Approximation Algorithms for Flexible Job Shop Problems

Klaus Jansen[1], Monaldo Mastrolilli[2], and Roberto Solis-Oba[3]

[1] Institut für Informatik und Praktische Mathematik, Universität zu Kiel, Germany
kj@informatik.uni-kiel.de †
[2] IDSIA Lugano, Switzerland, monaldo@idsia.ch ‡
[3] Department of Computer Science, The University of Western Ontario, Canada
solis@brown.csd.uwo.ca §

Abstract. The Flexible Job Shop Problem is a generalization of the classical job shop scheduling problem in which for every operation there is a group of machines that can process it. The problem is to assign operations to machines and to order the operations on the machines, so that the operations can be processed in the smallest amount of time. We present a linear time approximation scheme for the non-preemptive version of the problem when the number m of machines and the maximum number μ of operations per job are fixed. We also study the preemptive version of the problem when m and μ are fixed, and present a linear time $(2+\varepsilon)$-approximation algorithm for the problem with migration.

1 Introduction

The job shop scheduling problem is a classical problem in Operations Research [10] in which it is desired to process a set $\mathcal{J} = \{J_1, \ldots, J_n\}$ of n jobs on a group $M = \{1, \ldots, m\}$ of m machines in the smallest amount of time. Every job J_j consists of a sequence of μ operations $O_{1j}, O_{2j}, \ldots, O_{\mu j}$ which must be processed in the given order. Every operation O_{ij} has assigned a unique machine $m_{ij} \in M$ which must process the operation without interruption during p_{ij} units of time, and a machine can process at most one operation at a time.

In this paper we study a generalization of the job shop scheduling problem called the *flexible job shop problem* [1], which models a wide variety of problems encountered in real manufacturing systems [1, 13]. In the flexible job shop problem an operation O_{ij} can be processed by any machine from a given group $M_{ij} \subseteq M$. The processing time of operation O_{ij} on machine $k \in M_{ij}$ is p_{ij}^k. The goal is to choose for each operation O_{ij} an eligible machine and a starting time so that the maximum completion time $C_{\max}$ over all jobs is minimized. $C_{\max}$ is called the *makespan* or the *length* of the schedule.

† This research was done while the author was at IDSIA Lugano, Switzerland.
‡ This author was supported by the Swiss National Science Foundation project 21-55778.98.
§ This research was done while the author was at MPII Saarbrücken, Germany.

The flexible job shop problem is more complex than the job shop problem because of the need to assign operations to machines. Following the three-field $\alpha|\beta|\gamma$ notation suggested by Vaessens [13] and based on that of [4], we denote our problem as $m1m|chain, op \leq \mu|C_{\max}$. In the first field m specifies that the number of machines is a constant, 1 specifies that any operation requires at most one machine to be processed, and the second m gives an upper bound on the number of machines that can process an operation. The second field states the precedence constraints and the maximum number of operations per job, while the third field specifies the objective function. The following special cases of the problem are already NP-hard (see [13] for a survey): 2 1 2$|chain, n = 3|C_{\max}$, 3 1 2$|chain, n = 2|C_{\max}$, 2 1 2$|chain, op \leq 2|C_{\max}$.

The job shop scheduling problem has been extensively studied. The problem is known to be strongly NP-hard even if each job has at most three operations and there are only two machines [10]. Williamson et al. [14] proved that when the number of machines, jobs, and operations per job are part of the input there does not exist a polynomial time approximation algorithm with worst case bound smaller than $\frac{5}{4}$ unless $P = NP$. On the other hand the preemptive version of the job shop scheduling problem is NP-complete in the strong sense even when $m = 3$ and $\mu = 3$ [3]. Jansen et al. [8] have designed a linear time approximation scheme for the case when m and μ are fixed. When m and μ are part of the input the best known result [2] is an approximation algorithm with worst case bound $O([\log(m\mu)\log(\min\{m\mu, p_{max}\})/\log\log(m\mu)]^2)$, where p_{max} is the largest processing time among all operations.

Scheduling jobs with chain precedence constraints on unrelated parallel machines is a special case of the flexible job shop problem. For the first problem, Shmoys et al. [12] designed a polynomial-time randomized algorithm that, with high probability, finds a schedule of length at most $O((\log^2 n/\log\log n)C^*_{\max})$, where $C^*_{\max}$ is the optimal makespan.

In this work we study the preemptive and non-preemptive versions of the flexible job shop scheduling problem when the number of machines m and the number of operations per job μ are fixed. We generalize the techniques described in [8] for the job shop scheduling problem and design a linear time approximation scheme for the flexible job shop problem. In addition, each job J_j has a *delivery time* q_j. If in a schedule J_j completes its processing at time C_j, then its *delivery completion time* is equal to $C_j + q_j$. The problem now is to find a schedule that minimizes the maximum delivery completion time L_{max}. We notice that by using the same techniques we can also handle the case in which each job J_j has a *release time* r_j when it becomes available for processing and the objective is to minimize the makespan.

Our techniques allow us also to design a linear time approximation scheme for the preemptive version of the flexible job shop problem without migration. No migration means that each operation must be processed by a unique machine. So if an operation is preempted, its processing can only be resumed on the same machine on which it was being processed before the preemption. Due to space limitations we do not describe this algorithm here. We also study the preemptive

flexible job shop problem with migration, and present a $(2+\varepsilon)$-approximation algorithm for it. The last algorithm handles release and delivery times, and both of them produce solutions with only a constant number of preemptions.

2 The Non-preemptive Flexible Job Shop Problem

Consider an instance of the flexible job shop problem with release and delivery times. Let $L^*_{\max}$ be the length of an optimum schedule. For every job J_j, let $P_j = \sum_{i=1}^{\mu}[\min_{s \in M_{kj}} p^s_{ij}]$ denote its minimum processing time. Let $P = \sum_{J_j \in \mathcal{J}} P_j$. Let r_j be the release time of job J_j and q_j be its delivery time. We define $t_j = r_j + P_j + q_j$ for all jobs J_j, and we let $t_{\max} = \max_j t_j$.

Lemma 1.

$$\max\left\{\frac{P}{m}, t_{\max}\right\} \leq L^*_{\max} \leq P + t_{\max}. \tag{1}$$

We divide all processing, release, and delivery times by $\max\left\{\frac{P}{m}, t_{\max}\right\}$, and thus by Lemma 1,

$$1 \leq L^*_{\max} \leq m+1, \text{ and } t_{\max} \leq 1. \tag{2}$$

We observe that Lemma 1 holds also for the preemptive version of the problem with or without migration. Here we present an algorithm for the non-preemptive flexible job shop problem that works for the case when all release times are zero. The algorithm works as follows. First we show how to transform an instance of the flexible job shop problem into another instance without delivery times. Then we define a set of time intervals and assign operations to the intervals so that operations from the same job that are assigned to different intervals appear in the correct order, and the total length of the intervals is no larger than the length of an optimum schedule. We perform this step by first fixing the position of the operations from a constant number of jobs (which we call the long jobs), and then using linear programming to determine the position of the remaining operations.

Next we use an algorithm by Sevastianov [11] to find a feasible schedule for the operations within each interval. Sevastianov's algorithm finds for each interval a schedule of length equal to the length of the interval plus $m\mu^3 p_{\max}$, where $p_{\max}$ is the largest processing time of any operation in the interval. In order to keep this enlargement small, we remove from each interval a subset $\mathcal{V}$ of jobs with large operations before running Sevastianov's algorithm. Those operations are scheduled at the beginning of the solution, and by choosing carefully the set of long jobs we can show that the total length of the operations in $\mathcal{V}$ is very small compared to the overall length of the schedule.

2.1 Getting Rid of the Delivery Times

We use a technique by Hall and Shmoys [6] to transform an instance of the flexible job shop problem into another with only a constant number of different

delivery times. Let $q_{\max}$ be the maximum delivery time and let $\varepsilon > 0$ be a constant value. The idea is to round each delivery time down to the nearest multiple of $\frac{\varepsilon}{2}q_{\max}$ to get at most $1 + 2/\varepsilon$ distinct delivery times. Next, apply a $(1 + \varepsilon/2)$-approximation algorithm for the flexible job shop problem that can handle $1+2/\varepsilon$ distinct delivery times (this algorithm is described below). Finally, add $\frac{\varepsilon}{2}q_{\max}$ to the completion time of each job; this increases the length of the solution by $\frac{\varepsilon}{2}q_{\max}$. The resulting schedule is feasible for the original instance, so this is a $(1 + \varepsilon)$-approximation algorithm for the original problem. In the remainder of this section, we restrict our attention to the problem for which the delivery times $q_1, ..., q_n$ can take only $\chi \leq 1 + \frac{2}{\varepsilon}$ distinct values, which we denote by $\delta_1 > ... > \delta_\chi$.

The delivery time of a job can be interpreted as an additional *delivery operation* that must be processed on a *non-bottleneck machine* after the last operation of the job. A non-bottleneck machine is a machine that can process simultaneously any number of operations. Moreover, every feasible schedule for the jobs $\mathcal{J}$ can be transformed into another feasible schedule, in which all delivery operations finish at the same time, without increasing the schedule length: simply shift the delivery operations to the end of the schedule. Therefore, we only need to consider a set $\mathcal{D} = \{d_1, ..., d_\chi\}$ of χ different delivery operations, where d_i has processing time δ_i.

2.2 Relative Schedules

Assume that the jobs are indexed so that $P_1 \geq P_2 \geq ... \geq P_n$. Let $\mathcal{L} \subset \mathcal{J}$ be the set formed by the first k jobs, i.e., the k jobs with longest minimum processing time, where k is a constant to be defined later. We call $\mathcal{L}$ the set of *long* jobs. An operation from a long job is called a long operation, regardless of its processing time. Let $\mathcal{S} = \mathcal{J} \setminus \mathcal{L}$ be the set of *short* jobs.

Consider any feasible schedule for the jobs in $\mathcal{J}$. This schedule assigns a machine to every operation and it also defines a relative ordering for the starting and finishing times of the operations. A *relative schedule* R for $\mathcal{L}$ is an assignment of machines to long operations and a relative ordering for the starting and finishing times of the long operations and the delivery operations, such that there is a feasible schedule for $\mathcal{J}$ that respects R. This means that for every relative schedule R there is a feasible schedule for $\mathcal{J}$ that assigns the same machines as R to the long operations and that schedules the long and the delivery operations in the same relative order as R. Since there is a constant number of long jobs, then there is also a constant number of different relative schedules.

Lemma 2. *The number of relative schedules is at most* $m^{\mu k}(2(\mu k + \chi))!$.

If we build all relative schedules, one of them must be equal to the relative schedule defined by some optimum solution. Since it is possible to build all relative schedules in constant time, we might assume without loss of generality that we know how to find a relative schedule R such that some optimum schedule for $\mathcal{J}$ respects R.

Fix a relative schedule R as described above. The ordering of the starting and finishing times of the operations divide the time line into intervals that we call *snapshots*. We can view a relative schedule as a sequence of snapshots $M(1), M(2), \ldots, M(g)$, where $M(1)$ is the unbounded snapshot whose right boundary is the starting time of the first operation according to R, and $M(g)$ is the snapshot bounded on the right by the finishing time of the delivery operations. The number of snapshots g is at most $2\mu k + \chi + 1$ because the starting and finishing times of every operation might bound a snapshot.

2.3 Scheduling the Small Jobs

Given a relative schedule R as described above, to obtain a solution for the flexible job shop problem we need to schedule the small operations within the snapshots defined by R. We do this in two steps. First we use a linear program $LP(R)$ to assign small operations to snapshots and machines, and second, we find a feasible schedule for the small operations within every snapshot.

To formulate the linear program we need first to define some variables. For each snapshot $M(\ell)$ we use a variable t_ℓ to denote its length. For each $J_j \in \mathcal{S}$ we define a set of decision variables $x_{j,(i_1,\ldots,i_\mu),(s_1,\ldots,s_\mu)}$ with the following meaning: $x_{j,(i_1,\ldots,i_\mu),(s_1,\ldots,s_\mu)} = f$ iff for all $q = 1, \ldots, \mu$, an f fraction of the q-th operation of job J_j is completely scheduled in the i_q-th snapshot and on machine s_q.

Let α_j be the snapshot where the delivery operation of job J_j starts. For every variable $x_{j,(i_1,\ldots,i_\mu),(s_1,\ldots,s_\mu)}$ we need $1 \le i_1 \le i_2 \le \cdots \le i_\mu < \alpha_j$ to ensure that the operations of J_j are scheduled in the proper order. Let $A_j = \{(i,s) \mid i = (i_1,\ldots,i_\mu)\ 1 \le i_1 \le \ldots \le i_\mu < \alpha_j,\ s = (s_1,\ldots,s_\mu)\ s_q \in M_{qj}$ and no long operation is scheduled by R at snapshot i_q on machine s_q, for all $q = 1, \ldots, \mu\}$.

The load $L_{\ell,h}$ on machine h in snapshot $M(\ell)$ is the total processing time of the operations from small jobs assigned to h during $M(\ell)$, i.e.,

$$L_{\ell,h} = \sum_{J_j \in \mathcal{S}} \sum_{(i,s) \in A_j} \sum_{\substack{q=1 \\ i_q = \ell, s_q = h}}^{\mu} x_{j\,is}\, p_{qj}^{s_q}, \tag{3}$$

where i_q and s_q are the q-th components of tuples i and s respectively.

For every long operation O_{ij} let α_{ij} and β_{ij} be the indices of the first and last snapshots where the operation is scheduled. Let p_{ij} be the processing time of long operation O_{ij} according to the machine assignment defined by the relative schedule R. We are ready to describe the linear program $LP(R)$ that assigns small operations to snapshots.

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

s.t.

(1)	$\sum_{\ell=\alpha_{ij}}^{\beta_{ij}} t_\ell = p_{ij}$,	for all $J_j \in \mathcal{L}$, $i = 1, \ldots, \mu$,
(2)	$\sum_{\ell=\alpha_j}^{g} t_\ell = \delta_j$,	for all delivery operations d_j ,
(3)	$\sum_{(i,s)\in A_j} x_{j\,is} = 1$,	for all $J_j \in \mathcal{S}$,
(4)	$L_{\ell,h} \le t_\ell$,	for all $\ell = 1, \ldots, g$, $h = 1, \ldots, m$.
(5)	$t_\ell \ge 0$,	for all $\ell = 1, \ldots, g$,

(6) $x_{jis} \geq 0\,,$ for all $J_j \in \mathcal{S}$, $(i,s) \in A_j$.

Lemma 3. *An optimum solution of $LP(R)$ has value no larger than the length of an optimum schedule S^* that respects the relative schedule R.*

One can solve $LP(R)$ optimally in polynomial time and get only a constant number of jobs with fractional assignments, since a basic feasible solution of $LP(R)$ has at most $k\mu + n - k + mg + \chi$ variables with positive value. By constraint (3) every small job has at least one positive variable associated with it, and so there are at most $mg + k\mu + \chi$ jobs with fractional assignments. We show later how to get rid of any constant number of fractional assignments by only slightly increasing the length of the solution.

The drawback of this approach is that solving the linear program might take a very long time. Since we want to get an approximate solution to the flexible job shop problem it is not necessary to find an optimum solution for $LP(R)$, an approximate solution would suffice.

2.4 Approximate Solution of the Linear Program

A *convex block-angular resource sharing problem* has the form:

$$\min\left\{\lambda \;\middle|\; \sum_{k=1}^{K} f_i^k(x^k) \leq \lambda, \text{for all } i = 1,\ldots,N, \text{and } x^k \in B^k, k = 1,\ldots,K\right\}$$

where $f_i^k : B^k \rightarrow \Re^+$ are N non-negative continuous convex functions, and B^k are disjoint convex compact nonempty sets called *blocks*, $1 \leq k \leq K$. The Potential Price Directive Decomposition Method of Grigoriadis and Khachiyan [5] can find a $(1+\rho)$-approximate solution to this problem for any $\rho > 0$. This algorithm needs $O(N(\rho^{-2}\ln\rho^{-1} + \ln N)(N\ln\ln(N/\rho) + KF))$ time, where F is the time needed to find a ρ-approximate solution to the following problem on any block B^k, for some vector $(p_1,\ldots,p_N) \in \Re^N$: $\min\left\{\sum_{i=1}^{N} p_i f_i^k(x^k) \,\middle|\, x^k \in B^k\right\}$.

We can write $LP(R)$ as a convex block-angular resource sharing problem as follows. First we guess the value s of an optimum solution for $LP(R)$, and add the constraint: $\sum_{\ell=1}^{g} t_\ell \leq s$ to the linear program. Note that $s \leq m+1$. Then we replace constraint (4) by constraint (4′), where λ is a non-negative value:

(4') $L_{\ell,h} - t_\ell + m + 1 \leq \lambda,$ for all $\ell = 1,\ldots,g,\ h = 1,\ldots,m.$

This new linear program, that we denote as $LP(R,s,\lambda)$, has the above block-angular structure. The blocks $B_j = \{x_{jis} \mid$ constraints (3) and (6) hold$\}$, are $(mg)^\mu$-dimensional *simplicies*. The block $B_{|\mathcal{S}|+1} = \{t_\ell \mid \sum_{\ell=1}^{g} t_\ell \leq s$ and constraints (1),(2), and (5) hold$\}$ has also constant dimension. Let $f_{\ell,h} = L_{\ell,h} - t_\ell + m + 1$. Since $t_\ell \leq s \leq m+1$, these functions are non-negative. Each block B_i has constant dimension, and so the above block optimization problem can be solved in constant time. Therefore the algorithm of [5] finds a $(1+\rho)$-approximate solution for $LP(R,s,\lambda)$ in $O(n)$ time for any value $\rho > 0$. This gives a feasible solution of $LP(R,s,(m+1+\rho'))$ for $\rho = \rho'/(m+1)$.

Let $L^*_{\max}$ be the length of an optimum schedule and assume that R is a relative schedule for $\mathcal{L}$ in an optimum schedule. We can use binary search on the interval $[1, 1+m]$ to find a value $s \leq (1+\frac{\epsilon}{8})L^*_{\max}$ such that $LP(R, s, (m+1+\rho'))$ has a solution for $\rho' = \frac{\epsilon}{8g}$. This search can be performed in $O(\log(\frac{1}{\varepsilon}\log m))$ iterations by performing the binary search only on the following values,

$$(1+\frac{\varepsilon}{8}), (1+\frac{\varepsilon}{8})^2, ..., (1+\frac{\varepsilon}{8})^{b-1}, m+1 \tag{4}$$

where b is the smallest integer such that $(1+\varepsilon/8)^b \geq m+1$. Thus, $b \leq \ln(m+1)/\ln(1+\varepsilon/8)+1 = O(\frac{1}{\varepsilon}\log m)$, since $\ln(1+\varepsilon/8) \geq \frac{\varepsilon/8}{1+\varepsilon/8}$. To see that this search yields the desired value for s, note that there exists a nonnegative integer $i \leq b$ such that $L^*_{\max} \in [(1+\frac{\varepsilon}{8})^i, (1+\frac{\varepsilon}{8})^{i+1}]$ and therefore with the above search we find a value $s \leq (1+\frac{\epsilon}{8})L^*_{\max}$ for which $LP(R, s, m+1+\rho')$ has a feasible solution. Linear program $LP(R, s, m+1+\rho')$ assumes that the length of each snapshot is increased by ρ', and therefore the total length of the solution is $(1+\frac{\varepsilon}{8})L^*_{\max} + g\rho' \leq (1+\frac{\varepsilon}{4})L^*_{\max}$.

Lemma 4. *A solution for $LP(R, s, m+1+\rho')$, with $s \leq (1+\frac{\epsilon}{8})L^*_{max}$ and $\rho' = \frac{\epsilon}{8g}$, of value at most $(1+\frac{\epsilon}{4})L^*_{\max}$ can be found in linear time.*

By using a similar technique as in [8] we can modify any feasible solution for $LP(R, s, m+1+\rho')$ to get a new feasible solution in which all but a constant number of variables x_{jis} have integer values. Moreover we can do this rounding step in linear time.

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

2.5 Generating a Feasible Schedule

To get a feasible schedule from the solution of the linear program we need to remove all jobs $\mathcal{F}$ that received fractional assignment. These jobs are scheduled sequentially at the beginning of the schedule.

For every operation of the small jobs, consider its processing time according to the machine selected for it by the solution of the linear program. Let $\mathcal{V}$ be the set formed by the small jobs containing at least one operation with processing time larger than $\tau = \frac{\varepsilon}{8\mu^3 mg}$. Note that $|\mathcal{V}| \leq \frac{m(m+1)}{\tau} = \frac{8\mu^3 m^2(m+1)}{\varepsilon} g$. We remove from the snapshots all jobs in $\mathcal{V}$ and place them sequentially at the beginning of the schedule.

Let $O(\ell)$ be the set of operations from small jobs that remain in snapshot $M(\ell)$. Let $p_{max}(\ell)$ be the maximum processing time among the operations in $O(\ell)$. Every snapshot $M(\ell)$ defines an instance of the job shop problem, since the solution of the linear program assigns a unique machine to every operation. Hence we can use Sevastianov's algorithm [11] to find in $O(n^2\mu^2m^2)$ time a feasible schedule for the operations $O(\ell)$; this schedule has length at most $\bar{t}_\ell =$

$t_\ell + \rho' + \mu^3 m p_{max}(\ell)$. We must increase the length of every snapshot $M(\ell)$ to $\bar{t}_\ell$ to accommodate the schedule produced by Sevastianov's algorithm. Summing up all these enlargements, we get:

Lemma 6.

$$\sum_{\ell=1}^{g} \mu^3 m p_{max}(\ell) \leq \mu^3 m g \tau = \frac{\varepsilon}{8} \leq \frac{\varepsilon}{8} L^*_{\max}. \tag{5}$$

The total length of the snapshots $M(\alpha_{ij}), \ldots, M(\beta_{ij})$ that contain a long operation O_{ij} 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. Let $P(J') = \sum_{J_j \in J'} P_j$ be the total processing time of all jobs in some set $J' \subset \mathcal{J}$ when the operations of those jobs are assigned to the machines with the lowest processing times.

Lemma 7. *A feasible schedule for the jobs $\mathcal{J}$ of length at most $(1 + \frac{3}{8}\epsilon)L^*_{\max} + P(\mathcal{F} \cup \mathcal{V})$ can be found in $O(n^2)$ time.*

We can choose the number k of long jobs so that $P(\mathcal{F} \cup \mathcal{V}) \leq \frac{\epsilon}{8} L^*_{\max}$.

Lemma 8. *[7] Let $\{d_1, d_2, \ldots, d_n\}$ be positive values and $\sum_{j=1}^{n} d_j \leq m$. Let q be a nonnegative integer, $\alpha > 0$, and $n \geq (q+1)^{\lceil \frac{1}{\alpha} \rceil}$. There exists an integer k such that $d_{k+1} + \ldots + d_{k+qk} \leq \alpha m$ and $k \leq (q+1)^{\lceil \frac{1}{\alpha} \rceil}$.*

Let us choose $\alpha = \frac{\epsilon}{8m}$ and $q = \left(\frac{8\mu^3 m(m+1)}{\varepsilon} + 1 \right) m(2\mu + \chi + 1)$. By Lemma 5, $|\mathcal{F} \cup \mathcal{V}| \leq mg + \frac{8\mu^3 m^2 (m+1)}{\varepsilon} g \leq qk$. By Lemma 8 it is possible to choose a value $k \leq (q+1)^{\lceil \frac{1}{\alpha} \rceil}$ so that the total processing time of the jobs in $\mathcal{F} \cup \mathcal{V}$ is at most $\frac{\varepsilon}{8} \leq \frac{\epsilon}{8} L^*_{\max}$. This value of k can clearly be computed in constant time. We select the set $\mathcal{L}$ of large jobs as the set consisting of the k jobs with largest processing times $P_j = \sum_{i=1}^{\mu} [\min_{s \in M_{kj}} p^s_{ij}]$.

Lemma 9.

$$P(\mathcal{F} \cup \mathcal{V}) \leq \frac{\epsilon}{8} L^*_{\max}. \tag{6}$$

Theorem 1. *For any fixed m and μ, there is an algorithm for the flexible job shop scheduling problem that computes for any value $\varepsilon > 0$, a feasible schedule of length at most $(1+\epsilon)L^*_{\max}$ in $O(n)$ time.*

Proof. By Lemmas 7 and 9, the above algorithm finds in $O(n^2)$ a schedule of length at most $(1 + \frac{1}{2}\epsilon)L^*_{\max}$. This algorithm can handle $1 + \frac{2}{\epsilon}$ distinct delivery times. By the discussion at the beginning of Section 3.1 it is easy to modify the algorithm so that it handles arbitrary delivery times and it yields a schedule of length at most $(1+\varepsilon)L^*_{\max}$. For every fixed m, μ, 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. The latter can be sped up to get linear time by "glueing" pairs of small jobs together as described in [8]. □

3 Preemptive Flexible Job Shop Problem with Migration

Let t a nonnegative variable, with $t \geq \delta_1$, that denotes the length of a schedule (with delivery times). Consider χ time intervals defined as follows, $[0, t-\delta_1], [t-\delta_1, t-\delta_2], ..., [t-\delta_{\chi-1}, t-\delta_\chi]$ where $\delta_1 > \ldots \delta_\chi$ are the delivery times. First we ignore the release times and select the machines and the time intervals in which the operations of every job are going to be processed. To do this we define a linear program LP (similar to that of Sect. 2.3) that minimize the value of t. The optimum solution of the LP has value no larger than $L^*_{\max}$. Again, by using the Logarithmic Potential Price Directive Decomposition Method [5] and the rounding technique [8], we can compute in linear time a $(1+\frac{\varepsilon}{4})$-approximate solution $\tilde{S}$ of the LP such that the size of the set $\mathcal{F}$ of jobs that receive fractional assignments is bounded by $m\chi$.

Let $\mathcal{P}$ denote the set of jobs from $\mathcal{J}\setminus\mathcal{F}$ for which at least one operation has processing time greater than $\frac{\varepsilon \tilde{t}}{4\chi\mu^3 m(1+\frac{\varepsilon}{8})}$, where $\tilde{t}$ is the LP approximate solution value. Let $\mathcal{L} = \mathcal{F} \cup \mathcal{P}$ and $\mathcal{S} = \mathcal{J}\setminus\mathcal{L}$. According to $\tilde{S}$, find a feasible schedule $\sigma_{\mathcal{S}}$ for the jobs from $\mathcal{S}$ applying Sevastianov's algorithm. We use the algorithm of Sevastianov to find a schedule for the operations assigned to each time interval. The maximum delivery completion time (when release times are ignored) of $\sigma_{\mathcal{S}}$ is at most $(1+\varepsilon)L^*_{\max}$. By adding release times the length of $\sigma_{\mathcal{S}}$ is at most $(2+\varepsilon)L^*_{\max}$, since the maximum release time cannot be more than $L^*_{\max}$. Again, the algorithm of Sevastianov can be sped up to take $O(n)$ time, and computing schedule $\sigma_{\mathcal{S}}$ takes linear time.

Now, we ignore the delivery times for jobs from $\mathcal{L}$ (they are considered later). We note that the cardinality of set $\mathcal{L}$ is bounded by $O(\frac{\mu^3 m^2}{\varepsilon^2})$. As for delivery times, the release times can be interpreted as additional operations of jobs that have to be processed on a non-bottleneck machine. Because of this interpretation, we can add to the set $\mathcal{O}_{\mathcal{L}}$ of operations from $\mathcal{L}$ a set $\mathcal{R}$ of release operations O_{0j} with processing times r_j. Each job $J_j \in \mathcal{L}$ has to perform its release operation O_{0j} on a non-bottleneck machine at the beginning. A *relative order* R is an ordered sequence of the starting and finishing times of all operations from $\mathcal{O}_{\mathcal{L}} \cup \mathcal{R}$, such that there is a feasible schedule for $\mathcal{L}$ that respects R. The ordering of the starting and finishing times of the operations divide the time line into small intervals. We observe that the number of small intervals g is bounded by $O(\frac{\mu^4 m^2}{\varepsilon^2})$. Note that a relative order (in this part) is defined without assigning operations of long jobs to machines. For every relative order R we define a linear program to assign (fractions of) operations to machines and small intervals that respects R. We build all relative orders R and solve the corresponding linear programs in constant time. At the end we select a relative schedule R^* with the smallest solution value. We can show that the value of this solution is no larger than the length of an optimum preemptive schedule for $\mathcal{L}$. This solution is in general not a feasible schedule for $\mathcal{L}$ since the order of fractions of operations within an interval could be incorrect. However the set of operations assigned to each interval gives an instance of the preemptive open shop problem, which can be solved exactly in constant time [9]. This can be done without increasing the

length of each interval and with in total only a constant number of preemptions (at most $O(m^2)$ preemptions for each interval). By adding the delivery times the length of the computed schedule $\sigma_{\mathcal{L}}$ is at most $2L^*_{\max}$. The output schedule is obtained by appending $\sigma_{\mathcal{S}}$ after $\sigma_{\mathcal{L}}$.

Theorem 2. *For any fixed m, μ, and $\varepsilon > 0$, there is a $(2+\varepsilon)$-linear-time approximation algorithm for the preemptive flexible job shop scheduling problem with migration.*

References

1. P. Brandimarte, Routing and scheduling in a flexible job shop by tabu search, *Annals of Operations Research*, 22, 158-183, 1993.
2. 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.
3. T. Gonzales and S. Sahni, Flowshop and jobshop schedules: complexity and approximation, *Operations Research* 26 (1978), 36-52.
4. R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnoy Kan, Optimization and approximation in deterministic sequencing and scheduling, *Ann. Discrete Math.* 5 (1979), 287-326.
5. M.D. Grigoriadis and L.G. Khachiyan, Coordination complexity of parallel price-directive decomposition, *Mathematics of Operations Research* 21 (1996), 321-340.
6. L.A. Hall and D.B. Shmoys, Approximation algorithms for constrained scheduling problems, *Proceedings of the IEEE 30th Annual Symposium on Foundations of Computer Science* (FOCS 89), 134-139.
7. 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*, (SODA 99), 490-498.
8. K. Jansen, R. Solis-Oba and M.I. Sviridenko, A linear time approximation scheme for the job shop scheduling problem, Proceedings of the Second International Workshop on Approximation Algorithms (APPROX 99), 177-188.
9. E. L. Lawler, J. Labetoulle, On Preemptive Scheduling of Unrelated Parallel Processors by Linear Programming, *Journal of the ACM*, vol. 25, no. 4, pp. 612–619, October 1978.
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.V. Sevastianov, Bounding algorithms for the routing problem with arbitrary paths and alternative servers, *Cybernetics* 22 (1986), 773-780.
12. D.B. Shmoys, C. Stein, and J. Wein, Improved approximation algorithms for shop scheduling problems, *SIAM Journal on Computing* 23 (1994), 617-632.
13. R.J.M. Vaessens, Generalized job shop scheduling: complexity and local search, Ph.D. thesis (1995), Eindhoven University of Technology.
14. D. Williamson, L. Hall, J. Hoogeveen, C.. Hurkens, J.. Lenstra, S. Sevastianov, and D. Shmoys, Short shop schedules, *Operations Research* 45 (1997), 288-294.