7] and Rybko and Stolyar [20] 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, Bertsimas and Ricard [3], Atkins and Chen [2], Chen and Yao [6], Eng, Humphrey and Meyn [8], Meyn [16, 17]. 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 [17], none of these papers presents any performance analysis of the derived discrete policy (except in fairly restricted settings). Meyn [17] 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 [15], building on the BIGSTEP approach of Harrison [11], 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 [15] 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 [15] to obtain asymptotically optimal schedules. (The scheme presented in Maglaras [15] appears to use the fact that effective arrival rates 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; and second, we solve the fluid relaxation *once*, and do not resolve it at intermediate points in time.

An interesting recent development is the work of Queyranne and Sviridenko [19] in which they consider approximation algorithms for shop scheduling problems with a minsum 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. (In fact, 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 satisfy always satisfy the hypothesis of their statement: the work of Queyranne and Sviridenko [19] 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, Schuurman and Woeginger [12] show that the job shop problem with the objective of minimizing weighted completion time does not have a polynomial-time approximation scheme unless $P = NP$.

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 provides an asymptotically optimal schedule. In §5, we present computational results on a variety of job shop instances from the OR library. §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 $1 \le J_i \le J_{\max}$. 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 $j = 1, 2, \ldots, J$

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

There are n_i jobs for each type i initially present at their corresponding first stage. Let $w_{i,k}$ be non-negative 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 non-preemptive. 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 $(k-1)^{\text{st}}$ 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 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 $\sigma_k^i, k = 1, 2, \ldots J_i$ it has to be processed on; 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 $x_{i,k}(t)$ to 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 non-negative 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 non-negative 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.

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

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

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

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

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 non-negative. 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 [1]), Luo & Bertsimas [14], based on earlier work by Pullan [18], propose provably convergent discretization-based methods that are able to solve large scale instances in practice fast. The algorithm of Luo & Bertsimas [14] 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 & Nash [1].

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

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

Since $T_{i,k}(t)$ is piecewise linear, Eq. (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 $T_{i,k}(t)$. Therefore, Proposition 1 guarantees the existence of an optimal fluid solution with piecewise constant control. This property enables us to use the machinery developed in Bertsimas and Sethuraman [5] for the makespan objective repeatedly 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)** $Z_{\mathrm{F}}(N) \;\; = C\, N^2$.

(b) $Z_{\mathrm{JS}}(N) \;\; \geq \;\; Z_{\mathrm{F}}(N) - O(N)$.

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

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

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

(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

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

∎

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.

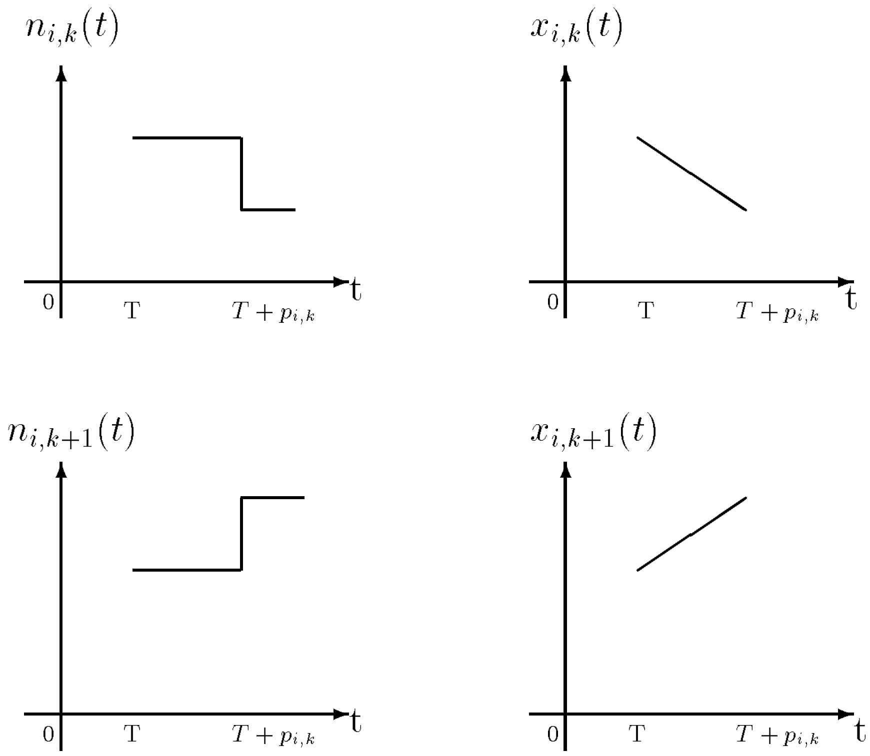

Figure 1: "Fluidizing" a discrete job shop schedule.

3.3 A Suboptimal Fluid Solution with Integral Queue Lengths at the Break Points

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.

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* $M \geq J_{\max}$*, 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

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

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 $v_{i,k} \geq 0$. Note

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

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

Now fix an arbitrary station σ_j. We have

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

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

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

Also

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

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

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

We now show that the solution ends with queue lengths in class (i,k) equal to $\lfloor y_{i,k} \rfloor$. In fact, since 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), $k \geq 2$ 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

$$\begin{aligned} 1/2(\lfloor x_{i,k} \rfloor + \lfloor y_{i,k} \rfloor)\hat{L} &\leq 1/2(x_{i,k} + y_{i,k} + 2)(L + 2\sigma_{\max} P_{\max} J_{\max}) \\ &= 1/2(x_{i,k} + y_{i,k})L + (x_{i,k} + y_{i,k})\sigma_{\max} P_{\max} J_{\max} + 2\sigma_{\max} P_{\max} J_{\max} + L. \end{aligned}$$

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

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

But for all i, k, $1/2(x_{i,k} + y_{i,k})L$ 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

$$\begin{aligned} \hat{c} &\leq c + \sum_{i,k} (x_{i,k} + y_{i,k})\sigma_{\max} P_{\max} J_{\max} + I J_{\max}(2\sigma_{\max} P_{\max} J_{\max} + L) + I(2M - J_{\max})(2\sigma_{\max} P_{\max} J_{\max} + L) \\ &= c + \sum_{i,k} (x_{i,k} + y_{i,k})\sigma_{\max} P_{\max} J_{\max} + 2IM(2\sigma_{\max} P_{\max} J_{\max} + L). \end{aligned}$$

This completes the proof of the lemma. ■

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 r^{th} piece is $N x_{i,k}^r$. 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 r^{th} piece is $\lfloor x_{i,1}^r \rfloor + (R - r)J_{\max}$ for $k = 1$ and $\lfloor x_{i,k}^r \rfloor$, for $k > 1$, and the cost of this solution $\hat{c}$ satisfies*

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

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. ■

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 Revised Fluid Synchronization Algorithm ($RFSA$)) introduced by Bertsimas and Sethuraman in [5]. 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. Since 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 [5]. 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 n^{th} (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 n^{th} (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 *Revised Fluid Synchronization Algorithm* ($RFSA$). In order to introduce it we adopt certain definitions from [5].

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

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

and

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

We also introduce

Discrete Start time ($DS_{i,k}(n)$): This is the start time of the n^{th} (i,k) job in the discrete network, i.e., the time at which the n^{th} (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 n^{th} (i,k) job in the discrete network. In particular,

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

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

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

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

Fluid Completion time ($FC_{i,k}(n)$): This is the completion time of the n^{th} (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 n^{th} (i,k) job is defined as follows.

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

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

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

Nominal Completion time ($NC_{i,k}(n)$): The nominal completion time of the n^{th} (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 n^{th} (i,k) job at time t is:

- *not available*, if $0 \le t < \max\{DC_{i,k-1}(n), NS_{i,k}(n)\}$.
- *available*, if $\max\{DC_{i,k-1}(n), NS_{i,k}(n)\} \le t < DS_{i,k}(n)$.
- *in progress*, if $DS_{i,k}(n) \le t < DC_{i,k}(n)$.
- *departed*, if $t \ge DC_{i,k}(n)$.

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 n^{th} (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,*

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

and

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

Proof. Eq. (21) was proved in Bertsimas and Sethuraman [5] under the FSA. A careful examination of the proof in Bertsimas and Sethuraman [5] reveals that the exact same argument holds for the $RFSA$ as well.

Eq. (22) follows by the definition of the $RFSA$ as follows.

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

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

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 break points 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 $1 \leq i \leq R$. 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 $\hat{T}^0 = 0, \hat{T}^1, \hat{T}^2, \ldots, \hat{T}^R$ such that the vector of queue lengths at $\hat{T}^i$ 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:** $x^r_{i,k}$ 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 break points are $NX^r_{i,k}$ then

$$x^r_{i,1} = \lfloor NX^r_{i,1} \rfloor + (R-r)J_{\max}, \qquad x^r_{i,k} = \lfloor NX^r_{i,k} \rfloor, \quad k = 2, \ldots J_i. \tag{23}$$

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

- **Number of jobs processed in piece r:** $y^r_{i,k}$ denotes the number of (i,k) jobs processed by the modified fluid solution in piece r; clearly, $y^r_{i,k} = \mu_{i,k} u^r_{i,k} L^r$, where $u^r_{i,k}$ 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 r^{th} piece of the fluid relaxation. Let z^r_{ikl} be the number of such jobs. For convenience, we define z^r_{ikk} 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 r^{th} piece of the fluid relaxation. Let z^r_{ikE} be the number of such jobs. We next illustrate the computation of z^r_{ikl} in an example, in order to motivate a formal algorithm to compute these quantities that follows next.

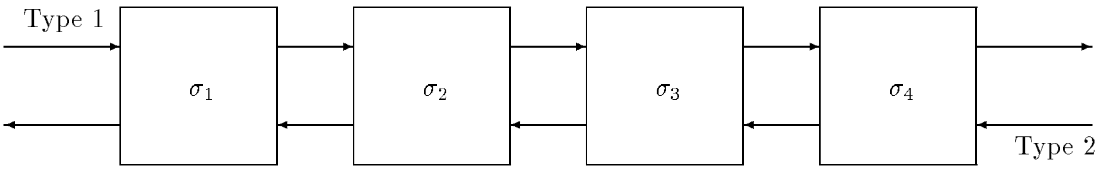

Figure 2: A four station network.

Consider the following example (see Figure 2): There are four machines and two types of jobs. Type 1 jobs 1 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 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 r^{th} 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

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

Table 1: Holding costs and processing times.

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 2: A near-optimal fluid solution.

	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

Table 3: Auxiliary variables z_{ikl}^r for the fluid solution of Table 2.

stage $k' > k$ at T^r. (Recall that if a job is waiting at stage k, it has undergone processing upto its $(k-1)^{\text{st}}$ 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 k^{th} stage is processed in piece r. From these two observations, we have

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

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

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

Computing the auxiliary variables z_{ikl}^r (for $l \le k$)) now reduces to the problem of allocating the $outflow(i,k;r)$ to the stages $l \ge k$ 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 $x_{i,k}^{r-1} - outflow(i,k;r)$. For $l > k$, clearly, z_{ikl}^r cannot exceed $outflow(i,k;r)$ or $inflow(i,l;r)$. For $l = k+1$, we set

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

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. 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)

1. Solve the fluid control problem, and obtain the optimal fluid relaxation. This can be accomplished by applying the algorithm of Luo and Bertsimas [14]. The optimal fluid relaxation has R pieces, and breakpoints $0, T^1, \ldots, T^R$ and the corresponding lengths of the pieces are $L^r = T^{r+1} - T^r$.

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.

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

For $i = 1, 2, \ldots, I$:

For $k = 1, 2, \ldots, J_i$:

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

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

$$outflow(i,E;r) \;=\; 0; \;\; inflow(i,E;r) = \sum_{p=1}^{J_i} (x_{i,p}^{r-1} - x_{i,p}^{r}).$$

For $k = 1, 2, \ldots, J_i$:

$$z_{ikk}^{r} \;=\; x_{i,k}^{r-1} - outflow(i,k;r)$$

For $l = k+1, k+2, \ldots, J_i, E$:

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

$$outflow(i,k;r) := outflow(i,k;r) - z_{ikl}^{r};$$

$$inflow(i,l;r) := inflow(i,l;r) - z_{ikl}^{r};$$

Figure 3: Computing the values of the auxiliary variables z_{ikl}^{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 r^{th} 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 r^{th} piece.

(b) Compute the number z^r_{ikl} and z^r_{ikE} of such jobs by applying the Algorithm shown in Figure 3.

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

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^r_{ikl} 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 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 r^{th} 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 r^{th} 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.

Lemma 4 *The cost of the r^{th} 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 r^{th} 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 r^{th} 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 Eq. (25) over all possible job types, we obtain Eq. (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 r^{th} 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

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

Proof. The cost of the r^{th} 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 $k \le p \le l$, recall that $DC_{ikl,p}(n)$ is the completion time of the n^{th} 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 Eq. (28) separately. First, consider the last term in Eq. (28). Using Eq. (21) for type $(i,k,l;r)$ jobs, we conclude that

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

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

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

Combining Eqs. (29) and (30), we obtain

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

From Eq. (22), we obtain

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

Using Eqs. (31) and (32), we obtain

$$\begin{aligned}
\sum_{n=1}^{z^r_{ikl}} \sum_{p=k+1}^{l-1} w_{i,p}(DC_{ikl,p}(n) - DC_{ikl,p-1}(n)) \;&\le\; \sum_{n=1}^{z^r_{ikl}} \sum_{p=k+1}^{l-1} w_{i,p}\Big(n\frac{L^r}{z^r_{ikl}} - (n-1)\frac{L^r}{z^r_{ikl}} \\
&\qquad + (p-k+1)\ (2P_{\max} + U_{\max})\Big) \\
&= \sum_{n=1}^{z^r_{ikl}} \sum_{p=k+1}^{l-1} w_{i,p}\Big(\frac{L^r}{z^r_{ikl}} + (p-k+1)\ (2P_{\max} + U_{\max})\Big). \\
&= \sum_{p=k+1}^{l-1} w_{i,p}\Big(L^r + z^r_{ikl}(p-k+1)\ (2P_{\max} + U_{\max})\Big).
\end{aligned}$$

We next consider the second term of Eq. (28). From Theorem 9 in Bertsimas and Sethuraman [5], we know that the discretization of the r^{th} piece finishes at time $\hat{L}^r$, such that

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

Using Eqs. (33) and (32), we obtain

$$\sum_{n=1}^{z^r_{ikl}} w_{i,l}(\hat{L}_r - DC_{ikl,l-1}(n)) \;\;\le\;\; \sum_{n=1}^{z^r_{ikl}} w_{i,l}\left(L_r + J_{\max}(2\ P_{\max} + U_{\max}) - \frac{(n-1)L^r}{z^r_{ikl}}\right)$$

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

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

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

The cost of type $(i, k, l; r)$ jobs in the r^{th} piece in the discrete network is obtained by adding Eqs. (33)-(35), which yields Eq. (27). ■

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*

$$
Z_{\mathrm{D}}(N) \ \leq \ Z_{\mathrm{F}}(N) + O(N). \qquad (36)
$$

In particular,

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

and thus

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

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 Eqs. (24) and (26), we have

$$
\begin{aligned}
C^r_d - C^r_f \ \leq \ & \sum_{i=1}^{I} \sum_{k=1}^{J_i} \sum_{l=k}^{J_i} \Bigg\{ \frac{w_{i,k} L^r}{2} + w_{i,k} z^r_{ikl} (2\ P_{\max} + U_{\max}) \\
& + \frac{w_{i,l} L^r}{2} + w_{i,l} z^r_{ikl} J_{\max} (2\ P_{\max} + U_{\max}) \\
& + \sum_{p=k+1}^{l-1} w_{i,p} \Big(L^r + z^r_{ikl} (p-k+1)\ (2P_{\max} + U_{\max}) \Big) \Bigg\}.
\end{aligned}
$$

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

$$C^r_d - C^r_f \ \leq \ AN,$$

for some (large enough) constant A. Thus,

$$\begin{aligned} Z_{\mathrm{D}}(N) - Z'_{\mathrm{F}}(N) &= \sum_{r=1}^{R}(C^r_d - C^r_f) \\ &\leq ARN, \end{aligned}$$

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

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

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

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 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. 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

Benchmark	Z_F (N = 1)	$\frac{Z_D - N^2 Z_F}{N^2 Z_F}$ N = 1	N = 2	N = 5	N = 10	N = 100	N = 500
abz5	4154.54	1.731	1.663	1.302	0.876	0.087	0.014
abz6	3116.64	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	2740.45	2.117	1.987	1.671	1.037	0.436	0.022
ft20	9493.73	1.989	1.700	1.481	1.002	0.210	0.019
la01	2837.45	1.965	1.573	1.320	1.129	0.313	0.016
la02	2802.26	1.270	1.113	0.912	0.614	0.128	0.023
la03	2471.49	1.961	1.672	1.475	1.131	0.254	0.014
la04	2473.30	2.114	1.842	1.386	1.141	0.195	0.014
la05	2501.91	1.320	1.219	1.214	1.067	0.411	0.016
la06	5732.63	2.630	2.315	2.059	1.254	0.193	0.008
la10	5998.61	1.767	1.645	1.323	1.006	0.255	0.011
la11	10000.16	2.749	2.119	1.346	1.043	0.197	0.009
la13	9715.28	2.643	2.216	1.414	1.095	0.351	0.011
la15	10097.26	2.891	2.148	1.730	1.533	0.471	0.021
la17	2983.00	2.653	2.351	1.985	1.438	0.336	0.021
la19	3072.54	2.717	2.185	1.754	1.324	0.372	0.019
orb01	3013.75	2.018	1.811	1.439	1.007	0.221	0.007
orb03	2831.91	2.005	1.837	1.601	1.105	0.119	0.016
orb05	2719.82	1.882	1.473	1.338	0.903	0.143	0.009

Table 4: Job Shop instances in OR-Library—Weighted completion time.

relaxation, $Z_{\mathrm{F}}(1)$, is shown in the second column, and is valid for $N = 1$; the lower bound for $N = n$ is $n^2\, Z_F(1)$. The subsequent columns report the value of the relative error,

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

We can make the following observations from the results reported in Table 4.

1. 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 [5] 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 more jobs in the system in order to obtain the same accuracy. In order to obtain an error of 1%, for example, we need $N = 100$ for makespan, while we need $N = 500$ for the holding cost objective. 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. Especially if one considers that the relative error is between the performance of the $FSA-HC$ and the fluid lower bound, the performance of the $FSA-HC$ compared to the true optimal value will be even better.

2. 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 practical and attractive alternative** for solving job shop scheduling problems of moderate to high multiplicity.

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. Given that especially in a manufacturing environment, jobs **do** have high multiplicity, the $FSA-HC$ should be considered a candidate for practical application.

References

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

[2] D. Atkins and H. Chen. Performance evaluation of scheduling control of queueing networks: fluid model heuristics. *Queueing Systems and Applications*, 21:391–413, 1995.

[3] 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.

[4] D. Bertsimas and D. Gamarnik. Asymptotically optimal algorithms for job shop scheduling and packet routing. *Journal of Algorithms*, 1998. to appear.

[5] D. Bertsimas and J. Sethuraman. From fluid relaxations to practical algorithms for job shop scheduling: the makespan objective. Technical report, Operations Research Center, MIT, 1999.

[6] H. Chen and D. Yao. Dynamic scheduling of a multiclass fluid network. *Operations Research*, 41(6):1104–1115, 1993.

[7] J. G. Dai. On positive harris recurrence of multiclass queueing networks: A unified approach via fluid limit models. *Annals of Applied Probability*, 5:49–77, 1995.

[8] D. Eng, J. Humphrey, and S. P. Meyn. Fluid network models: Linear programs for control and performance bounds. In 13^{th} *World Congress of International Federation of Automatic Control*. San Francisco, 1996.

[9] L. Hall. Approximation algorithms for scheduling. In D. Hochbaum, editor, *Approximation Algorithms for $\mathcal{NP}$-hard problems*. PWS Publishing company, 1997.

[10] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: off-line and on-line approximation algorithms. *Mathematics of Operations Research*, 22(3):513–544, 1997.

[11] 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.

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

[13] D. Karger, C. Stein, and J. Wein. Scheduling algorithms. In *CRC Handbook on Algorithms*, 1997.

[14] 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.

[15] C. Maglaras. Discrete-review policies for scheduling stochastic networks: Trajectory tracking and fluid-scale asymptotic optimality. submitted to *Annals of Applied Probability*, 1997.

[16] S. P. Meyn. The policy improvement algorithm for markov decision processes with general state space. *IEEE Transactions on Automatic Control*, 42(12):1663–1680, 1997.

[17] 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.

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

[19] M. Queyranne and M. Sviridenko. Approximation algorithms for shop scheduling problems with minsum criteria. Technical report, Faculty of Commerce, University of British Columbia, April 1999.

[20] A. N. Rybko and A. L. Stolyar. Ergodicity of stochastic processes describing the operations of open queueing networks. *Problems of Information Transimission*, 28:199–220, 1992.