Operations Research Letters 44 (2016) 469–473

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Splitting versus setup trade-offs for scheduling to minimize weighted completion time

José Correa [a], Victor Verdugo [a,b,*], José Verschae [c]

[a] Departamento de Ingeniería Industrial, Universidad de Chile, Chile
[b] Departement d'Informatique, École normale supérieure, France
[c] Facultad de Matemáticas & Escuela de Ingeniería, Pontificia Universidad Católica de Chile, Chile

ARTICLE INFO

Article history:
Received 6 May 2014
Received in revised form
30 April 2016
Accepted 30 April 2016
Available online 7 May 2016

Keywords:
Scheduling
Split jobs
Weighted completion time

ABSTRACT

We study scheduling problems when jobs can be split and a setup is required before processing each part, to minimize the weighted sum of completion times. Using a simple splitting strategy and a reduction to an orders scheduling problem we derive a 2-approximation algorithm for the case with uniform weights and setups, improving upon previous work. We extend this idea to the general identical machine case and conclude by designing a constant factor approximation algorithm when machines are unrelated.

1. Introduction

Scheduling problems in which jobs are allowed to be split into many parts and be processed independently and concurrently on different machines naturally model situations in which jobs consist of a large number of small identical operations. When processing a job (or its small operations) the setup cost is negligible, while when switching to a different job a setup time is required, during which the machine cannot process or setup any other job (see Fig. 1). This family of scheduling problems was introduced by Potts and Van Wassenhove [9] to study batching and lot-sizing integrated with scheduling decisions in manufacturing.

In this paper we consider a stylized version of the model in which jobs can be arbitrarily split, introduced by Serafini [11] as a scheduling model for the textile industry. More specifically, our model considers a set M of m machines and a set J of n jobs under two different machine environments: unrelated and identical machines. In the unrelated machine setting, each job $j \in J$ has a machine dependent setup time s_{ij} and processing time p_{ij}, while for identical machines the processing and setup times are machine independent, so $p_{ij} = p_j$ and $s_{ij} = s_j$ for all $j \in J$ and $i \in M$. On the other hand, each job $j \in J$ is associated with a nonnegative weight w_j and the objective is to minimize the weighted sum of job completion times. Thus, the problems studied are denoted by $R|\text{split}|\sum w_jC_j$ (unrelated machines) and $P|\text{split}|\sum w_jC_j$ (identical parallel machines).

A number of scheduling problems with split jobs are polynomially solvable in the absence of setup times [11,13], however when setup times are present these problems become NP-hard. Indeed, Schalekamp et al. [10] show that $P|\text{split}|\sum w_jC_j$ is NP-hard even with uniform setup times ($s_j = s$). They also design a 2.781-approximation algorithm for the uniform setup time version of $P|\text{split}|\sum C_j$ and an exact polynomial time algorithm for the case $m = 2$, though the complexity is open for $m \geq 3$. Xing and Zhang [13] consider the problem of minimizing makespan on identical machines with splitting jobs and setup times, $P|\text{split}|C_{\max}$, obtaining a $(1.75 - 1/m)$-approximation algorithm. This was later improved to an algorithm with a worst-case factor of 5/3 by Chen et al. [2]. Recently, Correa et al. [3] obtain a $(2.618 + \varepsilon)$-approximation algorithm for the case of unrelated machines, $R|\text{split}|C_{\max}$. We refer to the survey by Allahverdi et al. [1] for further related results.

In this paper, an outgrowth of [12], we first improve upon the results of Schalekamp et al. [10], who design a greedy strategy for $P|\text{split}|\sum C_j$ and uniform setups. In their algorithm jobs are sorted by their size and split among an appropriate number of machines such that the job finishes the earliest. The algorithm and its analysis combine the decision of how to split and how to schedule simultaneously. This provokes extra technical difficulties in the analysis. In a broad sense, the completion time of each job

* Corresponding author at: Departamento de Ingeniería Industrial, Universidad de Chile, Chile.
E-mail addresses: correa@uchile.cl (J. Correa), victor.verdugo@ens.fr (V. Verdugo), jverschae@uc.cl (J. Verschae).

http://dx.doi.org/10.1016/j.orl.2016.04.011
0167-6377/

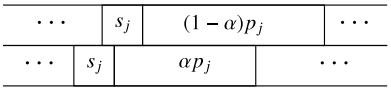

Fig. 1. Example where a job is split in two pieces.

is bounded by using the average load of jobs processed before it. The bound on the total cost obtained by this method naturally yields a term that depends purely on the processing time of jobs, and another depending on the setup times. Afterwards, each term is bounded separately. Bounding the term related to the setup times requires some technical effort and it is not clear how to generalize it to other cases, for example, for weighted jobs or job-dependent setup times. In contrast, we propose a simple but new splitting scheme that separates the decision of splitting and scheduling. This provides a more robust strategy that extends naturally to different problem specifications and yields very clean and simple arguments. Specifically we split jobs into uniform pieces of size essentially equal to the setup. We then interpret the problem we are left with as a problem of scheduling orders, where orders correspond to jobs and jobs correspond to pieces, and by applying existing machinery [7,8] we obtain a factor 4 approximation algorithm for $P|\text{split}|\sum w_jC_j$, which improves to $(2+\varepsilon)$ if the setups are uniform, and further to 2 if also the objective is unweighted. In the latter case the analysis becomes remarkably simple.

To wrap-up we consider the most general version of the problem for which we can design a constant factor approximation algorithm. This turns out to be the unrelated machine problem $R|\text{split}, r_{ij}|\sum w_jC_j$ in which additionally jobs have machine-dependent release dates. In this setting the splitting concept is that of Correa et al. [3] where if a fraction x_{ij} of job j is assigned to machine i it requires time $x_{ij}p_{ij}+s_{ij}$ to be processed.

2. Identical machines

Clearly in the identical machine case we can consider that $p_j, w_j, s_j \in \mathbb{Z}_+$. Recall that a job may be split into several parts, which can be processed simultaneously on different machines and before processing any part of a job a setup time is required, during which the machine cannot process or setup any other job. These setup times affect C_j, the completion time of job $j \in J$, and therefore also $\sum_{j\in J} w_jC_j$, the objective to be minimized.

2.1. Splitting jobs into orders

Key to our algorithms is a rule that pre-specifies the splitting pattern of every job j. If a job j has a positive setup time, the rule splits the job into parts of size s_j, which together with their setups need a total processing time of $2s_j$ each. If the setup time is zero, the rule splits the job into m parts of size p_j/m. The variant obtained, where jobs are split and each part is not splittable, can be interpreted as the following scheduling problem with *orders*. Consider a set of jobs J that is partitioned into sets $L_1 \cup \cdots \cup L_k$ with $L_s \cap L_{s'} = \emptyset$ for all $s \neq s'$. Each set L_s corresponds to an order, that is completed when the last of its jobs is completed. Thus, the completion time of order L is $C_L := \max\{C_j : j \in L\}$. Given weights $w_L \in \mathbb{Z}_+$ for each order L, we aim to find a schedule on m machines that minimizes $\sum_L w_LC_L$. We denote this problem as $P|\text{part}|\sum w_LC_L$ [4].

Given an instance I of $P|\text{split}|\sum w_jC_j$, we transform it to an instance I' of $P|\text{part}|\sum w_LC_L$ as follows: each job with $s_j > 0$ is split into $\lceil p_j/s_j\rceil$ parts, each of size $2s_j$. A job j with $s_j = 0$ is split into m parts of size p_j/m. Together, the parts of a given job j are interpreted as an order L_j where the parts correspond to jobs and the weight of L_j is $w_{L_j} := w_j$. We remark that the number of jobs in I' might be pseudo-polynomial. In Section 2.2 we discuss how we deal with this technical issue. The next two lemmas show how to transform feasible schedules for an instance I of $P|\text{split}|\sum w_jC_j$ into schedules for the transformed instance I' of $P|\text{part}|\sum w_LC_L$ and back.

Lemma 1. *A feasible schedule for I' with completion times C_{L_j} can be transformed into a feasible schedule to I with completion times $C_j \le C_{L_j}$ for each $j \in J$.*

Proof. Consider a feasible schedule for I'. Since each job in L_j has processing time $2s_j$, we can use the time slots occupied by jobs in L_j to process job j. In this slot we can schedule the setup time s_j plus s_j processing time units of job j. The chosen number of $\lceil p_j/s_j\rceil$ parts for job j guarantees that job j can be processed for $s_j \cdot \lceil p_j/s_j\rceil \ge p_j$ units of time, and thus fully completed before C_{L_j}. □

Lemma 2. *A feasible schedule for I with completion times C_j can be transformed into a feasible schedule for I' with completion times $C_{L_j} \le 2C_j$ for each $j \in J$.*

Proof. By a simple swapping argument, we can restrict ourselves to consider a schedule for I that processes each job contiguously on each machine. Let t_{ij} be the amount of time used to schedule job j on machine i (without considering the setup). We will see that duplicating the total time spent to process job j on each machine i, that is $s_{ij}+t_{ij}$, is enough to process the complete order L_j.

If the job j has a positive setup time, doing this gives a total of $2(s_j+t_{ij}) = 2s_j\left(1+t_{ij}/s_j\right) \ge 2s_j\lceil t_{ij}/s_j\rceil$ units of time available for order L_j on machine i. Hence, we can process at least $\lceil t_{ij}/s_j\rceil$ jobs of L_j on machine i. For this case, the lemma follows by noticing that the total number of jobs of L_j that we can process is

$$\sum_{i\in M}\left\lceil\frac{t_{ij}}{s_j}\right\rceil \ge \left\lceil\sum_{i\in M}\frac{t_{ij}}{s_j}\right\rceil \ge \left\lceil\frac{p_j}{s_j}\right\rceil = |L_j|.$$

When the job has a setup time equal to zero, by duplicating the time we obtain a total of $2t_{ij} \ge (p_j/m)\cdot\lceil 2m\cdot t_{ij}/p_j - 1\rceil$ units of time available for order L_j on machine i. Therefore, we can process $\lceil 2m\cdot t_{ij}/p_j - 1\rceil$ jobs of order L_j on machine i and thus in total we can process at least m jobs of L_j:

$$\sum_{i\in M}\left\lceil\frac{2m\cdot t_{ij}}{p_j}-1\right\rceil \ge \left\lceil\sum_{i\in M}\left(\frac{2m\cdot t_{ij}}{p_j}-1\right)\right\rceil = m. \quad \square$$

Corollary 3. *For any $\alpha \ge 1$, an α-approximation algorithm for $P|\text{part}|\sum w_LC_L$ implies a pseudo-polynomial 2α-approximation algorithm for $P|\text{split}|\sum w_jC_j$.*

The last result is an immediate consequence of Lemmas 1 and 2 and it naturally raises the question of whether the proposed way of splitting can be done more effectively. We show that if each job is split into several parts independently to the rest of the instance, then our splitting procedure is best possible regarding the worst-case loss in the objective function. More precisely, we say that $F(\cdot,\cdot)$ is a *splitting function* if $F(p,s)$ is a vector $(f_1(p,s),\ldots,f_{r(p,s)}(p,s))$ such that $\sum_{\ell=1}^{r(p,s)} f_\ell(p,s) = p$. Then, any splitting function defines a feasible way of splitting job j in $r(p_j,s_j)$ many pieces, of sizes $s_j+f_\ell(p_j,s_j)$ for $\ell \in \{1,\ldots,r(p_j,s_j)\}$.

Lemma 4. *Consider an instance of $P|\text{split}|\sum w_jC_j$ and a splitting function F. If each job is split according to F, then there exists an instance for which the transformation increases the optimal value by a factor of at least 2.*

Proof. Consider a single job with processing time $p \in \mathbb{Z}_+$ and setup time $s = 1$. Suppose there exists an entry in $F(p, s)$ with value $f \geq 1$ and there are $m = p^2$ machines. Any solution with this splitting has a cost of at least $1 + f \geq 2$. In an optimal schedule the job is split uniformly over the p^2 machines achieving a cost of $1 + 1/p$. Therefore, the splitting scheme induced by F increases the cost by at least a factor of $2/(1 + 1/p)$, which converges to 2 when $p \to \infty$.

Now suppose that every entry in the vector $F(p, s)$ is at most $\alpha < 1$, which implies that $r(p, s) \geq p/\alpha$. If the instance contains a single machine, then the cost of scheduling the pieces on the machine is at least $p/\alpha + p$. In an optimal schedule the job is not split, yielding a cost of $1 + p$. We obtain that cost increases by a factor of at least

$$\frac{p/\alpha + p}{1 + p} > \frac{2p}{1 + p} \xrightarrow{p \to \infty} 2. \quad \square$$

2.2. A 4-approximation for $P|split|\sum w_jC_j$

Now we obtain a 4-approximation algorithm for the general $P|\text{split}|\sum w_jC_j$ problem using Corollary 3 and a result of Leung et al. [7] for $P|\text{part}|\sum w_LC_L$.

Leung et al. [7] design a list-scheduling algorithm for $R|\text{part}|\sum w_jC_j$ which specialized to the identical parallel machine setting leads to a 2-approximation algorithm. The algorithm works as follows. First relabel the orders such that $p(L_1)/w_{L_1} \leq \cdots \leq p(L_n)/w_{L_n}$, where $p(L) := \sum_{j \in L} p_j$. For each $k = 1, \ldots, n$, take all jobs in L_k in arbitrary order and schedule them iteratively in the machine with the smallest load. In the instance of $P|\text{part}|\sum w_LC_L$ constructed in Corollary 3 the number of jobs is pseudopolynomial and all jobs within an order have the same processing time. We now argue how we can use this fact in order to implement this algorithm in polynomial time. In this setting, a schedule is described by specifying the number of jobs n_{ik} of each order L_k assigned to machine i.

Lemma 5. *The list-scheduling algorithm by Leung et al. [7] can be implemented in polynomial time if all jobs in an order have the same processing times and the number of jobs per order is pseudopolynomial.*

Proof. Let us assume that we have scheduled orders $L_1, \ldots, L_{k-1}$, and let T_i be the completion time (load) of machine i in that schedule. We denote by C_k the completion time of order L_k in the list-scheduling algorithm and by p_k the processing time of a job in L_k. We notice the following property: after scheduling order L_k each machine that processes a job in L_k has load in $[C_k - p_k, C_k]$. Thus, we have that

$$n_{ik} \in \left\{ \left\lceil \frac{C_k - p_k - T_i}{p_k} \right\rceil, \left\lceil \frac{C_k - p_k - T_i}{p_k} \right\rceil + 1 \right\}.$$

Consider the function $h(C) = \sum_{i \in M} \lceil (C - p_k - T_i)/p_k \rceil$, which is an estimate of the number of jobs of size p_k that can be used to greedily schedule jobs having a completion time at most C. In particular it holds that $0 \leq |L_k| - h(C_k) \leq m$. Moreover, we know that $\min_{i \in M} T_i \leq C_k \leq \min_{i \in M} T_i + |L_k| p_k$. Since h is non-decreasing, by running a binary search procedure we can find C^* such that $|L_k| - m \leq h(C^*) \leq |L_k|$. Then we assign $\lceil (C^* - p_k - T_i)/p_k \rceil$ many jobs of L_k to each machine i. The at most m missing jobs in L_k can be assigned greedily. $\square$

We conclude the following result.

Theorem 6. *The problem $P|split|\sum w_jC_j$ admits a 4-approximation algorithm.*

2.3. Uniform setup times

We now turn to the special case in which the setup times are independent of the jobs. For the corresponding problem with orders, this translates into the case in which all jobs have the same processing times, $P|p_j = p, \text{part}|\sum w_LC_L$. In what follows we argue that this problem admits a PTAS. To this end we first prove that for $P|p_j = p, \text{part}|\sum w_LC_L$ we can focus on solutions given by a list-scheduling algorithm. However, unlike the list-scheduling algorithm studied in Section 2.2, the sequence of orders is not necessarily given by Smith's rule. Without loss of generality we assume that $p = 1$.

Lemma 7. *For each instance of $P|p_j = 1, part|\sum w_LC_L$ there exists a list-scheduling optimal solution.*

Proof. Consider an optimal schedule S with completion times C_L and relabel the orders such that $C_{L_1} \leq C_{L_2} \leq \cdots \leq C_{L_n}$. We feed the list-scheduling algorithm with the sequence of orders $L_1, \ldots, L_n$. The completion time of order L_j in this schedule is $C'_{L_j} = \lceil \frac{1}{m} \sum_{k \leq j} |L_k| \rceil$. On the other hand, in the original schedule S all the orders $L_1, \ldots, L_j$ have been completed up to time C_{L_j}, and then we have that $mC_{L_j} \geq \sum_{k \leq j} |L_k|$. Because of the integrality of C_{L_j} it holds that $C_{L_j} \geq \lceil \frac{1}{m} \sum_{k \leq j} |L_k| \rceil = C'_{L_j}$. $\square$

Lemma 7 teaches us an important lesson. Optimal solutions to $P|p_j = p, \text{part}|\sum w_LC_L$ are characterized by the sequence of orders. We next show that we can further reduce the problem to a single machine problem with a non-linear objective, namely $\sum w_j \lceil C_j/m \rceil$.

Lemma 8. *The instances of $P|p_j = p, part|\sum w_LC_L$ are in a cost-preserving one-to-one correspondence to instances of $1 \parallel \sum w_j \lceil C_j/m \rceil$.*

Proof. We map each instance I_o of the problem with orders to an instance I_s for the single machine problem as follows. Each order L of I_o corresponds to a job $j(L)$ in I_s with processing time $|L|$ and weight w_L. Clearly this mapping is bijective. By Lemma 7 we know that there exists an optimal schedule for I_o given by a list-scheduling algorithm. Let $L_1, \ldots, L_n$ be any sequence of orders. If we schedule the jobs in I_s according to this sequence, the total cost is

$$\sum_{j=1}^{n} w_{L_j} \left\lceil \frac{C_{L_j}}{m} \right\rceil = \sum_{j=1}^{n} w_{L_j} \left\lceil \frac{1}{m} \sum_{k \leq j} |L_k| \right\rceil,$$

which coincides with the cost of the solution for I_o following the corresponding list-scheduling solution. Thus, any solution for I_s is in one-to-one correspondence to a list-scheduling solution I_o of equal cost. $\square$

With Lemma 8 we can design approximation algorithms for the single machine problem, which turns out to be strongly NP-hard [6], and then transfer them to the problem with orders. To this end we use the PTAS for $1 \parallel \sum_j w_j f(C_j)$, where f is any computable non-negative non-decreasing function, obtained by Megow and Verschae [8], and directly obtain a PTAS for $P|p_j = p, \text{part}|\sum w_LC_L$, and a $(2 + \varepsilon)$-approximation algorithm for our split job problem with uniform setup times.

We remark that the fact that the transformation in Corollary 3 creates a pseudopolynomial number of jobs per order does not affect the polynomiality of the algorithm just described. Indeed, the number of jobs in the instance of $P|p_j = p, \text{part}|\sum w_LC_L$ corresponds to the processing time of jobs of the instance of $1 \parallel \sum w_j f(C_j)$, which can be described with polynomially many bits.

Theorem 9. *There exists a $(2+\varepsilon)$-approximation for the problem with splittable jobs and equal setup times.*

Finally, when all setup times are equal and all weights are unit, i.e., $P|s_j = s, \text{split}|\sum C_j$. The same reasoning allows us to reduce the problem to $1 \parallel \sum \lceil C_j/m \rceil$ by losing a factor of 2 in the approximation guarantee. For the latter problem a simple swapping argument amounts to conclude that the SPT rule yields an optimal schedule, and therefore we obtain a 2-approximation algorithm.

Theorem 10. *The split scheduling problem admits a 2-approximation in the case of uniform setup times and uniform weights.*

3. Unrelated machines

We now focus on the unrelated machine case, $R|\text{split}|\sum w_j C_j$. Here, a fraction y of job j on machine i uses $s_{ij} + y \cdot p_{ij}$ time units in total. We consider the problem even under machine-dependent release dates r_{ij} for each machine–job pair i, j. For technical reasons we scale the processing times and setups such that $p_{ij} \geq m$ for all nonzero p_{ij}'s. Additionally, also by scaling, we can assume that if $p_{ij} = 0$ then $s_{ij} \geq 1$. Thus, any feasible solution satisfies $C_j \geq 1$ for all j.

Our algorithm is based on an interval-indexed LP relaxation introduced by Hall et al. [5] and the extension in [4]. Let T be an upper bound on the makespan of the optimal schedule, e.g., $T = \sum_{i,j} r_{ij} + s_{ij} + p_{ij}$. Given $1 < \alpha \leq 2$, let q be such that $\alpha^{q-1} \geq T$. We partition the time axis in intervals $[\tau_0, \tau_1], (\tau_1, \tau_2], \ldots, (\tau_{q-1}, \tau_q]$, where $\tau_0 = 1$ and $\tau_k = \alpha^{k-1}$. Our LP-relaxation considers variables $y_{ijk} \in [0, 1]$ that represent the fraction of job j that finishes at interval $(\tau_{k-1}, \tau_k]$ for each $k \geq 2$, and y_{ij1} represents the fraction of job j finishing in $[0, 1]$. In what follows when it is not specified k ranges in $\{1, \ldots, q\}$, i ranges in M and $j \in J$.

$$\min \sum_j w_j C_j$$

$$\text{s.t.} \sum_{i,k} y_{ijk} = 1 \quad \text{for all } j, \tag{1}$$

$$\sum_j \sum_{\ell \leq k} y_{ij\ell}(p_{ij} + s_{ij}) \leq \tau_k \quad \text{for all } k, i, \tag{2}$$

$$\sum_{i,k} y_{ijk}\tau_{k-1} \leq C_j \quad \text{for all } j, \tag{3}$$

$$y_{ijk} = 0 \quad \text{for all } i, j, k : r_{ij} > \tau_k \text{ or } s_{ij} > \tau_k, \tag{4}$$

$$y_{ijk} \geq 0 \quad \text{for all } i, j, k.$$

It is easy to see that (1), (2), and (4) are valid inequalities for our problem. To see also that (3) is valid notice that for all k we have that $\tau_{k-1} \leq C_j$ if $y_{ijk} > 0$: this holds by definition of y_{ijk} for $k \geq 2$ and because $C_j \geq 1$ for $k = 1$. Thus (3) follows by (1). Hence, the previous LP is a relaxation to our problem.

Consider an optimal solution $(y_{ijk})_{ijk}$ and $(C_j^{\text{LP}})_j$ of this LP. Our rounding technique consists of two steps. First, since the LP solution can process a fraction of job j at a time much larger than C_j^{LP}, we truncate the solution so that no job finishes later than $\gamma \cdot C_j^{\text{LP}}/(\gamma - 1)$, for some $\gamma > 1$ that will be chosen appropriately. Unfortunately, the resulting fractional solution it is still not a feasible schedule since the LP only reserved a fraction of the needed setup for each piece of a job. To overcome this difficulty, we interpret each interval–machine pair (i, k) as a machine and round using the technique of [3] that studies the split job problem in the minimum makespan setting. It is worth mentioning that both transformations maintain variables $y_{ijk} = 0$ untouched, and thus (4) is always satisfied. Finally we construct the final schedule with a greedy algorithm based on the rounded LP solution.

The next lemma specifies how to truncate the solution given by the LP.

Lemma 11. *Let $\gamma > 1$ and $Y_j = \sum_{i,k:\tau_{k-1} \leq \gamma C_j^{\text{LP}}} y_{ijk}$ for each job j. Then,*

$$y'_{ijk} = \begin{cases} 0 & \text{if } \tau_{k-1} > \gamma \cdot C_j^{\text{LP}}, \\ y_{ijk}/Y_j & \text{if } \tau_{k-1} \leq \gamma \cdot C_j^{\text{LP}}, \end{cases}$$

satisfies (1) and (4). Also, (2) is satisfied if the right-hand-side is multiplied by $\gamma/(\gamma - 1)$.

Proof. The vector y' satisfies (1) and (4) by definition. We will show that for all i, j, k it holds that $y'_{ijk} \leq y_{ijk} \cdot \gamma/(\gamma - 1)$. This is enough to conclude the lemma since y satisfies (2). Notice that $Y_j \geq (\gamma - 1)/\gamma$,

$$\sum_{i,k} y_{ijk}\tau_{k-1} \geq \sum_{i,k:\tau_{k-1} \geq \gamma C_j^{\text{LP}}} y_{ijk} \cdot \gamma \cdot C_j^{\text{LP}} = \gamma C_j^{\text{LP}}(1 - Y_j) > C_j^{\text{LP}},$$

contradicting (3). We conclude that $y'_{ijk} \leq y_{ijk}/Y_j \leq y_{ijk} \cdot \gamma/(\gamma - 1)$ for all i, j, k, which implies the lemma. □

We further round the solution by using the following recent result for $R|\text{split}|C_{\max}$, implicitly proven in [3]. The theorem gives a way of rounding a fractional solution into a fractional solution in which the setups are fully considered, while the total load of the machine is at most twice the original fractional load plus one setup.

Theorem 12 ([3])**.** *Consider an instance of $R|split|C_{\max}$. For any non-negative assignment vector x such that $\sum_i x_{ij} = 1$ for all j, there exists a non-negative solution $\tilde{x}$ such that: $\sum_i \tilde{x}_{ij} = 1$ for each j, if $x_{ij} = 0$ then $\tilde{x}_{ij} = 0$, and for each machine i*

$$\sum_{j:\tilde{x}_{ij}>0} (\tilde{x}_{ij}p_{ij} + s_{ij}) \leq 2\sum_j x_{ij}(p_{ij} + s_{ij}) + \max_{j:x_{ij}>0} s_{ij}.$$

The next step of our algorithm consists in interpreting a machine–interval pair as a machine in the previous theorem and applying it to solution y' given by Lemma 11. This leads to a solution $\tilde{y}$ satisfying for all i, k:

$$\begin{aligned} \sum_{j:\tilde{y}_{ijk}>0} (\tilde{y}_{ijk} \cdot p_{ij} + s_{ij}) &\leq 2\sum_j y'_{ijk}(p_{ij} + s_{ij}) + \max_{ij:y'_{ijk}>0} s_{ij} \\ &\leq 2\sum_j y'_{ijk}(p_{ij} + s_{ij}) + \tau_k, \end{aligned} \tag{5}$$

where the last inequality follows from (4) since $y'_{ijk} > 0$ implies that $y_{ijk} > 0$.

Let $J_{ik} = \{j : \tilde{y}_{ijk} > 0\}$. The final step of the algorithm is the following. For every $k = 1, \ldots, q$ and every machine i, our algorithm takes each job j (in arbitrary order) in J_{ik} and processes its setup time s_{ij} and a fraction of $\tilde{y}_{ijk}$ as early as possible while obeying the release dates.

Theorem 13. *The problem $R|r_{ij}, split|\sum_j w_j C_j$ admits a 19.7864-approximation algorithm.*

Proof. Let $\bar{\tau}_k = 1/(\alpha - 1) + \sum_{\ell=1}^{k}(2\sum_j y'_{ijk}(p_{ij} + s_{ij}) + \tau_\ell)$. We prove that all pieces of jobs corresponding to J_{ik} can be completely processed (including setup) within $[\bar{\tau}_{k-1}, \bar{\tau}_k]$. Indeed, for all $j \in J_{ik}$ we have that $\tilde{y}_{ijk} > 0$ and thus $y_{ijk} > 0$, which in turn implies that

$r_{ij} \le \tau_k$ by (4). Thus, since $1 < \alpha \le 2$, we have that

$$r_{ij} \le \tau_k \le \frac{1}{\alpha-1} + \frac{\tau_k - 1}{\alpha - 1} = \frac{1}{\alpha-1} + \sum_{\ell=1}^{k-1} \tau_\ell \le \bar{\tau}_{k-1}.$$

Therefore all jobs in J_{ik} are available at time $\bar{\tau}_{k-1}$. Moreover, since $\bar{\tau}_k - \bar{\tau}_{k-1} = 2\sum_j y'_{ijk}(p_{ij}+s_{ij}) + \tau_k$, Eq. (5) implies that all the pieces in J_{ik}, together with their setups, can be processed within $[\bar{\tau}_{k-1}, \bar{\tau}_k]$. Hence, all pieces of J_{ik} are finished by time

$$\begin{aligned}\bar{\tau}_k &= \frac{1}{\alpha-1} + \sum_{\ell=1}^{k} 2\left(\sum_j y'_{ijk}(p_{ij}+s_{ij}) + \tau_\ell\right) \\ &\le \frac{1}{\alpha-1} + 2\frac{\gamma}{\gamma-1}\tau_k + \sum_{\ell=1}^{k}\tau_\ell \\ &\le \alpha\left(2\frac{\gamma}{\gamma-1} + \frac{\alpha}{\alpha-1}\right)\tau_{k-1}.\end{aligned}$$

Here, the first inequality follows since by Lemma 11 y' satisfies (2) with the right-hand-side amplified by a factor $\gamma/(\gamma-1)$, and the second by the definition of τ_k. We conclude that all pieces of a job $j \in J_{ik}$ finish by time $\alpha\left(2\frac{\gamma}{\gamma-1} + \frac{\alpha}{\alpha-1}\right)\tau_{k-1}$. Moreover, if $j \in J_{ik}$ then $\tilde{y}_{ijk} > 0$, and thus by Theorem 12 $y'_{ijk} > 0$. By the definition of y' in Lemma 11 this implies that $\tau_{k-1} \le \gamma \cdot C_j^{\mathrm{LP}}$. Therefore job j is completely finished by time $\gamma\alpha(2\frac{\gamma}{\gamma-1} + \frac{\alpha}{\alpha-1})C_j^{\mathrm{LP}}$. Choosing $\alpha = 1.39775$ and $\gamma = 1.60225$ we obtain that each job j finishes by times $19.7864 \cdot C_j^{\mathrm{LP}}$ and thus the overall cost is at most $19.7864\sum_j w_j C_j^{\mathrm{LP}}$. □

The values of α and γ are computed numerically in order to approximately minimize the approximation factor. The exact optimal values cannot be determined analytically since they correspond to the solutions of higher order polynomial equations.

Acknowledgments

This work was supported by Nucleo Milenio Información y Coordinación en Redes ICM/FIC P10-024F, by EU-IRSES grant EUSACOU, and by FONDECYT project No. 11140579.

References

[1] A. Allahverdi, C. Ng, T. Cheng, M. Kovalyov, A survey of scheduling problems with setup times or costs, European J. Oper. Res. 187 (2008) 985–1032.
[2] B. Chen, Y. Ye, J. Zhang, Lot-sizing scheduling with batch setup times, J. Sched. 9 (2006) 299–310.
[3] J.R. Correa, A. Marchetti-Spaccamela, J. Matuschke, O. Svensson, L. Stougie, V. Verdugo, J. Verschae, Strong LP formulations for scheduling splittable jobs on unrelated machines, Math. Program.-Ser. B 154 (2015) 305–328.
[4] J.R. Correa, M. Skutella, J. Verschae, The power of preemption on unrelated machines and applications to scheduling orders, Math. Oper. Res. 37 (2012) 379–398.
[5] L.A. Hall, A.S. Schulz, D.B. Shmoys, J. Wein, Scheduling to minimize average completion time: off-line and on-line approximation algorithms, Math. Oper. Res. 22 (1997) 513–544.
[6] W. Höhn, T. Jacobs, On the performance of Smith's rule in single-machine scheduling with nonlinear cost, ACM Trans. Algorithms 11 (2015) 25.
[7] J.Y.T. Leung, H. Li, M. Pinedo, J. Zhang, Minimizing total weighted completion time when scheduling orders in a flexible environment with uniform machines, Inform. Process. Lett. 103 (2007) 119–129.
[8] N. Megow, J. Verschae, Dual techniques for scheduling on a machine with varying speed, in: Automata, Languages, and Programming, ICALP 2012, in: LNCS, vol. 7965, 2013, pp. 745–756.
[9] C.N. Potts, L.N.V. Wassenhove, Integrating scheduling with batching and lot sizing: A review of algorithms and complexity, J. Oper. Res. Soc. 43 (1992) 395–406.
[10] F. Schalekamp, R. Sitters, S. van der Ster, L. Stougie, V. Verdugo, A. van Zuylen, Split scheduling with uniform setup times, J. Sched. 18 (2014) 119–129.
[11] P. Serafini, Scheduling jobs on several machines with the job splitting property, Oper. Res. 44 (1996) 617–628.
[12] V. Verdugo, Algoritmos de aproximación para la programación de trabajos divisibles con tiempos de instalación en máquinas paralelas (Master's thesis), Department of Industrial Engineering and Department of Mathematical Engineering, University of Chile, 2014.
[13] W. Xing, J. Zhang, Parallel machine scheduling with splitting jobs, Discrete Appl. Math. 103 (2000) 259–269.