Asymptotically optimal algorithms for job shop scheduling and packet routing

Dimitris Bertsimas * David Gamarnik †

January 1998, revised June 1999

Abstract

We propose asymptotically optimal algorithms for the job shop scheduling and packet routing problems. We propose a fluid relaxation for the job shop scheduling problem, in which we replace discrete jobs with the flow of a continuous fluid. We compute an optimal solution of the fluid relaxation in closed form, obtain a lower bound $C_{\max}$ to the job shop scheduling problem, and construct a feasible schedule from the fluid relaxation with objective value at most $C_{\max}+O\left(\sqrt{C_{\max}}\right)$, where the constant in the $O(\cdot)$ notation is independent of the number of jobs, but it depends on the processing times of the jobs, thus producing an asymptotically optimal schedule as the total number of jobs tends to infinity. If the initially present jobs increase proportionally, then our algorithm produces a schedule with value at most $C_{\max}+O(1)$. For the packet routing problem with fixed paths the previous algorithm applies directly. For the general packet routing problem we propose a linear programming relaxation that provides a lower bound $C_{\max}$, and an asymptotically optimal algorithm that uses the optimal solution of the relaxation with objective value at most $C_{\max}+O\left(\sqrt{C_{\max}}\right)$. Unlike asymptotically optimal algorithms that rely on probabilistic assumptions, our proposed algorithms make no probabilistic assumptions, and they are asymptotically optimal for all instances with a large number of jobs (packets). In computational experiments our algorithms produce schedules which are within 1% of optimality even for moderately size problems.

*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.

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

1 Introduction

The job shop scheduling and the packet routing problems are fundamental problems in Operations Research and Computer Science. The job shop scheduling problem is the the problem of scheduling a set of I job types on J machines. Job type i consists of J_i stages, each of which must be completed on a particular machine. The pair (i, j) represents the jth stage of the ith job, and has processing time $p_{i,j}$. The completion time of job type i is the completion time of the last stage J_i of job type i. Assuming that we have n_i jobs of type i, the objective is to find a schedule that minimizes the maximum completion time, called the makespan, subject to the following restrictions:

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 a classical $\mathcal{NP}$-hard problem notoriously difficult to solve even relatively small instances. As an example, a specific instance involving 10 machines and 10 jobs posed in a book by Muth and Thompson [11] in 1963 remained unsolved for over 20 years until solved by Carlier and Pinson [2] in 1985.

The packet routing problem in a communication network $(V, \mathcal{A})$ is the problem of routing a collection of packets from a source node to a destination node. It takes one time unit for a packet to traverse an edge in $\mathcal{A}$, and only one packet can traverse a given edge at a time. As in the job shop scheduling problem, the objective is to find a schedule that minimizes the time, called the makespan, that all packets are routed to their destinations. For the case that the paths along which packets need to be routed are given, the problem can be modeled exactly as a job shop scheduling problem. However, when we can select the paths along which to route packets, the problem is more complicated as it involves both routing (path selection) as well as sequencing (which packet each edge processes) decisions.

Our overall approach for these problems relies on two ideas from two distinct communities. First, we consider a relaxation for the job shop scheduling problem called the fluid control problem, 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. Multiclass queueing networks are stochastic and dynamic versions of job shops. In recent years there has been considerable progress in solving the fluid control problem in multiclass queueing networks. Focusing on objective functions that minimize a weighted combination of number of jobs at the various machines, as opposed to makespan, Avram, Bertsimas, and Ricard [1] show that by using the Pontryagin maximum principle, we can find the optimal control explicitly. However, the description of the optimal control, while insightful for the original problem, involves the enumeration of an exponential number of cases. Luo and Bertsimas [10], building upon the work of Pullan [12], 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. For the objective we consider (minimize the length of the schedule, i.e., the maximum completion time) the optimal solution of the fluid control problem can be computed in closed form and provides a lower bound $C_{\max}$ to the job shop scheduling problem. Weiss [17] has considered and solved the makespan objective for a fluid control problem with arrivals. Our proof of the fluid control problem without arrivals follows along similar lines.

The second idea of the paper is motivated by the considerable progress in the deterministic scheduling community in providing approximation algorithms for scheduling problems that rounds the solution of a linear programming relaxation of the scheduling problem. Shmoys, Stein and Wein [15], Goldberg et. al. [4] and Feige and Scheideler [3] provide algorithms that are within a multiplicative logarithmic guarantee from the optimal solution value. Very recently Jansen, Solis-Oba, and Sviridenko [6] provide a polynomial time approximation scheme. For a review of this approach see Hall [5] and Karger, Stein and Wein [7]. However, the paper closest in spirit to the current work is a scheduling algorithm for job shop problems constructed by Sevast'janov [13] (see also [14]). Sevast'janov's algorithm is based on an interesting geometric method, unrelated with the methods of the current paper, and produces a schedule with length $C_{\max} + O(1)$, and as a result, is asymptotically optimal as the number of jobs tends to infinity. Our algorithm is significantly simpler than Sevast'janov's and produces superior bounds for a variety of instances. For example, for the 10 by 10 instance defined in Muth and Thompson [11] with the same number n of jobs for every job type, then the bound for our algorithm is always stronger for all n. We compare the bounds given by our methods and those by Sevast'janov in Section 4.3.

We use the optimal solution of the fluid control problem to construct a feasible schedule with objective value $C_{\max} + O(\sqrt{C_{\max}})$. If the initially present jobs increase proportionally, then our algorithm produces a schedule with value at most $C_{\max} + O(1)$. Similarly, for the packet routing problem we propose a linear programming relaxation that provides a lower bound $C_{\max}$ and use its solution to construct a feasible schedule with objective value $C_{\max} + O(\sqrt{C_{\max}})$. We note that the constant in the $O(\cdot)$ notation is independent of the number of jobs, but it does depend on the processing times of the jobs. This implies that as the total number of jobs (packets respectively) tends to infinity, the proposed algorithm is asymptotically optimal. Unlike asymptotically optimal algorithms that rely on probabilistic assumptions, the above algorithm makes no probabilistic assumptions, and it is asymptotically optimal for all instances with a large number of jobs (packets respectively). The classical result in this area is the work of Karp [8], who provided an asymptotically optimal algorithm for the traveling salesman problem when the points are randomly and uniformly distributed in the unit square in the Euclidean plane.

The combinatorial structure of the job scheduling problem makes the problem very complicated to solve when there is a small number of jobs in the system. Interestingly, the results of the 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. Similarly, the packet routing problem has an even richer combinatorial structure. The results of the paper also imply that a continuous approximation to the problem is asymptotically exact.

The paper is structured as follows. In Section 2, we formulate the job shop scheduling problem, and describe the notation. In Section 3, we introduce the fluid control problem for the job shop scheduling problem, and solve it in closed form. In Section 4, we present and analyze the rounding algorithm, called the synchronization algorithm. We also provide some computational results and contrast our bounds with those by Sevast'janov [13]. In Section 5, we address packet routing in communication networks with fixed paths as an application of job shop scheduling. In Section 6, we propose an asymptotically optimal algorithm for the general packet routing problem in communication networks. Section 7 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 $\sigma_1^i, \sigma_2^i, \ldots, \sigma_{J_i}^i$ in that order. Let $J_{\max} = \max_i J_i$. The time to process a type i job on machine σ_k^i is denoted by $p_{i,k}$. Throughout the paper 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." The number of jobs in stages $k \geq 2$ in any specific machine σ_j is called noninitial queue in machine σ_j. In particular, at time zero all noninitial queues are empty.

We will also think of each machine σ_j as a collection of all type and stage pairs that it processes. Namely, for each $j = 1, 2, \ldots, J$

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

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 $n = n_1 + n_2 + \cdots + n_I$ jobs on machines $\sigma_1, \ldots, \sigma_J$, so that the time it takes to process all the jobs is minimized.

Each machine σ has a certain processing time required to process jobs that eventually come to this machine. Specifically, for machine σ_j this time 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:*

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

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 the next section). For the fluid control problem we show that a simple algorithm leads to a makespan equal to $C_{\max}$, and therefore is optimal.

3 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 processing machines $\sigma_1, \sigma_2, \ldots, \sigma_J$, I job types, each specified by the sequence of machines σ_k^i, $k = 1, 2, \ldots, J_i$, 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 control problem 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 control problem of minimizing makespan can be formulated as follows:

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

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

$$x_{i,k}(t) = \mu_{i,k-1} T_{i,k-1}(t) - \mu_{i,k} T_{i,k}(t), \qquad k = 2, \ldots, J_i, \ i = 1, 2, \ldots, I, \ t \ge 0, \tag{3}$$

$$\sum_{(i,k) \in \sigma_j} (T_{i,k}(t_2) - T_{i,k}(t_1)) \le t_2 - t_1, \qquad \forall \ t_2 t_1, \ t_1, t_2 \ge 0, \ j = 1, 2, \ldots, J. \tag{4}$$

$$x_{i,k}(t) \ge 0, \quad T_{i,k}(t) \ge 0. \tag{5}$$

The objective function (1) 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 (2), (3) 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 $k1$) 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 (4) is just the aggregate feasibility constraint for machine σ_j.

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

$$C_j = \sum_{(i,k)\in\sigma_j} p_{i,k}x_i, \tag{6}$$

and the maximal congestion as

$$C_{\max} = \max_{1\le j\le J} C_j. \tag{7}$$

We next show that the fluid control problem can be solved in closed form.

Proposition 2 *The fluid control problem (1) 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 (2), (3):

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

For each station 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 (4) 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 optimal control problem 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

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

For all $tC_{\max}$ 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. (2) is satisfied for all $t \leq C_{\max}$. In particular, $x_{i,1}(C_{\max}) = 0$, $i = 1, 2, \ldots, I$. Moreover, for all i, $k = 2, 3, \ldots, J_i$ and $t \leq C_{\max}$ 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. (3) is satisfied. Finally, for any $t_1 < t_2 \leq C_{\max}$ and for any machine σ_j, we have:

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

and constraint (4) 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 $tC_{\max}$ follows trivially. ■

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 the fluid solution to construct an asymptotically optimal solution for the original discrete job shop scheduling problem.

4 An Algorithm for the Job Shop Scheduling Problem

In this section, we consider the original job shop scheduling problem, described in Section 2. Recall that we are initially given n_i jobs for each job type i, $i = 1, 2, \ldots, I$, where n_i is some nonnegative integer. Station σ_j has congestion C_j given by $\sum_{(i,k)\in\sigma_j} p_{i,k} n_i$. Let again $C_{\max}$ denote the maximal congestion. Let Ω be a certain positive real value. An exact value for Ω will be specified later. For each job type i we let

$$a_i = \left\lceil \frac{n_i \Omega}{C_{\max}} \right\rceil, \tag{8}$$

For each station σ_j let

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

Namely, U_j is the workload of station σ_j when only one job per type is present. Finally, let

$$U_{\max} = \max_{1\leq j\leq J} U_j. \tag{9}$$

The proposed algorithm revisits the schedule in time intervals of length $\Omega + U_{\max}$.

The Synchronization Algorithm.

For each interval $[m(\Omega+U_{\max}), (m+1)(\Omega+U_{\max})]$, $m = 0, 1, \ldots$, of length $\Omega+U_{\max}$, each machine σ_j, and each pair $(i,k) \in \sigma_j$, machine σ_j processes exactly a_i jobs of type i (which takes $p_{i,k}a_i$ time units) and idles

$$\Omega + U_{\max} - \sum_{(i,k)\in\sigma_j} p_{i,k}a_i$$

time units. If for some i, at time $m(\Omega+U_{\max})$, the number of type i jobs in machine σ_j is less than a_i, then machine σ_j processes all the available type i jobs and idles for the remaining time.

Note that the synchronization algorithm produces a feasible schedule, since for each machine σ_j and each $(i,k) \in \sigma_j$, it takes $p_{i,k}a_i$ time units to process a_i jobs of type i. Since

$$\sum_{(i,k)\in\sigma_j} p_{i,k}a_i \leq \sum_{(i,k)\in\sigma_j} \Big(\frac{n_i\Omega}{C_{\max}} + 1\Big)p_{i,k} \leq \Omega + U_j \leq \Omega + U_{\max}$$

it follows that the schedule is indeed feasible.

In the fluid relaxation, each job (i,k) receives $\frac{p_{i,k}n_i}{C_{\max}}$ % of the effort from the corresponding machine. The synchronization algorithm over each interval of length $\Omega + U_{\max}$ allocates time $a_i p_{i,k}$ on jobs (i,k). Thus, job (i,k) receives

$$\frac{a_i p_{i,k}}{\Omega + U_{\max}} \approx \frac{p_{i,k}n_i}{C_{\max}}\%$$

of the effort from the corresponding machine. We used approximation in the last equality, since in the synchronization algorithm we deal with discrete jobs. Intuitively, the synchronization algorithm gives essentially the same amount of effort as in the fluid relaxation.

The next theorem shows that the algorithm does a good job of synchronizing and pipelining the total workload in the system, and achieves asymptotic optimality.

Theorem 1 *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 synchronization algorithm with $\Omega = \sqrt{C_{\max}U_{\max}/J_{\max}}$ produces a schedule with makespan time C_{H} such that*

$$C_{\max} \leq C^* \leq C_{\mathrm{H}} \leq C_{\max} + 2\sqrt{C_{\max}U_{\max}J_{\max}} + U_{\max}J_{\max} \tag{10}$$

where $U_{\max}$ is defined by (9). In particular,

$$\frac{C_{\mathrm{H}}}{C^*} \le \frac{C_{\mathrm{H}}}{C_{\max}} \to 1, \tag{11}$$

as

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

where C^* is the optimal makespan. In addition, all the noninitial queue lengths at each station σ_j are at most

$$\sqrt{\frac{U_{\max} C_{\max}}{J_{\max}}} + U_{\max}. \tag{12}$$

Proof: For each $i \le I, k \le J_i$ and each integer time t, let $N_{i,k}(t)$ denote the number of type i jobs in stage k (waiting to be processed on the machine σ_k^i). Note that $N_{i,k}(0) = 0$ for all $k \ge 2$. For each i, machine σ_1^i will process exactly a_i jobs during the interval $[0, \Omega + U_{\max}]$.

Therefore,

$$N_{i,1}(\Omega + U_{\max}) = n_i - a_i, \quad N_{i,2}(\Omega + U_{\max}) = a_i, \quad N_{i,k}(\Omega + U_{\max}) = 0, k \ge 3.$$

Also, for each i machines σ_1^i, σ_2^i will process exactly a_i jobs during the interval $[\Omega + U_{\max}, 2(\Omega + U_{\max})]$. Therefore,

$$N_{i,1}(2(\Omega{+}U_{\max})) = n_i{-}2a_i,\ \ N_{i,2}(2(\Omega{+}U_{\max})) = a_i,\ \ N_{i,3}(2(\Omega{+}U_{\max})) = a_i,\ \ N_{i,k}(2(\Omega{+}U_{\max})) = 0,\ \ k \ge 3.$$

Similarly, we observe that for each $(i,k) \in \sigma_j$, machine σ_j does not receive type i jobs until time $t = (k-1)(\Omega + U_{\max})$, and during each subsequent interval $[m(\Omega + U_{\max}), (m+1)(\Omega + U_{\max})]$, $m \ge k-1$ it receives and processes exactly a_i jobs of type i, until no new jobs of type i arrive. Let

$$T = \left\lceil \frac{C_{\max}}{\Omega} \right\rceil (\Omega + U_{\max}). \tag{13}$$

Then

$$N_{i,1}(T) \le n_i - a_i \left\lceil \frac{C_{\max}}{\Omega} \right\rceil \le n_i - \frac{n_i \Omega}{C_{\max}} \frac{C_{\max}}{\Omega} = 0 \tag{14}$$

for all i. Therefore, all jobs leave the initial stage during the time interval $[0, T]$.

We next estimate the time to process all $\sum_{i=1}^{I} n_i$ jobs initially present in the system. We tag a given job θ of type i. This job, as shown in (14), leaves the first stage $(i, 1)$ at some time not bigger than T.

Given any $k \leq J_i$, suppose job θ arrives at the k-th stage at some time interval $[(m-1)(\Omega + U_{\max}), m(\Omega + U_{\max})]$. All the type i jobs that arrived at stage k before time $(m-1)(\Omega + U_{\max})$ have been processed, and job θ is one of the a_i jobs of type i that arrived at stage k (machine σ_k^i) during the time interval $[(m-1)(\Omega + U_{\max}), m(\Omega + U_{\max})]$, from the previous stage. During the time interval $[m(\Omega+U_{\max}), (m+1)(\Omega+U_{\max})]$ machine σ_k^i processes a_i jobs of type i, so job θ is in stage $k+1$ at time $(m+1)\Omega$. We conclude, that the delay for job θ between leaving the first stage and being processed at the last machine $\sigma_{J_i}^i$ is at most $(J_i - 1)(\Omega + U_{\max}) \leq (J_{\max} - 1)(\Omega + U_{\max})$.

Combining this with (14), we conclude, that the total delay for the job θ is at most

$$T + (J_i - 1)(\Omega + U_{\max}) = \left\lceil \frac{C_{\max}}{\Omega} \right\rceil (\Omega + U_{\max}) + (J_i - 1)(\Omega + U_{\max})$$

$$\leq \Big(\frac{C_{\max}}{\Omega} + 1\Big)(\Omega + U_{\max}) + (J_{\max} - 1)(\Omega + U_{\max}) =$$

$$C_{\max} + J_{\max}U_{\max} + \frac{U_{\max}C_{\max}}{\Omega} + J_{\max}\Omega.$$

Recall, that we have not specified the value of Ω. We now select Ω to be the minimizer of the expression above, i.e.,

$$\Omega = \sqrt{\frac{U_{\max}C_{\max}}{J_{\max}}}.$$

Therefore, the total makespan time under the algorithm is at most

$$C_{\mathrm{H}} \leq C_{\max} + 2\sqrt{C_{\max}U_{\max}J_{\max}} + U_{\max}J_{\max}.$$

This proves the bound (10).

Notice that the maximum congestion $C_{\max}$ tends to infinity, as the initial total number of jobs $\sum_{i=1}^{I} n_i$ tends to infinity. Therefore,

$$\frac{C_{\mathrm{H}}}{C_{\max}} \to 1$$

as $\sum_{i=1}^{I} n_i \to \infty$. From Proposition 1, $C_{\max}$ is a lower bound on the optimal makespan time C^*. Therefore, (11) follows, i.e., the synchronization algorithm is asymptotically optimal.

Finally, as we have seen, the number of type i jobs in machine σ_k^i is never more than a_i for $k \geq 2$. As a result, the noninitial queue length in machine σ_j is at most

$$\sum_{(i,k)\in\sigma_j, k\geq 2} a_i.$$

From the integrality of processing times, it follows that the noninitial queue length in machine σ_j is at most

$$\sum_{(i,k)\in\sigma_j,k\geq 2} a_i p_{i,k} \leq \sum_{(i,k)\in\sigma_j,k\geq 2} \Big(\frac{n_i}{C_{\max}}\sqrt{\frac{U_{\max}C_{\max}}{J}} p_{i,k} + p_{i,k}\Big) \leq \sqrt{\frac{U_{\max}C_{\max}}{J}} + U_{\max}.$$

This proves (12). ■

Note that the makespan C_{H} of the synchronization algorithm satisfies:

$$C_{\mathrm{H}} = C_{\max} + O\Big(\sqrt{C_{\max}}\Big).$$

4.1 The Proportional Case

In this section, we address the case with $n_i = b_i n$, and b_i are integers. In this case, we let $\Omega = \max_j \sum_{(i,k)\in\sigma_j} p_{i,k} b_i$. Then $C_{\max} = n\Omega$. We modify slightly the synchronization algorithm in this case: For each interval $[m\Omega, (m+1)\Omega]$, $m = 0, 1, \ldots$, of length Ω, each machine σ_j, and each pair $(i,k) \in \sigma_j$, machine σ_j processes exactly b_i jobs of type i (which takes $p_{i,k} b_i$ time units) and idles

$$\Omega - \sum_{(i,k)\in\sigma_j} p_{i,k} b_i$$

time units.

By following the same analysis as in Theorem 1, we show that the modified synchronization algorithm produces a schedule with makespan at most

$$C_{\max} + (J_{\max} - 1)\Omega = C_{\max} + O(1).$$

4.2 Computational Results

We have implemented three algorithms for the job scheduling problem. The first algorithm is the synchronization algorithm (called original fluid tracking heuristic in Figures 1 and 2), the second is the modified synchronization algorithm outlined in the proportional case (called new heuristic for Uniform N in Figures 1 and 2), and a final modification of the synchronization algorithm, in which the machines do not idle if they do not have work to do (called variable omega heuristic in Figures 1 and 2). We run these algorithms on the 10 by 10 instances in Muth and Thompson [11] with $n_i = N$ jobs present and varied N. Figure 1 shows the performance of the synchronization algorithm and its modification for $N \leq 50$, while Figure 2 shows the performance of all three algorithms for $N \leq 2500$. It is interesting that for $N \geq 500$, the variable Ω heuristic produces solutions within 1% from the lower bound.

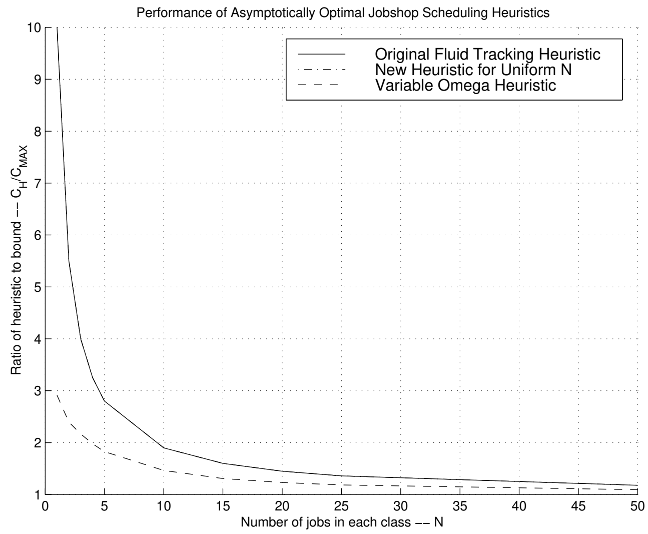

Figure 1: The performance of the synchronization algorithm and its modification for $N \leq 50$.

4.3 Comparison With Sevast'janov's Algorithm

Sevast'janov [13] has constructed a scheduling algorithm with makespan time not exceeding $C_{\max} + (J_{\max} - 1)(JJ_{\max}^2 + 2J_{\max} - 1)p_{\max}$, where $p_{\max} \equiv \max\{p_{i,k}\}$ is the maximal processing time of a single job. We now compare this performance bound with the one given by Theorem 1.

Proposition 3 **(a)** *If the total number of jobs n satisfies:* $n \leq n_0 = \dfrac{(J_{\max} - 1)(JJ_{\max}^2 + 2J_{\max} - 1)}{3\sqrt{J_{\max}}}$, *then*

$$C_{\max} + 2\sqrt{C_{\max}U_{\max}J_{\max}} + U_{\max}J_{\max} \leq C_{\max} + (J_{\max} - 1)(JJ_{\max}^2 + 2J_{\max} - 1)p_{\max}, \quad (15)$$

i.e., the upper bound on the makespan time C_{H} corresponding to the synchronization algorithm is superior to the upper bound on the Sevast'janov scheduling algorithm.

(b) *If all job types have the same number n of jobs initially present, there are at most $J_{\max}$ job types, and each job type is processed by any given machine at most once, then the bound given by*

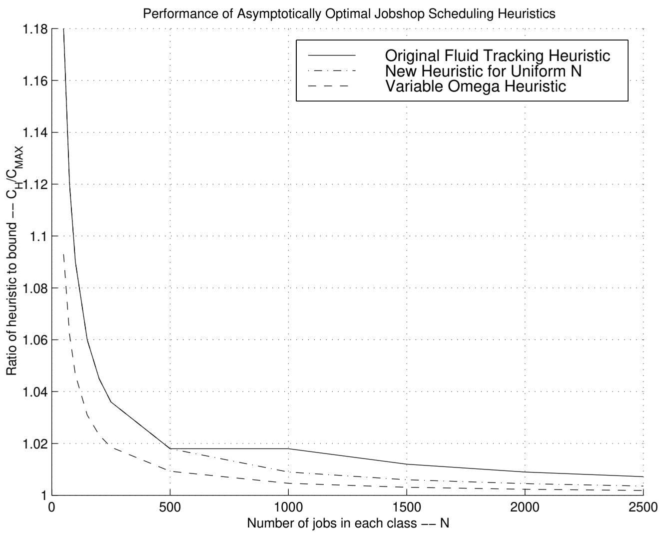

Figure 2: The performance of all three algorithms for $N \leq 2500$.

the synchronization algorithm is always stronger than the bound given by Sevast'janov scheduling algorithm.

Proof:

(a) Trivially, $U_{\max} \leq C_{\max} \leq np_{\max}$. Then, the left hand side of (15) is smaller than

$$C_{\max} + 2np_{\max}\sqrt{J_{\max}} + np_{\max}J_{\max}$$

Therefore, the left hand side of (15) is smaller than

$$C_{\max} + 2np_{\max}J_{\max},$$

which is less than or equal to

$$C_{\max} + (J_{\max} - 1)(JJ_{\max}^2 + 2J_{\max} - 1)p_{\max},$$

if $n \leq n_0$.

(b) If all job types have the same number n of jobs initially present, i.e., $n_i = n$, the performance bound of the synchronization algorithm is

$$C_{\max} + (J_{\max} - 1)U_{\max}.$$

If there are at most $J_{\max}$ job types, and each job type is processed by any given machine at most once, then $U_{\max} \leq J_{\max} p_{\max}$, and thus the bound given by the synchronization algorithm is always stronger. ■

For example, for the 10 by 10 instance defined in Muth and Thompson [11] with the same number n of jobs for every job type, then the bound for the synchronization algorithm is always stronger for all n.

5 The Packet Routing Problem with Fixed Paths

In this section, we apply our results on the job shop scheduling problem to the problem of packet routing in communication networks. Given a directed graph $(V, \mathcal{A})$ that represents a communication network, there is a collection of packets that need to be sent from a source node to a destination node along given paths. Each packet is given with a prespecified simple path (each node visited at most once), connecting the source to the destination, i.e., each packet is represented by the triplet (s_k, t_k, P_k), where s_k is the source node, t_k is the destination node and P_k is the prespecified simple path. It takes one time unit for a packet to traverse an edge in $\mathcal{A}$, and only one packet can traverse a given edge at a time. We will consider two versions of the packet routing problem: in this section, we address the problem where the paths are given, and in the next section, we address the problem, where we need to select the paths. For the packet routing problem with fixed paths there are n_P packets that need to be sent along path P, for each simple path P. Let $\mathcal{P}$ denote the collection of all simple paths, for which $n_P > 0$.

A scheduler decides which packets traverse any given edge and which packets wait in queue. The goal is to find a schedule which routes all packets from their sources to their destinations in minimal possible (makespan) time, given the initial number of packets n_P for each simple path P.

Extensive research has been conducted on this problem (see Leighton [9]). It is easy to see that the packet routing problem with fixed paths is a special case of the job shop scheduling problem. Each edge can be seen as a processing machine. Each path is a sequence of machines (edges) that jobs (packets) need to follow. All the processing (traversing) times are equal to one. The job types correspond to paths in $\mathcal{P}$, and the stages correspond to edges within the path. Also the quantity

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

in the job shop scheduling problem corresponds simply to the number of paths crossing any given edge e. This number is at most $|\mathcal{P}|$. The congestion C_e of a given edge $e \in \mathcal{A}$ is simply the number of packets that eventually cross e (the corresponding paths contain e):

$$C_e = \sum_{P:e\in P} n_P$$

The maximal congestion $C_{\max}$ is $\max_e C_e$, and is clearly a lower bound on the optimal makespan time.

Applying the synchronization algorithm for the job shop scheduling problem we obtain the following result.

Theorem 2 *Given a directed graph $(V, \mathcal{A})$, suppose for each $P \in \mathcal{P}$ there are n_P packets that need to follow path P. There exists a schedule that brings all the packets to their destinations in time C_{H} at most*

$$C_{\max} + 2\sqrt{C_{\max}|\mathcal{P}|L} + |\mathcal{P}|L,$$

where $|\mathcal{P}|$ is the cardinality of the set $\mathcal{P}$ and L is the size of the longest simple path in the graph. In particular,

$$\frac{C_{\mathrm{H}}}{C^*} \to 1$$

as

$$\sum_{P\in\mathcal{P}} n_P \to \infty,$$

and C^* is the optimal makespan time.

6 The Packet Routing Problem with Path Selection

In this section, we consider a more general version of the packet routing problem, in which the collection of paths is not given a priori, but needs to be determined. Given a directed network $(V, \mathcal{A})$, for each pair of nodes $k, l \in V$ there is a number of packets n_{kl} (called packets of type (k, l)) that need to be routed from source k to destination l via some path in the network. Let $\mathcal{P}$ denote the collection of all types in the network

$$\mathcal{P} = \{(k, l) : n_{kl} 0\}.$$

The scheduler is free to choose a path for each packet. The objective is to construct a schedule (which selects paths and chooses packets to traverse any given edge) so as to minimize the total time it takes to route all the packets to their destinations. Srinivasan and Teo [16] provide an algorithm that uses a linear programming problem that finds a schedule within a constant factor from the minimum makespan.

In this section, we construct a schedule that has makespan $C_H \leq C^* + O(\sqrt{C^*})$, where C^* denotes the minimum makespan time. Therefore, our algorithm is asymptotically optimal as the total number of packets increases to infinity.

In the previous sections we used a dynamic fluid relaxation to construct a schedule. We will now use a static multicommodity flow relaxation of the problem. For any feasible schedule, we define decision variables x_{ij}^{kl} to be the total number of type (k, l) packets that traverse the edge (i, j). We can assume without loss of generality that origin and destination nodes are not revisited by any packet. In particular, $x_{ik}^{kl} = x_{lj}^{kl} = 0$ for all $k, l, i, j \in V$.

For each edge $(i, j) \in \mathcal{A}$ the value

$$C_{i,j} = \sum_{(k,l)\in\mathcal{P}} x_{ij}^{kl} \tag{16}$$

represents the total time that edge (i, j) is processing packets. Clearly, the optimal makespan time C^* is at least $\max_{(i.j)\in\mathcal{A}} C_{i,j}$. Therefore, the following multicommodity flow problem provides a lower bound on C^*:

$$\begin{aligned}
&\text{minimize} && C_{\max} && && (17)\\
&\text{subject to} && \sum_{i:(k,i)\in\mathcal{A}} x_{ki}^{kl} = n_{kl}, && (k,l)\in\mathcal{P}, && (18)\\
& && \sum_{i:(i,l)\in\mathcal{A}} x_{il}^{kl} = n_{kl}, && (k,l)\in\mathcal{P}, && (19)\\
& && \sum_{j:(j,i)\in\mathcal{A}} x_{ji}^{kl} = \sum_{r:(i,r)\in\mathcal{A}} x_{ir}^{kl}, && (k,l)\in\mathcal{P},\ i\neq k,l, && (20)\\
& && C_{i,j} = \sum_{(k,l)\in\mathcal{P}} x_{ij}^{kl}, && (i,j)\in\mathcal{A}, && (21)\\
& && C_{i,j} \leq C_{\max}, && (i,j)\in\mathcal{A}, && (22)\\
& && x_{ij}^{kl},\ C_{ij} \geq 0, && (i,j)\in\mathcal{A},\ (k,l)\in\mathcal{P}. && (23)
\end{aligned}$$

Eqs. (18)-(20) represent conservation of flow. The objective function value of this linear programming problem, denoted also by $C_{\max}$ is clearly a lower bound on the optimal makespan time C^*.

The linear programming problem has $|\mathcal{A}||\mathcal{P}|$ variables and $|V||\mathcal{P}|+|\mathcal{A}|$ constraints. Thus it can be solved in polynomial time even if the n_{kl} are large. We next propose an algorithm that constructs a schedule with performance close to $C_{\max}$ when the number of packets is large.

Packet Routing Synchronization Algorithm

1. Calculate the optimal value of $C_{\max}$ of the linear programming problem (17).

2. Let Ω be a positive real value, to be specified later. For each interval $[m\Omega, (m+1)\Omega], m = 0, 1, \ldots, \left\lceil \frac{C_{\max}}{\Omega} \right\rceil - 1$, each edge (i, j) processes

$$a_{ij}^{kl} = \left\lfloor \frac{x_{ij}^{kl}\Omega}{C_{\max}} \right\rfloor \tag{24}$$

packets of type $(k, l) \in \mathcal{P}$ and idles the remaining time. If there are less than a_{ij}^{kl} jobs of type (k, l) available, then all available packets are processed and the rest of the dedicated time the edge idles, i.e., the edge does not process other packets.

3. At time

$$T \equiv \left\lceil \frac{C_{\max}}{\Omega} \right\rceil \Omega \leq C_{\max} + \Omega, \tag{25}$$

we process all remaining M packets sequentially, taking $MP_{\max}$ time units, where $P_{\max}$ is the length of the maximal simple path in the network.

Note that the last step is inefficient, but as we will see the number of packets left in the network after time T is small. Let us first show that the algorithm is feasible. For each edge $(i, j) \in \mathcal{A}$ we have:

$$\sum_{(k,l)\in\mathcal{P}} a_{ij}^{kl} \leq \sum_{(k,l)\in\mathcal{P}} \frac{x_{ij}^{kl}\Omega}{C_{\max}} = \frac{C_{i,j}\Omega}{C_{\max}} \leq \Omega.$$

We will show first that the number of packets left in the network at time T is $O(\sqrt{C_{\max}})$, by selecting Ω appropriately. For each node i, let $d(i)$ denote the outdegree of node i.

Proposition 4 *Let*

$$\Omega = \sqrt{C_{\max}|\mathcal{P}|}.$$

Then the total number of packets present in the network at time T defined in (25) is at most

$$2|\mathcal{A}|\sqrt{C_{\max}|\mathcal{P}|} + |\mathcal{A}||\mathcal{P}|.$$

Proof: We first show that for each $(k,l) \in \mathcal{P}$, the total number of packets of type (k,l) present in node k at time T is not bigger than $d(k)\frac{C_{\max}}{\Omega}$. During each interval $[m\Omega, (m+1)\Omega]$, $m = 0, 1, 2, \ldots, \left\lceil \frac{C_{\max}}{\Omega} \right\rceil - 1$, the number of type (k,l) packets processed from node k is equal to

$$\sum_{i:(k,i)\in\mathcal{A}} a_{ki}^{kl} \geq \sum_{i:(k,i)\in\mathcal{A}} \left(\frac{x_{ki}^{kl}\Omega}{C_{\max}} - 1 \right) \geq \frac{n_{kl}\Omega}{C_{\max}} - d(k),$$

where the second inequality follows from (18). Therefore, the number of type (k,l) packets present in node k at time $T = \left\lceil \frac{C_{\max}}{\Omega} \right\rceil \Omega$ is at most

$$n_{kl} - \Big(\frac{n_{kl}\Omega}{C_{\max}} - d(k) \Big) \frac{C_{\max}}{\Omega} = d(k)\frac{C_{\max}}{\Omega}.$$

We next consider any node $i \in V$ and any type (k,l) such that $i \neq k$. Initially, there are no type (k,l) packets at node i. Let m_0 be the largest integer such that there are no type (k,l) packets at node i at time $m_0\Omega$. The total number of type (k,l) packets that arrive into i during $[m_0\Omega, (m_0+1)\Omega]$ is at most

$$\sum_{j:(j,i)\in\mathcal{A}} a_{ji}^{kl} \leq \sum_{j:(j,i)\in\mathcal{A}} \frac{x_{ji}^{kl}\Omega}{C_{\max}}. \tag{26}$$

During each subsequent interval $[m\Omega, (m+1)\Omega]$, $m \geq m_0$ the total number of type (k,l) packets that arrive into i during $[m\Omega, (m+1)\Omega]$ is also at most

$$\sum_{j:(j,i)\in\mathcal{A}} \frac{x_{ji}^{kl}\Omega}{C_{\max}}.$$

The schedule will allocate at least

$$\sum_{j:(i,j)\in\mathcal{A}} a_{ij}^{kl} \geq \sum_{j:(i,j)\in\mathcal{A}} \left(\frac{x_{ij}^{kl}\Omega}{C_{\max}} - 1 \right) = \sum_{j:(i,j)\in\mathcal{A}} \frac{x_{ij}^{kl}\Omega}{C_{\max}} - d(i)$$

time units to type (k,l) during each interval $[m\Omega, (m+1)\Omega]$, $m \geq m_0 + 1$. Thus, during each subsequent interval $[m\Omega, (m+1)\Omega]$, $m = m_0+1, m_0+2, \ldots$ the number of type (k,l) packets in node i increases by at most

$$\sum_{j:(j,i)\in\mathcal{A}} \frac{x_{ji}^{kl}\Omega}{C_{\max}} - \sum_{j:(i,j)\in\mathcal{A}} \frac{x_{ji}^{kl}\Omega}{C_{\max}} + d(i) = d(i),$$

where the equality follows from (20). Combining with (26), the total number of type (k,l) packets at node i at time T s at most

$$\left\lceil \frac{C_{\max}}{\Omega} \right\rceil d(i) + \sum_{j:(j,i)\in\mathcal{A}} \frac{x_{ji}^{kl}\Omega}{C_{\max}} \leq \frac{C_{\max}}{\Omega} d(i) + d(i) + \sum_{j:(j,i)\in\mathcal{A}} \frac{x_{ji}^{kl}\Omega}{C_{\max}}$$

By summing over all nodes $i \in V$ and types $(k,l) \in \mathcal{P}$, we obtain that the total number of packets in the network at time T is at most

$$\sum_{i \in V} \sum_{(k,l) \in \mathcal{P}} \left(\frac{C_{\max}}{\Omega} d(i) + d(i) + \sum_{j:(j,i) \in \mathcal{A}} \frac{x_{ji}^{kl} \Omega}{C_{\max}} \right) = \left(\frac{C_{\max}}{\Omega} + 1 \right) |\mathcal{A}||\mathcal{P}| + \sum_{(j,i) \in \mathcal{A}} \frac{C_{j,i} \Omega}{C_{\max}} \leq \left(\frac{C_{\max}}{\Omega} + 1 \right) |\mathcal{A}||\mathcal{P}| + |\mathcal{A}| \Omega.$$

We now select Ω so as to minimize the quantity above. Namely, set $\Omega = \sqrt{C_{\max}|\mathcal{P}|}$. Then the total number of packets in the network at time T is at most

$$2|\mathcal{A}|\sqrt{C_{\max}|\mathcal{P}|} + |\mathcal{A}|\mathcal{P}|.$$

■

We next apply Proposition 4 to obtain an upper bound on the makespan time realized by the algorithm.

Theorem 3 *The packet routing synchronization algorithm routes all n_{kl} packets with origin k and destination l in time C_{H} satisfying*

$$C_{\max} \leq C^* \leq C_H \leq C_{\max} + \sqrt{C_{\max}|\mathcal{P}|} + 2|\mathcal{A}|\sqrt{C_{\max}|\mathcal{P}|}|V| + |\mathcal{A}||\mathcal{P}||V|,$$

where C^* is the optimal makespan time. In particular,

$$C_{rmH} = C_{\max} + O\left(\sqrt{C_{\max}}\right),$$

and

$$\frac{C_H}{C^*} \leq \frac{C_H}{C_{\max}} \to 1$$

as

$$\sum_{(k,l) \in \mathcal{P}} n_{kl} \to \infty.$$

Proof: From Proposition 4, at time T we have at most

$$2|\mathcal{A}|\sqrt{C_{\max}|\mathcal{P}|} + |\mathcal{A}||\mathcal{P}|$$

packets in the network. These packets can be routed to their destination by any trivial algorithm (as in the Step 3 of the Algorithm) within time at most

$$\left(2|\mathcal{A}|\sqrt{C_{\max}|\mathcal{P}|} + |\mathcal{A}||\mathcal{P}|\right) P_{\max},$$

where $P_{\max}$ is the length of the maximal simple path in the network. Since $P_{\max} \leq |V|$, we obtain that the total makespan time of the algorithm is, using the bound (25), at most

$$T + \left(2|\mathcal{A}|\sqrt{C_{\max}|\mathcal{P}|} + |\mathcal{A}||\mathcal{P}|\right)|V| \leq C_{\max} + \sqrt{C_{\max}|\mathcal{P}|} + 2|\mathcal{A}|\sqrt{C_{\max}|\mathcal{P}|}|V| + |\mathcal{A}||\mathcal{P}||V|.$$

∎

7 Concluding Remarks

We presented algorithms for the job shop scheduling and packet routing problems that are asymptotically optimal as the number of jobs (packets respectively) in the system approaches infinity. Unlike asymptotically optimal algorithms that rely on probabilistic assumptions, the proposed algorithms make no probabilistic assumptions, and they are asymptotically optimal for all instances with a large number of jobs (packets respectively).

The algorithm for job shop scheduling and its analysis underscores the importance of the fluid control problem and it shows that for instances of the problem with a large number of jobs, it is the dynamic and not the combinatorial character of the problem that dominates. Interestingly, the dynamic character of the problem that can be captured by the fluid control problem has a very simple structure. The algorithm for packet routing underscores the importance of the idea already observed in other discrete optimization problems that continuous relaxations carry information about the discrete optimization problem that can be used to construct near optimal solutions.

Finally, the results of the paper imply that in the limit of a large number of jobs (packets) the combinatorial structure of the problems, which is the essential difficulty of the problems, becomes increasingly unimportant as both problems are well approximated by continuous relaxations that are efficiently solvable.

Acknowledgement

We would like to thank Leon Hsu for his implementation of the algorithms proposed in the paper regarding the job shop scheduling problem.

References

[1] F. Avram, D. Bertsimas, and M. Ricard. Optimization of multiclass queueing networks: a linear control approach. In *Stochastic Networks, Proceedings of the IMA*, pages 199–234. F. Kelly and R. Williams eds., 1995.

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

[3] U. Feige and C. Scheideler. Improved bounds for acylic job shop scheduling. *ACM Symposium on the Theory of Computing*, pages 624–633, 1998.

[4] L. A. Goldberg, M. S. Peterson, A. Srinivasan, and E. Sweedyk. Better approximation guarantees for job-shop scheduling. *ACM-SIAM Symposium on Discrete Algorithms*, pages 599–608, 1997.

[5] L. Hall. Approximation algorithms for scheduling. In *Approximation algorithms for scheduling $\mathcal{NP}$-hard problems*, pages 1–45. PWS Publishing company (D. Hochbaum, editor), 1997.

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

[7] D. Karger, C. Stein, and J. Wein. Scheduling algorithms. In *CRC handbook of theoretical computer science*. to appear, 1999.

[8] R. M. Karp. Probabilistic analysis of partitioning algorithms for the traveling salesman problem in the plane. *Mathematics of Operations Research*, 2:209–224, 1977.

[9] F. T. Leighton. *Introduction to parallel algorithms and architectures: arrays, trees, and hypercubes.* Morgan Kaufmann, San Mateo, California, 1992.

[10] X. Luo and D. Bertsimas. A new algorithm for state constrained separated continuous linear programs. *SIAM J. Control Optim.*, 36:177–210, 1998.

[11] J. F. Muth and G. L. Thompson (eds.). *Industrial scheduling.* Prentice-Hall, Englewood Cliffs, NJ., 1963.

[12] M. C. Pullan. An algorithm for a class of continuous linear programming problems. *SIAM J. Control Optim.*, 31:1558–1577, 1993.

[13] S. V. Sevast'janov. An algorithm with an estimate for a problem with routing of parts of arbitrary shape and alternative executors. *Cybernetics*, 22:773–781, 1986.

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

[15] D. B. Shmoys, C. Stein, and J. Wein. Improved approximation algorithms for shop scheduling problems. *SIAM J. Comp.*, 23:617–632, 1994.

[16] A. Srinivasan and C. P. Teo. A constant-factor approximation algorithm for packet routing, and balancing local vs. global criteria. *ACM Symposium on the Theory of Computing*, pages 636–643, 1997.

[17] G. Weiss. On the optimal draining of re–entrant fluid lines. In *Stochastic Networks, Proceedings of the IMA*, pages 91–104. F. Kelly and R. Williams eds., 1995.