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

Dimitris Bertsimas * Jay Sethuraman †

August 1999, revised December 2000

Abstract

We design an algorithm, called the fluid synchronization algorithm (FSA), for the job shop scheduling problem with the objective of minimizing the makespan. We round an optimal solution to a fluid relaxation, in which we replace discrete jobs with the flow of a continuous fluid, and use ideas from fair queueing in the area of communication networks in order to ensure that the discrete schedule is close to the one implied by the fluid relaxation. FSA produces a schedule with makespan at most $C_{\max} + (I+2)P_{\max}J_{\max}$, where $C_{\max}$ is the lower bound provided by the fluid relaxation, 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. We report computational results based on all benchmark instances chosen from the OR library that suggest that FSA is a practical algorithm for solving job shop scheduling problems of even moderate multiplicity. We extend our results to job shops with arriving jobs.

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

†Columbia University, Department of Industrial Engineering and Operations Research, New York. The research was completed while at the Operations Research Center, Massachusetts Institute of Technology.

1 Introduction

The job shop scheduling problem is a central $\mathcal{NP}$-hard problem in Operations Research and Computer Science that has been studied extensively from a variety of perspectives in the last thirty years, and is defined as follows: We are interested in scheduling 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 [1]. The pair (i, k) represents the k^{th} stage of the i^{th} job type, and has processing time $p_{i,k}$. Suppose that we have n_i jobs of type i. Our objective is to find a schedule that minimizes the makespan, which is defined as the maximum completion time of the jobs; in the standard scheduling notation, this problem is denoted as $J||C_{\max}$. An alternative objective is to minimize the weighted completion time or more generally to minimize the total holding cost. We address this objective in Bertsimas, Gamarnik and Sethuraman [4].

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. The stages of each job must be completed in order.

The classical job shop scheduling problem involves exactly one job from each type, i.e., the initial vector of job types is $(1, 1, \ldots, 1)$. The job shop scheduling problem is notoriously difficult to solve exactly, even if the sizes of the instances are relatively small. As an example, a specific instance involving 10 machines and 10 jobs posed in a book by Muth and Thompson [18] in 1963 remained unsolved for over 20 years until solved by Carlier and Pinson [5] in 1985.

Our overall approach for the problem draws on ideas from two distinct communities, and is inspired by the recent paper of Bertsimas and Gamarnik [3] who first introduced the idea of rounding a fluid relaxation to the job shop scheduling problem.

First, we consider a relaxation for the job shop scheduling problem called the fluid relaxation, in which we replace discrete jobs with the flow of a continuous fluid. The motivation for this approach comes from optimal control of multiclass queueing networks, which are stochastic and dynamic generalizations of job shops. For the makespan objective, the optimal solution of the fluid control problem

[1] We allow multiple stages of a job to be processed by the same machine; this model is referred to as a "re-entrant" job shop in the literature

can be computed in closed form and provides a lower bound $C_{\max}$ to the job shop scheduling problem; see Weiss [26], Bertsimas and Gamarnik [3].

Our second idea is motivated by the literature on *fair queueing*, which addresses the question of emulating a *given* head-of-the-line processor sharing discipline without preempting jobs. Processor sharing disciplines were originally proposed as an idealization of time-sharing in computer systems. In a time-sharing discipline, the processor cycles through the jobs, giving each job a small quantum of service; processor-sharing is the discipline obtained as the quantum length approaches zero. Processor sharing disciplines are attractive from the point of view of congestion control in large scale networks because of their inherent fairness. For this reason, the question of emulating a given processor sharing discipline using a non-preemptive discipline (while retaining its attractive features) has received a lot of attention in the flow control literature; several simple and elegant schemes, classified under the generic name of *fair queueing,* have been proposed (see Demers, Keshav and Shenker [6], Greenberg and Madras [10], Parekh and Gallager [19, 20]). Of particular relevance to our work is a fair queueing discipline called fair queueing based on start times (FQS). Under this discipline, whenever a scheduling decision is to be made, the job selected is the one that *starts earliest* in the underlying processor sharing discipline. A comprehensive review of related work appears in the survey of Zhang [27].

Our algorithm can be viewed as a natural outcome of combining these two ideas. We use appropriate fluid relaxations to compute the underlying processor sharing discipline (i.e., the rate at which the machines work on various job classes), and then draw on ideas from the fair queueing literature. An important difficulty that must be addressed is that the fluid relaxation approximates jobs by a "continuous fluid," whereas in reality, jobs are "discrete entities." This necessitates a more careful definition of "start times," while attempting to use a discipline like FQS.

In recent years, considerable progress has been made in the deterministic scheduling community in providing approximation algorithms for scheduling problems based on linear programming relaxations. In this framework, a natural linear programming relaxation of the scheduling problem is solved first, which results in LP start/completion times for each job. A typical heuristic is to then schedule the jobs in the order of their LP start times or LP completion times. For a review of this approach, we refer the readers to the papers by Hall [11], Karger, Stein and Wein [15], Hall, Schulz, Shmoys and Wein [12], and the references therein. As we shall see, our scheduling algorithm can be viewed as a generalization of this idea to a dynamic setting in which the "LP start times" are computed on-the-fly. In other words, the "LP start times" at time t are computed based on both the continuous relaxation, and all of the jobs that have been scheduled prior to t.

Results. We describe an efficient algorithm to round an optimal fluid solution such that the resulting schedule is asymptotically optimal. Essentially, we show that for the job shop problem with a *fixed* number of job types, rounding an optimal fluid solution appropriately results in a schedule that incurs $O(1)$ extra cost compared to a trivial lower bound; this trivial lower bound coincides with the optimal cost of the fluid relaxation. To put our result in perspective, consider the classical job shop scheduling problem, which has exactly one job of each type. In this case, the combinatorial structure of the job scheduling problem makes the problem very complicated to solve. Interestingly, the results of this paper 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 becomes increasingly exact. The results of this paper also imply that a continuous approximation to the job shop problem is asymptotically exact. Our results are consistent with the conclusions of several earlier papers that consider the job shop problem with a fixed number of machines; in particular, the fact that such instances become "easier" to solve with increasing number of jobs was recognized earlier in the Soviet literature (see Sevast'janov [24]).

Related work. There is an extensive literature on both job shop scheduling problems, and fluid relaxations. We next provide a brief overview of some of these results, which will also serve to place our results in perspective.

In spite of the intense attention, very little was known about approximation algorithms for job shops until recently. Gonzalez and Sahni [9] proved that any algorithm in which at least one machine is operating at any point in time is within a factor of J of the optimal. Interesting approximation algorithms for shop scheduling problems appeared in the Soviet literature in the mid seventies: these were based on geometric arguments, and were discovered independently by Belov and Stolin [2], Sevast'janov [21], and Fiala [7]. The results using this approach are in the spirit of our results, although based on entirely different methods. These approaches typically produce schedules of length $C_{\max} + \epsilon$, where $C_{\max}$ is a lower-bound on the optimal makespan, and the error term ϵ is independent of the number of jobs. The strongest of these results is by Sevast'janov [22, 23] who proposed a polynomial-time algorithm that delivers a schedule with additive error at most $(J_{\max} - 1)(JJ_{\max}^2 + 2J_{\max} - 1)P_{\max}$, where J is the number of machines, $J_{\max}$ is the maximum number of stages of any job-type, and $P_{\max}$ is the maximum processing time over all tasks. Our algorithm, while delivering a schedule with an additive error independent of the number of jobs, has two distinct advantages: (i) the error bound is substantially better (in fact, it is at most $(I+2)P_{\max}J_{\max}$) when the number of job types is small, and (ii) our algorithm is substantially simpler to implement. We note that the earlier algorithm of Bertsimas and Gamarnik [3], while based on similar methods, produces a schedule with makespan $C_{\max} + O(\sqrt{C_{\max}})$.

Leighton, Maggs and Rao [16], motivated by a packet routing application, considered a restricted version of the job shop problem in which all of the processing times are identically equal to one. (The job shop problem remains strongly $\mathcal{NP}$-hard even under this restriction.) Leighton, Maggs and Rao [16] showed the existence of a schedule with length $O(C_{\max}+P_{\max})$. Unfortunately, their result was not algorithmic, as it relied on a non-constructive probabilistic argument based on the Lovász Local Lemma; they also discovered a randomized algorithm that delivers a schedule of length at most $O(C_{\max}+P_{\max}\log N)$ with high probability, where N is the number of jobs to be scheduled. Subsequently, Leighton, Maggs and Richa [17], using an algorithmic form of the Lovász Local Lemma discovered by Beck[1], showed show how to find a schedule of length $O(C_{\max}+P_{\max})$ in $O(J(C_{\max}+P_{\max})(logL)(loglogL))$ time, with probability $1-1/L-\beta$ for any positive constant β, where L is the sum of the processing times of the jobs. Shmoys, Stein and Wein [25] described a polynomial-time randomized algorithm for job shop scheduling that, with high probability, yields a schedule of length $O\left(\frac{\log^2(JP_{\max})}{\log\log(JP_{\max})}C_{\max}\right)$. They describe a $(2+\epsilon)$-approximation algorithm when J and $P_{\max}$ are constants (as is the case in our setting). These results were subsequently improved by Goldberg, Paterson, Srinivasan, and Sweedyk [8], who present an algorithm for the job shop problem that constructs a schedule of length $O((P_{\max}+C_{\max})\rho)$, with $\rho = \frac{\log(JJ_{\max})}{\text{loglog}(JJ_{\max})}\left\lceil\frac{\log(\min\{JJ_{\max},P_{\max}\})}{\text{loglog}(JJ_{\max})}\right\rceil$. An interesting recent development is the discovery of polynomial time approximation schemes for the job shop scheduling problem; this result was discovered by Jansen, Solis-Oba, and Sviridenko [14, 13]. While all of these algorithms serve an important role — that of classifying these problems in the complexity hierarchy according to the ease of their approximability — none of these algorithms seems practical. In contrast, the algorithm proposed in this paper is accompanied by a guarantee of a small additive error, is easy to implement, and appears to be practical, as demonstrated by extensive computational results.

Structure of the paper. In §2, we consider the makespan objective, and describe an algorithm that provides an asymptotically optimal schedule. These results are extended in §3 to a model in which deterministic arrivals occur over a finite horizon. In §4, we present computational results on a variety of job shop instances from the OR library. §5 contains some concluding remarks.

2 An Algorithm for the Makespan Objective

This section considers the job shop problem with the objective of minimizing makespan, and is structured as follows: In §2.1, we define the job shop scheduling problem formally, and discuss the notation. In §2.2, we describe the fluid relaxation for the job shop scheduling problem, and discuss its solution; this section is reproduced from Bertsimas and Gamarnik [3] and is included here primarily for the sake of completeness. In §2.3, we provide an algorithm, called the *fluid synchronization algorithm* (FSA). In §2.4 we analyze the performance of algorithm FSA and prove that it yields a schedule that is asymptotically optimal.

2.1 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 $\sigma_1^i, \sigma_2^i, \ldots, \sigma_{J_i}^i$ in that order, where $1 \leq J_i \leq J_{\max}$. The time to process a type i job on machine σ_k^i is denoted by $p_{i,k}$. Throughout, we assume that $p_{i,k}$ are integers.

The jobs of type i that have been processed on machines $\sigma_1^i, \ldots, \sigma_{k-1}^i$ but not on machine σ_k^i, are queued at machine σ_k^i 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 $j = 1, 2, \ldots, J$

$$\sigma_j = \{(i,k) : \sigma_j = \sigma_k^i,\ 1 \leq i \leq I,\ 1 \leq k \leq J\}.$$

There are n_i jobs for each type i initially present at their corresponding first stage. Our objective is to minimize the makespan, i.e., to process all the $n_1 + n_2 + \cdots + n_I$ jobs on machines $\sigma_1, \ldots, \sigma_J$, so that the time taken to process all the jobs is minimized.

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

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

The following proposition is immediate.

Proposition 1 *The minimum makespan C^* of the job shop scheduling problem satisfies:*

$$C^* \geq C_{\max}.$$

We define a few other useful quantities. For machine σ_j, let

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

and

$$P_j = \max_{(i,k)\in\sigma_j} p_{i,k}. \tag{1}$$

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

$$U_{\max} = \max_{1\leq j\leq J} U_j, \tag{2}$$

and

$$P_{\max} = \max_{1\leq j\leq J} P_j. \tag{3}$$

In the next section we consider a fluid (fractional) version of this problem, in which the number of jobs n_i of type i can take arbitrary positive real values, and machines are allowed to work simultaneously on several types of jobs (the formal description of the fluid job shop scheduling problem is provided in §2.2). For the fluid relaxation, we show that a simple algorithm leads to a makespan equal to $C_{\max}$, and is, therefore, optimal.

2.2 The Fluid Job Shop Scheduling Problem

In this section we describe a fluid version of the job shop scheduling problem. The input data for the fluid job shop scheduling problem is the same as for the original problem. There are J machines $\sigma_1, \sigma_2, \ldots, \sigma_J$, I job types, each specified by the sequence of machines σ_k^i, $k = 1, 2, \ldots, J_i$, $J_i \leq J$ and the sequence of processing times $p_{i,k}$ for type i jobs in stage k. We introduce the notation $\mu_{i,k} = 1/p_{i,k}$ that represents the rate of machine σ_k^i on a type i job. The number of type i jobs initially present, denoted by x_i, takes nonnegative real values.

In order to specify the fluid relaxation we introduce some notation. We let $x_{i,k}(t)$ be the total (fractional in general) number of type i jobs in stage k at time t. We call this quantity the fluid level of type i in stage k at time t. We denote by $T_{i,k}(t)$ the total time the machine σ_k^i works on type i jobs in stage k during the time interval $[0, t)$. Finally $1\{A\}$ denotes the indicator function for the set A.

The fluid relaxation associated with the problem of minimizing makespan can be formulated as follows:

$$\text{minimize} \quad \int_0^\infty 1\left\{ \sum_{1\le i\le I, 1\le k\le J} x_{i,k}(t) > 0 \right\} dt \tag{4}$$

$$\text{subject to} \qquad x_{i,1}(t) = x_i - \mu_{i,1}T_{i,1}(t), \qquad\qquad i = 1,2,\ldots,I,\ t\ge 0, \tag{5}$$

$$x_{i,k}(t) = \mu_{i,k-1}T_{i,k-1}(t) - \mu_{i,k}T_{i,k}(t), \quad k = 2,\ldots,J_i,\ i = 1,2,\ldots,I,\ t\ge 0, \tag{6}$$

$$0 \le \sum_{(i,k)\in\sigma_j} (T_{i,k}(t_2) - T_{i,k}(t_1)) \le t_2 - t_1,\ \forall\, t_2 > t_1,\ t_1,t_2\ge 0,\ j = 1,2,\ldots,J, \tag{7}$$

$$x_{i,k}(t) \ge 0, \quad T_{i,k}(t) \ge 0. \tag{8}$$

The objective function (4) represents the total time that at least one of the fluid levels is positive. It corresponds to the minimum makespan schedule in the discrete problem. Equations (5) and (6) represent the dynamics of the system. The fluid level of type i in stage k at time t is the initial number of type i jobs in stage k (x_i for $k=1$, zero for $k>1$) plus the number of type i jobs processed in stage $k-1$ during $[0,t)$ (given by $\mu_{i,k-1}T_{i,k-1}(t)$), minus the number of type i jobs processed in stage k during $[0,t)$ (given by $\mu_{i,k}T_{i,k}(t)$). Constraint (7) is just the aggregate feasibility constraint for machine σ_j.

Similar to the definition for the discrete problem, we define congestion in machine σ_j as

$$C_j = \sum_{(i,k)\in\sigma_j} p_{i,k}x_i, \tag{9}$$

and the maximal congestion as

$$C_{\max} = \max_{1\le j\le J} C_j. \tag{10}$$

We next show that the fluid relaxation can be solved in closed form; see Weiss [26], Bertsimas and Gamarnik [3].

Proposition 2 *The fluid relaxation (4) has an optimal value equal to the maximum congestion* $C_{\max}$.

Proof: We first show that the maximum congestion $C_{\max}$ is a lower bound on the optimal value of the fluid relaxation. For any positive time t and for each $i\le I$, $k\le J_i$, we have from (5), (6):

$$\sum_{l=1}^{k} x_{i,l}(t) = x_i - \mu_{i,k}T_{i,k}(t).$$

For each machine σ_j we obtain:

$$\sum_{(i,k)\in\sigma_j} p_{i,k}\sum_{l=1}^{k} x_{i,l}(t) = \sum_{(i,k)\in\sigma_j} p_{i,k}x_i - \sum_{(i,k)\in\sigma_j} T_{i,k}(t) \ge C_j - t,$$

where the last inequality follows from the definition of C_j and Constraint (7) applied to $t_1 = 0$, $t_2 = t$. It follows then, that the fluid levels are positive for all times t smaller than C_j. Therefore, the objective value of the fluid relaxation is at least $\max_j C_j = C_{\max}$.

We now construct a feasible solution that achieves this value. For each $i \le I$, $k \le J_i$ and each $t \le C_{\max}$ we let

$$\begin{aligned}
T_{i,k}(t) &= \frac{p_{i,k}x_i}{C_{\max}}t, \\
x_{i,1}(t) &= x_i - \mu_{i,1}T_{i,1}(t) = x_i - \frac{x_i}{C_{\max}}t, \quad i = 1,\ldots,I, \\
x_{i,k}(t) &= 0, \qquad\qquad\qquad\qquad\qquad k = 2,3,\ldots,J_i, \ i = 1,\ldots,I.
\end{aligned}$$

For all $t > C_{\max}$ we set $T_{i,k}(t) = p_{i,k}x_i$, $x_{i,k}(t) = 0$. Clearly, this solution has an objective value equal to $C_{\max}$. We now show that this solution is feasible. It is nonnegative by construction. Also by construction, Eq. (5) is satisfied for all $t \le C_{\max}$. In particular, $x_{i,1}(C_{\max}) = 0$, $i = 1,2,\ldots,I$. Moreover, for all i, $k = 2,3,\ldots,J_i$ and $t \le C_{\max}$ we have:

$$\mu_{i,k-1}T_{i,k-1}(t) - \mu_{i,k}T_{i,k}(t) = p_{i,k-1}^{-1}T_{i,k-1}(t) - p_{i,k}^{-1}T_{i,k}(t) = \frac{x_i}{C_{\max}}t - \frac{x_i}{C_{\max}}t = 0 = x_{i,k}(t),$$

and Eq. (6) is satisfied. Finally, for any $t_1 < t_2 \le C_{\max}$ and for any machine σ_j, we have:

$$\sum_{(i,k)\in\sigma_j} (T_{i,k}(t_2) - T_{i,k}(t_1)) = \sum_{(i,k)\in\sigma_j} \left(\frac{p_{i,k}x_i}{C_{\max}}t_2 - \frac{p_{i,k}x_i}{C_{\max}}t_1\right) = \frac{C_j}{C_{\max}}(t_2 - t_1) \le t_2 - t_1,$$

and Constraint (7) is satisfied. Note, that for the constructed solution $x_{i,k}(C_{\max}) = 0$ for all $i \le I$, $k \le J_i$. Therefore the feasibility for times $t > C_{\max}$ follows trivially. ■

Let $u_{i,k}$ be the fraction of effort allocated by machine σ_k^i to processing (i,k) jobs in the constructed optimal solution. Clearly,

$$u_{i,k} = \frac{d\, T_{i,k}(t)}{dt} = \frac{p_{i,k}x_i}{C_{\max}}.$$

The constructed solution has a structure resembling a processor sharing policy. It calculates the maximal congestion $C_{\max}$ and allocates a proportional effort to different job types within each machine to achieve the target value $C_{\max}$. Such an optimal policy is possible, since we relaxed the integrality constraint on the number of jobs and allowed machines to work simultaneously on several job types. In the following section we use this fluid solution to construct an asymptotically optimal solution for the original discrete job shop scheduling problem.

2.3 The Fluid Synchronization Algorithm

We now provide an efficient algorithm to discretize a fluid solution. We start with some definitions, followed by a formal description of our discretization algorithm. We illustrate our algorithm on a small example, and also discuss the motivation behind our approach. In the rest of this section the term "discrete network" refers to the (discrete) job shop scheduling problem, and the term "fluid relaxation" refers to the fluid job shop scheduling problem. The term "discrete schedule" will refer to the schedule of the jobs in the discrete network. Finally, whenever we use σ_j to refer to a machine, we assume that the reference is to the machine in the discrete network, unless stated otherwise.

Definitions. We first present some useful definitions, and describe our algorithm; the motivation behind these definitions will be described subsequently.

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.

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,

$$DC_{i,k}(n) = DS_{i,k}(n) + p_{i,k}. \tag{11}$$

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

$$FS_{i,k}(1) = 0, \tag{12}$$

$$FS_{i,k}(n) = FS_{i,k}(n-1) + \frac{C_{\max}}{n_i}, \quad n > 1. \tag{13}$$

Fluid Completion time ($FC_{i,k}(n)$): This is the completion time of the n^{th} (i,k) job in the fluid relaxation, and is given by

$$FC_{i,k}(n) = FS_{i,k}(n) + \frac{C_{\max}}{n_i}. \tag{14}$$

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

$$NS_{i,1}(n) = FS_{i,1}(n), \tag{15}$$

$$NS_{i,k}(1) \;=\; DS_{i,k-1}(1) + p_{i,k-1}, \qquad k > 1, \tag{16}$$

$$NS_{i,k}(n) \;=\; \max\left\{NS_{i,k}(n-1) \;+\; \frac{C_{\max}}{n_i},\;\; DS_{i,k-1}(n) \;+\; p_{i,k-1}\right\}, \qquad n, k > 1. \tag{17}$$

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

$$NC_{i,k}(n) \;=\; NS_{i,k}(n) \;+\; \frac{C_{\max}}{n_i}. \tag{18}$$

Remark. 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 $0 \;\leq\; t \;<\; DC_{i,k-1}(n)$.
- *available*, if $DC_{i,k-1}(n) \;\leq\; t \;<\; DS_{i,k}(n)$.
- *in progress*, if $DS_{i,k}(n) \;\leq\; t \;<\; DC_{i,k}(n)$.
- *departed*, if $t \;\geq\; DC_{i,k}(n)$.

We define the *queue-length* of class (i,k) jobs at time t to be the total number of class (i,k) jobs that are *available* or *in progress* at time t.

The following lemma is an easy consequence of our definitions.

Lemma 1

(a) The fluid start time and fluid completion time of the n^{th} class (i,k) job satisfy

$$FS_{i,k}(n) \;=\; (n-1)\frac{C_{\max}}{n_i}, \qquad n = 1, 2, \ldots, n_i. \tag{19}$$

$$FC_{i,k}(n) \;=\; n\frac{C_{\max}}{n_i}, \qquad n = 1, 2, \ldots, n_i. \tag{20}$$

(b) The nominal start time, $NS_{i,k}(n)$, and the nominal completion time, $NC_{i,k}(n)$, of the n^{th} (i,k) job can be computed at time $DS_{i,k-1}(n)$. Specifically,

$$NS_{i,k}(n) \;=\; \max_{1\le r\le n}\left\{DS_{i,k-1}(r)+p_{i,k-1}+(n-r)\frac{C_{\max}}{n_i}\right\}.$$

In particular, $NS_{i,k}(n)$ and $NC_{i,k}(n)$ can be computed no later than the time at which the n^{th} (i,k) job becomes available.

(c) The nominal completion time of the n^{th} class (i,k) job satisfies

$$NC_{i,1}(n) \;=\; n\frac{C_{\max}}{n_i}, \qquad n=1,2,\ldots,n_i, \tag{21}$$

$$NC_{i,k}(1) \;=\; DC_{i,k-1}(1) \;+\; \frac{C_{\max}}{n_i}, \qquad k>1, \tag{22}$$

$$NC_{i,k}(n) \;=\; \max\left\{NC_{i,k}(n-1),\;\; DC_{i,k-1}(n)\right\} \;+\; \frac{C_{\max}}{n_i}, \qquad n,k>1. \tag{23}$$

Proof. Part (a) is an immediate consequence of the definitions of $FS_{i,k}(n)$ and $FC_{i,k}(n)$. For part (b), we first suppose that $k>1$, and expand the recurrence relation given by Eqs. (16) and (17) to obtain

$$\begin{aligned} NS_{i,k}(n) \;&=\; \max\left\{NS_{i,k}(n-1) \;+\; \frac{C_{\max}}{n_i},\;\; DS_{i,k-1}(n) \;+\; p_{i,k-1}\right\}, \\ &=\; \max_{1\le r\le n}\left\{DS_{i,k-1}(r)+p_{i,k-1}+(n-r)\frac{C_{\max}}{n_i}\right\}. \end{aligned} \tag{24}$$

All of the terms involved in Eq. (24) become known when the n^{th} $(i,k-1)$ job is scheduled for service in the discrete network, i.e., at time $DS_{i,k-1}(n)$; this proves part (b) for $k>1$. For $k=1$, the nominal start times are determined (at time zero) by Eq. (15), which completes the proof. Part (c) follows from the definition of $NS_{i,k}(n)$ and $NC_{i,k}(n)$. ■

Description of algorithm FSA. 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 or some job arrives to 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, our algorithm 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 (by part (b) of Lemma 1).

Lemma 1 ensures that the nominal start time of a job is determined no later than the time at which it becomes available. Thus, our algorithm is well-defined — every job that is *available* for scheduling will have its *nominal start time* already determined. A complete description of the algorithm appears in Figure 1.

Initialization:

Set $NS_{i,1}(n) = (n-1)\frac{C_{\max}}{n_i}$, for $n = 1, 2, \ldots, n_i$, $\quad i = 1, 2, \ldots, I$.

Declare all jobs in stage 1 as *available*.

For $j = 1, 2, \ldots, J$: set machine j to have a scheduling epoch.

While (jobs remain to be processed) {

Process job completions

For $j = 1, 2, \ldots, J$:

if machine j has a scheduling epoch at *current-time*

if the n^{th} job $(i,k) \in \sigma_j$ just completed service

if $k < J_i$, declare job n of class $(i, k+1)$ as *available*.

if $(i, k+1) \in \sigma'_j$ and j' is idle, set machine j' to have

a scheduling epoch at *current-time*.

Schedule jobs at machines that have scheduling epochs

For $j = 1, 2, \ldots, J$:

if machine j does not have any jobs available

next-epoch(j) $= \infty$

else

if machine j has a scheduling epoch at *current-time*

Schedule an available job with the smallest *nominal start time*.

If the n^{th} (i,k) job is scheduled

Set $DC_{i,k}(n) =$ *current-time* $+ p_{i,k}$.

if $k < J_i$,

If $n = 1$ set $NS_{i,k+1}(n) = DC_{i,k}(1)$

else set $NS_{i,k+1}(n) = \max\left\{DC_{i,k}(n),\ NS_{i,k+1}(n-1) + C_{\max}/n_i\right\}$

next-epoch(j) $=$ *current-time* $+ p_{i,k}$.

else

next-epoch(j) $= \infty$.

Prepare for the next epoch

next-time $= \min_{j \in \{1,2,\ldots,J\}}$ *next-epoch(j)*.

current-time := *next-time*

For $j = 1, 2, \ldots, J$:

if *next-epoch(j)* $=$ *current-time*,

set machine j to have a scheduling epoch at *current-time*.

}

Figure 1: The Discretization Algorithm FSA.

Example: Consider the network of Figure 2: there are two machines and two types of jobs. Type 1 jobs require 2 units of processing at machine 1, followed by 4 units of processing at machine 2. Type 2 jobs require 1 unit of processing at machine 2, followed by 4 units of processing at machine 1. Initially, we are given 3 jobs of type 1 and 6 jobs of type 2; our objective is to find a schedule that minimizes the makespan. As before, we use (i, k) to denote type i jobs at stage k. The optimal makespan of the associated fluid job shop scheduling problem, with the corresponding optimal fluid controls are given by:

$$C_{\max} = 30,$$

$$\text{(at machine 1)} \quad u_{1,1} = 0.2, \quad u_{2,2} = 0.8,$$

$$\text{(at machine 2)} \quad u_{2,1} = 0.2, \quad u_{1,2} = 0.4.$$

We now step through algorithm FSA and illustrate how a discrete schedule is determined.

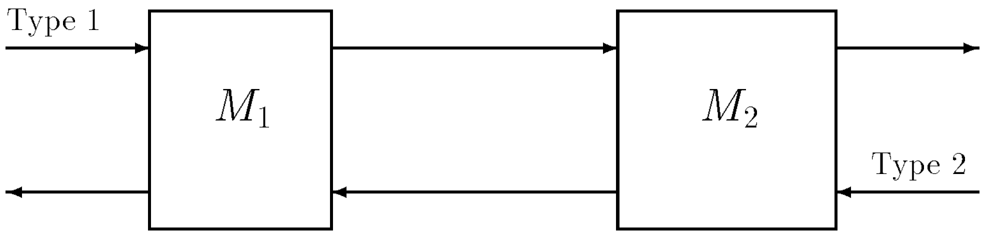

Figure 2: A two station network.

$n_1 = 3; p_{1,1} = 2, p_{1,2} = 4$

$n_2 = 6; p_{2,1} = 1, p_{2,2} = 4$

Optimal Fluid Solution: $C_{\max} = 30; u_{1,1} = 0.2, u_{1,2} = 0.4, u_{2,1} = 0.2, u_{2,2} = 0.8$

$\mathbf{t = 0}$**:** We first perform the initialization step: We compute $NS_{1,1}(n)$ for $n = 1, 2, 3$; and all three jobs of type $(1, 1)$ are declared available at M_1. Similarly, at M_2, we compute $NS_{2,1}(n)$ for $n = 1, \ldots, 6$, and all six jobs of type $(2, 1)$ are declared available. The "state" of the system seen by the machines M_1 and M_2 is shown in Table 1.

Remark: The tables shown at time t present the state of the system as seen by the machines *prior* to the scheduling decisions made at time t. As a consequence of the scheduling decisions made at time t, some additional nominal start times may get defined — we shall explicitly state these in our discussion. The "status" row in each table indicates the status of each job, and is one of "unavailable"

At	(1,1)			(2,2)					
M_1	job 1	job 2	job 3	job 1	job 2	job 3	job 4	job 5	job 6
$NS_{i,k}(n)$	0	10	20	-	-	-	-	-	-
status	a	a	a	na	na	na	na	na	na

At	(1,2)			(2,1)					
M_2	job 1	job 2	job 3	job 1	job 2	job 3	job 4	job 5	job 6
$NS_{i,k}(n)$	-	-	-	0	5	10	15	20	25
status	na	na	na	a	a	a	a	a	a

Table 1: State of the system at $t = 0$.

(na), "available" (a), "in progress" (p) or "departed" (d). In illustrating the algorithm on this example, we shall exhibit similar tables for each of the machines at all scheduling epochs.

Example (contd.): We now return to our example, and consider how the machines M_1 and M_2 make their scheduling decisions at time $t = 0$. At M_1, job 1 of type (1,1) has the smallest nominal start time among all available jobs, and so is scheduled. Similarly, at M_2, job 1 of type $(2,1)$ has the smallest nominal start time among all available jobs, and so is scheduled. These decisions determine the values of $NS_{1,2}(1)$ and $NS_{2,2}(1)$; using Eq. (16), we see that $NS_{1,2}(1) = 2$ and $NS_{2,2}(1) = 1$. The next scheduling epoch is for M_2 at $t = 1$.

t = 1: The state of the system is summarized in Table 2. Only M_2 has a scheduling epoch. Among all the available jobs at M_2, the second $(2,1)$ job has the smallest nominal start time, and so is scheduled for service. This determines the value $NS_{2,2}(2)$, which is computed (using Eq. (17)) as follows.

$$\begin{aligned} NS_{2,2}(2) &= \max\left\{NS_{2,2}(1) + \frac{C_{\max}}{n_i}, DS_{2,1}(2) + p_{2,1}\right\}, \\ &= \max\{1+5, 1+1\} = 6. \end{aligned}$$

The next scheduling epoch is at $t = 2$, for both M_1 and M_2.

t = 2: As before, the state of the system at $t = 2$ is summarized in Table 3. The available job with the smallest nominal start time at M_1 is the first $(2,2)$ job; so this job is scheduled for service at M_1. Similarly, the available job with the smallest nominal start time at M_2 is the first $(1,2)$ job, which is scheduled for service. Since both of the jobs scheduled leave the network, no additional nominal start times need to be computed. The next scheduling epoch is at $t = 6$, for both M_1 and M_2.

At	(1, 1)			(2, 2)					
M_1	job 1	job 2	job 3	job 1	job 2	job 3	job 4	job 5	job 6
$NS_{i,k}(n)$	0	10	20	1	-	-	-	-	-
status	p	a	a	a	na	na	na	na	na

At	(1, 2)			(2, 1)					
M_2	job 1	job 2	job 3	job 1	job 2	job 3	job 4	job 5	job 6
$NS_{i,k}(n)$	2	-	-	0	5	10	15	20	25
status	na	na	na	d	a	a	a	a	a

Table 2: State of the system at $t = 1$.

At	(1, 1)			(2, 2)					
M_1	job 1	job 2	job 3	job 1	job 2	job 3	job 4	job 5	job 6
$NS_{i,k}(n)$	0	10	20	1	6	-	-	-	-
status	d	a	a	a	a	na	na	na	na

At	(1, 2)			(2, 1)					
M_2	job 1	job 2	job 3	job 1	job 2	job 3	job 4	job 5	job 6
$NS_{i,k}(n)$	2	-	-	0	5	10	15	20	25
status	a	na	na	d	d	a	a	a	a

Table 3: State of the system at $t = 2$.

t = 6: The state of the system is summarized in Table 4. The available job with the smallest nominal start time at M_1 is the second $(2,2)$ job; so this job is scheduled for service at M_1. Since this job leaves the network after its service, it does not determine any additional nominal start times. Similarly, the job with the smallest nominal start time at M_2 is the third $(2,1)$ job, which is scheduled for service; this determines $NS_{2,2}(3) = 11$. The next scheduling epoch is at $t = 7$, for M_2.

t = 7: The state of the system is summarized in Table 5. Machine M_2 schedules job 4 of class $(2,1)$, which forces $NS_{2,2}(4) = 16$. The next scheduling epoch is at $t = 8$, for M_2.

At	(1, 1)			(2, 2)					
M_1	job 1	job 2	job 3	job 1	job 2	job 3	job 4	job 5	job 6
$NS_{i,k}(n)$	0	10	20	1	6	-	-	-	-
status	d	a	a	d	a	na	na	na	na
At	(1, 2)			(2, 1)					
M_2	job 1	job 2	job 3	job 1	job 2	job 3	job 4	job 5	job 6
$NS_{i,k}(n)$	2	-	-	0	5	10	15	20	25
status	d	na	na	d	d	a	a	a	a

Table 4: State of the system at $t = 6$.

At	(1, 1)			(2, 2)					
M_1	job 1	job 2	job 3	job 1	job 2	job 3	job 4	job 5	job 6
$NS_{i,k}(n)$	0	10	20	1	6	11	-	-	-
status	d	a	a	d	p	a	na	na	na
At	(1, 2)			(2, 1)					
M_2	job 1	job 2	job 3	job 1	job 2	job 3	job 4	job 5	job 6
$NS_{i,k}(n)$	2	-	-	0	5	10	15	20	25
status	d	na	na	d	d	d	a	a	a

Table 5: State of the system at $t = 7$.

At	(1,1)			(2,2)					
M_1	job 1	job 2	job 3	job 1	job 2	job 3	job 4	job 5	job 6
$NS_{i,k}(n)$	0	10	20	1	6	11	16	-	-
status	d	a	a	d	p	a	a	na	na

At	(1,2)			(2,1)					
M_2	job 1	job 2	job 3	job 1	job 2	job 3	job 4	job 5	job 6
$NS_{i,k}(n)$	2	-	-	0	5	10	15	20	25
status	d	na	na	d	d	d	d	a	a

Table 6: State of the system at $t = 8$.

At	(1,1)			(2,2)					
M_1	job 1	job 2	job 3	job 1	job 2	job 3	job 4	job 5	job 6
$NS_{i,k}(n)$	0	10	20	1	6	11	16	21	-
status	d	a	a	d	p	a	a	a	na

At	(1,2)			(2,1)					
M_2	job 1	job 2	job 3	job 1	job 2	job 3	job 4	job 5	job 6
$NS_{i,k}(n)$	2	-	-	0	5	10	15	20	25
status	d	na	na	d	d	d	d	d	a

Table 7: State of the system at $t = 9$.

t = 8: The state of the system is summarized in Table 6. Machine M_2 schedules job 5 of class $(2,1)$, which forces $NS_{2,2}(5) = 21$. The next scheduling epoch is at $t = 9$, for M_2.

t = 9: The state of the system is summarized in Table 7. Machine M_2 schedules job 6 of class $(2,1)$, which forces $NS_{2,2}(6) = 26$. The next scheduling epoch is at $t = 10$, for both M_1 and M_2.

t = 10: The state of the system is summarized in Table 8. Machine M_2 does not have any jobs to process and hence idles. The job with the smallest nominal start time at machine M_1 is job 2 of class $(1,1)$. The next scheduling epoch is at $t = 12$ for M_1.

At	(1, 1)			(2, 2)					
M_1	job 1	job 2	job 3	job 1	job 2	job 3	job 4	job 5	job 6
$NS_{i,k}(n)$	0	10	20	1	6	11	16	21	26
status	d	a	a	d	d	a	a	a	a

At	(1, 2)			(2, 1)					
M_2	job 1	job 2	job 3	job 1	job 2	job 3	job 4	job 5	job 6
$NS_{i,k}(n)$	2	-	-	0	5	10	15	20	25
status	d	na	na	d	d	d	d	d	d

Table 8: State of the system at $t = 10$.

By now, the mechanics of the algorithm are clear, and so we end the discussion of this example at this point. We note that the rest of the schedule can be computed easily: observe that the nominal start times of all the jobs that require processing at M_1 have been determined already; this dictates the order in which jobs get processed. At M_2, only jobs 2 and 3 of class $(1, 2)$ require processing, and they will be scheduled in that order. The schedule determined by algorithm FSA appears in Table 9. We note that in this example, the schedule determined by algorithm FSA has a makespan of 30, which equals the lower bound of $C_{\max} = 30$; thus the schedule shown in Table 9 is clearly optimal. ■

Running Time: The running time of FSA is linear in the number of jobs, since each job is only addressed a constant number of times. Strictly speaking, in the case that there is multiplicity of jobs, the algorithm is not polynomial, as we do not describe each job separately as part of the input.

Motivation: The key motivation behind algorithm FSA is to schedule jobs in a way that keeps the discrete schedule "close" to the optimal fluid solution. Since the optimal fluid cost is a lower bound, we expect this to be a good strategy. The notion of "closeness" is formalized in our definition of *nominal start times* of jobs. The *nominal start time*, $NS_{i,k}(n)$, of the n^{th} (i, k) job reflects the ideal time by which it should have been scheduled.

Since our main objective is to get "close" to the optimal fluid solution, a natural idea is to set the *nominal start time* of the n^{th} (i, k) job to be its start time in the fluid relaxation $(FS_{i,k}(n))$. While this is reasonable in a single machine setting, this does not give much information in a network setting. To illustrate this, consider the n^{th} job of class (i, k). Its fluid start time, $FS_{i,k}(n)$, is identically equal to its fluid start times at stages $1, 2, \ldots k - 1$, and is an artifact of the continuous nature of the fluid

Time	Queue	length		
t	(1,1)	(1,2)	(2,1)	(2,2)
0	3^*	0	6^*	0
1	3	0	5^*	1
2	2	1^*	4	2^*
6	2	0	4^*	1^*
7	2	0	3^*	2
8	2	0	2^*	3
9	2	0	1^*	4
10	2^*	0	0	4
12	1	1	0	4^*
16	1	1^*	0	3^*
20	1^*	0	0	2
22	0	1^*	0	2^*
26	0	0	0	1^*
30	0	0	0	0

Table 9: Discrete schedule computed by algorithm FSA.

(* indicates a job was scheduled at that time) relaxation. In contrast, in the actual problem, even if the machine at stage (i,k) could process arrivals continuously, job n cannot start at stage k unless it has completed processing at stage at $k-1$ in the discrete network! Our definition of *nominal start time* can be viewed as a correction to the *fluid start time* to account for this effect. Another way to understand the relationship is to observe the similarity in the definitions of $FS_{i,k}(n)$ and $NS_{i,k}(n)$, which are reproduced below.

$$\begin{aligned}
FS_{i,k}(1) &= 0, \\
FS_{i,k}(n) &= FS_{i,k}(n-1) + \frac{C_{\max}}{n_i}, \quad n>1, \\
NS_{i,1}(n) &= FS_{i,1}(n), \\
NS_{i,k}(1) &= DC_{i,k-1}(1), \quad k>1, \\
NS_{i,k}(n) &= \max\left\{NS_{i,k}(n-1) + \frac{C_{\max}}{n_i},\ DC_{i,k-1}(n)\right\}, \quad n>1, k>1.
\end{aligned}$$

In the definition of nominal start times, if we ignore the terms involving the discrete network, we obtain exactly the fluid start times! Our approach is inspired by (and related to) research that deals with generalized processor sharing approaches to flow control (see Parekh and Gallager [19, 20]), fair queueing (see Demers, Keshav and Shenker [6], Greenberg and Madras [10]). In fact, our definition of nominal start times can be viewed as a natural adaptation of the notion of *virtual start times*, used by Greenberg and Madras [10], to this setting.

2.4 Analysis of Algorithm FSA

In this section, we provide a complete analysis of the algorithm shown in Figure 1, and prove our main result, which is stated as Theorem 9. In what follows we will make repeated use of the start times $FS_{i,k}(n)$, $NS_{i,k}(n)$, and $DS_{i,k}(n)$, and the completion times $FC_{i,k}(n)$, $NC_{i,k}(n)$, and $DC_{i,k}(n)$; these quantities are defined by Eqs. (12)-(18), and further simplified in Eqs. (19)-(23). The proof of Theorem 9 involves understanding the relationships between the various start times and completion times for a fixed job. In particular, suppose we consider job n of class (i,k); our objective is to establish a strong relationship between $FC_{i,k}(n)$ and $DC_{i,k}(n)$ — these are, respectively, the completion time of this job in the fluid relaxation and in the discrete network. We establish such a relationship using using the nominal completion time, $NC_{i,k}(n)$, as an intermediary.

Our first result relates the the discrete start time, $DS_{i,k}(n)$, to the nominal start time, $NS_{i,k}(n)$. Specifically, we show that

$$DS_{i,k}(n) \le NS_{i,k}(n) + (P_{\sigma_k^i} + U_{\sigma_k^i}), \tag{25}$$

where σ^i_k is the machine that processes class (i,k) jobs. A result in this spirit, but for completion times, appears in the literature; it was first proved by Greenberg and Madras [10] for uniform processor sharing systems, and was generalized by Parekh and Gallager [19] to generalized processor sharing systems.

First, we develop the machinery necessary to prove Eq. (25). To this end, we focus on the particular machine $\sigma^i_k = \sigma_j$ at which class (i,k) is processed: as we shall see, only the classes that are processed at machine σ_j play a role in establishing Eq. (25). To avoid cumbersome notation, we drop the usual (i,k) notation for classes; instead we let R be the set of classes processed by machine σ_j, and use r to denote a generic element of R; these conventions will be in effect until we establish Eq. (25).

We start with a few definitions. Let $r \in R$; we define $T^c_r(t)$ and $T^d_r(t)$ as follows:

$$T^c_r(t) = \begin{cases} (n-1)\ p_r + u_r\ (t - NS_r(n)), & \text{for } NS_r(n) \le t < NC_r(n); \\ n\ p_r, & \text{for } NC_r(n) \le t < NS_r(n+1). \end{cases} \tag{26}$$

$$T^d_r(t) = \begin{cases} (n-1)\ p_r + (t - DS_r(n)), & \text{for } DS_r(n) \le t < DC_r(n); \\ n\ p_r, & \text{for } DC_r(n) \le t < DS_r(n+1). \end{cases} \tag{27}$$

Thus, $T^d_r(t)$ can be interpreted as the total amount of time devoted to processing class r jobs in $[0,t)$ in the discrete network; And $T^c_r(t)$ admits the same interpretation for the "continuous" schedule defined by the nominal start times, in which a job of class r is processed continuously at rate u_r. We also note that $T^c_r(t)$ is continuous: to prove this, we need only check continuity at $t = NC_r(n)$. Suppose $r = (i,k)$; Recall that

$$u_r \ = \ \frac{n_i p_r}{C_{\max}},$$

and

$$NC_r(n) - NS_r(n) \ = \ \frac{C_{\max}}{n_i}.$$

Using these observations in Eq. (26) at $t = NC_r(n)$, we have

$$\begin{aligned} T^c_r(t) &= (n-1)\ p_r + u_r(NC_r(n) - NS_r(n)) \\ &= (n-1)\ p_r + \frac{n_i p_r}{C_{\max}} \frac{C_{\max}}{n_i} \\ &= n\ p_r, \end{aligned}$$

which is consistent with Eq. (26) when $t = NC_r(n)$.

We define a potential function, $\phi_r(t)$, for class r jobs at time t as follows:

$$\phi_r(t) \ = \ \max\{T^c_r(t) - T^d_r(t), -p_r\}.$$

Our main motivation in defining the potential function, $\phi_r(t)$, is to capture the extent to which the "discrete" schedule is *behind* the "continuous" schedule on class r jobs up to time t. For this reason, we penalize the discrete schedule for falling behind the continuous schedule, but give credit for *at most one job* when the discrete schedule is ahead of the continuous schedule. We now proceed to derive some useful properties of $\phi_r(t)$, stated as a series of lemmas.

Lemma 2 *Let t be a scheduling epoch at machine σ_j. Then the following two statements are equivalent.*

(a) For some $n \geq 1$, job n of class r is such that $NS_r(n) < t \leq DS_r(n)$.

(b) $\phi_r(t) > 0$.

Proof:

(**a**) $\Rightarrow$ (**b**):

From Eq. (26), we have

$$NS_r(n) < t \quad \Rightarrow \quad T_r^c(t) > (n-1)p_r. \tag{28}$$

Similarly, from Eq. (27), we have

$$DS_r(n) \geq t \quad \Rightarrow \quad T_r^d(t) \leq (n-1)p_r. \tag{29}$$

Simplifying Eqs. (28) and (29), we obtain

$$T_r^c(t) - T_r^d(t) > 0.$$

Noting that $\phi_r(t) = \max\{T_r^c(t) - T_r^d(t), -p_r\}$, we conclude that $\phi_r(t) > 0$.

(**b**) $\Rightarrow$ (**a**):

By definition,

$$\phi_r(t) > 0 \ \Rightarrow \ T_r^c(t) - T_r^d(t) > 0.$$

Since t is a scheduling epoch in the discrete network, $T_r^d(t)$ should be an integral multiple of p_r, i.e., $T_r^d(t) = lp_r$ for some $l \geq 0$. Since $T_r^c(t) > T_r^d(t)$, the $(l+1)^{\text{st}}$ job of class r satisfies $NS_r(l+1) < t \leq DS_r(l+1)$. ∎

Lemma 3 *Let t be a scheduling epoch at machine σ_j, and suppose $\phi_r(t) > 0$ for some r. Suppose the job scheduled to start at machine σ_j at time t belongs to class r'. Then $\phi_{r'}(t) > 0$.*

Proof: Since $\phi_r(t) > 0$, by Lemma 2, there exists n such that $NS_r(n) < t \leq DS_r(n)$. In particular, σ_j cannot starve as there is at least one job waiting to be served. If $r' = r$, we are done, as $\phi_r(t) > 0$ by assumption. If not, let n' be the job of class r' that was scheduled at time t. Since algorithm FSA selected n' over n, $NS_{r'}(n') \leq NS_r(n)$; this is because, at any scheduling epoch, algorithm FSA schedules a job with the smallest nominal start time. In particular, $NS_{r'}(n') < t = DS_{r'}(n')$. Using Lemma 2, we conclude that $\phi_{r'}(t) > 0$. ∎

A *busy period* for machine σ_j begins when a job arrival (from its previous stage) to σ_j finds it idle; similarly, a *busy period* for machine σ_j ends when a job departure from σ_j leaves it idle.

Lemma 4 *Let t be a scheduling epoch at machine σ_j that begins a busy period. Then,*

$$\phi_r(t) \ \leq \ 0. \tag{30}$$

Proof: Let $W_r(t)$ is the sum of the processing times of all the class r jobs that arrive prior to time t. Since t is a scheduling epoch at σ_j that begins a busy period, $T_r^d(t) \ = \ W_r(t)$. Also, $T_r^c(t) \ \leq \ W_r(t)$, because $W_r(t)$ represents the total amount of class r work that has arrived up to time t. Thus, $T_r^c(t) - T_r^d(t) \ \leq \ 0$, for every r. By definition,

$$\phi_r(t) \ = \ \max\{T_r^c(t) - T_r^d(t), -p_r\} \ \leq \ 0.$$

∎

Lemma 5 *Let t be a scheduling epoch at machine σ_j. Then,*

$$\sum_{r=1}^{R} \phi_r(t) \ \leq \ 0. \tag{31}$$

Proof: Let $t_1, t_2, \ldots, t_b$ be the list of scheduling epochs during an arbitrary busy period in σ_j. (Note that t_b is the epoch that concludes the busy period.) We will prove this lemma using induction on scheduling epochs. By Lemma 4, the result is true at t_1, the beginning of the busy period. For $l \leq b$, suppose that Eq. (31) holds for $t \ = \ t_1, t_2, \ldots t_{l-1}$. We now prove that Eq. (31) holds for $t = t_l$. Since t_{l-1} does not conclude a busy period, some job is scheduled to start in machine σ_j at time t_{l-1}; let r' be the class of this job. By definition, $t_l \ = t_{l-1} + p_{r'}$. We next relate the potential functions at time t_{l-1} and t_l for each job class. In doing this, we shall use the following inequalities that $\phi_r(t)$ satisfies:

$$\phi_r(t) \ \ \geq \ \ T_r^c(t) - T_r^d(t), \tag{32}$$

$$\phi_r(t) + p_r \ \ \geq \ \ 0. \tag{33}$$

Since class r jobs are allocated a fraction u_r of effort in the continuous schedule, we have:

$$T_r^c(t_l) \;\leq\; T_r^c(t_{l-1}) \;+\; u_r p_{r'}. \tag{34}$$

Also, at the machine σ_j, since a job of class r' is scheduled in the interval $[t_{l-1}, t_l)$, we have

$$T_{r'}^d(t_l) = T_{r'}^d(t_{l-1}) + p_{r'}. \tag{35}$$

and

$$T_r^d(t_l) = T_r^d(t_{l-1}), \quad \text{for } r \neq r'. \tag{36}$$

Thus,

$$\begin{aligned} \phi_{r'}(t_l) &= \max\{T_{r'}^c(t_l) - T_{r'}^d(t_l), -p_{r'}\} \\ &\leq \max\{T_{r'}^c(t_{l-1}) - T_{r'}^d(t_{l-1}) + u_{r'}p_{r'} - p_{r'}, -p_{r'}\} \\ &\qquad\qquad\qquad \text{(by Eq. (34) for } r = r'\text{, and Eq. (35))} \\ &\leq \max\{\phi_{r'}(t_{l-1}) - p_{r'}(1 - u_{r'}), -p_{r'}\}. \qquad \text{(by Eq. (32)).} \end{aligned} \tag{37}$$

For $r \neq r'$, we have

$$\begin{aligned} \phi_r(t_l) &= \max\{T_r^c(t_l) - T_r^d(t_l), -p_r\} \\ &\leq \max\{T_r^c(t_{l-1}) - T_r^d(t_{l-1}) + u_r p_{r'}, -p_r\} \\ &\qquad\qquad\qquad \text{(by Eq. (34) for } r \neq r'\text{, and Eq. (36))} \\ &\leq \max\{\phi_r(t_{l-1}) + u_r p_{r'}, -p_r\} \qquad \text{(by Eq. (32)).} \\ &= \phi_r(t_{l-1}) + u_r p_{r'}. \qquad \text{(by Eq. (33)).} \end{aligned} \tag{38}$$

We now consider two possibilities depending on whether $\phi_{r'}(t_{l-1}) > 0$ or not.

Case 1: $\phi_{r'}(t_{l-1}) > 0$:

Since $\phi_{r'}(t_{l-1}) > 0$, and $u_{r'} \geq 0$,

$$\phi_{r'}(t_{l-1}) - p_{r'}(1 - u_{r'}) \;>\; -p_{r'}. \tag{39}$$

From Eqs. (39) and (37), we obtain

$$\phi_{r'}(t_l) \;\leq\; \phi_{r'}(t_{l-1}) - p_{r'}(1 - u_{r'}). \tag{40}$$

Thus,

$$\begin{aligned}
\sum_{r=1}^{R} \phi_r(t_l) &= \phi_{r'}(t_l) + \sum_{r:r\neq r'} \phi_r(t_l) \\
&\leq \phi_{r'}(t_{l-1}) - p_{r'}(1-u_{r'}) + \sum_{r:r\neq r'} (\phi_r(t_{l-1}) + u_r p_{r'}) \\
&\qquad \text{(by Eqs. (38) and (40))} \\
&= \sum_{r=1}^{R} \phi_r(t_{l-1}) - p_{r'}\Big(1 - \sum_{r=1}^{R} u_r\Big) \\
&\leq \sum_{r=1}^{R} \phi_r(t_{l-1}) \qquad \Big(\sum_{r=1}^{R} u_r \leq 1\Big) \\
&\leq 0. \qquad \text{(by the induction hypothesis).}
\end{aligned}$$

Case 2: $\phi_{r'}(t_{l-1}) \leq 0$:

Since a job of class r' is scheduled at t_{l-1}, and since $\phi_{r'}(t_{l-1}) \leq 0$, we use Lemma 3 to conclude that there cannot be any job class with positive potential, i.e.,

$$\phi_r(t_{l-1}) \leq 0, \quad \text{for all } r. \tag{41}$$

From Eqs. (41) and (38), we have

$$\phi_r(t_l) \leq u_r p_{r'}, \quad r \neq r'. \tag{42}$$

Similarly, from Eqs. (41) and (37), we obtain

$$\begin{aligned}
\phi_{r'}(t_l) &\leq \max\{-p_{r'}(1-u_{r'}), -p_{r'}\} \\
&= -p_{r'}(1-u_{r'}).
\end{aligned} \tag{43}$$

Adding Eqs. (42) and (43), we have

$$\begin{aligned}
\sum_{r=1}^{R} \phi_r(t_l) &\leq p_{r'}\Big(\sum_{r=1}^{R} u_r - 1\Big) \\
&\leq 0. \qquad \Big(\sum_{r=1}^{R} u_r \leq 1\Big).
\end{aligned}$$

In either case, we have shown that

$$\sum_{r=1}^{R} \phi_r(t_l) \leq 0,$$

completing the induction. ■

We are now ready to establish Eq. (25).

Theorem 6 *Let $NS_{i,k}(n)$ be the nominal start time of the n^{th} (i,k) job; let $DS_{i,k}(n)$ be its start time in the discrete network. Then,*

$$DS_{i,k}(n) \;\leq\; NS_{i,k}(n) + (P_{\sigma^i_k} + U_{\sigma^i_k}). \tag{44}$$

Proof: We let $r = (i,k)$, and let R be the set of all job classes processed by machine σ^i_k. For convenience, we also let $\sigma_j = \sigma^i_k$. If $DS_r(n) \leq NS_r(n)$, the lemma is trivially true. Suppose $DS_r(n) > NS_r(n)$. Let t be the first time instant in $[NS_r(n), \infty)$ at which the discrete network has a scheduling epoch. Note that $t \leq DS_r(n)$, since, by definition, t is the first time instant at which the job under consideration could be scheduled in the discrete network. Our plan for the proof is to consider the sum of the processing times, S, of all jobs that are processed at machine σ_j in the discrete network in the interval $[t, DS_r(n))$. Clearly, $DS_r(n) \;=\; t + S$, since algorithm FSA does not idle when jobs are available to be processed. We will show that all such jobs have nominal start times at most t, and then proceed to find an upper bound on the number of such jobs, thus providing an upper bound on S.

Consider a job, say job n' of class r', that was scheduled in the discrete network in the interval $[t, DS_r(n))$. Since job n of class r was a candidate during this period, and since it was not selected, we have

$$NS_{r'}(n') \leq NS_r(n). \tag{45}$$

This is because algorithm FSA always selects a job with the smallest nominal start time. From Eq. (45) and the fact that $NS_r(n) \leq t$, we obtain

$$NS_{r'}(n') \leq t. \tag{46}$$

Thus, we have established that the jobs processed during the interval $[t, DS_r(n))$ have nominal start times at most t. By Lemma 3,

$$\phi_{r'}(t) \;>\; 0, \tag{47}$$

if a job belonging to class r' is scheduled in the discrete network in the interval $[t, DS_r(n))$.

Let

$$B_{r'}(t) \;=\; \{n' \mid NS_{r'}(n') < t \leq DS_{r'}(n')\}.$$

In other words, $B_{r'}(t)$ consists of those class r' jobs that have started in the continuous schedule before time t, but start in the discrete network at or after t. From Eq. (46), the set of jobs that are processed in the discrete network during $[t, DS_r(n))$ is a subset of $\cup_{r'} B_{r'}(t)$. Thus, we are naturally led to considering the cardinality of $B_{r'}(t)$.

If $B_{r'}(t) \neq \emptyset$, let

$$B_{r'}(t) \;=\; \{a+1, a+2, \ldots, a+l\}, \qquad a \geq 1,$$

i.e., $|B_{r'}(t)| = l$. (The particular form of $B_{r'}(t)$ follows from the fact that jobs within a class are served in FCFS manner.) Since the $(a+l)^{\text{th}}$ job has started service in the continuous schedule, the $(a+l-1)^{\text{st}}$ job has completed service in the continuous schedule. Thus,

$$T^c_{r'}(t) \;>\; (a+l-1)\; p_{r'},$$

and

$$T^d_{r'}(t) \;=\; a\; p_r.$$

As a result, we obtain

$$\phi_{r'}(t) \;=\; \max\{T^c_{r'}(t) - T^d_{r'}(t), -p_{r'}\} \;=\; T^c_{r'}(t) - T^d_{r'}(t) \;>\; (l-1)p_{r'},$$

and hence,

$$(l-1) \;<\; \frac{\phi_{r'}(t)}{p_{r'}},$$

which implies

$$|B_{r'}(t)| \;=\; l \;\leq\; \frac{\phi_{r'}(t)}{p_{r'}} + 1. \tag{48}$$

From Eq. (48), the total time required to process all the jobs in $B_{r'}(t)$ in the discrete network is either zero (if $B_{r'}(t) = \emptyset$) or at most $|B_{r'}(t)|p_{r'} \;\leq\; \phi_{r'}(t) + p_{r'}$. Thus, the total time required to process all the jobs in $B_{r'}(t)$ is at most $\max\{\phi_{r'}(t) + p_{r'}, 0\}$, which, by Eq. (33), is $\phi_{r'}(t) + p_{r'}$.

Thus, the total time S required to process all the jobs scheduled in the discrete network during $[t, DS_r(n))$ satisfies:

$$\begin{aligned} DS_r(n) - t \;\leq\; S \;\;&\leq\;\; \sum_{r'=1}^{R} \phi_{r'}(t) + p_{r'} \\ &\leq\;\; \sum_{r'=1}^{R} p_{r'}. \qquad \text{(by Lemma 5).} \end{aligned} \tag{49}$$

Moreover, t is defined as the first scheduling epoch for the discrete network after time $NS_r(n)$. In the interval $[NS_r(n), t]$, if some job, say of class $\hat{r}$, is being processed, then

$$t - NS_r(n) \;\leq\; p_{\hat{r}}. \tag{50}$$

From Eqs. (49) and (50), we obtain

$$\begin{aligned} DS_r(n) - NS_r(n) &\leq \sum_{r'=1}^{R} p_{r'} + \max_{r'=1,\ldots R} p_{r'} \\ &= U_j + P_j. \end{aligned}$$

■

We now use Theorem 6 to establish a relationship between $NC_{i,k}(n)$ and $FC_{i,k}(n)$.

Theorem 7 *Let $NC_{i,k}(n)$ be the nominal completion time of the n^{th} (i,k) job, and let $FC_{i,k}(n)$ be its completion time in the fluid relaxation. Then,*

$$NC_{i,k}(n) \;\leq\; FC_{i,k}(n) + \sum_{l=1}^{k-1}(2\; P_{\sigma_l^i} + U_{\sigma_l^i}). \tag{51}$$

Proof: We fix a job type i, and prove this result by induction on the stage number. The base case for the induction is easy: for $k=1$, $NC_{i,k}(n)$ and $FC_{i,k}(n)$ are identical (Eqs. (14), (15), and (18)). Suppose the lemma is true for all $(i,k-1)$ jobs, $k \geq 2$. Consider the first (i,k) job.

$$\begin{aligned} NC_{i,k}(1) &= DC_{i,k-1}(1) + \frac{C_{\max}}{n_i} \qquad \text{(by Eq. (22))} \\ &= DS_{i,k-1}(1) + p_{i,k-1} + \frac{C_{\max}}{n_i} \qquad \text{(by Eq. (11))} \\ &\leq NS_{i,k-1}(1) + (P_{\sigma_{k-1}^i} + U_{\sigma_{k-1}^i}) + p_{i,k-1} + \frac{C_{\max}}{n_i} \qquad \text{(by Theorem 6)} \\ &= NC_{i,k-1}(1) + (P_{\sigma_{k-1}^i} + U_{\sigma_{k-1}^i}) + p_{i,k-1} \qquad \text{(by Eq. (18))} \\ &\leq FC_{i,k-1}(1) + \sum_{l=1}^{k-2}(2\; P_{\sigma_l^i} + U_{\sigma_l^i}) + (P_{\sigma_{k-1}^i} + U_{\sigma_{k-1}^i}) + p_{i,k-1} \\ &\qquad\qquad\qquad\qquad \text{(by the induction hypothesis)} \\ &\leq FC_{i,k}(1) + \sum_{l=1}^{k-1}(2\; P_{\sigma_l^i} + U_{\sigma_l^i}). \qquad \text{(by Eq. (20))} \end{aligned}$$

Thus, the Lemma is true for the first (i,k) job. Suppose it is true for the first $(n-1)$ (i,k) jobs. Consider the n^{th} (i,k) job $(n > 1,\ k > 1)$. From Eq. (23),

$$NC_{i,k}(n) \;=\; \max\left\{NC_{i,k}(n-1),\;\; DC_{i,k-1}(n)\right\} \;+\; \frac{C_{\max}}{n_i}.$$

We consider the two cases.

Case 1: $NC_{i,k}(n) \;=\; NC_{i,k}(n-1) + C_{\max}/n_i$.

In this case, we have:

$$\begin{aligned}
NC_{i,k}(n) &= NC_{i,k}(n-1) + \frac{C_{\max}}{n_i} \\
&\le FC_{i,k}(n-1) + \sum_{l=1}^{k-1}(2\ P_{\sigma_l^i} + U_{\sigma_l^i}) + \frac{C_{\max}}{n_i} \\
&\qquad\qquad\qquad\qquad \text{(by the induction hypothesis)} \\
&\le FC_{i,k}(n) + \sum_{l=1}^{k-1}(2\ P_{\sigma_l^i} + U_{\sigma_l^i}). \qquad \text{(by Eq. (20))}
\end{aligned}$$

Case 2: $NC_{i,k}(n) \;=\; DC_{i,k-1}(n) + C_{\max}/n_i$.

In this case, we have:

$$\begin{aligned}
NC_{i,k}(n) &= DC_{i,k-1}(n) + \frac{C_{\max}}{n_i} \\
&= DS_{i,k-1}(n) + p_{i,k-1} + \frac{C_{\max}}{n_i} \qquad \text{(by Eq. (11))} \\
&\le NS_{i,k-1}(n) + (P_{\sigma_{k-1}^i} + U_{\sigma_{k-1}^i}) + p_{i,k-1} + \frac{C_{\max}}{n_i} \qquad \text{(by Theorem 6)} \\
&= NC_{i,k-1}(n) + (P_{\sigma_{k-1}^i} + U_{\sigma_{k-1}^i}) + p_{i,k-1} \qquad \text{(by Eq. (18))} \\
&\le FC_{i,k-1}(n) + \sum_{l=1}^{k-2}(2\ P_{\sigma_l^i} + U_{\sigma_l^i}) + (P_{\sigma_{k-1}^i} + U_{\sigma_{k-1}^i}) + p_{i,k-1} \\
&\qquad\qquad\qquad\qquad \text{(by the induction hypothesis)} \\
&\le FC_{i,k}(n) + \sum_{l=1}^{k-1}(2\ P_{\sigma_l^i} + U_{\sigma_l^i}). \qquad \text{(by Eq. (20))}
\end{aligned}$$

■

The following theorem relates $DC_{i,k}(n)$ to $FC_{i,k}(n)$, and is an immediate consequence of Theorems 6 and 7.

Theorem 8 *Let $FC_{i,k}(n)$ ($DC_{i,k}(n)$) be the completion time of the n^{th} (i,k) job in the fluid relaxation (discrete network). Then,*

$$DC_{i,k}(n) \;\le\; FC_{i,k}(n) \;+\; \sum_{l=1}^{k}(2\ P_{\sigma_l^i} + U_{\sigma_l^i}). \tag{52}$$

Proof: From Theorem 6, we have

$$DS_{i,k}(n) \le NS_{i,k}(n) + (P_{\sigma_k^i} + U_{\sigma_k^i}),$$

and so,

$$\begin{aligned} DC_{i,k}(n) &= DS_{i,k}(n) + p_{i,k} \\ &\le NS_{i,k}(n) + (P_{\sigma_k^i} + U_{\sigma_k^i}) + p_{i,k} \\ &= NC_{i,k}(n) - \frac{C_{\max}}{n_i} + (P_{\sigma_k^i} + U_{\sigma_k^i}) + p_{i,k} \\ &\le NC_{i,k}(n) + (P_{\sigma_k^i} + U_{\sigma_k^i}). \qquad (\text{since } p_{i,k} \le C_{\max}/n_i). \end{aligned} \tag{53}$$

From Eqs. (53) and (51), we obtain Eq. (52). ■

We are now ready to state our main result.

Theorem 9 *Consider a job shop scheduling problem with I job types and J machines $\sigma_1, \sigma_2, \ldots, \sigma_J$. Given initially n_i jobs of type $i = 1, 2, \ldots, I$, the algorithm FSA produces a schedule with makespan time C_{D} such that*

$$C_{\max} \le C^* \le C_{\mathrm{D}} \le C_{\max} + \max_{i=1,2,\cdots,I} \sum_{l=1}^{J_i} (2\ P_{\sigma_l^i} + U_{\sigma_l^i}). \tag{54}$$

In particular,

$$\frac{C_{\mathrm{D}}}{C^*} \le \frac{C_{\mathrm{D}}}{C_{\max}} \to 1, \tag{55}$$

as

$$\sum_{i=1}^{I} n_i \to \infty,$$

where C^* is the optimal makespan.

Proof: From Eqs. (12), (13) and (14), we see that

$$FC_{i,J_i}(n_i) = C_{\max}$$

for all $i \in I$. Using Theorem 8,

$$\begin{aligned} DC_{i,J_i}(n_i) &\le FC_{i,J_i}(n_i) + \sum_{l=1}^{J_i} (2\ P_{\sigma_l^i} + U_{\sigma_l^i}) \\ &= C_{\max} + \sum_{l=1}^{J_i} (2\ P_{\sigma_l^i} + U_{\sigma_l^i}). \end{aligned}$$

The result follows by observing that

$$C_D = \max_{i \in I} \{DC_{i,J_i}(n_i)\}.$$

■

From Theorem 9, we see that the additive error of the schedule computed by algorithm FSA is bounded from above by $J_{\max}(2\ P_{\max} + U_{\max})$; using $U_{\max} \leq IP_{\max}$, we note that the additive error of the schedule constructed by algorithm FSA is at most $J_{\max}P_{\max}(I+2)$, which is substantially smaller than the guarantees provided by Sevast'janov's algorithm [22, 23] when I is small. We note that an additive error of $(J_{\max} - 1)P_{\max}$ is necessary for any algorithm that uses the optimal fluid cost for comparison purposes: for example, consider a simple flow shop with J stages, and let the processing time at each stage be P. If there are N jobs to start with, the optimal fluid cost is NP, whereas the optimal makespan is $(N+J-1)P$. An interesting open problem is to find the "optimal" additive error for algorithms based on fluid relaxations, and to design algorithms that achieve this additive error.

3 Makespan with Deterministic Arrivals

This section generalizes the results of §2 to a model in which external arrivals are permitted over a (suitably restricted) finite horizon $[0, T^*]$. The objective is to minimize the time required to process all the initial jobs plus the jobs that arrive during $[0, T^*]$. This section is structured as follows: In §3.1, we formally define the model considered; The associated fluid relaxation and its solution is discussed in §3.2. In §3.3, we prove that the *fluid synchronization algorithm* (FSA) yields a schedule that is asymptotically optimal.

3.1 Problem Formulation

The model considered here is identical to that of §2.2, except that external arrivals are permitted. We assume that type i jobs arrive to the network in a deterministic fashion at rate λ_i. The traffic intensity at machine σ_j, denoted by ρ_j, is defined as

$$\rho_j = \sum_{(i,k)\in\sigma_j} p_{i,k}\lambda_i. \tag{56}$$

Our objective is to minimize the time required to process all the $n_1 + n_2 + \cdots + n_I$ jobs, plus the jobs that arrive in the interval $[0, T^*]$, where

$$T^* = \max_j \frac{\sum_{(i,k)\in\sigma_j} p_{i,k} n_i}{1-\rho_j}. \tag{57}$$

Remarks:

- Observe that since arrivals to the network after T^* are irrelevant for our objective, we may assume that arrivals occur over a finite horizon $[0, T^*]$.
- In considering the asymptotics, we let $n \to \infty$; this will result in $T^* \to \infty$ as well, as specified in Eq. (57). We emphasize that T^* is implicitly determined by the choice of λ_i and n_i, and is *not* part of the input.

As before, we define the congestion of machine σ_j as

$$C_j = \sum_{(i,k)\in\sigma_j} p_{i,k}(n_i + \lambda_i T^*). \tag{58}$$

We denote the maximum congestion by

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

The following proposition is immediate.

Proposition 3 *The minimum makespan C^* of the job shop scheduling problem satisfies:*

$$C^* \geq C_{\max}.$$

We next show that $C_{\max} = T^*$.

Proposition 4 *The maximum congestion $C_{\max}$ of the job shop satisfies:*

$$C_{\max} = T^*.$$

Proof: Let j be such that

$$T^* = \frac{\sum_{(i,k)\in\sigma_j} p_{i,k} n_i}{1-\rho_j}.$$

Then,

$$\begin{aligned}
C_j &= \sum_{(i,k)\in\sigma_j} p_{i,k}(n_i + \lambda_i T^*) \\
&= \sum_{(i,k)\in\sigma_j} p_{i,k} n_i + \rho_j T^* \qquad \text{(by Eq. (56))} \\
&= \sum_{(i,k)\in\sigma_j} p_{i,k} n_i + \rho_j \frac{\sum_{(i,k)\in\sigma_j} p_{i,k} n_i}{1-\rho_j} \qquad \text{(by choice of } j\text{)} \\
&= T^*,
\end{aligned}$$

which shows that $C_{\max} \geq T^*$. Now, we show that $C_{j'} \leq T^*$ for an arbitrary machine j'. We have

$$\begin{aligned}
C_{j'} &= \sum_{(i,k)\in\sigma_{j'}} p_{i,k}(n_i + \lambda_i T^*) \\
&= \sum_{(i,k)\in\sigma_{j'}} p_{i,k} n_i + \rho_{j'} T^* \qquad \text{(by Eq. (56))} \\
&\leq (1-\rho_{j'})T^* + \rho_{j'} T^* \qquad \text{(by definition of } T^*\text{)} \\
&= T^*,
\end{aligned}$$

which proves that $C_{\max} \leq T^*$. ■

We next consider the associated fluid relaxation and show that a simple algorithm leads to a makespan equal to $C_{\max}$, and is, therefore, optimal.

3.2 The Fluid Job Shop Scheduling Problem

The fluid relaxation associated with the model considered in §3.1 is as follows:

$$\text{minimize} \quad \int_0^\infty 1\left\{\sum_{1\leq i\leq I, 1\leq k\leq J} x_{i,k}(t) > 0\right\} dt \tag{59}$$

$$\text{subject to} \quad x_{i,1}(t) = x_i + \lambda_i \min(t, T^*) - \mu_{i,1} T_{i,1}(t), \qquad i = 1, 2, \ldots, I,\ t \geq 0, \tag{60}$$

$$x_{i,k}(t) = \mu_{i,k-1} T_{i,k-1}(t) - \mu_{i,k} T_{i,k}(t), \quad k = 2, \ldots, J_i,\ i = 1, 2, \ldots, I,\ t \geq 0, \tag{61}$$

$$0 \leq \sum_{(i,k)\in\sigma_j} (T_{i,k}(t_2) - T_{i,k}(t_1)) \leq t_2 - t_1, \ \forall\ t_2 > t_1,\ t_1, t_2 \geq 0,\ j = 1, 2, \ldots, J, \tag{62}$$

$$x_{i,k}(t) \geq 0, \quad T_{i,k}(t) \geq 0. \tag{63}$$

The objective function (59) represents the total time that at least one of the fluid levels is positive, and corresponds to the minimum makespan schedule in the discrete problem. The only difference from the model of §2.2 is in Eq. (60), where the additional term $\lambda_i \min(t, T^*)$ represents the (fractional) number of external arrivals of type i jobs up to time t.

Similar to the definition for the discrete problem, we define congestion in machine σ_j as

$$C_j = \sum_{(i,k)\in\sigma_j} p_{i,k}(x_i + \lambda_i T^*) \tag{64}$$

and the maximal congestion as

$$C_{\max} = \max_{1\leq j\leq J} C_j. \tag{65}$$

We next show that the fluid relaxation can be solved explicitly.

Proposition 5 *The fluid relaxation (59) has an optimal value equal to the maximum congestion* $C_{\max}$.

Proof: We first show that the maximum congestion $C_{\max}$ is a lower bound on the optimal value of the control problem. For any positive time t and for each $i \le I$, $k \le J_i$, we have from Eqs. (60), (61):

$$\sum_{l=1}^{k} x_{i,l}(t) = x_i + \lambda_i \min(t, T^*) - \mu_{i,k} T_{i,k}(t).$$

Let $t \le T^*$, and let j be an index that achieves the minimum in Eq. (57). For machine σ_j we obtain:

$$\begin{aligned}
\sum_{(i,k)\in\sigma_j} p_{i,k} \sum_{l=1}^{k} x_{i,l}(t) &= \sum_{(i,k)\in\sigma_j} p_{i,k}(x_i + \lambda_i \min(t, T^*)) - \sum_{(i,k)\in\sigma_j} T_{i,k}(t) \\
&\ge \sum_{(i,k)\in\sigma_j} p_{i,k}(x_i + \lambda_i t) - t \qquad \text{(Constraint (62) applied to} \\
&\qquad\qquad\qquad\qquad\qquad\qquad t_1 = 0, t_2 = t) \\
&= \sum_{(i,k)\in\sigma_j} p_{i,k} x_i - (1 - \rho_j)t \qquad \text{(by Eq. (56))} \\
&= (1 - \rho_j)(T^* - t) \qquad \text{(by the choice of } j\text{, and Eq. (57))}
\end{aligned}$$

It follows then, that the fluid levels at σ_j are positive for all times t smaller than T^*. Therefore, the objective value of the fluid relaxation is at least T^*, which, by Proposition 4, equals $C_{\max}$.

We now construct a feasible solution that achieves this value. For each $i \le I$, $k \le J_i$ and each $t \le C_{\max}$ we let

$$\begin{aligned}
T_{i,k}(t) &= \frac{p_{i,k}(x_i + \lambda_i C_{\max})}{C_{\max}} t, \\
x_{i,1}(t) &= x_i + \lambda_i t - \mu_{i,1} T_{i,1}(t) = x_i + \lambda_i t - \frac{(x_i + \lambda_i C_{\max})}{C_{\max}} t, \quad i = 1, \ldots, I, \\
x_{i,k}(t) &= 0, \qquad\qquad k = 2, 3, \ldots, J_i, \ i = 1, \ldots, I.
\end{aligned}$$

For all $t > C_{\max}$ we set $T_{i,k}(t) = p_{i,k}(x_i + \lambda_i C_{\max})$, $x_{i,k}(t) = 0$. Clearly, this solution has an objective value equal to $C_{\max}$. We now show that this solution is feasible. It is nonnegative by construction. Also by construction, Eq. (60) is satisfied for all $t \le C_{\max}$. In particular, $x_{i,1}(C_{\max}) = 0$, $i = 1, 2, \ldots, I$. Moreover, for all i, $k = 2, 3, \ldots, J_i$ and $t \le C_{\max}$ we have:

$$\mu_{i,k-1} T_{i,k-1}(t) - \mu_{i,k} T_{i,k}(t) = \frac{(x_i + \lambda_i C_{\max})}{C_{\max}} t - \frac{(x_i + \lambda_i C_{\max})}{C_{\max}} t = 0 = x_{i,k}(t),$$

and Eq. (61) is satisfied. Finally, for any $t_1 < t_2 \le C_{\max}$ and for any machine σ_j, we have:

$$\begin{aligned}
\sum_{(i,k)\in\sigma_j} (T_{i,k}(t_2) - T_{i,k}(t_1)) &= \sum_{(i,k)\in\sigma_j} \left(\frac{p_{i,k}(x_i + \lambda_i C_{\max})}{C_{\max}} t_2 - \frac{p_{i,k}(x_i + \lambda_i C_{\max}}{C_{\max}} t_1 \right) \\
&= \frac{C_j}{C_{\max}} (t_2 - t_1) \\
&\le t_2 - t_1,
\end{aligned}$$

and Constraint (62) is satisfied. Note, that for the constructed solution $x_{i,k}(C_{\max}) = 0$ for all $i \leq I, k \leq J_i$. Therefore the feasibility for times $t > C_{\max}$ follows trivially. ■

In the following section we prove that algorithm FSA yields an asymptotically optimal solution for the original discrete job shop scheduling problem.

3.3 The Fluid Synchronization Algorithm

Recall that algorithm FSA of §2.3 relied on the notion of *nominal start times*. Our plan for this section is quite simple: we describe an analogous definition of *nominal start times* for the model with arrivals, and argue that all of the results of §2.3 carry over under this new definition also.

Definitions: The definitions of $DS_{i,k}(n)$ and $DC_{i,k}(n)$ are the same as before. We now present the definitions of $FS_{i,k}(n)$ and $NS_{i,k}(n)$. In the following, we let n index the jobs; the first n_i jobs are the ones that are initially present in the network. Jobs $n_i+1,\ldots,n_i+e_i$ are the type i jobs that arrived from outside in the interval $[0, T^*]$. Let $a_i(l)$ be the arrival time of the l^{th} external arrival of type i, for $l = 1, 2, \ldots, e_i$.

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

$$FS_{i,k}(1) = 0, \tag{66}$$

$$FS_{i,k}(n) = FS_{i,k}(n-1) + \frac{C_{\max}}{(n_i + \lambda_i C_{\max})}, \quad n > 1. \tag{67}$$

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

$$NS_{i,1}(n) = FS_{i,1}(n), \quad n = 1, 2, \ldots, n_i, \tag{68}$$

$$NS_{i,1}(n_i + l) = \max\{FS_{i,1}(n_i + l), a_i(l)\} \quad l = 1, 2, \ldots, e_i, \tag{69}$$

$$NS_{i,k}(1) = DS_{i,k-1}(1) + p_{i,k-1}, \quad k > 1, \tag{70}$$

$$NS_{i,k}(n) = \max\left\{NS_{i,k}(n-1) + \frac{C_{\max}}{(n_i + \lambda_i C_{\max})}, DS_{i,k-1}(n) + p_{i,k-1}\right\}, \quad n, k > 1. \tag{71}$$

As before, at every scheduling epoch for the discrete network, algorithm FSA schedules a job with the smallest nominal start time. To prove that FSA yields an asymptotically optimal schedule, we need to prove analogs of Theorems 6 and 7. Clearly, Theorem 6 remains true in this setting as well. In the proof of Theorem 7, we used $NS_{i,1}(n) = FS_{i,1}(n)$ for all i, n to establish the basis for the induction; this must be established for the model under consideration for $n_i < n \leq n_i + e_i$. It is easy to see that the proof of Theorem 7 would follow if we can prove that $NS_{i,1}(n) = FS_{i,1}(n)$ for all i, n.

Lemma 10 *Let $FS_{i,k}(n)$ and $NS_{i,k}(n)$ be defined as in Eqs. (66) and (71). Then,*

$$NS_{i,1}(n) = FS_{i,1}(n), \qquad \forall\ i, n. \tag{72}$$

Proof: Fix a job type i. The Lemma is trivially true (by Eq. (68)) for $n \leq n_i$. To establish Eq. (72) for $n > n_i$, we only need to show that for any $1 \leq l \leq e_i$,

$$FS_{i,1}(n_i + l) \geq a_i(l) \tag{73}$$

Since arrivals are deterministic,

$$a_i(l) = \frac{l-1}{\lambda_i}. \tag{74}$$

From Eq. (66),

$$FS_{i,1}(n_i + l) = (n_i + l - 1)\,\frac{C_{\max}}{n_i + \lambda_i C_{\max}}. \tag{75}$$

Thus,

$$\begin{aligned}
\frac{FS_{i,1}(n_i + l)}{a_i(l)} &= (n_i + l - 1)\,\frac{C_{\max}}{n_i + \lambda_i C_{\max}}\,\frac{\lambda_i}{l-1} \\
&= \frac{n_i C_{\max}\lambda_i + (l-1)C_{\max}\lambda_i}{n_i(l-1) + (l-1)C_{\max}\lambda_i} \\
&\geq 1 \qquad (\text{because } e_i \leq \lambda_i C_{\max} + 1).
\end{aligned}$$

■

Thus, Theorems 6 and 7 remain true for this model, which in turn imply Theorem 8. These observations prove the following analog of Theorem 9.

Theorem 11 *Consider a job shop scheduling problem with I job types and J machines $\sigma_1, \sigma_2, \ldots, \sigma_J$. Suppose we are given initially n_i jobs of type $i = 1, 2, \ldots, I$; suppose also that external arrivals of type i jobs occur deterministically at rate λ_i over the horizon $[0, T^*]$, where T^* is given by Eq. (57). Then, the algorithm FSA produces a schedule with makespan time C_{D} such that*

$$C_{\max} \leq C^* \leq C_{\mathrm{D}} \leq C_{\max} + \max_{i=1,2,\cdots,I} \sum_{l=1}^{J_i} (2\ P_{\sigma_l^i} + U_{\sigma_l^i}). \tag{76}$$

In particular,

$$\frac{C_{\rm D}}{C^*} \leq \frac{C_{\rm D}}{C_{\max}} \to 1, \tag{77}$$

as

$$\sum_{i=1}^{I} n_i \to \infty,$$

where C^* is the optimal makespan.

From Theorem 11, we see that the additive error of the schedule computed by algorithm FSA is bounded from above by $J_{\max}(2\ P_{\max} + U_{\max})$. As before, considering a flow shop establishes that an an additive error of $(J_{\max} - 1)P_{\max}$ is necessary for any algorithm that uses the optimal fluid cost for comparison purposes. Improved results for the model of §2 will directly result in improvements for the model with arrivals, which makes the problem of designing algorithms that achieve the optimal additive error both interesting and important.

N	$C_{\max}$	C_D	$C_D - C_{\max}$	$(C_D - C_{\max})/C_{\max}$
1	631	1189	558	0.8843
2	1262	1820	558	0.4421
3	1893	2546	653	0.3450
4	2524	3073	549	0.2175
5	3155	3740	585	0.1854
6	3786	4313	527	0.1392
7	4417	4975	558	0.1263
8	5048	5608	560	0.1109
9	5679	6170	491	0.0865
10	6310	6842	532	0.0843
20	12620	13159	539	0.0427
30	18930	19509	579	0.0306
40	25240	25779	539	0.0214
50	31550	32041	491	0.0156
100	63100	63639	539	0.0085
1000	631000	631584	584	0.0009

Table 10: Computational results for the 10×10 example.

4 Computational Results

We present in this section computational results based on algorithm FSA. In our first experiment, we consider the famous 10 by 10 instance in Muth and Thompson [18] with $n_i = N$ jobs present and vary N. Table 10 shows the performance of algorithm FSA for N ranging from 1 to 1000. This class of instances has $J_{\max} = 10$, $I = 10$ and $P_{\max} = 98$. As Theorem 9 predicts the gap between C_D and $C_{\max}$ is insensitive to N. What is striking is that the actual gap is substantially smaller than the bound predicted in Eq. (54). While bound (54) gives an upper bound to the error of 5,000, the actual error has never been higher than 653. Moreover, the actual gap is even smaller than $J_{\max}P_{\max} = 980$. Finally, note for $N = 100$ jobs present for each type, the relative error is less than 1%, while for $N = 1000$, the error is less than 0.1%.

We have performed an extensive computational study of all benchmark job shop instances available as part of the OR library (http://mscmga.ms.ic.ac.uk/info.html). The results reported in Table 11 are for 82 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 report results for $N = 1$, $N = 2$, $N = 5$, $N = 10$ and $N = 100$, and $N = 500$. The lower bound based on the fluid relaxation, $C_{\max}$, is shown in the second column, and is valid for $N = 1$; the lower bound for $N = n$ is $n\ C_{\max}$. The subsequent columns report the value of $C_D - n\ C_{\max}$. In the final four columns we report the values of $P_{\max}$, $J_{\max}$, the value of the error (54), and the value of $(J_{\max} - 1)P_{\max}$.

We can make the following observations from the results reported in Table 11.

1. The gap between C_D and $nC_{\max}$ is insensitive to n as predicted by Theorem 9. Moreover, the actual gap is significantly smaller than the one predicted by the bound (54) by an order of magnitude.

2. Asymptotic results usually require very large values of n in order for their performance to be in a practical range. This is not the case for algorithm FSA. The relative error is about 10% for $n = 10$, 1% for $n = 100$, 0.05% for $n = 500$. Especially if one considers that the gap is between the performance of algorithm FSA and the trivial lower bound $C_{\max}$, the performance of algorithm FSA compared to the true optimal value will be even better.

3. 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, algorithm FSA represents in our opinion **a practical** algorithm for solving job shop scheduling problems of even moderate multiplicity. Every one of the instances in Table 11 was solved in under five seconds on a Ultra 10 Sun workstation.

Benchmark	$C_{\max}$ (N = 1)	$C_D - N\ C_{\max}$ N = 1	N = 2	N = 5	N = 10	N = 100	N = 500	$P_{\max}$	$J_{\max}$	Error (54)	$(J_{\max}-1)P_{\max}$
abz5	868	599	452	457	443	567	475	99	10	9725	891
abz6	688	368	350	412	234	270	256	98	10	7758	882
abz7	556	233	255	227	194	43	79	40	15	8506	560
abz8	566	294	269	140	131	84	93	40	15	8766	560
abz9	563	366	234	135	122	110	102	40	15	8614	560
ft06	43	19	33	21	21	21	21	10	6	305	50
ft10	631	558	558	585	532	539	545	99	10	6883	891
ft20	1119	526	477	426	341	54	54	99	5	6055	396
la01	666	106	44	85	86	86	86	98	5	3765	392
la02	635	219	98	100	100	100	100	99	5	3579	396
la03	588	174	115	46	0	0	0	91	5	3225	364
la04	537	158	153	114	100	132	132	98	5	3429	392
la05	593	17	0	0	0	0	0	97	5	3163	388
la06	926	0	0	0	0	0	0	98	5	4916	392
la07	869	219	105	67	67	67	67	97	5	4655	388
la08	863	117	92	0	0	0	0	98	5	4753	392
la09	951	67	23	0	0	0	0	99	5	5207	396
la10	958	48	7	7	7	7	7	97	5	4950	388
la11	1222	50	0	0	0	0	0	98	5	6275	392
la12	1039	0	0	0	0	0	0	98	5	5606	392
la13	1150	49	0	0	0	0	0	97	5	6144	388
la14	1292	0	0	0	0	0	0	97	5	6276	388
la15	1207	380	232	46	44	44	44	99	5	6411	396
la16	660	520	477	328	335	277	277	98	10	7151	882
la17	683	260	242	181	259	259	259	98	10	6402	882
la18	623	421	373	346	262	219	264	97	10	7010	873
la19	685	298	148	32	34	34	34	97	10	7156	873
la20	744	528	345	175	160	160	160	99	10	7317	891
la21	935	471	432	172	171	160	160	99	10	9900	891
la22	830	482	374	439	439	439	439	98	10	9176	882
la23	1032	250	49	142	0	0	0	97	10	9951	873
la24	857	331	186	129	129	59	59	99	10	9615	891
la25	864	344	408	99	23	29	29	99	10	9389	891
la26	1218	154	236	11	0	4	4	99	10	12421	891
la27	1188	456	421	401	280	121	121	99	10	12748	891
la28	1216	258	149	60	157	157	157	99	10	12614	891
la29	1105	473	413	226	236	215	215	99	10	11831	891
la30	1355	293	14	0	0	0	0	99	10	12562	891
la31	1784	150	0	0	0	0	0	99	10	17107	891
la32	1850	260	79	0	0	0	0	99	10	18527	891
la33	1719	154	73	66	66	66	66	99	10	16881	891

continued on next page

continued from previous page											
	$C_{\max}$	$C_D - N\,C_{\max}$						$P_{\max}$	$J_{\max}$	Error (54)	$(J_{\max}-1)P_{\max}$
Benchmark	(N = 1)	N = 1	N = 2	N = 5	N = 10	N = 100	N = 500				
la34	1721	240	41	0	0	0	0	99	10	17275	891
la35	1888	254	8	8	8	8	8	99	10	17403	891
la36	1028	488	393	468	388	409	406	99	15	14507	1386
la37	980	881	711	530	330	406	406	99	15	15381	1386
la38	876	590	610	345	373	99	99	99	15	14083	1386
la39	1012	520	497	557	604	679	679	99	15	14314	1386
la40	1027	504	346	278	233	193	161	99	15	14260	1386
orb01	643	736	720	720	740	740	740	99	10	7317	891
orb02	671	336	322	246	271	134	243	99	10	7117	891
orb03	624	781	792	829	798	819	819	99	10	7162	891
orb04	759	566	447	337	337	371	371	98	10	7506	882
orb05	630	525	557	397	390	390	390	99	10	6698	891
orb06	659	671	731	696	746	746	746	99	10	7476	891
orb07	286	189	162	162	140	132	132	59	10	3305	531
orb08	585	640	823	763	755	774	725	97	10	6376	873
orb09	661	528	492	554	448	448	448	99	10	7049	891
orb10	652	651	553	415	415	408	408	99	10	7431	891
swv01	1219	935	996	996	996	996	996	100	10	11595	900
swv02	1259	898	828	828	828	828	828	100	10	11945	900
swv03	1178	841	772	834	834	834	834	100	10	11809	900
swv04	1161	964	895	842	737	608	645	100	10	12192	900
swv05	1235	783	782	767	767	767	767	100	10	11991	900
swv06	1229	1290	1161	1130	1103	1067	1067	100	15	18111	1400
swv07	1128	1111	1096	1156	999	977	977	100	15	17472	1400
swv08	1330	1174	1062	983	1009	1091	1091	100	15	18679	1400
swv09	1266	1232	1235	1189	1189	1189	1127	100	15	17790	1400
swv10	1159	1147	1659	1353	1370	1286	1286	100	15	18599	1400
swv11	2808	1619	1619	1619	1619	1619	1619	100	10	27383	900
swv12	2829	1854	1830	1742	1602	888	902	100	10	27510	900
swv13	2977	1852	1774	1752	1736	1752	1752	100	10	27820	900
swv14	2842	1779	1684	1684	1684	1684	1684	100	10	26985	900
swv15	2762	1858	1931	1877	1787	898	1013	100	10	26959	900
swv16	2924	27	54	54	54	54	54	100	10	27313	900
swv17	2794	168	171	146	114	114	114	100	10	26411	900
swv18	2852	122	99	108	108	108	108	100	10	26641	900
swv19	2843	252	67	0	0	0	0	100	10	27581	900
swv20	2823	45	0	0	0	0	0	100	10	26585	900
yn1	643	473	409	287	226	222	232	49	20	13660	931
yn2	686	509	418	357	333	288	318	49	20	13652	931
yn3	659	476	392	292	187	213	215	49	20	13475	931
											continued on next page

continued from previous page

Benchmark	$C_{\max}$ (N = 1)	$C_D - N\ C_{\max}$ N = 1	N = 2	N = 5	N = 10	N = 100	N = 500	$P_{\max}$	$J_{\max}$	Error (54)	$(J_{\max}-1)P_{\max}$
yn4	676	521	563	517	430	340	326	49	20	14000	931

Table 11: Computational results for Job Shop instances in OR-Library.

5 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. FSA is attractive from a practical perspective. First, it is very simple to implement and it is very fast. Second, its performance on all problems in the OR library 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, FSA should be considered a candidate for practical application.

An interesting open problem is to find the tightest upper bound on the error given in (54). Given a worst case example with additive gap $(J_{\max}-1)P_{\max}$, it is tempting to conjecture that the answer might be $(J_{\max}-1)P_{\max}$; we refer the reader to Sevast'janov [24] for related conjectures and open problems.

Acknowledgements

We thank David Gamarnik and a referee for helpful comments on the paper.

References

[1] J. Beck. An algorithmic approach to the lovász local lemma. *Random Structures and Algorithms*, 2:343–365, 1991.

[2] I. S. Belov and Ya. N. Stolin. An algorithm in a single path operations scheduling problem. *Mathematical Economics and Functional Analysis*, pages 248–257, 1974. (in Russian).

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

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

[5] J. Carlier and E. Pinson. An algorithm for solving the job-shop problem. *Management Science*, 35:164–176, 1985.

[6] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Internetworking Research and Experience*, 1, 1990.

[7] T. Fiala. Kozelito algorithmus a harom gep problemeara. *Alkalmazott Matematikai Lapok*, 3:389–398, 1977.

[8] L. A. Goldberg, M. Paterson, A. Srinivasan, and E. Sweedyk. Better approximation guarantees for job-shop scheduling. In *Proceedings of the eighth ACM-SIAM symposium on Discrete Algorithms*, pages 599–608, 1997.

[9] T. Gonzalez and S. Sahni. Flowshop and jobshop schedules: complexity and approximation. *Operations Research*, 26:36–52, 1978.

[10] A. G. Greenberg and N. Madras. How fair is fair queueing? *Journal of the Association for Computing Machinery*, 39:568–598, 1992.

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

[12] 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.

[13] K. Jansen, R. Solis-Oba, and M. Sviridenko. A linear time approximation scheme for job shop scheduling problems. In *Proceedings of APPROX '99*, 1999. (to appear).

[14] K. Jansen, R. Solis-Oba, and M. Sviridenko. Makespan minimization in job shops: a polynomial time approximation scheme. In *Proceedings of the Symposium on Theory of Computing*, pages 394–399, 1999.

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

[16] F. T. Leighton, B. Maggs, and S. Rao. Universal packet routing algorithms. In *Proceedings of the* 29^{th} *Annual Symposium on Foundations of Computer Science*, pages 256–269, 1988.

[17] F. T. Leighton, B. Maggs, and A. Richa. Fast algorithms for finding o(congestion+dilation) packet routing schedules. *Combinatorica*, 19:375–401, 1999.

[18] J. F. Muth and G. L. Thompson, editors. *Industrial Scheduling*, Englewood Cliffs, NJ, 1963. Prentice-Hall.

[19] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, 1:344–357, 1993.

[20] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple-node case. *IEEE/ACM Transactions on Networking*, 2:137–150, 1994.

[21] S. V. Sevast'yanov. On an asymptotic approach to some problems in scheduling theory. In *Abstracts of papers at* 3^{rd} *All-Union Conference of Problems of Theoretical Cybernetics*, pages 67–69, Novosibirsk, 1974. Inst. Mat. Sibirsk. Otdel. Akad. Nauk SSSR.

[22] S. V. Sevast'yanov. Efficient construction of schedules close to optimal for the cases of arbitrary and alternative routes of parts. *Soviet Math. Dokl.*, 29(3):447–450, 1984.

[23] S. V. Sevast'yanov. Bounding algorithm for the routing problem with arbitrary paths and alternative servers. *Kibernetika*, 22(6):74–79, 1986. Translation in *Cybernetics*, 22:773–780.

[24] S. V. Sevast'yanov. On some geometric methods in scheduling theory: a survey. *Discrete Applied Mathematics*, 55:59–82, 1994.

[25] D. B. Shmoys, C. Stein, and J. Wein. Improved approximation algorithms for shop scheduling problems. *SIAM Journal on Computing*, 23(3):617–632, 1994.

[26] G. Weiss. On optimal draining of fluid re-entrant lines. In F. P. Kelly and R. J. Williams, editors, *Stochastic Networks*, volume 71 of *Proceedings of the International Mathematics Association*, pages 91–103. Springer-Verlag, New York, 1995.

[27] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, October 1995.