A linear time approximation scheme for job shop scheduling *

Klaus Jansen[1], Roberto Solis-Oba[2], and Maxim Sviridenko[3]

[1] IDSIA, Lugano, Switzerland klus@idsia.ch
[2] Max Planck Institut für Informatik, Saarbrücken, Germany solis@mpi-sb.mpg.de
[3] Sobolev Institute of Mathematics, Novosibirsk, Russia svir@math.nsc.ru

Abstract. In this paper we present a linear time approximation scheme for the non-preemptive job shop scheduling problem when the number of machines and the number of operations per job are fixed. We also show how to extend the approximation scheme for the preemptive version of the problem.

Let $p_{max} = \max_{ij} p_{ij}$ denote the maximum processing time of any operation. Let $\Pi_{max} = \max_{1 \leq i \leq m} \sum_{j,k|m_{kj}=i} p_{kj}$ be the maximum load or total processing time assigned to any machine.

1 Introduction

In the job shop scheduling problem there is a set $\mathcal{J} = \{J_1, \ldots, J_n\}$ of n jobs that must be processed on a group $M = \{1, \ldots, m\}$ of m machines. Each job J_j consists of a sequence of operations $O_{1j}, O_{2j}, \ldots, O_{\mu j}$, where O_{ij} must be processed without interruption on machine $m_{ij} \in \{1, \ldots, m\}$ during p_{ij} time units. The operations $O_{1j}, O_{2j}, \ldots, O_{\mu j}$ must be processed one after another in the given order and each machine can process at most one operation at a time. The problem is to schedule the jobs so as to minimize the overall *makespan* (or *schedule length*), which is the time by which all jobs are completed.

Two related problems are the open shop and the flow shop scheduling problems. In the open shop problem there is no specified ordering for the processing of the operations of a job. This means that the operations of a job can be processed in any order. In the flow shop problem the ordering of the operations is the same for all jobs. In both of these problems each job has exactly one operation per machine.

All three above problems are strongly NP-hard when the number of machines is not fixed [10]. The job shop problem is strongly NP-hard even if each job has at most three operations and there are only two machines [10]. The flow shop problem is strongly NP-hard as soon as $m = 3$ [2]. The open shop problem

* This work was supported in part by EU ESPRIT LTR Project 20244 (ALCOM-IT) and by the Swiss Office Fédéral de l'éducation et de la Science Project 97.0315 titled "Platform".

with a constant number of machines $m \geq 3$ is known to be weakly NP-hard [3], and the question of whether the problem is NP-hard in the strong sense is still open. Williamson et al. [19] proved that when the number of machines is part of the input, there does not exist a polynomial time approximation algorithm with worst case bound smaller than $\frac{5}{4}$ for any of the above three problems unless $P = NP$.

Consider an instance I of one of the above shop problems. Let $OPT(I)$ be the length of the minimum makespan for I, and let $\mathcal{A}(I)$ be the length of the schedule computed by some algorithm $\mathcal{A}$. A polynomial time approximation scheme (PTAS) for the problem is an algorithm $\mathcal{A}$ which for any (constant) $\epsilon > 0$ and input I outputs in time polynomial in the length of I a feasible schedule of length $R_{\mathcal{A}}(I, \epsilon) = \frac{\mathcal{A}(I)}{OPT(I)} \leq 1 + \epsilon$. We can view such an algorithm $\mathcal{A}$ as a family of algorithms $\{\mathcal{A}_\epsilon | \epsilon > 0\}$ such that $\mathcal{A}_\epsilon(I) \leq (1 + \epsilon)OPT(I)$. A fully polynomial approximation scheme is an approximation scheme $\{\mathcal{A}_\epsilon\}$ in which each algorithm $\mathcal{A}_\epsilon$ runs in time polynomial in the length of I and $\frac{1}{\epsilon}$.

When the number m of machines is fixed, Hall [7] developed a polynomial time approximation scheme for the flow shop problem, while Sevastianov and Woeginger [17] designed an $O(n \log n)$ time approximation scheme for the open shop problem. We [9] designed a polynomial time approximation scheme for the job shop problem when m and μ are fixed. When m and μ are part of the input the best known result [5] for the job shop problem is an approximation algorithm with worst case bound $O((\log(m\mu) \log(min(m\mu, p_{max}))/ \log\log(m\mu))^2)$.

In this paper we improve on the polynomial time approximation scheme in [9], which is only guaranteed to run in polynomial time, by presenting a linear time approximation scheme for the job shop scheduling problem when m and μ are constant. The idea is to divide the set of jobs $\mathcal{J}$ into two groups $\mathcal{L}$ and $\mathcal{S}$ formed by jobs with "large" and "small" total processing time, respectively. We restrict $\mathcal{L}$ to include only a constant number of jobs. We fix a relative ordering for the long jobs and find a (probably infeasible) scheduling for the small jobs using a linear program. Then we round the solution for the linear program so that only a small, constant, number of jobs from $\mathcal{S}$ are preempted. These jobs are removed from the schedule and placed at the end of the schedule. By choosing the set $\mathcal{L}$ appropriately we can show that the increase in the length of the schedule created by these jobs is very small.

The linear program that we propose might yield an infeasible schedule because two operations from the same job might be scheduled simultaneously on different machines. However, the solution computed with the linear program has the property that it partitions the set of jobs into groups which are scheduled in disjoint time intervals. Thus the only infeasibilities in the schedule might come from conflicts among the jobs in the same group. The key idea of the algorithm is to use cleverly an algorithm by Sevastianov [14] on each group to find a feasible schedule of length only "slightly" larger than the length of the schedule obtained with the linear program.

We show how to solve approximately the above linear program in $O(n)$ time, and we prove that an approximate solution is sufficient to get a polynomial time

approximation scheme for the problem. We also present a linear time rounding algorithm that rounds to zero or one all but a constant number of fractional values in any solution of the the linear program.

Even when Sevastianov's algorithm runs in $O(n^2)$ time, we can still have a linear time approximation scheme. To achieve this we carefully "glue" some small jobs together to reduce their number without increasing the machine load or the maximum number of operations per job. The running time of our algorithm is not polynomial in $\frac{1}{\epsilon}$, but this is not surprising, since being the problem strongly NP-hard no fully polynomial approximation scheme can be expected.

We show that the approximation scheme can be generalized also to the preemptive job shop problem. The approach is similar to the non-preemptive case, but we have to handle carefully the set $\mathcal{L}$ to ensure that the algorithm still runs in linear time. The preemptive job shop problem is also NP-complete in the strong sense for $m = 3$ and $\mu = 3$ [4]. Our approximation schemes can be generalized also for flexible job shops, dag shops, and job shop problems with release and delivery times.

2 Linear Program

In this section, we formulate the fractional job shop scheduling problem as a linear program. Let $S(O') = \sum_{O_{kj} \in O'} p_{kj}$ be the total processing time for some set of operations O'; and let $S(J')$ be the total processing time of all jobs in some set $J' \subset \mathcal{J}$.

2.1 Relative Schedules

Let $\mathcal{L} \subset \mathcal{J}$ be a set formed by a constant number of jobs. We call $\mathcal{L}$ the set of *long* jobs and we show later how to define it. Let $k = |\mathcal{L}|$. Set $\mathcal{S} = \mathcal{J} \setminus \mathcal{L}$ is the set of *short* jobs. Two operations O_{ij} and $O_{i'j'}$, $J_j, J_{j'} \in \mathcal{L}$ and $j \neq j'$, are compatible if they must be processed on different machines, i.e., $m_{ij} \neq m_{i'j'}$. A *snapshot* of $\mathcal{L}$ is a subset of compatible operations. A *relative schedule* of $\mathcal{L}$ is a sequence $M(1), \ldots, M(g)$ of snapshots of $\mathcal{L}$ such that

- $M(1) = \emptyset$ and $M(g) = \emptyset$,
- each operation O_{ij} for a job $J_j \in \mathcal{L}$ occurs in a subsequence of consecutive snapshots $M(\alpha_{ij}), \ldots, M(\beta_{ij})$, $2 \leq \alpha_{ij} \leq \beta_{ij} < g$, where $M(\alpha_{ij})$ is the first and $M(\beta_{ij})$ is the last snapshot that contain operation O_{ij},
- for two operations O_{ij} and O_{i+1j} of the same job $J_j \in \mathcal{L}$, $\beta_{ij} < \alpha_{i+1j}$, and
- consecutive snapshots are different.

We observe that g can be bounded by $2k\mu+1$. A relative schedule corresponds to an order of execution for the jobs in $\mathcal{L}$. One can associate a relative schedule to each non-preemptive schedule of $\mathcal{L}$ by looking at every time in the schedule when a task of $\mathcal{L}$ starts or ends and creating a snapshot right after that time. Notice, that we have included one further snapshot $M(1) = \emptyset$ without operations of jobs

in $\mathcal{L}$. This corresponds to the case when the schedule for $\mathcal{J}$ does not contain any operation O_{ij} of a job $J_j \in \mathcal{L}$ at the beginning. For any snapshot $M(\ell)$ let $P(\ell) = \bigcup_{O_{ij} \in M(\ell), J_j \in \mathcal{L}} \{m_{ij}\}$, be the set of machines that process operations from long jobs during snapshot $M(\ell)$.

For each relative schedule $R = (M(1), \ldots, M(g))$ of $\mathcal{L}$, we formulate a linear program $LP(R)$ as follows. For each job $J_j \in \mathcal{S}$ we use a set of decision variables $x_{j,(i_1,\ldots,i_\mu)} \in [0,1]$ for tuples $(i_1, \ldots, i_\mu) \in A$, where $A = \{(i_1, \ldots, i_\mu) | 1 \leq i_1 \leq i_2 \leq \ldots \leq i_\mu \leq g\}$.

The meaning of these variables is that $x_{j,(i_1,\ldots,i_\mu)} = 1$ if and only if each operation O_{kj} of job J_j is scheduled in snapshot i_k for each $1 \leq k \leq \mu$. Note that by the way in which we numbered the operations, any tuple $(i_1, \ldots, i_\mu) \in A$ represents a valid ordering for the operations. Let the load $L_{\ell,h}$ on machine h in snapshot $M(\ell)$ be defined as the total processing time of operations from short jobs that are executed by machine h during snapshot ℓ, i.e., $L_{\ell,h} = \sum_{J_j \in \mathcal{S}} \sum_{(i_1,\ldots,i_\mu) \in A} \sum_{k=1,\ldots,\mu | i_k = \ell, m_{kj} = h} x_{j,(i_1,\ldots,i_\mu)} p_{kj}$. Each variable t_ℓ, $1 \leq \ell \leq g$, denotes the length of snapshot $M(\ell)$. The linear program for a given relative schedule R is the following.

Minimize $\sum_{\ell=1}^{g} t_\ell$

s.t.
- **(1)** $t_\ell \geq 0$, for all $1 \leq \ell \leq g$
- **(2)** $\sum_{\ell=\alpha_{ij}}^{\beta_{ij}} t_\ell = p_{ij}$ for every operation O_{ij} $1 \leq i \leq \mu$, $J_j \in \mathcal{L}$,
- **(3)** $x_{j,(i_1,\ldots,i_\mu)} \geq 0$ for each $J_j \in \mathcal{S}$, $(i_1, \ldots, i_\mu) \in A$,
- **(4)** $\sum_{(i_1,\ldots,i_\mu) \in A} x_{j,(i_1,\ldots,i_\mu)} = 1$ for every $J_j \in \mathcal{S}$,
- **(5)** $L_{\ell,h} \leq t_\ell$, for every $1 \leq \ell \leq g$, $1 \leq h \leq m$ and $h \notin P(\ell)$,
- **(6)** $x_{j,(i_1,\ldots,i_\mu)} = 0$ for $J_j \in \mathcal{S}$ and $m_{kj} \in P(i_k)$.

To avoid that some operation O_{kj} of a short job $J_j \in \mathcal{S}$ is scheduled on a machine $m_{kj} \in P(i_k)$ occupied by a long job, we set $x_{j,(i_1,\ldots,i_\mu)} = 0$ (constraint (6)). Note that if a variable $x_{j,(i_1,\ldots,i_\mu)}$ has value one, then job $j \in \mathcal{S}$ has a unique snapshot assignment.

Lemma 1. *An optimum solution of $LP(R)$ has value no larger than the makespan of an optimum schedule for $\mathcal{J}$ that respects the relative schedule R.*

Proof. See Appendix.

Remark: One can solve the linear program $LP(R)$ optimally in polynomial time and get only a constant number $(m+1)g \leq (m+1)(2|\mathcal{L}|\mu + 1)$ of jobs with non-unique snapshot assignment (see also paper by Potts [12] and Hall [7]). However for our problem we do not need to solve exactly the linear program, an approximate solution suffices. We show below how to get this approximate solution in linear time.

Let $OPT(I)$ be the makespan of an optimum schedule, let $d_j = \sum_{i=1}^{\mu} p_{ij}$ be the total processing time of job $J_j \in \mathcal{J}$, and let $D = \sum_{J_j \in \mathcal{J}} d_j$. Then, we get directly the following bounds:

Lemma 2.

$$\frac{D}{m} \leq OPT(I) \leq D.$$

By dividing all execution times p_{ij} by D, we may assume that $D = 1$ and $\frac{1}{m} \leq OPT(I) \leq 1$. So we can use in our LP the bound $\sum_{\ell=1}^{g} t_\ell \leq 1$ as additional constraint. Given a relative schedule R, the minimum makespan $OPT_R(I)$ of a schedule respecting R satisfies also $\frac{1}{m} \leq OPT_R(I) \leq 1$.

3 The Algorithm

For each relative schedule R we approximately solve the linear program $LP(R)$. Then, we round the solution to get only few fractional variables. Next, we transform the solution of the linear program into a feasible schedule. Finally, we select one schedule that has the smallest makespan.

3.1 Approximate Solution of the Linear Program

We guess the length s of an optimum schedule and add the constraint $\sum_{\ell=1}^{g} t_\ell \leq s$ to $LP(R)$. Then we replace constraint (5) by constraint (5′), where λ is a non-negative value:

(5') $\quad L_{\ell,h} - t_\ell + 1 \leq \lambda, \quad$ for $h \notin P(\ell)$.

This new linear program will be denoted as $LP(R, s, \lambda)$. The linear program $LP(R, s, \lambda)$ has a special *block angular* structure (for a survey see [6, 11]). The blocks $B_j = \{x_{j,(i_1,\ldots,i_\mu)} \mid$ conditions (3), (4), and (6) are satisfied $\}$ for $J_j \in \mathcal{S}$ are g^μ-dimensional *simplicies*, while the block $B_{|\mathcal{S}|+1} = \{t_\ell \mid \sum_{\ell=1}^{g} t_\ell \leq s$ and conditions $(1)-(2)$ are satisfied$\}$ contains only a constant number of variables and constraints. Linear inequalities (5′) form a set of so called *coupling constraints*. Let $f_{\ell,h} = L_{\ell,h} - t_\ell + 1$. Since $t_\ell \leq s \leq 1$, these functions are non-negative.

The Logarithmic Potential Price Directive Decomposition Method [6] developed by Grigoriadis and Khachiyan for a large class of problems with block angular structure provides a ρ-relaxed decision procedure for $LP(R, s, 1)$. This procedure either determines that $LP(R, s, 1)$ is infeasible, or computes (a solution that is nearly feasible in the sense that it is) a feasible solution of $LP(R, s, (1 + \rho))$. The overall running time of the procedure for $LP(R, s, 1)$ is $O(n)$.

Recall that $OPT(I)$ is the optimum makespan for a given instance I of the job shop problem. For brevity in the following, we use OPT instead of $OPT(I)$ and OPT_R instead of $OPT_R(I)$.

Lemma 3. *The following assertions are true:*

(0) *$LP(R, 1, 1)$ is feasible for each relative schedule R,*
(1) *if $LP(R, s, 1)$ is feasible and $s \leq s'$, then $LP(R, s', 1)$ is feasible,*

(2) *if $LP(R,s,1)$ is infeasible, then there exists no schedule (that respects the relative schedule R for the long jobs in $\mathcal{L}$) with makespan at most s,*
(3) *$LP(R,OPT_R,1)$ is feasible for each relative schedule R,*

Let R be a relative schedule for $\mathcal{L}$ in an optimum schedule. This Lemma implies that we can use binary search on the interval $[\frac{1}{m},1]$ to find a value $s \leq (1+\frac{\epsilon}{4})OPT$ such that $LP(R,s,(1+\rho))$ is feasible for $\rho = \frac{\epsilon}{4mg}$. This search can be performed in $O(\log m/\epsilon)$ iterations. For this value s a solution to $LP(R,s,1+\rho)$ has value $\sum_{\ell=1}^{g} t_\ell \leq OPT(1+\frac{\epsilon}{4})$, and machine loads $L_{\ell,h} \leq t_\ell + \rho$. Since $\rho = \frac{\epsilon}{4mg}$ we obtain a fractional solution of value $\sum_{\ell=1}^{g}(t_\ell+\rho) \leq (1+\frac{\epsilon}{2})OPT$.

Lemma 4. *A solution for $LP(R,s,1+\rho)$, with $s \leq (1+\frac{\epsilon}{4})OPT$ and $\rho = \frac{\epsilon}{4mg}$, of value at most $(1+\frac{\epsilon}{2})OPT$ can be found in linear time.*

3.2 Rounding Step

In this section we show how we can modify any feasible solution for $LP(R,s,1+\rho)$ to get a new feasible solution in which all but a constant number of variables $x_{j,(i_1,\ldots,i_\mu)}$ have value 0 or 1. Moreover we can do this rounding step in linear time. Let $n' = |\mathcal{S}|$.

Let us write the linear program $LP(R,s,1+\rho)$ in matrix form as $Bx = b$, $x \geq 0$. We arrange the columns of B so that the last g columns contain the coefficients for variables t_ℓ and the first $n'g^\mu$ columns have the coefficients for variables $x_{j,(i_1,\ldots,i_\mu)}$. The key observation that allows us to perform the rounding in linear time is to note that matrix B is sparse. In particular, each one of the first $n'g^\mu$ columns has at most $\mu+1$ non-zero entries, and it is not difficult to show that there exists a constant size subset B' of these columns in which the number of non-zero rows is smaller than the number of columns. The non-zero entries of B' induce a singular matrix of constant size, so we can find a non-zero vector y in the null space of this matrix, i.e., $B'y = 0$.

Let $\delta > 0$ be the smallest value such that some component of the vector $x+\delta y$ is either zero or one (if the dimension of y is smaller than the dimension of x we augment it by adding an appropriate number of zero entries). Note that the vector $x+\delta y$ is a feasible solution of $LP(R,s,1+\rho)$. Let $x^0$and x^1 be respectively the zero and one components of vector $x+\delta y$. We update the linear program by making $x = x+\delta y$ and then removing from x all variables in x^0 and x^1 and all columns of B corresponding to such variables. If $x^1 \neq \emptyset$ then vector b is set to $b - \sum_{i \in x^1} B[*,i]$, where $B[*,i]$ is the column of B corresponding to variable i.

This process rounds the value of at least one variable $x_{j,(i_1,\ldots,i_\mu)}$ to either 0 or 1 and it can be carried out in constant time since the sizes of the submatrices involved are constant. We note that the value of δ can be found in constant time also since y has constant size. We can repeat this process until only a constant number of variables $x_{j,(i_1,\ldots,i_\mu)}$ have fractional values. Since there is a linear number of these variables then the overall time is linear.

Now we describe the rounding algorithm in more detail. Let us assume that the first $n'g^\mu$ columns of B are indexed so that the columns corresponding to

variables $x_{j,(i_1,\ldots,i_\mu)}$ for each job J_j appear in adjacent positions. We might assume that at all times during the rounding procedure each job J_j has associated at least two columns in B. This assumption can be made since if job J_j has only one associated column, then the corresponding variable $x_{j,(i_1,\ldots,i_\mu)}$ must have value either zero or one. Let B' be the set formed by the first $2mg+1$ columns of B. Note that at most $2mg$ rows of B' have non-zero entries. To see this observe that at most mg of these entries come from constraint (4), while at most mg non-zero entries come from constraint (5′) and by the above assumption on the number of columns for each job.

To avoid introducing more notation let B' be the matrix induced by the non-zero rows. Since B' has at most $2mg$ rows and exactly $2mg+1$ columns then B' is singular and hence its null space has at least one non-zero vector y. To find y we can use elementary row operations to convert B' to row-echelon form $\begin{pmatrix} I & B'' \\ 0 & 0 \end{pmatrix}$ where I is an identity matrix of rank equal to the rank of B' and B'' is a non-zero matrix. Then it is easy to find a vector y as described above.

After updating x, B, and b as described above, the procedure is repeated. This is done until there are at most $2mg$ columns in B corresponding to variables $x_{j,(i_1,\ldots,i_\mu)}$. Hence the total number of iterations is at most $n'g^\mu - 2mg$ and each iteration can be done in constant time.

Lemma 5. *A solution for $LP(R,s,1+\rho)$ can be transformed in linear time into another feasible solution for $LP(R,s,1+\rho)$ in which the set of jobs $\mathcal{F}$ that have fractional assignments in more than one snapshot after the rounding procedure has size $|\mathcal{F}| \leq mg \leq m(2k\mu+1)$.*

Proof. By the above argument at most $2mg$ variables $x_{j,(i_1,\ldots,i_\mu)}$ might have value different from 0 and 1. Since at the end of the rounding procedure each job has either 0 or at least 2 columns in B, then at most mg jobs receive fractional assignments.

3.3 Generating a Schedule

After rounding the solution for $LP(R,s,1+\rho)$ as described above, we remove all jobs $\mathcal{F}$ that received fractional assignment. These jobs will be placed at the end of the schedule. Using the approximate solution for the LP, our snapshots are enlarged from t_ℓ to at most $t'_\ell = t_\ell + \rho$.

Consider an snapshot $M(\ell)$, $\ell \in \{1,\ldots,g\}$. In this snapshot the set $P(\ell)$ contains machines that are reserved for the long jobs. The set of operations $O(\ell)$ from small jobs assigned by the solution of the linear program must be scheduled in machines $M \setminus P(\ell)$. Find in each machine $i \in M \setminus P(\ell)$ the set $O_\ell(i)$ formed by the $\mu^3 m$ largest operations assigned to it in $M(\ell)$. Let $\mathcal{V}_\ell(i)$ be the set of jobs corresponding to operations $O_\ell(i)$ and let $\mathcal{V}_\ell = \bigcup_{i \notin P(\ell)} \mathcal{V}_\ell(i)$. Remove from $M(\ell)$ all jobs in $\mathcal{V}_\ell$ (these jobs will be scheduled later).

Let $p_{max}(\ell)$ be the maximum processing time among the remaining operations from $O(\ell)$. Without loss of generality we may assume that the largest remaining operation is assigned to machine M_ℓ. Let $\mathcal{V} = \bigcup_{\ell=1}^{g} \mathcal{V}_\ell$.

Lemma 6.

$$|\mathcal{V}| \leq \mu^3 m^2 g.$$

Now, the load of machine M_ℓ is at most $t'_\ell - \mu^3 m p_{max}(\ell)$ and all operations have processing time no larger than $p_{max}(\ell)$. We notice that the load of the other machines $i \in M \setminus P(\ell)$, $i \neq M_\ell$, could be close to t'_ℓ. Since we have deleted the largest $\mu^3 m$ operations on machine M_ℓ (all with length $\geq p_{max}(\ell)$), we get the following bound:

Lemma 7.

$$\mu^3 m p_{max}(\ell) \leq S(\mathcal{O}_\ell(M_\ell)),$$

where $S(\mathcal{O}_\ell(M_\ell))$ is the total processing time of the operations $\mathcal{O}_\ell(M_\ell)$.

On the modified snapshot $M'(\ell)$ the remaining operations from the small jobs form an instance of the job shop problem with maximum machine load t'_ℓ and maximum operation length $p_{max}(\ell)$. We use the algorithm of Sevastianov [14] to find in $O(n^2\mu^2 m^2)$ time a schedule for the operations in $M'(\ell)$ of length at most $\bar{t}_\ell = t_\ell + \rho + \mu^3 m p_{max}(\ell)$. Hence we get an enlarged snapshot $M'(\ell)$ of length at most $\bar{t}_\ell$. Summing these enlargements among all snapshots, we get:

Lemma 8.

$$\sum_{\ell=1}^{g} \mu^3 m p_{max}(\ell) \leq \sum_{\ell=1}^{g} S(\mathcal{O}_\ell(M_\ell)) \leq S(\mathcal{V}).$$

The total length of the snapshots $M'(\alpha_{ij}), \ldots, M'(\beta_{ij})$ that contain an operation O_{ij} of a long job J_j might be larger than p_{ij}. This creates some idle times on machine m_{ij}. We start operations O_{ij} for long jobs $\mathcal{L}$ at the beginning of the enlarged snapshot $M'(\alpha_{ij})$. The resulting schedule is clearly feasible.

Lemma 9. *A feasible schedule for the jobs $\mathcal{J}$ of length at most $(1+\frac{\epsilon}{2})OPT_R(I) + S(\mathcal{V}) + \mathcal{S}(\mathcal{F} \cup \mathcal{V})$ can be found in $O(n^2)$ time.*

Proof. The small jobs $\mathcal{F}$ that received fractional assignment and the jobs in $\mathcal{V}$ are scheduled sequentially at the end of the schedule produced by the above algorithm. Thus, by Lemmas 4 and 8 the claim follows.

4 Analysis of the Algorithm

Lemma 10. *Let $k = |\mathcal{L}|$. The total number of relative schedules is at most $(2k^2\mu^2)^{k\mu}$.*

Since k, m and μ are constant, the number of relative schedules is also constant.

Lemma 11. *Let $d_1 \geq d_2 \geq \ldots d_n > 0$ be a sequence of real numbers and $1 = \sum_{j=1}^{n} d_j$. Let q be a nonnegative integer, α be a positive value, and assume that $n \geq (q+1)^{\lceil \frac{1}{\alpha} \rceil}$. There exists an integer k such that $d_k + \ldots + d_{k+qk-1} \leq \alpha$ and $k \leq (q+1)^{\lceil \frac{1}{\alpha} \rceil - 1}$.*

Let $\alpha = \frac{\epsilon}{4m}$ and $qk = 5\mu^4 m^2 k$, i.e. $q = 5\mu^4 m^2$. By this lemma, $k \leq (5\mu^4 m^2 + 1)^{\lceil 4m/\epsilon \rceil - 1}$. We choose $\mathcal{L}$ as the set of $k-1$ jobs with the largest values $d_j = \sum_{i=1}^{\mu} p_{ij}$.

Lemma 12.

$$S(\mathcal{F} \cup \mathcal{V}) \leq \frac{\epsilon}{4} OPT_R(I).$$

Proof. By Lemmas 5 and 6, $|\mathcal{F} \cup \mathcal{V}| \leq \mu^3 m^2 g + mg \leq 2\mu^4 m^2 k + \mu^3 m^2 + 2mk\mu + m \leq 5\mu^4 m^2 k$. By the preceding discussion on the choice of $\mathcal{L}$ and by Lemma 11, the total processing time for the jobs in $\mathcal{F} \cup \mathcal{V}$ is at most $\frac{\epsilon}{4m} \leq \frac{\epsilon}{4} OPT_R(I)$.

Theorem 1. *For any fixed m and μ, there is a polynomial-time approximation scheme for the job shop scheduling problem that computes for any fixed $\epsilon > 0$ a feasible schedule of makespan at most $(1+\epsilon) \cdot OPT$ in $O(n^2)$ time.*

Proof. By considering all relative schedules R for $\mathcal{L}$ the claim on the length of the makespan follows from Lemmas 9 and 12. For every fixed m and ϵ, all computations can be carried out in $O(n)$ time, with exception of the algorithm of Sevastianov that runs in $O(n^2)$ time.

4.1 Speed Up to Linear Time

We show how to speed up the algorithm of Sevastianov [14] to get linear time in Theorem 1. Given an instance of the job shop problem with m machines and μ operations per job, Sevastianov's algorithm finds in $O(n^2 \mu^2 m^2)$ time a schedule of length at most $\Pi_{max} + \mu^3 m p_{max}$.

We may assume that $\Pi_{max} \leq 1$ (by normalization). Let $d_j = \sum_{\ell=1}^{\mu} p_{\ell j}$. For a job J_j let $m_j = (m_{1j}, m_{2j}, \ldots, m_{\mu j})$ be a vector that describes the machines on which its operations must be performed. Let us partition the set of jobs $\mathcal{J}$ into m^μ groups $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_{m^\mu}$ such that all jobs J_j in some group $\mathcal{J}_i$ have the same machine vector m_j, and jobs from different groups have different machine vectors. Consider the jobs in one of the groups $\mathcal{J}_i$. Let γ be a constant to be defined later. Let J_j and J_h be two jobs from $\mathcal{J}_i$ such that for each one of them the total execution time of its operations is smaller than γ. We "glue" together these two jobs to form a composed job in which the processing time of the i-th operation is equal to the sum of the processing times of the i-th operations of J_j and J_h. We repeat this process until at most one job from $\mathcal{J}_i$ has processing time smaller than γ. At the end, all jobs in group $\mathcal{J}_i$ have processing times smaller than $max(p_{max}, 2\gamma)$.

The same procedure is performed in all other groups $\mathcal{J}_j$. At the end of this process, each one of the composed jobs has at most μ operations, and the total number of composed jobs is no larger than $\frac{m}{\gamma} + m^\mu$. Note that the procedure runs in linear time and a feasible schedule for the original jobs can be easily obtained from a feasible schedule for the composed jobs.

We run Sevastianov's algorithm on the set of composed jobs to get a schedule of length $\Pi_{max} + \mu^3 mr$, where $r = \max\{p_{max}, 2\gamma\}$. The time needed to get

this schedule is $O((\frac{m}{\gamma} + m^{\mu})^2 \mu^2 m^2)$. If we choose $\gamma = \frac{\epsilon}{8m^2\mu^3 g}$ we get a total lengthening for the snapshots of at most $\frac{\epsilon}{4}OPT$. By this choice of γ Sevastianov's algorithm needs only constant time plus linear pre-processing time.

Theorem 2. *For any fixed m and μ, there is a polynomial-time approximation scheme for the job shop scheduling problem that computes for any fixed $\epsilon > 0$ in $O(n)$ time a feasible schedule whose makespan is at most $(1+\epsilon) \cdot OPT$.*

5 Preemptive Job Shop Problem

As in the non-preemptive case we divide the set of jobs $\mathcal{J}$ into long jobs $\mathcal{L}$ and short jobs $\mathcal{S}$. Set $\mathcal{L}$ has a constant number of jobs, as in the non-preemptive case. Consider a preemptive schedule for $\mathcal{J}$. Look at the time at which any operation from a long job starts or ends. These times define a set of time intervals, similar to those defined in the non-preemptive case by the snapshots. For convenience we also call these time intervals snapshots.

Since $\mathcal{L}$ has a constant number of jobs, we can look at all relative orderings of the long jobs in the snapshots. Now an operation of a long job will be schedule in consecutive snapshots $i, i+1, \ldots, i+t$, *but* only a fraction (possible equal to zero) of the operation might be scheduled in any one of these snapshots. However in every snapshot there is at most one operation from any given long job.

Now we define a linear program as in the case of the non-preemptive job shop. For each long job J_j we define variables $x_{j,(i_1,\ldots,i_\mu)}$ for those tuples $(i_1, \ldots, i_\mu)$ corresponding to snapshots where the operations of J_j might be scheduled, as described above. Let g be the number of snapshots, t_ℓ be the length of the ℓ snapshot, and $L_{\ell,h}$ be as before. The new linear program is the following.

Minimize $\sum_{\ell=1}^{g} t_\ell$

s.t. **(1)** $t_\ell \geq 0$, for all $1 \leq \ell \leq g$

(3) $x_{j,(i_1,\ldots,i_\mu)} \geq 0$ for each $J_j \in \mathcal{S}$, $(i_1, \ldots, i_\mu) \in A$,

(3') $x_{j,(i_1,\ldots,i_\mu)} \geq 0$ for each $J_j \in \mathcal{L}$, and $(i_1, \ldots, i_\mu)$ as described above,

(4) $\sum_{(i_1,\ldots,i_\mu)\in A} x_{j,(i_1,\ldots,i_\mu)} = 1$ for every $J_j \in \mathcal{J}$,

(5) $L_{\ell,h} \leq t_\ell$, for every $1 \leq \ell \leq g$, $1 \leq h \leq m$,

In a solution for this linear program the scheduling for the long jobs is always feasible, since there is at most one operation of a given job in any snapshot. Now we find an approximate solution to the linear program and then apply our rounding procedure. After rounding there are at most mg small jobs that are preempted.

Now we find a feasible schedule for every snapshot. Consider a snapshot. First remove from the snapshot the operations belonging to long jobs. These operations will be added to the schedule later. Then we use Sevastianov's algorithm to find a feasible schedule for the (fractions of) small jobs in the snapshot. Finally we reintroduce the operations from the long jobs placing them in the

empty gaps left by this schedule. At the end we have a feasible schedule because there is at most one operation of each long job in the snapshot.

In this schedule the number of preemptions is at most mg (after rounding the linear program) + $n\mu + mg$ (since after introducing the operations from the long jobs, we might have this many preemptions for them). So there are in total $O(n)$ preemptions. Choosing the size of $\mathcal{L}$ as in the non-preemptive case we ensure that this algorithm is a polynomial time approximation scheme for the preemptive case. This algorithm also runs in linear time.

6 Conclusions

We have designed an approximation scheme for the job shop problem. The running time for the algorithm is $O(n)$. Using our approach we can improve the running time of approximation schemes for open shops [17] and flow shops [7] to $O(n)$ time. We have described also a linear time approximation scheme for the preemptive version of the problem. Our method can be generalized to flexible job shops, dag shops, and job shop problems with release and delivery times.

References

1. A.K. Amoura, E. Bampis, C. Kenyon and Y. Manoussakis, Scheduling independent multiprocessor tasks, *Proceedings of the 5th Annual European Symposium on Algorithms*, LNCS 1284 (1997), 1-12.
2. M.R. Garey, D.S. Johnson and R. Sethi, The complexity of flowshop and jobshop scheduling, *Mathematics of Operations Research* 1 (1976), 117-129.
3. T. Gonzales and S. Sahni, Open shop scheduling to minimize finish time, *Journal of the ACM* 23 (1976), 665-679.
4. T. Gonzales and S. Sahni, Flowshop and jobshop schedules: complexity and approximation, *Operations Research* 26 (1978), 36-52.
5. L.A. Goldberg, M. Paterson, A. Srinivasan, and E. Sweedyk, Better approximation guarantees for job-shop scheduling, *Proceedings of the 8th Symposium on Discrete Algorithms* SODA 97, 599-608.
6. M.D. Grigoriadis and L.G. Khachiyan, Coordination complexity of parallel price-directive decomposition, *Mathematics of Operations Research* 21 (1996), 321-340.
7. L.A. Hall, Approximability of flow shop scheduling, *Mathematical Programming* 82 (1998), 175-190.
8. K. Jansen and L. Porkolab, Linear-time approximation schemes for scheduling malleable parallel tasks, *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, to appear, 1999.
9. K. Jansen, R. Solis-Oba and M.I. Sviridenko, Makespan minimization in job shops: a polynomial time approximation scheme, *Proceedings of the 31th Annual ACM Symposium on Theory of Computing*, to appear, 1999.
10. E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, Sequencing and scheduling: Algorithms and complexity, in: Handbook in Operations Research and Management Science, Vol. 4, North-Holland, 1993, 445-522.
11. S.A. Plotkin, D.B. Shmoys and E. Tardos, Fast approximation algorithms for fractional packing and covering problems, *Mathematics of Operations Research* 20 (1995), 257-301.

12. C.N. Potts, Analysis of a linear programming heuristic for scheduling unrelated parallel machines, *Discrete Applied Mathematics* 10 (1985), 155-164.
13. J.P. Schmidt, A. Siegel, A. Srinivasan, Chernoff-Hoeffding bounds for applications with limited independence, *SIAM Journal Discrete Mathematics* 6 (1995), 223-250.
14. S.V. Sevastianov, Bounding algorithms for the routing problem with arbitrary paths and alternative servers, *Cybernetics* 22 (1986), 773-780.
15. S.V. Sevastianov, On some geometric methods in scheduling theory: a survey, *Discrete Applied Mathematics* 55 (1994), 59-82.
16. S.V. Sevastianov: Nonstrict vector summation in multi-operation scheduling, *Annals of Operations Research* 83 (1998), 179-212.
17. S.V. Sevastianov and G.J. Woeginger, Makespan minimization in open shops: A polynomial time approximation scheme, *Mathematical Programming* 82 (1998), 191-198.
18. D.B. Shmoys, C. Stein and J. Wein, Improved approximation algorithms for shop scheduling problems, *SIAM Journal on Computing* 23 (1994), 617-632.
19. D.P. Williamson, L.A. Hall, J.A. Hoogeveen, C.A.J. Hurkens, J.K. Lenstra, S.V. Sevastianov and D.B. Shmoys, Short shop schedules, *Operations Research* 45 (1997), 288-294.

7 Appendix

Proof of Lemma 2.1. Consider an optimum schedule S^* for $\mathcal{J}$ that respects the relative schedule R. We only need to show that for any job $J_j \in S$ there is a feasible solution of $LP(R)$ that schedules all operations of J_j in the same positions as S^*.

Assume that S^* schedules each operation O_{kj} on machine m_{kj} for a time interval p_{kj} that spans consecutive snapshots $M(s_{kj}), \ldots, M(e_{kj})$, where s_{kj} might be equal to e_{kj} (corresponding to the case when the operation is completely scheduled in a single snapshot). Let $f_{kj}(i)$ be the fraction of operation O_{kj} that is scheduled in snapshot $M(i)$.

We assign values to the variables $x_{j,(i_1,\ldots,i_\mu)}$ as follows. Set $x_{j,(s_{1j},s_{2j},\ldots,s_{\mu j})} = f$, where $f = \min\{f_{kj}(s_{kj}) \mid 1 \leq k \leq \mu\}$ is the minimum fraction of an operation of job J_j that is scheduled in the first snapshot assigned to it by S^*. Now we have to assign values to the other variables $x_{j,(i_1,\ldots,i_\mu)}$ to cover the remaining $1 - f$ fraction of each operation. To do this, for every operation O_{kj}, we make $f_{kj}(s_{kj}) = f_{kj}(s_{kj}) - f$. Clearly for at least one operation O_{kj} the new value of $f_{kj}(s_{kj})$ will be set to zero; for those operations with $f_{kj}(s_{kj}) = 0$ we make $s_{kj} = s_{kj} + 1$ since the beginning snapshot for the rest of the operation O_{kj} is snapshot $M(s_{kj} + 1)$. Then we assign value to the new $x_{j,(s_{1j},s_{2j},\ldots,s_{\mu j})}$ as above and repeat the process until $f = 0$.

Note that each iteration of this process assigns a value to a different variable $x_{j,(i_1,\ldots,i_\mu)}$ since from one iteration to the next at least one snapshot s_{kj} is changed. This assignment of values to variables $x_{j,(i_1,\ldots,i_\mu)}$ satisfies the constraints of the linear program and assigns the same time intervals than S^*. Thus the claim follows.