Fixed-Order Scheduling on Parallel Machines

Thomas Bosman, Dario Frascaria, Neil Olver, Rene Sitters, Leen Stougie

Fixed-Order Scheduling on Parallel Machines *

Thomas Bosman[1], Dario Frascaria[1], Neil Olver[1,2], René Sitters[1,2], and Leen Stougie[2,1]

[1] Department of Econometrics and Operations Research, Vrije Universiteit Amsterdam, The Netherlands
[2] CWI, Amsterdam, The Netherlands

Abstract. We consider the following natural scheduling problem: Given a sequence of jobs with weights and processing times, one needs to assign each job to one of m identical machines in order to minimize the sum of weighted completion times. The twist is that for machine the jobs assigned to it must obey the order of the input sequence, as is the case in multi-server queuing systems. We establish a constant factor approximation algorithm for this (strongly NP-hard) problem. Our approach is necessarily very different from what has been used for similar scheduling problems without the fixed-order assumption. We also give a QPTAS for the special case of unit processing times.

1 Introduction

We consider an extremely simple, yet challenging, scheduling principle that arises in many logistic and service applications. Given a sequence of jobs and a set of machines, we need to dispatch the jobs one by one over the machines, where for each machine the ordering of the original sequence is preserved. Hence, each machine must handle the jobs in a first-in first-out (FIFO) order. Each job has a processing time p_j and weight w_j and the goal is to minimize the weighted sum of completion times, where the completion time of a job j is the total processing time of the jobs preceding j (including j) on the same machine.

The FIFO-ordering restriction is common in queuing theory, where the *task assignment problem* [9, 8, 5] is concerned with the same question, except that jobs arrive stochastically over time. Our problem can be seen as asking for the optimal way of dispatching jobs from a single queue over m server queues under complete information of the processing times, essentially *unzipping* a single queue into m queues. The reverse problem of *zipping* m queues into a single queue, is the classic single machine scheduling problem: $1|\text{chains}|\sum w_j C_j$ (in the 3-field notation by Graham et al. [7]) and can be solved efficiently by greedily selecting a prefix of jobs with largest ratio of total weight over total processing time [12].

In the scheduling literature, the fixed-ordering scheduling problem can be seen as a special case of *scheduling problems with sequence dependent setup times*,

* This work was partially supported by the Netherlands Organisation for Scientific Research (NWO) through a VIDI grant (016.Vidi.189.087) and the Gravitation Programme Networks (024.002.003).

where for each pair of jobs there is a changeover cost c_{ij} that is paid if job j is the immediate successor of job i on a machine. Such setup times occur naturally in many industrial applications [1]. Our problem is precisely this, in the special case where $c_{ij} = \infty$ if i is later than j in the ordering, and $c_{ij} = 0$ otherwise. While the problem has received substantial attention (see [2, 3, 1] for a comprehensive literature review), almost nothing is known from a theoretical perspective (an exception is [10], but this is concerned with a rather unusual objective function). We believe our work may shed light on this more general problem.

We remark that the online version of the problem, where we need to assign a job before we get to know the next job in the sequence, does not admit a constant-competitive algorithm. Consider, for example, two machines and a sequence of three jobs (where job 1 should be completed first and job 3 last on any machine) with $p_1 = k^2, p_2 = k, p_3 = 1$ and $w_1 = 1, w_2 = k$ and w_3 is either zero or k^3, depending on the schedule of the first two jobs. Here, k is an arbitrarily large number. It is easy to see that a good schedule requires knowledge of the weight of job 3 before deciding whether to put jobs 1 and 2 on the same machine.

Our contribution. Scheduling problems with weighted completion times objective *without* ordering constraints typically admit good approximations algorithms. For identical machines there is a PTAS [19], while a slightly better than $\frac{3}{2}$-approximation [4] for unrelated machines is known. But even simpler approaches yield a constant factor approximation. On identical machines, it is a classic result due to Kawaguchi and Kyan [11] (see [17] for a modern proof) that scheduling greedily according to Smith ratio (see Section 2) is a $\frac{1+\sqrt{2}}{2}$-approximation. For unrelated machines, independent rounding of both a natural time-indexed LP [16] and a (nontrivial) convex quadratic program [18] works, and achieve approximation ratios of $\frac{3}{2} + \epsilon$ and $\frac{3}{2}$ respectively. Another approach is *α-point scheduling* [15], where jobs are sorted according to the time by which an α fraction of the job has been processed in some LP relaxation. The jobs are then scheduled greedily in that order. This method has enabled many algorithmic improvements in scheduling, since it can be modified to deal with additional complications, such as precedence constraints and release times [13].

Fixed-order scheduling appears highly resistant to all these techniques. A big obstacle is that moving even a single pair of jobs onto the same machine can have a catastrophic effect on the objective function if the order is fixed: think of a job with large processing time but minuscule weight, followed by a job with large weight and minuscule processing time. Thus in order to have any hope of a non-trivial performance guarantee, jobs must be assigned to machines in a highly dependent way. To achieve this, our approach radically departs from earlier ones.

We define an important partial order $\prec$: essentially, $j \prec k$ means that not only is j earlier than k in the FIFO ordering, but also this ordering is opposite to what would be preferred according to Smith's rule. Thus as alluded to earlier, if $j \prec k$ we should be particularly careful about assigning j and k to the same machine. Order the machines arbitrarily. A key idea is that at only a constant factor loss, we can restrict our attention to a class of schedules we call *Smith-monotone*, meaning that if $j \prec k$, then j is assigned to an earlier machine (or the same machine) as

the one that k is assigned to. Next, we relax the problem by computing each job's completion times partially, based only on jobs preceding it in the partial order $\prec$. While this potentially distorts completion times a lot, we can ensure the amortized effect is not too large by appropriately rounding weights and processing times. Finally, we formulate a new LP using these partial completion times and enforcing the strong structure imposed by Smith monotonicity. This LP can be rounded in a way that completely respects the pairwise probabilities of jobs being assigned to the same machine; Smith monotonicity is crucial here. The rounding is very appealing and natural, and can be seen an analog of α-point scheduling with respect to machine index rather than time.

Finally, we remark that for the case of unit processing times, the complexity of the problem is unknown, but it is unlikely to be APX-hard: we present a QPTAS in Section A of the appendix.

2 Problem Definition, Notation and NP-hardness

We have a set of identical machines $M = \{1, \ldots, m\}$, each of which can process one job at a time, and a totally ordered set of jobs $J = 1, ..., n$, where each job $j \in J$ has weight $w_j \in \mathbb{N}$ and processing time $p_j \in \mathbb{N}$. Because of its frequent use we also define notation for the so called *Smith ratio* $s_j := \frac{w_j}{p_j}$ of job j. A feasible schedule $\mu : J \to M$ assigns to every job j a machine $\mu(j)$. Each machine processes the jobs assigned to it in order of their number. The cost of the schedule μ is the sum of weighted completion times, i.e.,

$$\Gamma(\mu) = \sum_{k \in J} \sum_{\substack{j \in J : j \leq k, \\ \mu(j) = \mu(k)}} p_j \cdot w_k. \tag{1}$$

The objective is to minimize the cost of the schedule. We denote by OPT the optimal cost, by $\sigma_\mu : J \to \mathbb{N}$ the function that maps every job to its completion time under μ and by $\prec$ a partial order on J such that $j \prec k$ if and only if $j < k$ and $s_j \leq s_k$.

Note that we explicitly disallow $p_j = 0$. This is for convenience, and ensures that the Smith ratio w_j/p_j is always well defined. Our results easily extend to the case where jobs with zero processing time are allowed.

The problem of minimizing the sum of weighted completion times without ordering constraints is a classic problem that has long been known to be strongly NP-hard [6]. This result extends to fixed-order scheduling as well: given an assignment of jobs to a machine, it is always optimal to schedule them in decreasing order of Smith ratio, so the ordering constraints become redundant if $s_1 \geq s_2 \geq \cdots \geq s_n$.

3 Structural properties of optimal solutions

In this section we will provide a characterization of an optimal solution which will help us construct a constant-factor approximation algorithm in Section 4.

Unit processing times

Suppose all jobs have equal processing time and hence that the relation $j \prec k$ indicates that $j < k$ and $w_j \leq w_k$. W.l.o.g. we may then as well assume that the processing times are 1. An initial simplification is that we will assume that schedules are *staircase shaped*, in the sense that for every prefix of the jobs $1, \ldots, k$, the number of jobs assigned to each machine decreases monotonically with the machine index. We will use the following equivalent definition.

Definition 1. *A schedule μ is **staircase shaped** if for each job k, $\mu(k) = |\{j < k : \sigma_\mu(j) = \sigma_\mu(k)\}| + 1$.*

Given any schedule μ, we can clearly turn it into a staircase shaped schedule without changing the completion time of any job.

Clearly we want jobs with high weights to be completed early, but this may not always be possible because of the ordering on the jobs. Intuitively, it seems like a good idea to 'reserve' some of the good spots early in the schedule, for higher weight jobs later in the sequence. In staircase shaped schedules this means that early and low weight jobs should be put on low index machines as much as possible. Lemma 1 below makes this more precise.

Definition 2. *A schedule μ is **Smith-monotone** if, for every $j \prec k$, it holds that $\mu(j) \leq \mu(k)$.*

Lemma 1. *For unit processing times, there exists an optimal schedule that is Smith-monotone and staircase shaped.*

Proof. Let us define the potential function $\sum_{j \in J} \mu(j) j$. Let μ be a solution maximizing this potential function among those that are optimal and staircase shaped. Suppose μ is not Smith-monotone. We obtain a contradiction by showing there is a staircase shaped schedule with a higher potential but no higher cost.

Since μ is not Smith-monotone, there exists a pair $j \prec k$ that violates Smith monotonicity. Pick j, k so that there is no other violating pair $j' \prec k'$ between it, i.e with $j \leq j'$ and $k' \leq k$ and at least one of the inequalities strict. We call such a pair *tight*. For $g = j, k$, let $S_g = \{h \in \{j+1, \ldots, k-1\} : \mu(h) = \mu(g)\}$ be the set of jobs between j and k on the machine of job g. It follows that $h \in S_k \implies w_h \leq w_j$ and that $h \in S_j \implies w_h \geq w_k$ as, otherwise, the pair $j \prec k$ would not be tight. (In fact, $<$ holds.) By assigning j to the machine of k and vice-versa we get a schedule μ' that improves our potential function. We first show is that the new schedule μ' does not incur a higher cost than μ.

Since only the starting times (whence completion times) of j, k and jobs in S_j and S_k may change, all others remaining equal, it follows that

$$\sum_{h \in \{j\} \cup S_k} \sigma_{\mu'}(h) + \sum_{h \in \{k\} \cup S_j} \sigma_{\mu'}(h) = \sum_{h \in \{j\} \cup S_k} \sigma_\mu(h) + \sum_{h \in \{k\} \cup S_j} \sigma_\mu(h),$$

hence

$$\sum_{h \in \{j\} \cup S_k} (\sigma_{\mu'}(h) - \sigma_\mu(h)) = \sum_{h \in \{k\} \cup S_j} (\sigma_\mu(h) - \sigma_{\mu'}(h)). \tag{2}$$

Now note that the completion times of jobs j and S_k increase, while those of k and S_j decrease, since the schedule was staircase shaped. Combining this with the fact that $w_h \leq w_j$ for $h \in S_k$ and $w_h \geq w_k$ for $h \in S_j$, we can bound the increase in the cost as follows:

$$\sum_{h \in \{j,k\} \cup S_j \cup S_k} w_h(\sigma_{\mu'}(h) - \sigma_\mu(h)) =$$
$$\sum_{h \in \{j\} \cup S_k} w_h(\sigma_{\mu'}(h) - \sigma_\mu(h)) - \sum_{h \in \{k\} \cup S_j} w_h(\sigma_\mu(h) - \sigma_{\mu'}(h)) \leq$$
$$w_j \sum_{h \in \{j\} \cup S_k} (\sigma_{\mu'}(h) - \sigma_\mu(h)) - w_k \sum_{h \in \{k\} \cup S_j} (\sigma_\mu(h) - \sigma_{\mu'}(h)) \leq 0 \quad \text{by (2).}$$

So the new schedule has a higher potential function and no higher cost. The schedule may not be staircase shaped, however. But simply sorting the jobs fixes this without undoing our work. To see this, note that the set of timeslots occupied on each machine did not change when we modified the schedule. So the fact that the old schedule was staircase shaped, implies that the new schedule still has the following weaker property: if exactly k jobs are scheduled at time t, they are scheduled on the first k machines. By sorting all the jobs assigned to one timeslot by number and assigning them to the first available machine in that order, the potential function can only increase further, while completion times stay the same. Hence, we have found an optimal staircase shaped schedule with higher potential function than we started with, contradicting our choice of the original schedule and concluding the proof.

General processing times

Unfortunately, for general processing times we cannot hope for an equally nice structural result. Indeed, there may not be an optimal schedule that is Smith-monotone. However, as we will now show, we may impose this structure with the loss of only a constant factor in the objective.

The proof works by reducing a general instance to one with unit processing times, finding an optimal Smith-monotone schedule, and then rounding it back. Although this reduction is not polynomial time, it will suffice to prove the bound on the optimality ratio. Instead, in the next section we will find that we can bypass the reduction, and approximate such a schedule directly.

Given an instance I to the general problem, let I^{unit} be the instance with unit processing times obtained from I by replacing every job $j \in J$ with a set $U(j) = \{j^1, \ldots, j^{p_j}\}$ of p_j consecutive jobs, each having unit processing time and weight w_j/p_j. Let J^{unit} be the set of jobs of I^{unit} and OPT^{unit} be the optimal cost for I^{unit}.

Lemma 2. $\text{OPT}^{unit} \leq \text{OPT} - \sum_{j \in J} \frac{1}{2}(p_j - 1)w_j$

Proof. The statement follows for the schedule μ^{unit} obtained from the optimal schedule for I by putting all the jobs in $U(j)$ on the machine $\mu(j)$, for all $j \in J$.

The completion times of the jobs in $U(j)$ run from $\sigma(j) - p_j + 1$ to $\sigma(j)$, and hence the average completion time is $\sigma(j) - \frac{1}{2}(p_j - 1)$. The proof follows from multiplying by the total weight of the jobs in $U(j)$.

Lemma 3. *An optimal Smith-monotone schedule for I has cost at most $\text{OPT}^{unit} + \sum_{j \in J}(p_j - 1)w_j$.*

Proof. Given an optimal schedule μ^{unit} for I^{unit}, we can create a schedule for I by putting job j on machine i with probability $|\{h \in U(j) : \mu^{unit}(h) = i\}|/|U(j)|$, for all $j \in J$. The expected time spent processing job $j \in J$ on machine i is exactly $|\{h \in U(j) : \mu^{unit}(h) = i\}|$. As a consequence, the expected starting time of a job $j \in J$ is at most the average starting time of the jobs in $U(j)$, and thus the expected completion time is at most the expected completion time of jobs in $U(j)$ plus $p_j - 1$ (which is the difference between the processing time of job j and any job in $U(j)$).

What remains to show is that μ is Smith-monotone. By Lemma 1 we can assume, w.l.o.g., that the optimal solution of I^{unit} satisfies Smith monotonicity. Consider an arbitrary pair $j \prec k$. We have that $j' \prec k'$ for all jobs $j' \in U(j), k' \in U(k)$. This implies that, if any job in $U(j)$ is scheduled on machine i, all jobs in $U(k)$ are scheduled on machines with index not smaller than i. Hence it holds that the machine with highest index to which j may be assigned cannot have index larger than any machine to which k may be assigned. Therefore, for every possible realization of the random schedule, Smith monotonicity is satisfied, completing the proof.

Lemma 4. *An optimal Smith-monotone schedule has cost at most $\frac{3}{2}\text{OPT}$.*

Proof. By Lemma 3, we have that an optimal Smith-monotone schedule has cost at most $\text{OPT}^{unit} + \sum_{j \in J}(p_j - 1)w_j$, which in turn, by Lemma 2, is at most $\text{OPT} + \frac{1}{2}\sum_{j \in J}(p_j - 1)w_j \leq \frac{3}{2}\text{OPT}$.

Though the bound in Lemma 4 is unlikely to be tight, it cannot be improved much further. The Kawaguchi-Kyan bound of $\frac{1+\sqrt{2}}{2} \approx 1.207$ is known to be tight [11, 17], and this lower bound carries over to fixed-order scheduling: the worst-case example uses jobs with equal Smith ratios, and in that case reordering the jobs assigned to a machine does not change the cost.

4 A constant factor approximation algorithm

In this section we will describe an algorithm that proves our main result: Theorem 1. Our approach is as follows: first we round the instance such that all Smith ratios are powers of $\frac{1}{3}$ (by rounding up the weights as appropriate). Given that, we show that a certain relaxed objective function is always within a constant of the original objective function. We then use LP rounding to find a Smith-monotone schedule that is optimal with respect to the relaxed objective function.

Theorem 1. *Fixed-order scheduling can be approximated to within a factor* $\frac{27}{2} + 9\sqrt{3} < 29.1$.

From hereon assume that all Smith ratios are powers of some fixed $\gamma \in (0,1)$. Suppose we relax the objective function (1) to disregard the contribution of processing times of jobs with $j < k$ and $w_j > w_k$ (hence, $j \not\prec k$) in the completion time of job k. We claim this relaxation loses only a factor $(\frac{4}{1-\sqrt{\gamma}} - 3)$.

Definition 3. *The* ***partial completion time*** *of a job $k \in J$ under a schedule μ is*

$$\tilde{c}_k = \sum_{j \preceq k : \mu(j) = \mu(k)} p_j \ .$$

The ***partial cost*** *of a schedule μ is $\tilde{\Gamma}(\mu) = \sum_{k \in J} w_k \tilde{c}_k$.*

It is crucial that the Smith ratios are powers of γ; this ensures that either $j \preceq k$ or w_j is relatively large compared to w_k. Intuitively, in the latter case we don't care too much about the effect of j's processing time on the lighter-weight job k.

Theorem 2. *Take an instance where all Smith ratios are positive powers of $\gamma \in (0,1)$. Consider any schedule μ and denote its cost by $\Gamma(\mu)$. It holds that*

$$\tilde{\Gamma}(\mu) \leq \Gamma(\mu) \leq \left(\frac{4}{1-\sqrt{\gamma}} - 3 \right) \cdot \tilde{\Gamma}(\mu).$$

Since it is obvious that $\tilde{\Gamma}(\mu) \leq \Gamma(\mu)$ we prove the upper bound.

To simplify notation assume the instance has only one machine; the result can be applied to each machine individually. For $d \in \mathbb{N}$, let N_d be the set of jobs j with $s_j = \gamma^d$; let W_d and P_d be, respectively, the total weight and total processing time of jobs in N_d. We denote by $H_d = \frac{1}{W_d} \sum_{k \in N_d} w_k \tilde{c}_k$ the weighted average of the partial cost in N_d. It follows that

$$\tilde{\Gamma}(\mu) = \sum_{k \in J} w_k \tilde{c}_k = \sum_{d \in \mathbb{N}} W_d H_d = \sum_{d \in \mathbb{N}} \gamma^d P_d H_d. \tag{3}$$

Our goal is to bound $\Gamma(\mu)$ in the same terms. Since H_d only accounts for the contribution of jobs with Smith ratio at most γ^d in the completion time of jobs in N_d, we need to correct for the other jobs (that are in $N_1, \ldots, N_{d-1}$). In the worst case, all these jobs are scheduled first and hence their processing times need to be added. Therefore we get:

$$\Gamma(\mu) \leq \sum_{d \in \mathbb{N}} W_d (H_d + \sum_{i=1}^{d-1} P_i) \leq \sum_{d \in \mathbb{N}} \gamma^d P_d (H_d + \sum_{i=1}^{d-1} P_i). \tag{4}$$

We will prove that this value can be bounded by the desired constant times $\tilde{\Gamma}(\mu)$. Globally our strategy is to show that every newly introduced term $\gamma^d P_d P_i$ can be charged to a term in the expression for $\tilde{\Gamma}(\mu)$ such that no term gets charged more than a $\frac{4}{1-\sqrt{\gamma}} - 4$ fraction of its value.

Before we can proceed we will need the inequality in Lemma 5 below. It says that the average weighted completion time in N_d is at least half the total processing time in N_d, which can intuitively be seen as follows: think of all jobs $j \in N_d$ as blocks of equal width, length p_j and mass w_j. So, all blocks have equal density. Now stack the boxes on top of each other: then P_d corresponds to the total length, and H_d approximately to the center of mass, which is in the middle.

Lemma 5. *$P_d \leq 2H_d$.*

Proof.

$$H_d = \frac{1}{W_d} \sum_{k \in N_d} w_k \sum_{j \leq k \wedge s_j \leq s_k} p_j \geq \underbrace{\frac{1}{W_d} \sum_{k \in N_d} w_k \sum_{j \leq k \wedge j \in N_d} p_j}_{:=Q}$$

Suppose now that $Q < \frac{P_d}{2}$. Since $W_d = \gamma^d P_d$, it follows that:

$$Q = \frac{1}{P_d} \sum_{k \in N_d} p_k (P_d - \sum_{h > k \wedge h \in N_d} p_h) = P_d - \frac{1}{P_d} \sum_{h \in N_d} p_h \sum_{k < h \wedge k \in N_d} p_k \geq P_d - Q,$$

implying that $Q \geq P_d/2$, a contradiction. □

Lemma 6. *$\max\{\gamma^d P_d H_d, \gamma^i P_i H_i\} \geq \frac{1}{2}(\gamma^d P_d P_i)\gamma^{\frac{1}{2}(i-d)}$.*

Proof. Suppose that $\gamma^d P_d H_d < \frac{1}{2}\gamma^d P_d P_i \gamma^{\frac{1}{2}(i-d)}$. This implies that

$$H_d < \frac{1}{2} P_i \gamma^{\frac{1}{2}(i-d)}. \tag{5}$$

So we obtain

$$\gamma^d P_d P_i \gamma^{\frac{1}{2}(i-d)} \leq \gamma^d 2 H_d P_i \gamma^{\frac{1}{2}(i-d)} \overset{(5)}{<} \gamma^{d-i}\gamma^i P_i \gamma^{\frac{1}{2}(i-d)} P_i \gamma^{\frac{1}{2}(i-d)} \leq 2\gamma^i H_i P_i,$$

where the first and third inequalities follow from Lemma 5. □

We are now ready to prove Theorem 2.

Proof (Theorem 2). We will prove the following inequality, implying the theorem by (4):

$$\sum_{d \in \mathbb{N}} \gamma^d P_d H_d \big(\frac{4}{1-\sqrt{\gamma}} - 4\big) \geq \sum_{d \in \mathbb{N}} \gamma^d P_d \sum_{i=1}^{d-1} P_i.$$

Applying Lemma 6 and replacing the max by a sum we get:

$$\begin{aligned}
\sum_{d \in \mathbb{N}} \gamma^d P_d \sum_{i=1}^{d-1} P_i &\leq \sum_{d \in \mathbb{N}} \sum_{i=1}^{d-1} \gamma^{\frac{1}{2}(d-i)} 2 \max(\gamma^d P_d H_d, \gamma^i P_i H_i) \\
&\leq \sum_{d \in \mathbb{N}} \sum_{i=1}^{d-1} \gamma^{\frac{1}{2}(d-i)} 2(\gamma^d P_d H_d + \gamma^i P_i H_i) \\
&\leq \sum_{d \in \mathbb{N}} \sum_{i=1}^{\infty} \gamma^{\frac{1}{2}i} 4(\gamma^d P_d H_d) = \sum_{d \in \mathbb{N}} \gamma^d P_d H_d \big(\frac{4}{1-\sqrt{\gamma}} - 4\big). \square
\end{aligned}$$

Linear Programming Relaxation

Suppose all Smith ratios are positive powers of $\gamma \in (0,1)$. The following mixed-integer program captures the problem of finding a Smith-monotone ordering that minimizes the modified objective $\tilde{\Gamma}(\mu)$.

$$\begin{aligned}
\min \quad & \sum_{k \in J} w_k \tilde{c}_k \\
\text{s.t.} \quad & u_k \geq u_j + z_{jk} && \forall j \prec k && (6)\\
& u_k \leq m && \forall k \in J && (7)\\
& \tilde{c}_k \geq p_k + \sum_{j \prec k} (1 - z_{jk}) p_j && \forall k \in J && (8)\\
& u_k \in \mathbb{N}, z_{jk} \in \{0,1\} && \forall k \in J, j \prec k
\end{aligned}$$

Here, z_{jk} is the indicator variable for the event that j and k are assigned to different machines. The variable u_k indicates which machine job k is assigned to. Finally, $\tilde{c}_k$ represents the partial completion time of job k. The constraint (6) is valid since we require a Smith-monotone ordering.

Now consider the natural LP relaxation of this mixed-integer program, where we drop the integrality requirements and instead require $1 \leq u_k \leq m$, $0 \leq z_{jk} \leq 1$. Denote this relaxation by (LP). Let $(z^*, u^*, \tilde{c}^*)$ be an optimal solution to (LP), with cost OPT_{LP}.

Definition 4. *For $\beta \in (0,1)$, the **β-point schedule** associated to u^* is the schedule obtained by assigning job j to machine $\lceil u_j^* - \beta \rceil$.*

From now on, β will be chosen uniformly at random from $(0,1)$, making the β-point schedule a random schedule.

Let N_d be the set of jobs with Smith ratios γ^d and let $\tilde{C}_k$ be the (random) partial completion time of job k under the β-point schedule. The following statements are easy to verify.

Proposition 1. *For any pair of jobs $j \prec k$, the probability that jobs j and k are assigned to the same machine under the β-point schedule is at most $1 - z_{jk}^*$.*

Proof. This follows immediately from the constraint (6). □

Proposition 2. *For any $k \in J$, $\mathbb{E}[\tilde{C}_k] \leq \tilde{c}_k^*$. Hence*

$$\mathbb{E}\Big[\sum_{k \in J} w_k \tilde{C}_k\Big] \leq \sum_{k \in J} w_k \tilde{c}_k^* = \text{OPT}_{\text{LP}}.$$

Proof. This follows from Proposition 1 and (8). □

Note that a β-point schedule can be derandomized in polynomial time: a job j is always assigned to u_j^* when u_j^* is integral and to either $\lfloor u_j^* \rfloor$ or $\lceil u_j^* \rceil$ otherwise. Let b_j be the maximum value of β for which $\lceil u_j^* - \beta \rceil = \lfloor u_j^* \rfloor$. It follows that all possible schedules are the ones obtained by assigning to β the values in $\{b_j | j \in J\} \cup \{1\}$.

Constant factor approximation

We are now ready to complete the main proof of this section.

Proof (Theorem 1). Given an instance I where the jobs have arbitrary Smith ratios, let I^γ be the instance obtained from I by rounding up the weights such that the Smith ratios are powers of γ, where γ will be defined later, and then scaled to be in $(0, 1)$. Let OPT_I and OPT_{I^γ} be the the optimal cost for the instances I and I^γ respectively. Clearly the cost of any schedule μ on I^γ is at most γ^{-1} times the cost of μ on I and $\text{OPT}_I \leq \text{OPT}_{I^\gamma}$ since the cost of a schedule is linear in the weights (see (1)) and weights do not affect feasibility.

We denote by μ and OPT_{LP}, respectively, the β-point schedule and the optimal cost of (LP) on I^γ. By Lemma 4, Proposition 2 and Theorem 2, the cost of μ on I is at most

$$\frac{3}{2}\left(\frac{4}{1-\sqrt{\gamma}}-3\right)\cdot\text{OPT}_{\text{LP}} \leq \frac{3}{2}\left(\frac{4}{1-\sqrt{\gamma}}-3\right)\cdot\text{OPT}_{I^\gamma} \leq \frac{3}{2}\left(\frac{4}{1-\sqrt{\gamma}}-3\right)\cdot\text{OPT}_I\gamma^{-1}.$$

Minimizing over γ, this yields an approximation ratio of

$$\min_{\gamma\in(0,1)} \frac{3}{2}\left(\frac{4}{1-\sqrt{\gamma}}-3\right)\cdot\gamma^{-1} = \frac{3}{2}(9+6\sqrt{3}) = \frac{27}{2}+9\sqrt{3}$$

when $\gamma = 1/3$, concluding the proof. □

5 Epilogue

Our work suggests many further interesting and natural directions. One is to find a PTAS (or even a polynomial time exact algorithm) for unit processing times, perhaps expanding on the QPTAS in the appendix. Good approximation algorithms for all of the following problems remain open questions.

(1) There are k arrival lines that need to be dispatched over m servers, such that the FIFO ordering in each of the arrival lines is obeyed in each of the server queues.
(2) An arbitrary partial order on the jobs is given, and we require that if two jobs are assigned to the same machine, then the partial order is respected. (1) is exactly this problem, where the partial order is described by k disjoint chains.
(3) Instead of requiring that the order is exactly preserved, a natural relaxation is to allow a *reordering buffer* (see, e.g. [14]) of limited size. Jobs enter the buffer in the given FIFO order, but any job in the buffer can be chosen and assigned to one of the machines.

References

1. A. Allahverdi. The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 246(2):345 – 378, 2015.

2. A. Allahverdi, J. N. Gupta, and T. Aldowaisan. A review of scheduling research involving setup considerations. *Omega*, 27(2):219 – 239, 1999.
3. A. Allahverdi, C. Ng, T. Cheng, and M. Y. Kovalyov. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985 – 1032, 2008.
4. N. Bansal, A. Srinivasan, and O. Svensson. Lift-and-round to improve weighted completion time on unrelated machines. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing*, pages 156–167, 2016.
5. H. Feng, V. Misra, and D. Rubenstein. Optimal state-free, size-aware dispatching for heterogeneous M/G/-type systems. *Performance Evaluation*, 62(1):475 – 492, 2005.
6. M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, USA, 1979.
7. R. Graham, E. Lawler, J. Lenstra, and A. Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 287 – 326. Elsevier, 1979.
8. M. Harchol-Balter. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press, New York, NY, USA, 1st edition, 2013.
9. M. Harchol-Balter, M. E. Crovella, and C. D. Murta. On choosing a task assignment policy for a distributed server system. *IEEE Journal of Parallel and Distributed Computing*, 59(2):204–228, 1999.
10. K. Hiraishi, E. Levner, and M. Vlach. Scheduling of parallel identical machines to maximize the weighted number of just-in-time jobs. *Computers and Operations Research*, 29(7):841 – 848, 2002.
11. T. Kawaguchi and S. Kyan. Worst case bound of an LRF schedule for the mean weighted flow-time problem. *SIAM Journal on Computing*, 15(4):1119–1129, 1986.
12. E. L. Lawler. Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics*, 2:7590, 1978.
13. M. Queyranne and A. S. Schulz. Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. *SIAM Journal on Computing*, 35(5):1241–1253, 2006.
14. H. Räcke, C. Sohler, and M. Westermann. Online scheduling for sorting buffers. In R. Möhring and R. Raman, editors, *Proceedings of 10th Annual European Symposium on Algorithms*, pages 820–832, 2002.
15. A. S. Schulz and M. Skutella. The power of α-points in preemptive single machine scheduling. *Journal of Scheduling*, 5(2):121–133, 2002.
16. A. S. Schulz and M. Skutella. Scheduling unrelated machines by randomized rounding. *SIAM Journal on Discrete Mathematics*, 15(4):450–469, 2002.
17. U. Schwiegelshohn. An alternative proof of the Kawaguchi-Kyan bound for the Largest-Ratio-First rule. *Operations Research Letters*, 39(4):255–259, 2011.
18. M. Skutella. Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the ACM*, 48:206–242, 2001.
19. M. Skutella and G. J. Woeginger. A PTAS for minimizing the total weighted completion time on identical parallel machines. *Mathematics of Operations Research*, 25(1):63–75, 2000.

A A QPTAS for unit processing times

In this section we sketch a simple quasipolynomial time approximation scheme (QPTAS) for the problem under unit processing times. Note that we do not know if this version of the problem remains NP-hard. However, it seems to capture most of the difficulty, so we feel that tackling this case will help substantially in improving the upper bound for the general case. The QPTAS works by solving a relaxed problem by dynamic programming. We round the completion times to geometric intervals and then we consider schedules in which at any time point only one machine per completion time can accept jobs. This sufficiently reduces the solution space to get a quasipolynomial time algorithm.

The first step is to consider only a logarithmic number of distinct completion times. Let $R = \{\lfloor (1+\epsilon)^i \rfloor : i \in \mathbb{N}\}$ be the set of integers found by rounding down a geometric series growing with rate $1+\epsilon$. Then order the elements $1 = R_1 < R_2 < \dots$ and take $R_0 = 0$ by convention. Assume that s is the smallest index such that $R_s \geq n$, and note that $s = O(\log_{1+\epsilon}(n))$. Now consider the objective of minimizing the weighted sum of *rounded* completion times, where each completion time is rounded up to the nearest R_i. Call this the *rounded objective*; clearly the rounded objective of any schedule is at most $1+\epsilon$ times the actual objective. So, if we can find an optimal segmented schedule for the rounded objective, we immediately get a $(1+\epsilon)$-approximation to the original problem.

Now we define a restricted type of schedule, which we call a *segmented staircase schedule*. A segmented staircase schedule is similar to a staircase shaped schedule, except that the "steps" are now defined in terms of the rounded completion times. When j is assigned to a machine, it is assigned to the leftmost machine that gives it the same rounded completion time. In other words, if a job j gets assigned to $\mu(j)$ and gets completed at time $t \in (R_i, R_{i+1}]$, then $\mu(j)$ is the lowest index machine for which the number of jobs $k < j$ assigned to it does not exceed $R_{i+1} - 1$.

Lemma 7. *There is an optimal solution to the problem of minimizing the rounded objective that is a segmented staircase schedule.*

Proof. Take μ to be a schedule for which $(\mu(1), \mu(2), \dots, \mu(n))$ is lexicographically minimal amongst all solutions that are optimal for the rounded objective. Notice that μ must be staircase shaped; otherwise, transforming it into a staircase shaped schedule would yield a schedule μ' in which every job has the same completion time, but which is lexicographically smaller than μ.

Suppose for a contradiction that this schedule is not a segmented staircase schedule. Let j be the last (maximum index) job that violates the rule for a segmented staircase schedule: j is assigned to machine h', but $h < h'$ is the smallest index machine that gives it the same rounded completion time, ignoring all jobs after j. Let k be the first job $k > j$ scheduled on machine h. (If there is no such job, then moving j to h' reduces the lexicographical value and does not increase the total rounded completion time.) Note that since j was chosen maximally, no job ℓ with $j < \ell < k$ is scheduled on machine h'. Moreover,

$\sigma_\mu(k) > \sigma_\mu(j)$, since μ is staircase shaped, so it must be that j and k are both in the same segment $(R_i, R_{i+1}]$ for some i. So we can simply swap k and j to obtain a lexicographically smaller schedule of the same rounded objective value.

For segmented staircase schedules, we can compactly describe the loads on the machines just prior to assigning job j. Let X_i^j be the number of jobs on machines with load currently in the interval $[R_i, R_{i+1})$ just prior to assigning job j. Since only one of the machines with load in that interval can have strictly more than R_i jobs on it, this number also completely determines how many machines there are with loads exactly R_i. For each j, we have $n^{O(\log_{1+\epsilon} n)}$ options for the values of $X_1^j, X_2^j, \ldots, X_s^j$. Once we know the minimum cost of a schedule attaining each of those options, we can compute the cost of all the schedules up to job $j+1$. Hence, we get the main result of this section.

Theorem 3. *For any $\epsilon > 0$, there is a $(1+\epsilon)$-approximation algorithm for fixed-order scheduling with unit processing times with running time $n^{O(\log_{1+\epsilon} n)}$.*