Research Report 312

Better approximation guarantees for job-shop scheduling

Leslie Goldberg, Mike Paterson, Aravind Srinivasan, Elizabeth Sweedyk

RR312

Job-shop scheduling is a classical NP-hard problem. Shmoys, Stein & Wein presented the first polynomial-time approximation algorithm for this problem that has a good (polylogarithmic) approximation guarantee. We improve the approximation guarantee of their work and present further improvements for some important NP-hard special cases of this problem (eg in the *preemptive* case where machines can suspend work on operations and later resume.) Some of these improvements represent the first constant-factor approximation algorithms. We also present NC algorithms with improved approximation guarantees for some NP-hard special cases.

Department of Computer Science
University of Warwick
Coventry CV4 7AL
United Kingdom

August 1996

Better approximation guarantees for job-shop scheduling

Leslie Ann Goldberg [*] Mike Paterson [†] Aravind Srinivasan [‡]
Elizabeth Sweedyk [§]

Abstract

Job-shop scheduling is a classical NP-hard problem. Shmoys, Stein & Wein presented the first polynomial-time approximation algorithm for this problem that has a good (polylogarithmic) approximation guarantee. We improve the approximation guarantee of their work, and present further improvements for some important NP-hard special cases of this problem (e.g., in the *preemptive* case where machines can suspend work on operations and later resume). Some of these improvements represent the first constant-factor approximation algorithms. We also present NC algorithms with improved approximation guarantees for some NP-hard special cases.

1 Introduction

Job-shop scheduling is a classical NP-hard minimisation problem [7]. We improve the approximation guarantees for this problem and for some of its important special cases, both in the sequential and parallel algorithmic domains; the improvements are over the current-best algorithms of Leighton, Maggs & Rao [8] and Shmoys, Stein & Wein [14]. In job-shop scheduling, we have n jobs and m machines. A job consists of a sequence of operations, each of which is to be processed on a specific machine for a specified integral amount of time; a job can have more than one operation on a given machine. The operations of a job must be processed in the given sequence, and a machine can process at most one operation at any given time. The problem is to schedule the jobs so that the *makespan*, the time when all jobs have been completed, is minimised. An important special case of this problem is *preemptive* scheduling, wherein machines can suspend work on operations, switch to other operations, and later resume the suspended operations (if this is not allowed, we have the *non-preemptive* scenario, which we take as the default); in such a case, all operation lengths may be taken to be one. Even this special case with $n = m = 3$ is NP-hard! We present further improved approximation factors for preemptive scheduling and related special cases of job-shop scheduling; in particular, we present the first sequential and parallel *constant-factor* approximation algorithms for the case where m is fixed and the maximum length of any operation is bounded.

Formally, a job-shop scheduling instance consists of jobs $J_1, J_2, \ldots, J_n$, machines $M_1, M_2, \ldots, M_m$, and for each job J_j, a sequence of μ_j *operations* $(M_{j,1}, t_{j,1})$, $(M_{j,2}, t_{j,2}), \ldots, (M_{j,\mu_j}, t_{j,\mu_j})$. Each operation is a (machine, processing time) pair: each $M_{j,k}$ represents some machine M_i, and the pair $(M_{j,i}, t_{j,i})$ signifies that the corresponding operation of job J_j must be processed on machine $M_{j,i}$ for an *uninterrupted* integral amount of time of $t_{j,i}$. The problem that we focus on throughout is to come up with a schedule that has a small makespan, for general job-shop scheduling and for some of its important special cases.

*Dept. of Computer Science, University of Warwick, Coventry CV4 7AL, UK. leslie@dcs.warwick.ac.uk.

†Dept. of Computer Science, University of Warwick, Coventry CV4 7AL, UK. msp@dcs.warwick.ac.uk.

‡Dept. of Information Systems and Computer Science, National University of Singapore, Singapore 119260. aravind@iscs.nus.sg.

§Department of Computer and Information Sciences, University of Pennsylvania. zzz@saul.cis.upenn.edu.

1.1 Earlier work

As described earlier, even very restricted special cases of job-shop scheduling are NP-hard. Furthermore, the problem seems quite intractable in practice, even for relatively small instances. Call a job-shop instance *acyclic* if no job has more than one operation that needs to run on any given machine. A single instance of acyclic job-shop scheduling consisting of 10 jobs, 10 machines and 100 operations resisted attempts at exact solution for 22 years, until its resolution by Carlier & Pinson [4]. More such exact solutions for certain instances (with no more than 20 jobs or machines) were computationally provided by Applegate & Cook, who also left open the exact solution of certain acyclic problems, e.g., some with 15 jobs, 15 machines, and 225 operations [2].

Thus, efficient exact solution of all instances with, say, 30 jobs, 30 machines, and 900 operations, seems quite out of our reach at this point; an obvious next question is to look at efficient approximability. Define a ρ-approximation algorithm as a polynomial-time algorithm that always outputs a feasible schedule with a makespan of at most ρ times optimal; ρ is called the approximation guarantee. A negative result is known: if there is a ρ-approximation algorithm for job-shop scheduling with $\rho < 5/4$, then P = NP [15].

There are two simple lower bounds on the makespan of any feasible schedule: P_{max}, the maximum total processing time needed for any job, and Π_{max}, the maximum total amount of time for which any machine has to process operations. For the NP-hard special case of *acyclic* job-shop scheduling wherein all operations have unit length, a breakthrough was achieved in [8], showing that a schedule of makespan $O(P_{\text{max}} + \Pi_{\text{max}})$ always exists! Such a schedule can also be computed in polynomial time [9]. However, if we drop any one of the two above assumptions (unit operation lengths and acyclicity), it is not known whether such a good bound holds.

What about upper bounds for general job-shop scheduling? It is not hard to see that a simple greedy algorithm, which always schedules available operations on machines, delivers a schedule of makespan at most $P_{\text{max}} \cdot \Pi_{\text{max}}$; one would however like to aim for much better. Let $\mu = \max_j \mu_j$ denote the maximum number of operations per job, and let p_{max} be the maximum processing time of any operation. By invoking ideas from [8, 12, 13] and by introducing some new techniques, good approximation algorithms were developed in [14]. Their deterministic approximation bounds were slightly improved in [11] to yield the following proposition.[1]

Proposition 1.1 ([14, 11]) *There is a deterministic polynomial-time algorithm that delivers a schedule of makespan* $O((P_{\text{max}}+\Pi_{\text{max}})\cdot\frac{\log(m\mu)}{\log\log(m\mu)}\cdot\log(\min\{m\mu, p_{\text{max}}\}))$ *for general job-shop scheduling. If we replace* m *by* n *in this bound, then such a schedule can also be computed in RNC.*

This is a ρ-approximation algorithm with $\rho = O(\log(m\mu)\log(\min\{m\mu, p_{\text{max}}\})/\log\log(m\mu))$. See [14, 6] for further results on approximating some special cases of shop scheduling that are not discussed here.

1.2 Our results

Our first result improves Proposition 1.1 by a doubly logarithmic factor and provides further improvements for important special cases.

Theorem 1 *There are the following deterministic algorithms for general job-shop scheduling, delivering schedules of makespan* $O((P_{\text{max}} + \Pi_{\text{max}})\cdot\rho)$*:*

(a)
$$\rho = \frac{\log(m\mu)}{\log\log(m\mu)} \cdot \left\lceil \frac{\log(\min\{m\mu, p_{\text{max}}\})}{\log\log(m\mu)} \right\rceil \quad \textit{(polynomial-time algorithm).}$$

If we replace m *by* n *in this bound, then such a schedule can also be computed in NC.*

[1]To avoid problems with small positive numbers, henceforth let $\log x$ and $\log\log x$ denote $\log_2(x + 2)$ and $\log_2\log_2(x + 4)$ respectively, for $x \geq 0$.

(b)
$$\rho = \frac{\log m}{\log\log m} \cdot \log(\min\{m\mu, p_{\max}\}) \text{ \textit{(polynomial-time algorithm)}.}$$

(c)
$$\rho = \frac{\log m}{\log\log m} \cdot \log(\min\{n\mu, p_{\max}\}) \text{ \textit{(NC algorithm)}.}$$

Thus, part (a) improves on the previous approximation bound by a doubly logarithmic factor. The impact of parts (b) and (c) is best seen for preemptive scheduling, wherein $p_{\max} = 1$, and for the related situations where $p_{\max}$ is "small". Our motivations for focusing on these cases are twofold. First, preemptability is known to be a powerful primitive in various scheduling models, see, e.g., [3]. Second, the above result of [8] shows that preemptability is powerful for acyclic job-shops. It is a major open question whether there is a schedule of makespan $O(P_{\max} + \Pi_{\max})$ for general job-shop scheduling and if so, in what cases it can be found efficiently. In view of the above result of [8], one way to attack this question is to study (algorithmically) the problem parametrised by $p_{\max}$, focusing on the case of "small" $p_{\max}$. Recall that even the case of $n = m = 3$ with $p_{\max} = 1$ is NP-hard. Thus, parts (b) and (c) above present the first constant-factor approximation algorithms in such situations. In general, these two parts show that, as long as the number of machines is small or fixed, we get very good approximations. Note that for the case in which $p_{\max}$ is small, part (c) is both a derandomisation and an improvement of the previous-best parallel algorithm for job-shop scheduling (see Proposition 1.1).

We further explore the issue of when good approximations are possible, once again with a view to generalise the above key result of [8]; this is done by the somewhat-technical Theorem 2. We take *high probability* to mean a probability of $1 - \epsilon$, where ϵ is a fixed positive constant as small as we please. This can be amplified by repetition to give any ϵ which tends to zero polynomially in the size of the problem instance. Theorem 2 shows that if (a) no job requires too much of any given machine for processing, or if (b) repeated uses of the same machine by a given job are well-separated in time, then good approximations are possible. Say that a job-shop instance is *w-separated* if every distinct pair $((M_{j,\ell}, t_{j,\ell}), (M_{j,r}, t_{j,r}))$ of operations of the same job with the same machine (i.e., $M_{j,\ell} = M_{j,r}$) has $|\ell - r| \geq w$.

Theorem 2 *There is a randomised polynomial-time algorithm for job-shop scheduling that, with high probability, delivers a schedule of makespan $O((P_{\max} + \Pi_{\max}) \cdot \rho)$, where*

(a) if every job needs at most u time units on each machine then

$$\rho = \frac{\log u}{\log\log u} \cdot \left\lceil \frac{\log(\min\{m\mu, p_{\max}\})}{\log\log u} \right\rceil;$$

(b) if the job-shop instance is w-separated and $p_{\max} = 1$ then

$$\rho = 1 \qquad \text{\textit{if} } w \geq \log(P_{\max} + \Pi_{\max})/2;$$

$$\rho = \frac{\log(P_{\max}+\Pi_{\max})}{w\log(\log(P_{\max}+\Pi_{\max})/w)}, \qquad \text{\textit{otherwise.}}$$

Most of our results rely on probabilistic ideas: in particular, we exploit a "random delays" technique due to [8]. Theorem 1(a) is obtained by a better combinatorial solution to a packing problem considered in [14], and parts (b) and (c) of Theorem 1 follow from a careful look at the approximation obtained by introducing random delays. We de-randomise the sequential formulations using a technique of [1] and then parallelise. A simple but crucial ingredient of Theorem 1 is a new way to structure the operations of jobs in an initial (infeasible) schedule; we call this *well-structuredness*, and present it in Section 2. Theorem 2 comes about by introducing random delays and by using the Lovász Local Lemma [5], which is also done in [8]; our improvements arise from a study of the correlations involved and by using Theorem 1(a).

We have presented an improved approximation algorithm for general job-shop scheduling (Theorem 1(a)), and have shown further improvements for certain NP-hard special cases. In particular, parts (b) and (c) of Theorem 1 show the power of preemptability (or of small operation lengths) when the number of machines is fixed, leading to constant-factor approximation algorithms. Theorem 2 generalises the result in [8] showing the existence of an $O(P_{\max} + \Pi_{\max})$ makespan schedule. Its part (a) quantitatively shows, for instance, the advantages of having multiple copies of each machine; in such a case, we can try to spread out the operations of a job somewhat equitably to the various copies. Part (b) of Theorem 2 shows that if we have some (limited) flexibility in rearranging the operation sequence of a job, then it pays to spread out multiple usages of the same machine. The rest of this paper is organised as follows. Section 2 sets up some preliminary notions, Section 3 presents the proof of Theorem 1, and Theorem 2 is proved in Section 4. The proofs of many of our results are presented in the appendix, for lack of space.

2 Preliminaries

For any non-negative integer k, we let $[k]$ denote the set $\{1, 2, \ldots, k\}$. The base of the natural logarithm is denoted by e as usual and, for convenience, we may use $exp(x)$ to denote e^x.

As in [14], we assume throughout that all operation lengths are powers of two. This can be achieved by multiplying each operation length by at most two. This assumption on operation lengths will only affect our approximation factor and running time by a constant factor. Thus, $P_{\max}$, $\Pi_{\max}$ and $p_{\max}$ should be replaced by some $P'_{\max} \leq 2P_{\max}$, $\Pi'_{\max} \leq 2\Pi_{\max}$, and $p'_{\max} \leq 2p_{\max}$ respectively, in the sequel. We have avoided using such new notation, to retain simplicity.

Reductions. It is shown in [14] that in deterministic polynomial time, we can reduce the general shop-scheduling problem to the case where (i) $p_{\max} \leq n\mu$, and where (ii) $n \leq \text{poly}(m, \mu)$, while incurring an *additive* $O(P_{\max} + \Pi_{\max})$ term in the makespan of the schedule produced. The reduction (i) also works in NC. Thus, for our sequential algorithms we assume that $n \leq \text{poly}(m, \mu)$ and that $p_{\max} \leq \text{poly}(m, \mu)$; while for our NC algorithms we assume only that $p_{\max} \leq n\mu$.

Bounds. We use the following bounds on expectation and tails of distributions.

Fact 2.1 [Hoeffding] *Let $X_1, X_2, \ldots, X_\ell \in [0, 1]$ be* independent *random variables with $X \doteq \sum_i X_i$. Then for any $\delta > 0$, $\mathrm{E}[(1+\delta)^X] \leq e^{\delta \mathrm{E}[X]}$.*

We define $G(\mu, \delta)$ to be $G(\mu, \delta) \doteq (e^\delta/(1+\delta)^{1+\delta})^\mu$. Using Markov's inequality and Fact 2.1, we obtain Chernoff and Hoeffding's bounds on the tails of the binomial distribution (see [10]).

Fact 2.2 [Chernoff, Hoeffding] *Let $X_1, X_2, \ldots, X_\ell \in [0, 1]$ be* independent *random variables with $X \doteq \sum_i X_i$ and $\mathrm{E}[X] = \mu$. Then for any $\delta > 0$, $\Pr(X \geq \mu(1+\delta)) \leq G(\mu, \delta)$.*

Random delays. Our algorithms use *random initial delays* which were developed in [8] and used in [14]. A *B-delayed schedule* of a job-shop instance is constructed as follows. Each job J_j is assigned a delay d_j in $\{0, 1, \ldots, B-1\}$. In the resulting B-delayed schedule, the operations of J_j are scheduled consecutively, starting at time d_j. A *random B-delayed schedule* is a B-delayed schedule in which the delays have been chosen independently and uniformly at random from $\{0, 1, \ldots, B-1\}$. Our algorithms schedule a job-shop instance by choosing a random B-delayed schedule for some suitable B, and then expanding this schedule to resolve conflicts between operations that use the same machine at the same time.

For a B-delayed schedule $\mathcal{S}$, the *contention*, $C(M_i, t)$, is the number of operations scheduled on machine M_i in the time interval $[t, t+1)$. (Recall that operation lengths are integral.) For any job J_j, define the random variable $X_{i,j,t}$ to be 1 if some operation of J_j is scheduled on M_i in the time interval $[t, t+1)$ by $\mathcal{S}$, and 0 otherwise. Since no two operations of J_j contend for M_i simultaneously, $C(M_i, t) = \sum_j X_{i,j,t}$. If the delays are chosen uniformly at random and $B \geq \Pi_{\max}$, then $\mathrm{E}[X_{i,j,t}]$ is at most the total processing time of J_j on M_i divided by $\Pi_{\max}$. Thus, $\mathrm{E}[C(M_i,t)] = \sum_j \mathrm{E}[X_{i,j,t}] \leq \Pi_{\max}/\Pi_{\max} = 1$. We also note that the random variables $\{X_{i,j,t} \mid j \in [n]\}$ are mutually independent, for any given i and t. We record all this as follows.

Fact 2.3 *If $B \geq \Pi_{\max}$ and $\mathcal{S}$ is a random B-delayed schedule then for any machine M_i and any time t, $C(M_i,t) = \sum_j X_{i,j,t}$, where the 0-1 random variables $\{X_{i,j,t} \mid j \in [n]\}$ are mutually independent. Also, $\mathrm{E}[C(M_i,t)] \leq 1$.*

Well-structuredness. Recall that all operation lengths are assumed to be powers of two. We say that a delayed schedule $\mathcal{S}$ is *well-structured* if for each k, all operations with length 2^k begin in $\mathcal{S}$ at a time instant that is an integral multiple of 2^k. We shall use the following simple way of constructing such schedules from randomly delayed schedules. First create a new job-shop instance by replacing each operation $(M_{j,\ell}, t_{j,\ell})$ by the operation $(M_{j,\ell}, 2 \cdot t_{j,\ell})$. Suppose $\mathcal{S}$ is a random B-delayed schedule for this modified instance, for some B; we will call $\mathcal{S}$ a *padded random B-delayed schedule.* From $\mathcal{S}$, we can construct a well-structured delayed schedule, $\mathcal{S}'$, for the original job-shop instance: simply insert $(M_{j,l}, t_{j,l})$ with the correct boundary in the slot assigned to $(M_{j,l}, 2 \cdot t_{j,l})$ by $\mathcal{S}$. $\mathcal{S}'$ will be called a *well-structured random B-delayed schedule* for the original job-shop instance.

3 Proof of Theorem 1

In this section we prove Theorem 1. In Section 3.1 we give a randomised polynomial-time algorithm that proves part (b) of the theorem. In Section 3.2 we improve the algorithm to prove part (a). Finally we discuss the derandomisation and parallelisation of these algorithms in Section 3.3. Throughout, we shall assume upper bounds on n and $p_{\max}$ (i.e., $p_{\max} \leq n\mu$, $n \leq \mathrm{poly}(m,\mu)$ and $p_{\max} \leq \mathrm{poly}(m,\mu)$) as described earlier; this explains terms such as $\log(\min\{m\mu, p_{\max}\})$ in the bounds of Theorem 1. Given a delayed schedule $\mathcal{S}$, define $C(t) \doteq \max_i C(M_i, t)$.

Lemma 3.1 *There is a randomised polynomial-time algorithm that takes a job-shop instance and produces a* well-structured *delayed schedule which has a makespan $L \leq 2(P_{\max} + \Pi_{\max})$. With high probability, this schedule satisfies:*

(a) $\forall i \in [m]\ \forall t \in \{0, 1, \ldots, L-1\}$, $C(M_i,t) \leq \alpha$, and

(b) $\sum_{t=0}^{L-1} C(t) \leq \beta(P_{\max} + \Pi_{\max})$,

where $\alpha = c_1 \log(m\mu)/\log\log(m\mu)$ and $\beta = c_2 \log m/\log\log m$, for sufficiently large constants $c_1, c_2 > 0$.

3.1 Proof of Theorem 1(b)

Assume $\mathcal{S}$ is a delayed schedule satisfying the conditions of Lemma 3.1 with makespan $L = O(P_{\max} + \Pi_{\max})$. We begin by partitioning the schedule into *frames*, i.e., time intervals $\{[ip_{\max}, (i+1)p_{\max}), i = 0, 1, \ldots, \lceil L/p_{\max} \rceil - 1\}$. By the definition of $p_{\max}$ and the fact that $\mathcal{S}$ is well-structured, no operation straddles a frame. We construct a feasible schedule for the operations performed under schedule $\mathcal{S}$ for each frame. Concatenating these schedules yields a feasible schedule for the original problem. We give the frame-scheduling algorithm where, without loss of generality, we assume that its input is the first frame.

Let T be a rooted complete binary tree with $p_{\max}$ leaves labelled, from left to right, $0, 1, \ldots, p_{\max} - 1$. Let u be a node in T and let $l(u)$ and $r(u)$ be the labels, respectively, of the leftmost and rightmost leaves of the subtree rooted at u. We shall associate the operations scheduled during the frame with the nodes of T in a natural way. For $i = 1, \ldots, m$ we define $S_i(u)$ to be those operations O that are scheduled on M_i by $\mathcal{S}$ for precisely the time interval $[l(u), r(u) + 1)$; each O scheduled by $\mathcal{S}$ in the first frame is in exactly one $S_i(u)$. Let $p(u) = (r(u) - l(u) + 1) \cdot \max_i ||S_i(u)||$, $||S_i(u)||$ denoting the cardinality of $S_i(u)$; $p(u)$ is an upper bound on the time needed to perform the operations $\cup_i S_i(u)$ associated with u. Let the nodes of T be numbered as $u_1, u_2, \ldots$ in the *preorder* traversal of T. Define $f(u_1) = 0$ and for $j \geq 2$, let $f(u_j) = \sum_{k<j} p(u_k)$. The algorithm simply schedules the operations in $S_i(u)$ on machine M_i consecutively beginning at time $f(u)$ and concluding no later than $f(u) + p(u)$. Let $\mathcal{S}'$ be the resulting schedule. Part (b) of Theorem 1 follows from Lemma 3.1 and the following lemma.

Lemma 3.2 *$\mathcal{S}'$ is feasible and has makespan at most $\sum_{u \in T} p(u)$, which is at most $(1 + \log_2 p_{\max}) \cdot \sum_{j=0}^{p_{\max}-1} C(j)$, where $C(t)$ is the maximum contention at time t under schedule $\mathcal{S}$.*

Proof. By construction, no machine performs more than one operation at a time. Suppose O_1 and O_2 are distinct operations of job J scheduled in the first frame. Assume $O_1 \in S_i(u)$ and $O_2 \in S_j(v)$, where possibly $i = j$. Assume O_1 concludes before O_2 begins under $\mathcal{S}$; thus u and v are roots of disjoint subtrees of T and u precedes v in the preorder traversal of T. Thus O_1 concludes before O_2 begins in $\mathcal{S}'$ and the new schedule is feasible.

Clearly the makespan of $\mathcal{S}'$ is at most $\sum_{u \in T} p(u)$. Fix a node u at some height k in T. (We take leaves to have height 0.) Then $p(u) = 2^k \max_i ||S_i(u)||$. Since the maximum number of jobs scheduled at any time t on any machine under $\mathcal{S}$ is $C(t)$, we get that $\forall t \in [l(u), \ldots, r(u)]$, $\max_i ||S_i(u)|| \leq C(t)$. Thus,

$$p(u) \leq 2^k \max_i ||S_i(u)|| \leq \sum_{t \in [l(u), \ldots, r(u)]} C(t).$$

Since each leaf of T has $(1 + \log_2 p_{\max})$ ancestors, the makespan of $\mathcal{S}'$ is at most

$$\sum_{u \in T} p(u) \leq \sum_{u \in T} \sum_{t \in [l(u), \ldots, r(u)]} C(t) = (1 + \log_2 p_{\max}) \cdot \sum_{t=0}^{p_{\max}-1} C(t). \qquad \square$$

3.2 Proof of Theorem 1(a)

We give a slightly different frame-scheduling algorithm and show that the feasible schedule for each frame has makespan $O(p_{\max} \alpha \lceil \log(p_{\max}) / \log \alpha \rceil)$. (The parameter α is from Lemma 3.1, and is assumed to be a power of two without loss of generality.) Thus, under the assumption that $p_{\max} \leq poly(m, \mu)$, the final schedule satisfies the bounds of Theorem 1(a).

The difficulty with the algorithm given in the Section 3.1 is that the operations may be badly distributed to the nodes of T by $\mathcal{S}$ so that $\mathcal{S}'$ is inefficient. To clarify, consider the following situation. Suppose that u has left child v, $p(u)$ is determined by $S_i(u)$, and $p(v)$ is determined by $S_j(v)$. The troubling case is when $i \neq j$. If, for instance, $S_j(u) = \phi$ and $S_i(v) = \phi$, then M_i and M_j will have idle periods of $p(v)$ and $p(u)$, respectively. We can reduce the idle time by pushing some of the operations in $S_i(u)$ down to v.

We give a push-down algorithm that associates operations $S'_i(u)$ for machine i with node u. We begin by partitioning T into subtrees. Mark a node u if it is at height 0 mod $\log \alpha$ in T. Eliminating the edges between a marked node and its children partitions T into a collection of subtrees, each of height $\log \alpha$, except possibly the one rooted at the root of T, which may have height less than $\log \alpha$. The push-down algorithm will redistribute operations within each of these subtrees independently.

Let T' be one of the subtrees of the partition. Initially each $S'_i(u)$ is empty for all $u \in T'$. Let v be a node in T'. Assume v has height k in T' and that $||S_i(v)|| = 2^\ell$, padding with dummy operations if necessary. If $k \geq \ell$, the algorithm distributes one operation of $S_i(v)$ to each $S_i'(w)$, where w is a descendant of v at a distance ℓ below v. Otherwise it distributes $2^{\ell-k}$ to $S_i'(w)$, for each w that is a leaf in T' and a descendant of v. The algorithm repeats the procedure for each $i = 1, \ldots, m$ and for each v in T'.

Let $p(u)$ and $f(u)$ be defined as before but relative to $S_i'(u)$, $i = 1, \ldots, m$. Run the scheduling algorithm described above to produce a schedule $\mathcal{S}'$.

Lemma 3.3 *$\mathcal{S}'$ is a feasible schedule with makespan at most* $O(p_{\max}\alpha\lceil\log p_{\max}/\log\alpha\rceil)$.

Proof. The proof that $\mathcal{S}'$ is feasible follows exactly as before. The makespan of $\mathcal{S}'$ is no more than $\sum_{u\in T} p(u)$.

Consider a subtree T' of the partition. Assume the leaves of T' are at height j in T. Let w be a node in T' and let V be the subset of nodes of T' consisting of w and its ancestors in T'.

First suppose w is a leaf. Let v be a node in V and assume that v has height k in T' with $||S_i(v)|| = 2^\ell$. Then v contributes at most $2^{\ell-k}$ operations to $S_i'(w)$ and each has length 2^{j+k}. The time needed to perform these operations is $2^{\ell-k} \cdot 2^{j+k} = 2^j 2^\ell$. By Lemma 3.1, part (a), $\sum_{v\in V} ||S_i(v)|| \leq 2\alpha$. (The factor of 2 arises from the (possible) padding of $S_i(v)$ with dummy operations.) Thus $p(w) \leq 2^{j+1}\alpha$.

Now suppose w is at height $r > 0$ in T'. A $v \in V$ at height $r + k$ in T' contributes at most one operation to $S_i'(w)$ and its length is 2^{j+k+r}. Thus $p(w) \leq \sum_{k=0}^{\log\alpha - r} 2^{j+k+r} \leq 2^{j+1}\alpha$.

Thus, if node w is at height $r + j$ in T and is in the layer of the partition containing T', then $p(w) \leq 2^{j+1}\alpha$; also, there are $p_{\max}/2^{r+j}$ nodes at this height in T. The sum of these $p(w)$'s is thus at most $2\alpha p_{\max}/2^r$. Each layer therefore contributes at most $4\alpha p_{\max}$, and there are $\lceil\log p_{\max}/\log\alpha\rceil$ layers. Thus $\sum_{v\in T} p(v)$ satisfies the bound of the lemma. □

3.3 Derandomisation and parallelisation

Note that all portions of our algorithm are deterministic (and can be implemented in NC), but for the setting of the initial random delays. The sequential derandomisation is a simple application of the method of conditional probabilities to the proof of Lemma 3.1. The derandomisation/parallelisation of these algorithms follows from results of [1], as sketched in Appendix B.

4 Proof of Theorem 2

We just show the existence of the schedules guaranteed by Theorem 2; constructivisation is very similar to the approach of [9] and is omitted here. For Theorem 2(a), we take $\mathcal{S}$ to be a *padded* random $2(P_{\max} + \Pi_{\max})$-delayed schedule. We let $\mathcal{S}'$ be a *well-structured* random $2(P_{\max} + \Pi_{\max})$-delayed schedule that is derived from $\mathcal{S}$, as described in Section 2. The makespan of $\mathcal{S}'$ will be some $L \leq 4P_{\max} + 2\Pi_{\max}$. For some ℓ that is a multiple of $p_{\max}$, we partition $\mathcal{S}'$ into $\lceil L/\ell\rceil$ contiguous frames $F_1, F_2, \ldots$ of length ℓ each. Finally we reschedule the operations within each frame to yield a feasible schedule for that frame and then concatenate the schedules. Since $\mathcal{S}'$ is well-structured and since ℓ is a multiple of $p_{\max}$, no operation will straddle frames. For Theorem 2(b), we take $\mathcal{S}$ to be a random $\Pi_{\max}$-delayed schedule, and partition $\mathcal{S}$ into frames of length w each (recall that the job-shop instance is assumed to be w-separated here).

The *contention of machine M_i in frame F_k*, denoted $C_{i,k}$, is the total processing time needed for the machine by the operations scheduled within the frame. If the contention within a frame is sufficiently small for each machine, we give a probabilistic proof that bounds the makespan of a feasible schedule for the frame; we use the Lovász Local Lemma to show that for some choice of random delays for the frame (i.e., taking the operations scheduled in the frame as a new job-shop instance), the resulting delayed schedule is feasible. When the contention within a frame is too large, we divide it into subframes and solve the problem recursively. Because of the conditions of the theorem (i.e., the time a job needs a particular machine is bounded or its operations using the machine are well-separated) we can argue that the contention in the frames (and subframes) is adequately small so that the prescribed delays do not enlarge the schedule too much. Our use of the Lovász Local Lemma is broadly similar to that of [8]. We first recall the lemma:

Lemma 4.1 ([5]) *Let $E_1, E_2, \ldots, E_\ell$ be any events with* $\Pr(E_i) \leq p$ *for all i. If each E_i is mutually independent of all but at most d of the other events E_j and if $ep(d+1) \leq 1$, then* $\Pr(\bigwedge_{i=1}^{\ell} \overline{E_i}) > 0$.

First, a fairly simple application of the Lovász Local Lemma:

Lemma 4.2 *There is a constant $c' > 0$ such that for any job-shop scheduling instance and any $B \geq 2(P_{\max}+\Pi_{\max})$, there is a* well-structured *B-delayed schedule for the instance with the property that for any machine i and any time t, $C(M_i, t) \leq c' \log(P_{\max} + \Pi_{\max}) / \log\log(P_{\max} + \Pi_{\max})$.*

Replacing the use of Lemma 3.1 by Lemma 4.2 in our proof of Theorem 1(a), we get

Corollary 4.3 *For general job-shop scheduling, there is always a schedule of makespan*

$$O\left((P_{\max} + \Pi_{\max}) \cdot \frac{\log(P_{\max} + \Pi_{\max})}{\log\log(P_{\max} + \Pi_{\max})} \cdot \left\lceil \frac{\log(\min\{m\mu, p_{\max}\})}{\log\log(P_{\max} + \Pi_{\max})} \right\rceil\right).$$

For any positive λ, let $p(i, k, \lambda)$ denote $\Pr(C_{i,k} \geq \lambda)$ for $\mathcal{S}'$. The proof technique of Lemma 4.2 directly gives the following lemma.

Lemma 4.4 *Suppose that*

$$\forall i\ \forall k,\ p(i,k,\lambda) \leq (8e(2P_{\max} + \Pi_{\max})P_{\max}\Pi_{\max})^{-1}.$$

Then there is a $2(P_{\max} + \Pi_{\max})$-delayed well-structured *schedule $\mathcal{S}''$ of the instance such that when $\mathcal{S}''$ is partitioned into frames of length ℓ (some multiple of $p_{\max}$), $C_{i,k} \leq \lambda$ for all machines M_i and frames F_k.*

4.1 Proof of Theorem 2(a)

Lemma 4.5 *Consider a job-shop instance in which every job needs at most u time units on each machine. Let $\mathcal{S}'$ be partitioned into frames of size v, where $u \leq v \leq 2(P_{\max} + \Pi_{\max})$. Then for any machine M_i, any frame F_k and any $\delta > 0$, $p(i, k, v(1+\delta)) \leq G(v/u, \delta)$.*

From now on, let c be a suitably large positive constant. We give a recursive scheme to prove the existential version of Theorem 2(a).

Base Case: $u \geq c\log(P_{\max} + \Pi_{\max})$

If $u^2 \geq 2(P_{\max} + \Pi_{\max})$, then by Corollary 4.3 the theorem holds. If not, partition $\mathcal{S}'$ into frames of length v, where v is the smallest multiple of $p_{\max}$ that is no smaller than u^2. Since $u \geq p_{\max}$, we have $u^2 \leq v \leq u^2 + u$. Fix a machine M_i and frame F_k. By Lemma 4.5 and from the fact that $u \geq c\log(P_{\max} + \Pi_{\max})$,

$$p(i,k,2v) \leq (e/4)^u \leq (e/4)^{c\log(P_{\max}+\Pi_{\max})},$$

which can be made at most $(8e(2P_{\max} + \Pi_{\max})P_{\max}\Pi_{\max})^{-1}$ by taking c suitably large. Thus, by Lemma 4.4, there is a setting of the delays so that each frame has a maximum machine contention of at most $2v$. Choose such a setting. Now view each of the frames as a job-shop scheduling problem wherein $P_{\max}$ and $\Pi_{\max}$ have been replaced by at most $u^2 + u$ and $2(u^2 + u)$ respectively. Thus, by Corollary 4.3, each frame can independently be made a valid schedule of makespan $O\left(u^2 \cdot \frac{\log u}{\log\log u} \cdot \left\lceil \frac{\log(\min\{m\mu, p_{\max}\})}{\log\log u} \right\rceil\right)$. Thus, since there are $O((P_{\max} + \Pi_{\max})/u^2)$ frames, the concatenation of these legal schedules gives us a feasible schedule of makespan

$$O\left((P_{\max} + \Pi_{\max}) \cdot \frac{\log u}{\log\log u} \cdot \left\lceil \frac{\log(\min\{m\mu, p_{\max}\})}{\log\log u} \right\rceil\right).$$

Recursive Case: $u < c\log(P_{\max} + \Pi_{\max})$
We use the following fact.
Fact 4.6 *If $\delta \in [0,1]$, then $e^{\delta}/(1+\delta)^{(1+\delta)} \leq e^{-\delta^2/3}$.*
We now partition $\mathcal{S}'$ into frames of length v, where v is the smallest multiple of $p_{\max}$ that is no smaller than $c^2\log^3(P_{\max} + \Pi_{\max})$. We have $c^2\log^3(P_{\max} + \Pi_{\max}) \leq v \leq c^2\log^3(P_{\max} + \Pi_{\max}) + u \leq 2c^2\log^3(P_{\max} + \Pi_{\max})$. Once again, fix M_i and F_k. By Lemma 4.5 and Fact 4.6,

$$p(i,k,v(1+1/\sqrt{\log(P_{\max} + \Pi_{\max})})) \leq \exp(-c\log^2(P_{\max} + \Pi_{\max})/(3\log(P_{\max} + \Pi_{\max}))).$$

If c is suitably large, Lemma 4.4 ensures that there is a setting of the delays so that each frame has a maximum machine contention of at most $2c^2\log^3(P_{\max} + \Pi_{\max})(1+1/\sqrt{\log(P_{\max} + \Pi_{\max})})$. Choose such a setting, and view each of the frames as a job-shop problem in which $P_{\max}$ and $\Pi_{\max}$ are replaced by $2c^2\log^3(P_{\max} + \Pi_{\max})$ and $2c^2\log^3(P_{\max} + \Pi_{\max})(1+1/\sqrt{\log(P_{\max} + \Pi_{\max})})$ respectively. Recursively make each frame feasible, independently, and concatenate the resulting schedules to produce a feasible schedule for the original problem.

Finally we analyse the makespan of the final schedule. Let F be a frame constructed at some point in the procedure and let a be the "$P_{\max} + \Pi_{\max}$" value of the job-shop instance associated with F. If F is partitioned in a recursive call and F' is a subframe of F with "$P_{\max} + \Pi_{\max}$" value a', then $a' \leq 4c^2(\log^3 a)(1+1/\sqrt{\log a})$ and so $\log(a') \leq 3\log a + c_0$, where c_0 is a constant depending on c. Define a sequence $a_1, a_2, \ldots$, where $a_1 = \log(P_{\max} + \Pi_{\max})$, and $a_{i+1} = 3\log a_i + c_0$. Let r be the first index such that $u \geq c\log a_r$. Then the final schedule has makespan

$$\mathrm{O}\left((P_{\max} + \Pi_{\max}) \cdot \frac{\log u}{\log\log u} \cdot \left\lceil \frac{\log(\min\{m\mu, p_{\max}\})}{\log\log u} \right\rceil \cdot \prod_i (1 + \mathrm{O}(1/\sqrt{a_i}))\right).$$

Since $a_r, a_{r-1}, \ldots, a_1$ grows exponentially fast, $\prod_i(1 + \mathrm{O}(1/\sqrt{a_i})) \leq \exp(\sum_i \mathrm{O}(1/\sqrt{a_i})) = \mathrm{O}(1)$, thus yielding the bound of Theorem 2(a).

4.2 Proof of Theorem 2(b)

We first need an intuitive lemma that is a consequence of the work of [11].
Lemma 4.7 ([11]) *Let $X_1, \ldots, X_\ell \in \{0,1\}$ be random variables such that for any $I \subseteq [\ell]$, $\Pr(\bigwedge_{i\in I}(X_i = 1)) \leq \prod_{i\in I}\Pr(X_i = 1)$; i.e., loosely speaking, the X_i are "negatively correlated". Then if $X = \sum_i X_i$ with $\mathrm{E}[X] = \mu$, we have, for any $\delta > 0$, $\Pr(X \geq \mu(1+\delta)) \leq G(\mu,\delta)$.*
Now we prove Theorem 2(b). Suppose that we have a w-separated job-shop scheduling instance with $p_{\max} = 1$. Consider partitioning the random $\Pi_{\max}$-delayed schedule $\mathcal{S}$ into frames of length w. Fix a machine M_i and a frame F_k. For each operation O that needs to be done on machine M_i, introduce an indicator variable X_O for the event that this operations is scheduled in F_k on M_i. Thus, the contention for M_i in F_k is $C_{i,k} = \sum_O X_O$. Note that $\mathrm{E}[C_{i,k}] \leq w$. If O and O' are from the *same* job then the probability that they are both scheduled on M_i in F_k is zero, due to our given assumption on w. Furthermore, operations from different jobs are independent. Thus, the variables X_O are negatively correlated in the sense of Lemma 4.7; hence $\Pr(C_{i,k} > w(1+\delta)) \leq G(w,\delta)$, for any $\delta > 0$. Note that, for a suitably large constant c'', we can choose (i) $\delta = c''$ if $w \geq \log(P_{\max} + \Pi_{\max})/2$, and (ii) $\delta = c''\log(P_{\max} + \Pi_{\max})/(w\log(\log(P_{\max} + \Pi_{\max})/w))$ if $w < \log(P_{\max} + \Pi_{\max})/2$, to ensure that $G(w,\delta) \leq (8e(2P_{\max} + \Pi_{\max})P_{\max}\Pi_{\max})^{-1}$. Thus, by Lemma 4.4, there is a setting of the delays so that each frame has a maximum machine contention of at most $w(1+\delta)$. Choose such a setting, and focus on any frame. Note that every job can have at most one operation on any machine in this frame, and that all operations are of length one. Thus, by invoking the main result of [8], each frame can be made feasible with makespan $\mathrm{O}(w + w(1+\delta)) = \mathrm{O}(w(1+\delta))$. We finally concatenate all the feasible schedules, concluding the proof.

References

[1] N. Alon and A. Srinivasan. Improved parallel approximation of a class of integer programming problems. To appear in *Algorithmica*.

[2] D. Applegate and W. Cook. A computational study of the job-shop scheduling problem. *ORSA Journal of Computing*, 3:149–156, 1991.

[3] A. Bar-Noy, R. Canetti, S. Kutten, Y. Mansour, and B. Schieber. Bandwidth allocation with preemption. In *Proc. ACM Symposium on Theory of Computing*, pages 616–625, 1995.

[4] J. Carlier and E. Pinson. An algorithm for solving the job-shop problem. *Management Science*, 35:164–176, 1989.

[5] P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In A. Hajnal et. al., editors, *Infinite and Finite Sets*, pages 609–627. Colloq. Math. Soc. J. Bolyai 11, North Holland, Amsterdam, 1975.

[6] L. A. Hall. Approximability of flow shop scheduling. In *Proc. IEEE Symposium on Foundations of Computer Science*, pages 82–91, 1995.

[7] 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 et al., editors, *Handbooks in Operations Research and Management Science, Volume 4: Logistics of Production and Inventory*, pages 445–522. Elsevier Science Publishers, 1993.

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

[9] F. T. Leighton, B. Maggs, and A. Richa. Fast algorithms for finding O(congestion + dilation) packet routing schedules. Manuscript.

[10] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[11] J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. *SIAM J. Discrete Math.*, 8:223–250, 1995.

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

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

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

[15] D. P. Williamson, L. A. Hall, J. A. Hoogeven, C. A. J. Hurkens, J. K. Lenstra, S. V. Sevast'yanov, and D. B. Shmoys. Short shop schedules. *Operations Research*, to appear.

Acknowledgements

Aravind Srinivasan thanks David Shmoys for introducing him to this area, for sharing many of his insights, and for his several suggestions. He also thanks Cliff Stein and Joel Wein for their suggestions and many helpful discussions. We thank Bruce Maggs and Andréa Richa for sending us an updated version of [9].

A Proofs of lemmas

Convexity. Recall that function $f : \Re \rightarrow \Re$ is *convex* in the interval $[a, b]$ if and only for all a' and b' such that $a \leq a' \leq b' \leq b$ and for all $q \in [0, 1]$, $f(qa' + (1-q)b') \leq qf(a') + (1-q)f(b')$.

Lemma A.1 *If f is any convex* non-decreasing *function in the finite interval $[a, b]$ and if X is a random variable taking values only in $[a, b]$ such that $E[X] \leq \mu$, then*

$$E[f(X)] \leq \frac{b-\mu}{b-a} f(a) + \frac{\mu - a}{b-a} f(b).$$

Proof. Since f is convex in $[a, b]$, we have $f(x) \leq \frac{b-x}{b-a} f(a) + \frac{x-a}{b-a} f(b)$, for any $x \in [a, b]$. Thus, since X is distributed in $[a, b]$, we have

$$f(X) \leq \frac{b-X}{b-a} f(a) + \frac{X-a}{b-a} f(b). \tag{1}$$

The proof is completed by taking expectations on both sides of (1), and noting that $f(b) \geq f(a)$. □

Corollary A.2 *Let f be any convex non-decreasing function in the finite interval $[a, b]$. Suppose $x_1, x_2, \ldots, x_\ell$ are such that for each i, $x_i \in [a, b]$ and $\sum_j x_j \leq c$, then*

$$\sum_i f(x_i) \leq \frac{\ell b - c}{b-a} f(a) + \frac{c - \ell a}{b-a} f(b).$$

Proof. This follows immediately from Lemma A.1 by taking X to be the uniform distribution over the the multiset $\{x_1, x_2, \ldots, x_\ell\}$. □

A.1 Proof of Fact 2.1

By Lemma A.1 and the convexity of $f(x) = (1+\delta)^x$, $\mathrm{E}[(1+\delta)^{X_i}] \leq 1 - \mathrm{E}[X_i] + (1+\delta)\mathrm{E}[X_i] = 1 + \delta \mathrm{E}[X_i] \leq e^{\delta \mathrm{E}[X_i]}$. The fact then follows from the independence of $X_1, \ldots, X_\ell$.

A.2 Proof of Lemma 3.1

Let $B = 2\Pi_{\max}$ and let $\mathcal{S}$ be a *padded* random B-delayed schedule of the new instance. $\mathcal{S}$ has a makespan of at most $2(P_{\max} + \Pi_{\max})$. Let $\mathcal{S}'$ be the well-structured random B-delayed schedule for the original instance that can be constructed from $\mathcal{S}$, as described in Section 2. The contention on any machine at any time under $\mathcal{S}'$ is clearly no more than under $\mathcal{S}$. Thus $\mathcal{S}'$ satisfies (a) and (b) with high probability since, by the following, $\mathcal{S}$ does.

Part (a). The following proof is based on that in [14]. For any positive integer k, and any M_i,

$$\Pr(C(M_i, t) \geq k \text{ for any } t\ ) \leq \binom{2\Pi_{\max}}{k} (1/(2\Pi_{\max}))^{k-1} \leq 2\Pi_{\max}/k!.$$

But $\Pi_{\max} \leq n\mu p_{\max}$, which by our assumptions is poly(m, μ) (recall that the reductions of Section 2 ensure that n and $p_{\max}$ are both at most poly(m, μ)). Since $\lceil \alpha \rceil! > (m\mu)^{c_1/2}$ for sufficiently large m or μ, we can satisfy (a) with high probability if we choose c_1 sufficiently large.

Part (b). Let $\gamma = \beta\epsilon/4$, where ϵ is the desired constant in the probability bound. Let the constant c_2 in the definition of β be sufficiently large so that $\gamma > 1$. Fix any M_i and t, and let $\lambda = \mathrm{E}[C(M_i,t)]$. (By Fact 2.3, $\lambda \leq 1$.) By Fact 2.1, with $1+\delta = \gamma$,

$$\mathrm{E}[\gamma^{C(M_i,t)}] \leq e^{(\gamma-1)\lambda} \leq e^{(\gamma-1)}.$$

Hence, for any given t,

$$\mathrm{E}[\gamma^{C(t)}] \leq \sum_{i\in[m]} \mathrm{E}[\gamma^{C(M_i,t)}] \leq me^{\gamma-1}. \tag{2}$$

Since the function $x \mapsto \gamma^x$ is convex, by Jensen's inequality we get that $\mathrm{E}[\gamma^{C(t)}] \geq \gamma^{\mathrm{E}[C(t)]}$. If we choose c_2 sufficiently large then $\gamma^\gamma \geq me^{\gamma-1}$ and so, by (2), $\mathrm{E}[C(t)] \leq \gamma$. By linearity of expectation, $\mathrm{E}[\sum_t C(t)] \leq 4\gamma(P_{\max} + \Pi_{\max})$ and finally, by Markov's inequality, we have $\Pr(\sum_t C(t) > \beta(P_{\max} + \Pi_{\max})) \leq 4\gamma/\beta = \epsilon$.

A.3 Proof of Lemma 4.2

As in the proof of Lemma 3.1, it will suffice for us to prove the bounds for the *padded* schedule $\mathcal{S}$; these will immediately extend to the well-structured schedule $\mathcal{S}'$. We will take up this approach in proving Lemmas 4.2 and 4.5.

Assign the delays randomly and let $E_{i,t}$ be the event

$$C(M_i,t) > c' \log(P_{\max} + \Pi_{\max})/\log\log(P_{\max} + \Pi_{\max}),$$

and E_i be the event that for some t, $E_{i,t}$ occurs.

As in the proof of Lemma 2 part (a), we can show that the probability of E_i is at most $1/(4eP_{\max}\Pi_{\max})$ by taking c' as a suitably large constant. As [8] observed, the contention pattern of any machine M_i is independent of (any function of) the contention patterns of all the machines M_k such that M_i and M_k have no common job that uses both. Thus, since there are at most $2\Pi_{\max}$ jobs that need M_i and since each of them can use at most $2P_{\max} - 1$ other machines, the "dependency" among the set of events $\{E_i : i \in [m]\}$, in the sense of Lemma 4.1, is at most $2\Pi_{\max}(2P_{\max} - 1) \leq 4P_{\max}\Pi_{\max} - 1$. This shows that $\Pr(\bigwedge_{i=1}^m \overline{E_i}) > 0$ by an invocation of the Lovász Local Lemma.

A.4 Proof of Lemma 4.5

Let $X(i,j,k)$ be the random variable denoting the total amount of processing time of job J_j on machine M_i in time frame F_k. Let $x_{i,j}$ denote the total amount of processing that J_j needs on M_i, in the given job-shop instance. Then

$$\forall i \forall j \forall k,\ \mathrm{E}[X(i,j,k)] \leq x_{i,j}\ell/2(P_{\max} + \Pi_{\max});\ \text{also, } 0 \leq X(i,j,k) \leq x_{i,j} \leq u.$$

Thus, for any $\gamma > 0$, Lemma A.1 shows that

$$\mathrm{E}[\exp(\gamma X(i,j,k))] \leq \frac{\ell\exp(\gamma x_{i,j})}{2(P_{\max} + \Pi_{\max})} + 1 - \frac{\ell}{2(P_{\max} + \Pi_{\max})} \leq \exp(\frac{\ell(\exp(\gamma x_{i,j}) - 1)}{2(P_{\max} + \Pi_{\max})}). \tag{3}$$

Fix M_i and F_k. Note that the random variables $\{X(i,j,k) : j \in [j]\}$ are mutually independent, and that $C_{i,k} = \sum_j X(i,j,k)$. Thus we have, for $\lambda = \ell(1+\delta)$ and any $\gamma > 0$,

$$p(i,k,\lambda)\exp(\gamma\lambda) \quad \leq \quad \mathrm{E}[\exp(\gamma C_{i,k})] \quad \text{(by Markov's inequality)}$$

$$\begin{aligned} &= \prod_j \mathrm{E}[\exp(\gamma X(i,j,k))] \\ &\leq \exp\left(\frac{\ell}{2(P_{\max}+\Pi_{\max})}\sum_j(\exp(\gamma x_{i,j})-1)\right), \end{aligned} \tag{4}$$

by (3). Since $0 \leq x_{i,j} \leq u$ and since $\sum_j x_{i,j} \leq 2(P_{\max}+\Pi_{\max})$, Corollary A.2 shows that

$$\sum_j(\exp(\gamma x_{i,j})-1) \leq (2(P_{\max}+\Pi_{\max})/u)(\exp(\gamma u)-1).$$

Thus, by (4), we see that

$$p(i,k,\ell(1+\delta)) \leq \exp(-\gamma\ell(1+\delta)+\frac{\ell}{u}(\exp(\gamma u)-1)).$$

Choosing $\gamma = \ln(1+\delta)/u > 0$, we get the claimed bound on $p(i,k,\ell(1+\delta))$.

B Derandomisation and parallelisation

Note that all portions of our algorithm are deterministic, except for the setting of the initial random delays, which we show how to derandomise now. The sequential derandomisation (which we omit in this version) is a simple application of the method of conditional probabilities to the probabilistic argument of Lemma 3.1. In any case, it will follow from the NC algorithm that we now present. As said before, we assume without loss of generality that $p_{\max}$ is at most poly(m,μ) and at most poly(n,μ), for our sequential and parallel algorithms respectively. We begin with a technical lemma.

Lemma B.1 *Let $x_1, x_2, \ldots, x_\ell$ be non-negative integers such that $\sum_i x_i = \ell a$, for some $a \geq 1$. Let k be any positive integer such that $k \leq a$. Then,*

$$\sum_{i=1}^{\ell} \binom{x_i}{k} \geq \ell \cdot \binom{a}{k}.$$

Proof. For real x, we define, as usual, $\binom{x}{k} \doteq (x(x-1)\cdots(x-k+1))/k!$. We first verify that the function $f(x) = \binom{x}{k}$ is non-decreasing and convex for $x \geq k$, by a simple check that the first and second derivatives of f are non-negative for $x \geq k$. Think of minimising $\sum_i \binom{x_i}{k}$ subject to the given constraints. If $x_i \leq (k-1)$ for some i, then there should be an index j such $x_j \geq (k+1)$, since $\sum_i x_i \geq \ell k$. Thus, we can lessen the objective function by simultaneously setting $x_i := x_i + 1$ and $x_j := x_j - 1$. Hence we may now assume that all the integers x_i are at least k. Now, by the convexity of f for $x \geq k$, we see that the objective function is at least $\sum_{i=1}^{\ell}\binom{a}{k}$. □

Define, for $z = (z_1, z_2, \ldots, z_n) \in \Re^n$, a family of symmetric polynomials $S_j(z), j = 0, 1, \ldots, n$, where $S_0(z) \equiv 1$, and for $1 \leq j \leq n$, $S_j(z) \doteq \sum_{1 \leq i_1 < i_2 \cdots < i_j \leq n} z_{i_1} z_{i_2} \cdots z_{i_j}$. We now recall one of the main results of [1] (this is not explicitly presented in [1], but is an obvious corollary of the results of Section 4 in [1]).

Proposition B.2 ([1]) *Suppose we are given m* independent *random variables $y_1, \ldots, y_m$, each of which takes values* uniformly *in $R = \{0, 1, \ldots, 2^b - 1\}$ where $b = \mathrm{O}(\log N)$; N here is a parameter that roughly stands for "input length", and $m = N^{\mathrm{O}(1)}$. Suppose we are also given, for each $j \in [m]$, a finite set of binary random variables $\{z_{jt} : t = 1, 2, \ldots\}$ where z_{jt} is 1 iff y_j lies in some fixed subset R_{jt} of R. Also given are r random variables*

$$U_i = \sum_{j=1}^{m} z_{j,f(i,j)}, \ i \in [r],$$

where f is some arbitrary given function. Now if $\mathrm{E}[U_i] < 1$ *for each i, then given any positive integer k such that $k = \mathrm{O}(\log N)$, we can find, deterministically using $N^{\mathrm{O}(1)}$ processors and $\mathrm{O}(\log^{\mathrm{O}(1)} N)$ time on the EREW PRAM, a setting $y_1 := w_1, \ldots, y_m := w_m$ such that*

$$\sum_{i\in[r]} S_k(z_{1,f(i,1)}, \ldots, z_{m,f(i,m)}) \leq rG(1, k-1)(1+N^{-c}),$$

for any desired constant $c > 0$.

In our setting, the random variables y_i are the initial random choices of the machines. It is easy to verify that each random variable $C(M_i, t)$ is of the form of some U_j in the notation of Proposition B.2. Now, by giving the initial random delays in the range $\{0, 1, \ldots, 2\Pi_{\max}\}$ instead of from $\{0, 1, \ldots, 2\Pi_{\max} - 1\}$, we can ensure the condition $\mathrm{E}[U_j] < 1$ of Proposition B.2 ($\mathrm{E}[C(M_i, t)] \leq 2\Pi_{\max}/(2\Pi_{\max} + 1)$ now). Let α and β be as in Lemma 3.1, and note that both are logarithmically bounded in the length of the input, as required for the parameter k in Proposition B.2. Let the random variables $X_{i,j,t}$ be as in Fact 2.3. From the proof of part (a) of Lemma 3.1, we see that $\sum_{i,t} G(1, \alpha - 1)$ is smaller than 1; thus, by Proposition B.2, we can find a setting $\vec{w}$ for the initial delays in NC such that

$$\sum_{i,t} S_\alpha(X_{i,1,t}, X_{i,2,t}, \ldots, X_{i,n,t}) < 1. \tag{5}$$

Now, if the congestion of some machine M_i at some t were at least α due to the above setting of the initial delays to $\vec{w}$, then the left-hand-side of (5) would be at least 1, contradicting (5). Thus, we have an NC derandomisation of Theorem 1(a).

As for Theorem 1(b), we can similarly find an NC assignment of initial delays $\vec{w}$ such that

$$\sum_{i,t} S_\beta(X_{i,1,t}, X_{i,2,t}, \ldots, X_{i,n,t}) < \mathrm{O}((P_{\max} + \Pi_{\max})mG(1, \beta - 1)) = \mathrm{O}((P_{\max} + \Pi_{\max})). \tag{6}$$

Let $C(t)$ be the (deterministic) maximum contention at time t, due to this setting. Note that

$$\binom{C(t)}{\beta} \leq \sum_i S_\beta(X_{i,1,t}, X_{i,2,t}, \ldots, X_{i,n,t}).$$

Thus, by (6), we see that

$$\sum_t \binom{C(t)}{\beta} = \mathrm{O}((P_{\max} + \Pi_{\max})).$$

We now invoke Lemma B.1 to conclude that $\sum_t C(t) = \mathrm{O}((P_{\max} + \Pi_{\max})\beta)$; thus, we have an NC derandomisation of Theorem 1(b).