RC23273 (W0407-035) July 15, 2004
Computer Science

IBM Research Report

Job Shop Scheduling with Unit Processing Times

Nikhil Bansal, Tracy Kimbrel, Maxim Sviridenko
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Job shop scheduling with unit processing times

Nikhil Bansal* Tracy Kimbrel* Maxim Sviridenko*

Abstract

We consider randomized algorithms for the preemptive job shop problem, or equivalently, the case in which all operations have unit length. We give an improved approximation ratio of $O(\frac{\log m}{\log \log m})$ for an arbitrary number m of machines, the first $(2+\varepsilon)$-approximation for all constant numbers of machines, and an α-approximation for the case of two machines where $\alpha < 1.45$. The last result is via an approximation algorithm for a string matching problem which is of independent interest.

1 Introduction

Job shop scheduling is a widely studied and difficult combinatorial optimization problem [11]. We consider the preemptive case with the objective to minimize makespan. This problem is strongly NP-hard even with only two machines [9]. If the number of machines is part of the input then Williamson et al. [20] result implies that there is no polynomial-time approximation algorithm with performance guarantee better than 5/4 unless $P = NP$.

For the general nonpreemptive job shop scheduling problem the best known approximation algorithm has performance guarantee of $O(\log^2 m\mu / \log^2 \log m\mu)$ where m is the number of machines and μ is the maximum number of operations in a job [19, 7]. In the case when for every job there is at most one operation on each machine this bound could be improved to $O(\log^{1+\varepsilon} m)$ for every $\epsilon > 0$ [5, 6]. This variant of the problem is called acyclic job shop scheduling problem. In the case of acyclic job shop with unit processing times for every operations the famous papers [12, 13] design constant factor approximation algorithms.

In the case of preemptive job shop or, equivalently, the case in which all operations have unit length (we will explain this equivalence in the next section), the results [12, 19, 7] give $O(\log m\mu / \log \log m\mu)$. A polynomial-time approximation scheme (PTAS) is known for the special case of a constant number of machines and a constant number of operations per job [10] both for the preemptive and nonpreemptive problems. Sevastianov [16, 17] gave an algorithm that constructs a schedule with makespan $L + O(m\mu^3)p_{max}$, where L is maximum load on a machine and p_{max} is the maximum length of an operation. In particular, for the preemptive case, $p_{max} = 1$ and hence Sevastianov's result gives a schedule of length $L + O(m\mu^3)$.

In this paper we give the following results for the general preemptive job shop scheduling problem:

1. For arbitrary m: We give an algorithm with approximation factor of $O(\log m / \log \log m)$ for m machines. This eliminates the dependence on μ in previous results of [12, 19, 7].

   We give another very simple algorithm that constructs a schedule of length $(1+\epsilon)L + O(\mu \log m)p_{max}$ in the non-preemptive case, and hence a schedule of length $(1+\epsilon)L +$

*IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598. email:{*nikhil,kimbrel,sviri*}*@us.ibm.com*

$O(\mu \log m)$ in the preemptive case. This gives a substantially stronger guarantee than Sevastianov's result [16, 17] stated previously. In particular, if the instance is such that $\mu \leq \epsilon L / \log m$, then our algorithm gives a $(1+\epsilon)$ approximation.

2. For constant m: We show how our additive approximation stated above can be used to give a polynomial time $(2+\varepsilon)$-approximation algorithm for any constant number of machines (note that we allow number of operations per job to be part of the input). Previously, no algorithm with an approximation ratio independent of m was known for the problem.

3. For $m = 2$: Sevastianov and Woeginger [15] give a linear-time 1.5-approximation algorithm for minimizing the makespan in the two-machine case. An algorithm with a tighter approximation guarantee in terms of the maximum machine load L and the maximum job length l, but still 1.5 in the worst case, was given by [3]. All known approximation results for shop scheduling (other than approximation schemes) use as a lower bound the maximum of L and l. Sevastianov and Woeginger [15] note than any approximation algorithm with ratio better than 1.5 for the two machine case would require a new, non-trivial lower bound on the optimal makespan. In this paper, we give such a result based on the relationship between the preemptive job shop scheduling problem and a string matching problem over the binary alphabet. Our algorithm has an approximation ratio of less than 1.45.

2 Model and notation

In the job shop scheduling problem there is a set $J = \{J_1, \ldots, J_n\}$ of n jobs that must be processed on a given set $\mathcal{M} = \{M_1, \ldots, M_m\}$ of m machines. Each job J_j consists of a sequence of μ_j operations $O_{1j}, \ldots, O_{\mu_j j}$ that need to be processed in this order. Operation O_{kj} must be processed on machine $M_{\pi_{kj}}$, during p_{kj} time units. A machine can process at most one operation at a time, and each job may be processed by at most one machine at any time. For a given schedule, let C_{kj} be the completion time of operation O_{kj}. The objective is to find a schedule that minimizes the maximum completion time, $C_{max} = \max_{kj} C_{kj}$. The value of C_{max} is also called the *makespan* or the *length* of the schedule.

For a given instance of the job shop scheduling problem, the value of the optimum makespan will be denoted C^*_{max}. For each job j and machine i, ℓ_{ij} is the total amount of work in job j designated for machine i. Let $\ell_j = \sum_{i \in \mathcal{M}} \ell_{ij}$ denote the total length of job j, and let $\ell = \max_{j \in \mathcal{J}} \ell_j$. Let $L_i = \sum_{j \in \mathcal{J}} \ell_{ij}$ denote the load on machine i, and let L denote $\max_{i \in \mathcal{M}} L_i$. Clearly, $\max\{L, l\} \leq C^*_{max}$. Let $\mu = \max_j \mu_j$ be the maximum number of operations in any job.

In this paper we consider the *preemptive* job shop scheduling problem, in which every operation can be preempted during its execution and resumed later without any penalty. Of course, we must still obey precedence constraints between operations, and for every operation O_{kj}, the total time allocated on machine π_{kj} must be equal to its processing time p_{kj}. It is well-known that there exists an optimal schedule for the preemptive job shop problem where preemptions occur at integral times (see [4] for more general results of this sort). We may assume that all operation lengths are polynomially bounded since this assumption can be removed with only ε loss for any $\varepsilon > 0$ by standard scaling and rounding techniques. Thus, we will consider the preemptive job shop scheduling problem to be equivalent to the nonpreemptive job shop scheduling problem with unit processing times; i.e., we split every operation O_{kj} with processing time p_{kj} into p_{kj} unit length operations and assume that all inputs p_{kj}, L_i, l_j are polynomially bounded. We use ε throughout the paper to denote a constant that can be made arbitrarily small, where running times depend on $1/\varepsilon$.

We assume without loss of generality that for all $i \in \mathcal{M}$, $L_i = L$; this is easily achieved as follows. We add a dummy job for each machine i if needed, comprising $L - L_i$ unit length operations on machine i only. It is easy to see that this does not change the optimal makespan, since any schedule has length at least L and thus $L - L_i$ idle time steps in which the dummy job may be executed.

3 The two-machine case

In this section we consider the preemptive two-machine job shop problem $J2|pmtn|C_{max}$. The previously best known algorithms for this problem have a worst case approximation ratio of 1.5 [15, 3]. As mentioned previously, it is impossible to achieve a better ratio if we use the trivial lower bound of $\max\{L, l\}$ only. To see this, consider the instance with two identical jobs J_1 and J_2. Each of them consists of $L/2$ operations that require machine 1 followed by $L/2$ operations that require machine 2. Cleary, the trivial lower bound on the makespan is L. However, it is easy to see that any feasible schedule has makespan at least $1.5L$.

In order to get a ratio strictly better than 1.5, we adopt the following approach: We first note that in the cases when $l < 0.88L$ or when $l > 1.16L$ the trivial lower bound is enough to give a ratio better than 1.442. In the hard case with a job B of length about L (i.e. $l \approx L$), we attempt to maximize the number of operations of other jobs that are performed concurrently with B. This is most clearly stated in terms of a string matching matching which we describe in Sections 3.1 and 3.3 below. We show how to solve this string matching problem so that we can perform concurrently with B at least a $1-1/e$ fraction of the most possible. We then show that combining these different cases gives a worst case approximation ratio of about 1.442 in the general case.

3.1 Maximizing matches between a set of binary strings and one long one.

We consider the following problem, which is interesting on its own and which may have other applications. Let S be a binary string and $C = \{S_1, S_2, \ldots, S_n\}$ be a collection of binary strings. Let ℓ denote the length of S, and let ℓ_i denote the length of S_i. For a string X, let $X(i)$ denote the i^{th} character of X. We say that S_i has a matching E_i of value k in S if there exist indices $a_1 < a_2 < \ldots < a_k$ and $b_1 < b_2 < \ldots < b_k$ such that $S_i(a_j) = S(b_j)$ for all $1 \leq j \leq k$. We denote the value of E_i by $|E_i|$. We also associate E_i with the set of indices $b_1, \ldots, b_k$. Matchings E_i and E_j of S_i and S_j, respectively, in S are said to be disjoint if $E_i \cap E_j = \emptyset$.

Problem: Find a collection of matchings $E_1, \ldots, E_n$ of $S_1, \ldots, S_n$ in S such that the E_i are pairwise disjoint, i.e. $E_i \cap E_j = \emptyset$ for all $1 \leq i < j \leq n$. The goal is to maximize the cardinality of $\sum_{i=1}^{n} |E_i|$.

We show the following:

Theorem 1 *There is a randomized approximation algorithm for our string matching problem with expected ratio at least* $1 - 1/e$.

The proof is deferred until section 3.3.

3.2 Solving job shop using the string matching algorithm

Theorem 2 *For every* $\varepsilon > 0$*, there is a randomized* $(\alpha+\varepsilon)$*-approximation algorithm for* $J2|pmtn|C_{max}$*, where* $\alpha = 1 + \frac{e}{3e-2} \approx 1.442$*, with running time polynomial in the number* n *of jobs, the maximum number of operations in a job* μ*, and* $1/\varepsilon$.

Proof: Let $0 \leq \delta \leq 1$ be a constant to be determined later. We consider separately the following cases:

1. $l \leq 2\delta L$. In this case, we use the algorithm of [3] which finds a schedule of length at most $L + l/2$, so the trivial lower bound of L in this case is enough to give an approximation ratio of $1 + \delta$.

2. $l \geq \frac{2}{e-1}L$. Again we use the algorithm of [3]. In this case the trivial lower bound of l implies a ratio of at most $L/l + 1/2 \leq e/2 \approx 1.36$.

3. $\frac{2}{e-1}L > l > 2\delta L$. In this case, Theorem 1 can be used. We reduce $J2|pmtn|C_{max}$ to the string matching problem as follows. String S corresponds to a "big" job B job of length l (an arbitrary choice can be made if there are two or more such jobs), and all other jobs correspond to strings in the set C. Operations of B processed on the first machine correspond to ones in S and operations on the second machine correspond to zeros. For the other jobs, the correspondence is inverted: operations on the first machine correspond to zeros and operations on the second machine correspond to ones. We can obtain a schedule for a job shop instance from a solution to the matching problem such that the number of matches is the number of time units in which two operations are executed in parallel.

   Let V denote the optimal value of the string matching instance. V is the maximum possible overlap between job B and the remaining jobs, i.e., the maximum number of unit-length operations in B that can be executed concurrently with operations in other jobs. Note this is not necessarily the amount of overlap between B and the other jobs in any optimal schedule, but is an upper bound. The optimal makespan C^*_{max} is at least

$$l + \frac{1}{2}(\sum_{j \neq B} \ell_j - V) = L + l/2 - V/2$$

   even if the maximum overlap between B and the other jobs is achieved and the remaining operations in the other jobs are executed fully in parallel. Using Theorem 1 we can obtain a schedule with expected length at most $2L - (1 - 1/e)V$. Thus the approximation ratio is at most

$$\frac{2L - (1 - 1/e)V}{L + l/2 - V/2}.$$

   Consider the quantity

$$f(W) = \frac{2L - (1 - 1/e)W}{L + l/2 - W/2}.$$

   It can be verified that for $l \leq \frac{2}{e-1}L$, $f(W + 1) \geq f(W)$. Thus since $V \leq l$ the approximation ratio has maximum value at most $2 - 2(1 - 1/e)\delta$ with $l = V = 2\delta L$.

Choosing whichever of these two algorithms yields a smaller makespan, we obtain a schedule of length at most $\min\{2 - 2(1 - 1/e)\delta, 1 + \delta\}C^*_{max}$. Thus we set $\delta = \frac{e}{3e-2} \approx 0.442$, and the theorem follows.

□

3.3 Proof of Theorem 1

We first consider a linear program formulation for a relaxation of the problem. We view the relaxed problem as follows: Consider a character $S_i(j)$ and imagine traversing the string S one character

at a time starting from $S(1)$ and going all the way to $S(l)$. At step k we match a fraction $x_{i,j,k}$ of $S_i(j)$ to $S(k)$, and then we discard (i.e., leave unmatched) a fraction $y_{i,j,k}$ of $S_i(j)$ before moving on to stage $k+1$. We also assume that for each i, j, there is a step 0, where we can discard a fraction $y_{i,j,0}$ of $S_i(j)$.

Consider the following linear program:

$$Maximize \qquad \sum_{i=1}^{n}\sum_{j=1}^{\ell_i}\sum_{k=1}^{\ell} x_{i,j,k}$$

$$(1) \quad \forall i,j,k \qquad \sum_{t=0}^{k-1}(x_{i,j-1,t}+y_{i,j-1,t}) - \sum_{t=0}^{k-1}(x_{i,j,t}+y_{i,j,t}) - x_{i,j,k} \geq 0$$

$$(2) \quad \forall k \qquad \sum_{i,j} x_{i,j,k} \leq 1$$

$$(3) \quad x_{i,j,k} = 0, \qquad S_i(j) \neq S(k)$$

$$(4) \quad y_{i,j,k} \geq 0$$

$$(5) \quad x_{i,j,k} \geq 0$$

$$(6) \quad \forall i,j \qquad \sum_{t=0}^{l=|S|}(x_{i,j,t}+y_{i,j,t}) = 1$$

The first set of constraints models the precedence constraints implied by the ordering of the characters in the strings. They say that the amount of $S_i(j)$ matched up to $S(k)$ and discarded up to $S(k-1)$ is no more than total amount of $S_i(j-1)$ matched and discarded up to $S(k-1)$. The second set of constraints ensures that every character is S is matched to a total amount of at most 1. The third set of constraint ensures that 0s are not matched to 1s and vice versa. The fourth and fifth sets of constraints ensure that $x_{i,j,k}$ and $y_{i,j,k}$ are non-negative. The final set of constraints say that each character $S_i(j)$ is either matched or discarded.

We now show that the LP optimum is an upper bound on the integral optimum solution. Consider any integral solution (I), and suppose that $S_i(j_a)$ is matched to $S(k_a)$, for $a = 1, \ldots, b$. Then for $a = 1, \ldots, b$ we set $x_{i,j_a,k_a} = 1$ and $x_{i,j_a,t} = 0$ for all $t \neq k_a$. We also set $y_{i,j_a,t} = 0$ for all t. For j such that $j_a < j < j_{a+1}$, we set $y_{i,j,k_a} = 1$ and $x_{i,j,t} = 0$ for all t. If $j < j_1$, we set $y_{i,j,0} = 1$.

Clearly the value of the objective function for this assignment of variables is exactly the number of matches in I. We only need to show that this assignment satisfies all the constraints. Since $x_{i,j,k}$ and $y_{i,j,k}$ are $\{0,1\}$ variables in this assignment and every $S_i(j)$ is either matched or discarded, the last 3 sets of constraints are trivially satisfied. The third set of constraints is satisfied since I does not match a 0 to a 1 or vice versa. The second set of constraints holds because each $S(k)$ is matched to at most one character among the S_i.

For the first set of constraints, since $x_{i,j,k}$ and $y_{i,j,k}$ are $\{0,1\}$, such a constraint is violated only if $S_i(j-1)$ has not been matched or discarded by step $k-1$ and either

1. $S_i(j)$ is matched at step k.
2. $S_i(j)$ is matched or discarded by step $k-1$ or sooner.

However, it is easy to see that if $S_i(j)$ is matched at to $S(k)$, then either $j-1$ was matched to $S(h)$ for $h < k$ or discarded at step g, where g is the last step before k where some character from S_i was matched.

If $S_i(j)$ is matched or discarded by step $k-1$ or sooner, it is trivial to see by construction that $S_i(j-1)$ is matched or discarded before step $k-1$ as well.

We now give a randomized procedure to round this LP solution. We will show that it produces a feasible integral solution which has a number of matches equal to at least $1-1/e$ times the optimum LP value in expectation.

It is useful to view the LP solution in the following equivalent way. Let $0 \leq s_{i,j,k} \leq 1$ denote the extent to which $S_i(j)$ has been matched to $S(1), \ldots, S(k-1)$ or discarded during the first $k-1$ steps, i.e. $s_{i,j,k} = \sum_{t=1}^{k-1}(x_{i,j,t} + y_{i,j,t})$. Let $v_{i,j,k} = s_{i,j,k} + x_{i,j,k}$; thus $v_{i,j,k} - s_{i,j,k}$ is exactly the extent to which $S_i(j)$ is matched to $S(k)$. Also note that $y_{i,j,k} = s_{i,j,k+1} - v_{i,j,k}$. Note that the first set of constraints implies that $s_{i,j-1,k} \geq v_{i,j,k}$.

Rounding Procedure:

1. For each string S_i choose $u_i \in [0,1]$ uniformly at random.

2. For each i, j assign $S_i(j)$ to $S(k)$ if and only if $s_{i,j,k} \leq u_i < v_{i,j,k}$.

3. Let $N(k)$ denote the number of characters assigned to $S(k)$ at the end of the previous step. $N(k)$ is a random variable. If $N(k) = 0$, $S(k)$ is not matched to any character. If $N(k) = 1$, we match $S(k)$ to the unique $S_i(j)$ assigned to it. If $N(k) \geq 2$, we arbitrarily match $S(k)$ to one of the characters, and discard the remaining $N(k) - 1$ characters (never to be matched again).

We now study the properties of the obtained solution. Given two open intervals of numbers $I_1 = (l_1, u_1)$ and $I_2 = (l_2, u_2)$, we say that $I_1 > I_2$ if $l_1 \geq u_2$, i.e. each element in I_1 is greater than every element than I_2.

Fix i and j. Since $x_{i,j,k} \geq 0$ and $y_{i,j,k} \geq 0$ it trivially follows that $v_{i,j,k} \leq s_{i,j,k+1}$ and hence the intervals $I_k = (s_{i,j,k}, v_{i,j,k})$ are pairwise disjoint.

Thus, no $S_i(j)$ can be assigned to two or more characters in S. At the end of the third rounding step, no $S(k)$ is matched to two or more characters. Thus, to show the validity of the solution produced after rounding, we need only to show that no precedence constraints are violated for any S_i.

Lemma 1 *For a fixed i and k, let I_j denote the (possibly empty) interval $(s_{i,j,k}, v_{i,j,k})$. Then, $I_{j_1} > I_{j_2}$ for all $j_1 < j_2$ or equivalently, $v_{i,j_2,k} \leq s_{i,j_1,k}$. In particular this implies that $I_{j_1} \cap I_{j_2} = \emptyset$ for all $1 \leq j_1 < j_2 \leq \ell_i$*

Proof: It suffices to show that $I_{j-1} > I_j$ for all j. By the first set of constraints we know that $v_{i,j,k} \leq s_{i,j-1,k}$ for all i, j, k. This implies the desired result. □

We now show that no precedence constraints will be violated for any S_i. Suppose $S_i(j_1)$ is assigned to $S(k_1)$ and $S_i(j_2)$ to $S(k_2)$ for $j_1 < j_2$ and $k_1 \geq k_2$. Since we use the same random u_i for all characters in the string S_i, it must be the case that $u_i \in (s_{i,j_1,k_1}, v_{i,j_1,k_1})$ and $u_i \in (s_{i,j_2,k_2}, v_{i,j_2,k_2})$, which implies that $v_{i,j_2,k_2} > s_{i,j_1,k_1}$. Since $v_{i,j,k}$ is monotonically increasing in k, and $k_2 \leq k_1$, this implies that $v_{i,j_2,k_1} > s_{i,j_1,k_1}$. But this violates lemma 1. Thus we have shown that the rounding produces a feasible solution.

We now analyze the quality of the solution.

Lemma 2 *Let V denote the optimum LP value. The rounding procedure matches at least $V(1-1/e)$ characters in S in expectation.*

Proof: Let us consider the solution obtained after the first two steps of rounding. Let $N(i,k)$ be a random variable that denotes the number of characters from S_i that are assigned to the character $S(k)$. Let $p_{i,k}$ denote $\sum_j x_{i,j,k} = \sum_j (v_{i,j,k} - s_{i,j,k})$. By Lemma 1, we know that the intervals $I_j = (s_{i,j,k}, v_{i,j,k})$ are disjoint. Hence the probability that some character of S_i is matched to $S(k)$ is exactly equal to the probability that $u_i \in \cup_j I_j$, which is exactly $p_{i,k}$. Thus $N(i,k)$ is a Bernoulli random variable with parameter $p_{i,k}$, that is, it is 1 with probability $p_{i,k}$ and is 0 otherwise.

Thus, $N(k)$ is the sum of m Bernoulli random variables with parameters $p_{1,k}, \ldots, p_{n,k}$. Let I_0 denote the random variable that denote the number of characters in S that have zero matches. Thus, $I_0 = |\{k : N(k) = 0\}|$.

Clearly, the number of matches in the LP is $\sum_{i,j,k} x_{i,j,k} = \sum_{i,k} p_{i,k}$. The number of matches in the integral solution is the length of S minus the number of non-matches. Thus, the number of matches is $\ell - I_0$.

Now

$$Pr\left[N(k) = 0\right] = \Pi_{i=1}^{n}(1 - p_{i,k}) \leq e^{-\sum_{i=1}^{n} p_{i,k}}$$

Thus, $E[I_0] \leq \sum_{k=1}^{\ell} e^{-\sum_{i=1}^{n} p_{i,k}}$ Thus the expected number of matches is

$$l - E[I_0]$$

$$\geq \sum_{k=1}^{\ell}(1 - e^{-\sum_{i=1}^{n} p_{i,k}})$$

$$\geq (1 - 1/e)\sum_{k=1}^{\ell}\sum_{i=1}^{n} p_{i,k}$$

The last step holds since $1 - e^{-x} \geq x(1 - 1/e)$ for $0 \leq x \leq 1$. Note that $\sum_{i=1}^{n} p_{i,k} \leq 1$ follows from constraint set 2.

□

This completes the proof of theorem 1.

4 An improved bound for an arbitrary number of machines

4.1 An $O(\log m / \log\log m)$-approximation algorithm.

In this section we consider a simple randomized algorithm previously considered in [12, 19, 7]. Namely, the algorithm chooses an integer number t_j between 0 and $L-1$ independently at random for each job $J_j \in \mathcal{J}$. It then constructs an infeasible schedule processing each job J_j in no-wait fashion in the time interval $[t_j, l_j + t_j - 1]$. The problem with this schedule is that in some time steps, some machines must process more than one job each. Let τ_{it} be the number of operations assigned to a time step t on machine M_i by the above randomized procedure.

To obtain a feasible schedule from the infeasible one constructed by the randomized shifting procedure, we expand every time step t into $\max_{i=1,\ldots,m} \tau_{it}$ steps. Since all operations assigned to the same step belong to different jobs we can feasibly schedule all such operations using this interval of length $\max_{i=1,\ldots,m} \tau_{it}$. Finally, we concatenate the schedules so obtained for all time steps. The total length of the final feasible schedule is at most $\sum_{t=1}^{L+l} \max_{i=1,\ldots,m} \tau_{it}$ (where the upper limit of summation corresponds to the upper bound of $L + l$ on the length of the infeasible schedule).

By Chernoff bounds it can be shown that with high probability $\max_{i,t} \tau_{it} = O(\log mL / \log\log mL)$ [12, 19, 7]. Therefore, with high probability the length of the final schedule is $O(\log mL / \log\log mL) \max\{L, l\}$.

Instead of using this high probability result, we will try to estimate the expected schedule length which is $E(\sum_{t=1}^{L+l} \max_{i=1,\ldots,m} \tau_{it}) = \sum_{t=1}^{L+l} E(\max_{i=1,\ldots,m} \tau_{it})$. We will do this using the following lemma. The techniques are standard and similar results can be found in [8, 14]. We defer the proof of this lemma to Section 6.

Lemma 3 (Non-uniform Balls and Bins) *Suppose we have m bins and n balls. Every ball j chooses a bin i at random with probability λ_{ij}, i.e. $\sum_{i=1}^{m} \lambda_{ij} \leq 1$. The expected number of balls in every bin is at most one, i.e. $\sum_{j=1}^{n} \lambda_{ij} \leq 1$. Then the expected maximum number of balls in any of the m bins is $O(\log m / \log \log m)$.*

Lemma 3 immediately implies an upper bound for $\sum_{t=1}^{L+l} E(\max_{i=1,\ldots,m} \tau_{it})$. In every time interval of unit length we have an instance of the balls and bins problem with m bins and n balls. Ball j landing in bin i corresponds to an operation of job J_j being scheduled on machine M_i at time step t by the randomized shifting procedure. This occurs with probability at most $\min\{\frac{l_j}{L}, 1\} \cdot \frac{l_{ij}}{l_j} = l_{ij} / \max\{L, l_j\}$, which is an upper bound on the probability that the job is processed in the time interval t, multiplied by the probability that the job is processed on machine i conditioned on its being processed during time step t. Therefore, $E(\max_{i=1,\ldots,m} \tau_{it}) = O(\log m / \log \log m)$ and we obtain the following.

Theorem 3 *The expected makespan of the feasible schedule obtained by the above randomized algorithm is $O(\log m / \log \log m) \max\{L, l\}$.*

4.2 An approximation algorithm with an additive performance guarantee.

Our second randomized algorithm also chooses random integer shifts t_j in the interval $[0, L-1]$ for every job J_j. The difference is that instead of processing every job in the time interval $[t_j, t_j + l_j]$, we process job J_j in the time interval $[t_j, t_j + l_j \cdot K \log m]$ with equal delays of $K \log m$ between consecutive operations of the same job where K is a constant specified later. This schedule has length at most $L + (K \log m)l$. This schedule also may be infeasible, but unlike the no-wait schedule from the previous section we will show it can be transformed into a feasible schedule of length $(1+\varepsilon)L + O(\log m)l$ for an arbitrarily small constant $\varepsilon > 0$. Note that the constant K depends on ε, which means that higher precision in the first term is compensated by an increase in the second term of our bound.

We now show how to transform an infeasible schedule into a feasible one with $1+\varepsilon$ increase in the schedule length. Consider an infeasible schedule obtained by the modified randomized shifting procedure. This schedule has length at most $L + (K \log m)l$. We split the time interval $[0, L + (K \log m)l]$ into consecutive intervals of length $K \log m$; the last interval may have smaller length. We use another variant of the balls and bins lemma.

Lemma 4 *Let $0 < \varepsilon < 1$ and $K_\varepsilon = 4/\varepsilon^2$. Suppose we have m bins and n balls. Every ball j chooses a bin i at random with probability λ_{ij}, i.e. $\sum_{i=1}^{m} \lambda_{ij} \leq 1$. The expected number of balls in every bin is at most $\sum_{j=1}^{n} \lambda_{ij} \leq K_\varepsilon \log m$. Then the expected maximum number of balls in any of the m bins is $O((1+\varepsilon) K_\varepsilon \log m)$.*

Therefore, all jobs which appear in the same interval of length $K_\varepsilon \log m$ in the infeasible schedule can be scheduled in $(1+\varepsilon) K_\varepsilon \log m$ time steps (in expectation) since there are no two operations of the same job in such intervals. Thus we obtain the following.

Theorem 4 *The expected makespan of the feasible schedule obtained by the above randomized algorithm is $(1+\varepsilon)L + O(\log m)l$.*

Remark. Applying the same algorithm for the nonpreemptive job shop scheduling problem with general processing times and using delays of $(K \log m)p_{max}$ between consecutive operations of the same job, where p_{max} is maximum processing time in the instance, we can get a feasible schedule of length at most $(1+\varepsilon)L + O(\mu \log m)p_{max}$, where μ is maximum number of operations per job and a hidden constant depends on ε. Sevastianov [16, 17] obtained similar bounds by using so-called vector summation techniques. More precisely he described a polynomial time algorithm which always finds a schedule of length at most $L + O(m\mu^3)p_{max}$.

5 A $(2+\varepsilon)$-approximation for any constant number of machines

Our $(2+\varepsilon)$-approximation for for the job shop scheduling problem $Jm|pmtn|C_{max}$ with unit operations (or preemptions) is based on the following standard idea in scheduling. We split the set of jobs into two sets: $L = \{J_j | l_j \geq \varepsilon L/(K_\varepsilon \log m)$ is the set of "big" jobs and $S = \mathcal{J} \setminus L$ is the set of "small" jobs.

The set of small jobs is scheduled by using the randomized algorithm from the previous section. Theorem 4 guarantees that the schedule length is at most $(1+O(\varepsilon))\max\{L, l\}$.

To schedule the set of big jobs we employ a well-known technique for the job shop scheduling problem with constant number of jobs. It is known that the job shop problem with two jobs is polynomially solvable [1]. If the number of jobs is constant $k \geq 3$ then the problem is weakly NP-hard [18] when processing times are allowed to be exponentially large, but can be solved in pseudo-polynomial time by a straightforward generalization of Akers' technique to a grid of k dimensions rather than 2. With unit processing times, the problem is polynomially solvable. Therefore, we can schedule the big jobs with makespan at most C^*_{max} (or $(1+\varepsilon)C^*_{max}$ if we include the factor we lose when we apply rounding and scaling to decrease number of unit length operations).

Concatenating the two schedules, we obtain a schedule of length at most $(2+\varepsilon)C^*_{max}$.

6 Proofs of Lemmas 3 and 4

We use the following version of Chernoff bounds as given on page 267, Corollary A.1.10, [2].

Lemma 5 *Suppose $X_1, \ldots, X_n$, are 0-1 random variables, such that $Pr[X_i = 1] = p_i$. Let $\mu = \sum_{i=1}^n p_i$ and $X = \sum_{i=1}^n X_i$. Then*

$$Pr[X - \mu \geq a] \leq e^{a-(a+\mu)\ln(1+a/\mu)}$$

We will also need the following corollary.

Lemma 6

$$Pr[X - \mu \geq a] \leq e^{-a\min(1/5, a/4\mu)}$$

Proof: Let $x = \mu/a$. Then the right-hand side can be written as $e^{-a((x+1)\ln(1+1/x)-1)}$.

Now, $(x+1)\ln(1+1/x) - 1$ is decreasing in x. At $x = 2$, its value is $3\ln 5/3 - 1 \geq 1/5$. For $x > 2$, it is at least $(x+1)(1/x - 1/2x^2) - 1 = 1/2x - 1/2x^2 \geq 1/4x$. □

Proof: (of Lemma 3)

Let B_{ij} be a 0-1 random variable that is 1 iff ball j goes to bin i. Let B_i denote the number of balls in bin i. Finally, let $B = \max_i B_i$. As $E[B_i] \leq 1$, by Lemma 5,

$$Pr[B_i \geq 1 + a] \leq Pr[B_i \geq E[B_i] + a] \leq e^{a-(a+E[B_i])\ln(1+a/E[B_i])} \leq e^{a-a\ln(1+a)}.$$

For $a \geq 3\frac{\ln m}{\ln \ln m}$ and observing that $\ln(1+a) - 1 \geq \ln(\ln m / \ln \ln m) \geq 1/2 \ln \ln m)$, we have that

$$Pr[B_i \geq 1 + a] \leq e^{-a \ln \ln m/2}$$

Since,

By the union bound, $Pr[B > 1 + a] \leq me^{-a \ln \ln m/2}$. Now,

$$\begin{aligned} E[B] &= \sum_{x=1}^{\infty} Pr[B \geq x] \\ &\leq 3\frac{\ln m}{\ln \ln m} + \sum_{a \geq 3\frac{\ln m}{\ln \ln m}} Pr[B \geq 1 + a] \\ &\leq 3\frac{\ln m}{\ln \ln m} + m \sum_{a \geq 3\frac{\ln m}{\ln \ln m}} e^{-a \ln \ln m/2} \\ &\leq 3\frac{\ln m}{\ln \ln m} + 2\frac{m^{-1/2}}{\ln \ln m} \leq 5\frac{\ln m}{\ln \ln m} \end{aligned}$$

□

Proof: (of Lemma 4) By Lemma 6

$$Pr[B_i \geq K_\varepsilon \ln m + \delta] \leq Pr[B_i \geq E[B_i] + \delta] \leq e^{-\delta \min(1/5, \delta/4\mu)}$$

We will only be interested in $\delta \geq \varepsilon K_\varepsilon \ln m$. Thus, $Pr[B_i \geq K_\varepsilon \ln m + \delta] \leq e^{-\delta \cdot \varepsilon/4}$. and hence by the union bound $Pr[B \geq K_\varepsilon \ln m + \delta] \leq me^{-\delta \cdot \varepsilon/4}$. Thus, we have that

$$\begin{aligned} E[B] &\leq (1+\varepsilon) K_\varepsilon \ln m + \sum_{\delta > \varepsilon K_\varepsilon \ln m} me^{-\delta \cdot \varepsilon/4} \\ &\leq (1+\varepsilon) K_\varepsilon \ln m + m\frac{4}{\varepsilon} e^{-\varepsilon^2 K_\varepsilon \ln m/4} \end{aligned}$$

Recalling that $K_\varepsilon = 4/\varepsilon^2$, we have that $E[B] \leq (1+\varepsilon) K_\varepsilon \ln m + 4/\epsilon \leq (1+\epsilon) K_\varepsilon \ln m + \epsilon K_\varepsilon \leq (1+3\varepsilon) K_\varepsilon \ln m$.

□

7 Open Problems

Two outstanding open questions that remain for the general preemptive job shop problem are:

1. Is there an $O(1)$ approximation for the general preemptive job shop problem with an arbitrary number of machines?

   A natural way to show an approximation ratio is to use the trivial lower bound $\max\{L, l\}$. Moreover all other known lower bounds are provably within a constant factor of $\max\{L, l\}$ and therefore are useful only if we want to refine an approximation ratio (as we did in Section 3). It is known that for nonpreemptive job shop scheduling there is no $O(1)$ approximation algorithm with respect to the trivial lower bound $\max\{L, l\}$ even for acyclic instances [6]. We believe that resolving this question would require significant new insights into the structure of preemptive schedules.

2. Is there a polynomial time approximation scheme for the case of a constant number of machines?

   Our algorithm in this paper gives a PTAS if $\mu \leq \epsilon L / \log m$. On the other hand, if we restrict the problem to instances with only a few long jobs that comprise all but an ε fraction of the load, then we can get a PTAS via the generalization of Akers' technique mentioned earlier. Making progress on the general case, in which both long jobs and short jobs may make up significant portions of the load, would require an understanding of how small jobs and long jobs are "mixed" together in an optimal. A first step in this direction would be to understand the approximability of the string matching problem from Section 3.

References

[1] S. B. Akers, A graphical approach to production scheduling problems, Operations Research 4 (1956), 244–245.

[2] N. Alon, and J. Spencer, The Probabilistic Method. John Wiley & Sons, 2000.

[3] E.J. Anderson, T.S. Jayram, and T. Kimbrel. Tighter Bounds on Preemptive Job Shop Scheduling with Two Machines. *Computing* 67 (2001), pp. 83-90.

[4] P. Baptiste, J. Carlier, A. Kononov, M. Queyranne, S. Sevastianov and M. Sviridenko, Structural Properties of Preemptive Schedules,submitted for publication.

[5] A. Czumaj and C. Scheideler, A New Algorithmic Approach to the General Lovasz Local Lemma with Applications to Scheduling and Satisfiability Problems, Proc. 32 ACM Symposium on Theory of Computing (STOC), 2000.

[6] U. Feige and C. Scheideler, Improved bounds for acyclic job shop scheduling. Combinatorica 22 (2002), no. 3, 361–399.

[7] L.A. Goldberg, M. Paterson, A. Srinivasan, and E. Sweedyk. Better approximation guarantees for job-shop scheduling. SIAM J. Discrete Math. 14 (2001), 67–92.

[8] G. Gonnet, Expected length of the longest probe sequence in hash code searching. J. Assoc. Comput. Mach. 28 (1981), no. 2, 289–304.

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

[10] K. Jansen, R. Solis-Oba, and M. Sviridenko. Makespan minimization in job shops: a linear time approximation scheme. SIAM J. Discrete Math. 16 (2003), 288–300.

[11] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. Sequencing and Scheduling: Algorithms and Complexity. In S.C. Graves, A.H.G. Rinnooy Kan, and P.H. Zipkin (eds.), *Logistics of Production and Inventory*, Handbooks in Operations Research and Management Science 4, North-Holland, Amsterdam, 1993, 445-522.

[12] F.T. Leighton, B. Maggs, and S. Rao. Packet routing and jobshop scheduling in O(congestion + dilation) steps. *Combinatorica* 14, pp. 167–186, 1994.

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

[14] M. Raab and A. Steger, "Balls into bins"—a simple and tight analysis. Randomization and approximation techniques in computer science (Barcelona, 1998), 159–170, Lecture Notes in Comput. Sci., 1518, Springer, Berlin, 1998.

[15] S.V. Sevastianov and G.J. Woeginger. Makespan minimization in preemptive two machine job shops. *Computing* 60 (1998), 73-79.

[16] S. Sevast'janov, On some geometric methods in scheduling theory: a survey. Discrete Appl. Math. 55 (1994), no. 1, 59–82.

[17] S. Sevastianov, Bounding algorithm for the routing problem with arbitrary paths and alternative servers, Cybernetics **22** (1986), pp. 773–780.

[18] Y. Sotskov and N. Shakhlevich, NP-hardness of shop-scheduling problems with three jobs. Discrete Appl. Math. 59 (1995), no. 3, 237–266.

[19] D. Shmoys, C. Stein, and J. Wein. Improved Approximation Algorithms for Shop Scheduling Problems. *SIAM Journal on Computing* 23:3, 617-632, 1994.

[20] D. P. Williamson, L. A. Hall, J. A. Hoogeveen, C. A. J. Hurkens, J. K. Lenstra, S. V. Sevast'janov and D. B. Shmoys, Short shop schedules, Operation Research **45** (1997), pp. 288–294.