The Power of Preemption on Unrelated Machines and Applications to Scheduling Orders*

José R. Correa[1], Martin Skutella[2], and José Verschae[2]

[1] Departamento de Ingeniería Industrial, Universidad de Chile, Santiago, Chile
joser.correa@gmail.com
[2] Institute of Mathematics, TU Berlin, Germany
{skutella,verschae}@math.tu-berlin.de

Abstract. Scheduling jobs on unrelated parallel machines so as to minimize the makespan is one of the basic, well-studied problems in the area of machine scheduling. In the first part of the paper we prove that the power of preemption, i.e., the ratio between the makespan of an optimal nonpreemptive and an optimal preemptive schedule, is exactly 4. This result is a definite answer to an important basic open problem in scheduling. The proof of the lower bound is based on a clever iterative construction while the rounding technique we use to prove the upper bound is an adaptation of Shmoys and Tardos' rounding for the generalized assignment problem. In the second part of the paper we apply this adaptation to the more general setting in which orders, consisting of several jobs, have to be processed on unrelated parallel machines so as to minimize the sum of weighted completion times of the orders. We obtain the first constant factor approximation algorithms for the preemptive and nonpreemptive case, improving and extending a recent result by Leung et. al.

1 Introduction

Problem description and basic results. Consider the classical scheduling problem of minimizing the makespan on unrelated parallel machines. In this problem we are given a set of jobs $J = \{1, \ldots, n\}$ and a set of machines $M = \{1, \ldots, m\}$ to process the jobs. Each job $j \in J$ has associated processing times p_{ij}, denoting the amount of time that it takes to process job j on machine i. Every job has to be scheduled on exactly one machine without interruption and each machine can schedule at most one job at a time. The objective is to find a schedule minimizing the point in time at which the last job is completed, i.e., minimizing

* This work was partially supported by Berlin Mathematical School, by DFG research center Matheon in Berlin, by CONICYT, through grants FONDECYT 1060035 and Anillo en Redes ACT08. The authors thank Nikhil Bansal for stimulating discussions on the material in Section 2.

I. Dinur et al. (Eds.): APPROX and RANDOM 2009, LNCS 5687, pp. 84–97, 2009.

$C_{\max} := \max_{j \in J} C_j$, where C_j is the completion time of job j. In the standard three-field scheduling notation (see, e.g., Lawler et al. [14]) this problem is denoted by $R||C_{\max}$.

In a seminal work, Lenstra, Shmoys and Tardos [16] give a 2-approximation algorithm for $R||C_{\max}$, and show that the problem is NP-hard to approximate within a factor better than 3/2. On the other hand, Lawler and Labetoulle [13] show that the preemptive version of this problem, denoted $R|pmtn|C_{\max}$, where jobs can be interrupted and resumed later on the same or a different machine, can be formulated as a linear program and thus be solved in polynomial time.

Power of preemption. The power of preemption is the worst-case ratio between the makespan of an optimal preemptive and an optimal nonpreemptive solution. This ratio has been studied in the literature for various scheduling problems [4,21,22]. One contribution of this work is to prove that this ratio is exactly 4 for the considered problem on unrelated machines. The proof consists of two steps — proving an upper and a lower bound of 4. For the upper bound, we consider an optimal solution to the linear programming formulation of Lawler and Labetoulle [13] for $R|pmtn|C_{\max}$, and round it to obtain an assignment of jobs to machines in which the makespan is increased at most by a factor of 4. The rounding consists in setting to zero all variables whose corresponding processing time is too large compared to the makespan, and then amplifying the remaining values so that a feasible fractional assignment is maintained. Then, the technique of Shmoys and Tardos [23] is applied to obtain a nonpreemptive solution. The proof of the lower bound is based on a clever recursive construction, where in each iteration the gap of the instance is increased.

Scheduling orders of jobs. In the second part of the paper, we apply the rounding technique used for the previous result to a more general setting. Consider the natural scheduling problem where clients place orders, consisting of several products, to a manufacturer owning m unrelated parallel machines. Each product has a machine dependent processing requirement. The manufacturer has to find an assignment of products to machines (and a schedule within each machine) so as to give the best possible service to his clients.

More precisely, we are given a set of machines $M = \{1, \ldots, m\}$, a set of jobs $J = \{1, \ldots, n\}$ (as before) and a set of orders $O \subseteq 2^J$, such that $\bigcup_{L \in O} L = J$. Each job $j \in J$ takes p_{ij} units of time to be processed in machine $i \in M$, and each order L has a weight factor w_L depending on how important it is for the manufacturer and the client. Also, job j is associated with a release date r_{ij}, so it can only start being processed on machine i by time r_{ij}. An order $L \in O$ is completed once all its jobs have been processed. Therefore, if C_j denotes the time at which job j is completed, $C_L = \max\{C_j : j \in L\}$ denotes the completion time of order L. The goal of the manufacturer is to find a nonpreemptive schedule on the m available machines so as to minimize the sum of weighted completion times of orders, i.e., $\min \sum_{L \in O} w_L C_L$. Let us remark that in this general framework we are not restricted to the case where the orders are disjoint, and therefore one job may contribute to the completion time of more than one order.

We adopt the standard three-field scheduling notation by denoting this problem $R|r_{ij}|\sum w_L C_L$, or $R||\sum w_L C_L$ in case all release dates are zero. When the processing times p_{ij} do not depend on the machine, we replace "R" with "P". Also, when we impose the additional constraint that orders are disjoint subsets of jobs we will add *part* in the second field of the notation.

Relation to other scheduling problems. It is easy to see that this setting generalizes several classical machine scheduling problems. In particular our problem becomes $R||C_{\max}$ when the total number of orders is one. Thus, it follows from [16] that $R||\sum w_L C_L$ cannot be approximated within a factor better than $3/2$, unless $P = NP$. On the other hand, if orders are singletons our problem becomes $R||\sum w_j C_j$. In this setting each job $j \in J$ is associated with a processing time p_{ij} and a weight w_j, and the goal is to find a schedule of the jobs so as to minimize the sum of weighted completion times. In other words, if C_j denotes the completion time of job j in a given schedule, the goal is to minimize $\sum_{j=1}^{n} w_j C_j$. As in the makespan case, this problem was shown to be APX-hard [12] and therefore there is no PTAS, unless $P = NP$. Using randomized rounding techniques based on a linear relaxation, Schulz and Skutella [22] proposed an approximation algorithm for this problem with performance guarantee $3/2 + \varepsilon$ in the case without release dates, and $2 + \varepsilon$ in the more general case. Later, Skutella [25] slightly improved this result by using randomized rounding over a convex cuadratic relaxation, obtaining approximation algorithms with performance guarantee $3/2$ and 2, respectively.

However, for the more general setting $R|r_{ij}|\sum w_L C_L$, there is no constant factor approximation known. The best known result, due to Leung, Li, Pinedo, and Zhang [18], is an approximation algorithm for the special case of related machines without release dates, denoted $Q||\sum w_L C_L$, where $p_{ij} = p_i/s_i$ and s_i is the speed of machine i. The performance ratio of their algorithm is $1 + \rho(m-1)/(\rho + m - 1)$, where ρ is the ratio of the speed of the fastest machine to that of the slowest machine. In general this guarantee is not constant and can be as bad as $m/2$.

Identical parallel machines. For the special case of identical parallel machines, our problem $P||\sum w_L C_L$ also generalizes $P||C_{\max}$ and $P||\sum w_j C_j$. These two problems are well known to be NP-hard, even for the case of only two machines, since the well-known PARTITION problem can be reduced to them. For the makespan objective, Graham [9] showed that a simple list scheduling algorithm yields a 2-approximation algorithm. Furthermore, Hochbaum and Shmoys [11] present a PTAS for the problem. On the other hand, for the sum of weighted completion times objective, a sequence of approximations algorithms had been proposed until Skutella and Woeginger [24] found a PTAS (see also [1]).

On the even more restricted setting of a single machine, the two previously mentioned problems $1||C_{\max}$ and $1||\sum w_j C_j$ can be easily solved, the first one by any feasible solution with no idle time, and the second one by applying *Smith's rule* [26]. However, our problem $1||\sum w_L C_L$ is NP-hard, as it is equivalent to $1|prec|\sum w_j C_j$. In the latter problem, there is a partial order $\preceq$ over the jobs, meaning that job j must be processed before job k if $j \preceq k$.

Lemma 1. *The approximability thresholds of* $1|prec|\sum w_jC_j$ *and* $1||\sum w_LC_L$ *coincide.*

Due to space restrictions, the proof of the lemma is omitted. The scheduling problem $1|prec|\sum w_jC_j$ has attracted much attention since the sixties. Lenstra and Rinnooy Kan [15] showed that this problem is strongly NP-hard even with unit weights. On the other hand, several 2-approximation algorithms have been proposed [10,6,5,20]. Furthermore, the results in [2,7] imply that $1|prec|\sum w_jC_j$ is a special case of *vertex cover*. However, hardness of approximation results were unknown until recently Ambühl, Mastrolilli and Svensson [3] proved that there is no PTAS unless NP-hard problems can be solved in randomized subexponential time. In particular, the same result holds for $P||\sum w_LC_L$. Nonetheless, the reduction used in the previous lemma does not work on the more restrictive case where orders are disjoint, $P|part|\sum w_LC_L$, and thus the question whether there is a PTAS for this latter problem remains open. However, we were able to develope a PTAS for the special cases in which either the orders are of constant size, or there is a constant number of orders, or there is a constant number of machines. Due to space restrictions this result is left for the full version of the paper (see [27] for details).

Our Contribution. Our tight result on the power of preemption for unrelated parallel machine scheduling with makespan objective have already been outlined above. In addition to the result stated in Lemma 1, we present the first constant factor approximation algorithm for the general problem $R|r_{ij}|\sum w_LC_L$ and its preemptive variant $R|r_{ij}, pmtn|\sum w_LC_L$. This is achieved by considering the interval indexed linear programs proposed by Dyer and Wolsey [8] and Hall et al. [10], and then applying essentially the same rounding technique that is used to prove the upper bound on the power of preemption. This approximation result improves upon the previously mentioned result of Leung, Li, Pinedo, and Zhang [18] for the special case $Q||\sum w_LC_L$.

2 A Simple Rounding Technique

We start by showing that the power of preemption for $R||C_{\max}$ is at most 4. As shown by Lawler and Labetoulle [13], we can obtain the optimal value of the preemptive version of this problem by solving the following linear program, whose variables x_{ij} denote the fraction of job j that is processed on machine i, and C the makespan of the solution: [LL] minimize C such that $\sum_{i\in M} x_{ij} = 1$ for all $j \in J$, $\sum_{j\in J} p_{ij}x_{ij} \leq C$ for all $i \in M$, $\sum_{i\in M} p_{ij}x_{ij} \leq C$ for all $j \in J$ and $x_{ij} \geq 0$ for all i, j.

Let x_{ij} and C be any feasible solution to [LL]. To round this fractional solution we proceed in two steps: First, we eliminate fractional variables whose corresponding processing time is too large; Then, we use the rounding technique developed by Shmoys and Tardos [23] for the general assignment problem. In the general assignment problem, we are given m machines and n jobs with machine dependant processing times p_{ij}. We also consider a cost of assigning job j to

machine i, denoted by c_{ij}. Given a total budget B and makespan C, the question is to decide whether there exists a schedule with total cost at most B and makespan at most C. The main result of [23] is subsumed in the next theorem.

Theorem 1 (Shmoys and Tardos [23]). *Given a nonnegative fractional solution to the following system of equations:*

$$\sum_{j\in J}\sum_{i\in M} c_{ij}x_{ij} \le B, \tag{1}$$

$$\sum_{i\in M} x_{ij} = 1, \qquad \text{for all } j \in J, \tag{2}$$

there exists an integral solution $\hat{x}_{ij} \in \{0,1\}$ satisfying (1),(2), and also,

$$x_{ij} = 0 \implies \hat{x}_{ij} = 0 \qquad \text{for all } i \in M, j \in J, \tag{3}$$

$$\sum_{j\in J} p_{ij}\hat{x}_{ij} \le \sum_{j\in J} p_{ij}x_{ij} + \max\{p_{ij} : x_{ij} > 0\} \qquad \text{for all } i \in M. \tag{4}$$

Furthermore, such integral solution can be found in polynomial time.

To proceed with our rounding, let $\beta > 1$ be a fixed parameter that we will specify later. We first define a modified solution x'_{ij} as follows:

$$x'_{ij} = \begin{cases} 0 & \text{if } p_{ij} > \beta C, \\ \frac{x_{ij}}{X_j} & \text{else,} \end{cases} \quad \text{where } X_j = \sum_{i:p_{ij}\le \beta C} x_{ij} \text{ for all } j \in J.$$

Note that,

$$1 - X_j = \sum_{i:p_{ij}>\beta C} x_{ij} < \sum_{i:p_{ij}>\beta C} x_{ij}\frac{p_{ij}}{\beta C} \le 1/\beta,$$

where the last inequality follows from [LL]. Therefore, x'_{ij} satisfies that $x'_{ij} \le x_{ij}\beta/(\beta-1)$ for all $j \in J$ and $i \in M$, and thus $\sum_{j\in J} x'_{ij}p_{ij} \le C\beta/(\beta-1)$ for all $i \in M$. Also, note that by construction $\sum_{i\in M} x'_{ij} = 1$ for all $j \in J$, and $x'_{ij} = 0$ if $p_{ij} > \beta C$. Then, we can apply Theorem 1 to x'_{ij} (for $c_{ij} = 0$), to obtain a feasible integral solution $\hat{x}_{ij}$ to [LL], and thus a feasible solution to $R||C_{\max}$, such that for all $i \in M$,

$$\sum_{j\in J} \hat{x}_{ij}p_{ij} \le \sum_{j\in J} x'_{ij}p_{ij} + \max\{p_{ij} : x_{ij} > 0\} \le \frac{\beta}{\beta-1}C + \beta C = \frac{\beta^2}{\beta-1}C.$$

Therefore, by optimally choosing $\beta = 2$, the makespan of the rounded solution is at most $\beta^2/(\beta-1) = 4$ times larger than the makespan of the fractional solution.

Power of Preemption for $R||C_{\max}$

We now give a family of instances showing that the integrality gap of [LL] is arbitrarily close to 4. Surprisingly, this implies that the rounding technique showed

in the last section is best possible. Note that this is equivalent to saying that the optimal nonpreemptive schedule is within a factor of 4, and no better than 4, of the optimal preemptive schedule.

Let us fix $\beta \in [2,4)$, and $\varepsilon > 0$ such that $1/\varepsilon \in \mathbb{N}$. We now construct an instance $I = I(\beta, \varepsilon)$ such that its optimal nonpreemptive makespan is at most $(1+\varepsilon)C$, and that any nonpreemptive solution of I has makespan at least βC. The construction is done iteratively, maintaining at each iteration a preemptive schedule of makespan $(1+\varepsilon)C$, and where the makespan of any nonpreemptive solution is increased at each step. Due to the equivalence between [LL] and $R|pmtn|C_{\max}$ we can use assignment variables to denote preemptive schedules.

Base Case. We begin by constructing an instance I_0, which will later be our first iteration. To this end consider a set of $1/\varepsilon$ jobs $J_0 = \{j(0;1), j(0;2), \ldots, j(0;1/\varepsilon)\}$ and a set of $1/\varepsilon + 1$ machines $M_0 = \{i(1), i(0;1), \ldots, i(0;1/\varepsilon)\}$. Every job $j(0;\ell)$ can only be processed in machine $i(0;\ell)$, where it takes βC units of time to process, and in machine $i(1)$, where it takes a very short time. More precisely, for all $\ell = 1, \ldots, 1/\varepsilon$ we define,

$$p_{i(0;\ell)j(0;\ell)} := \beta C \text{ and } p_{i(1)j(0;\ell)} := \varepsilon C \frac{\beta}{\beta - 1}.$$

The rest of the processing times are defined as infinity. Note that a feasible fractional assignment is given by setting $x_{i(0;\ell)j(0;\ell)} = 1/\beta$, $x_{i(1)j(0;\ell)} := f_0 := (\beta-1)/\beta$ and setting to zero all other variables. The makespan of this fractional solution is exactly $(1+\varepsilon)C$. Indeed, the load of each machine $i \in M_0$ is exactly C, and the load associated to each job in J_0 equals $C + \varepsilon C$. Furthermore, any nonpreemptive solution with makespan less than βC must process all jobs $j(0;\ell)$ in $i(1)$. This yields a makespan of $C/f_0 = \beta C/(\beta-1)$. Therefore, the makespan of any nonpreemptive solution is $\min\{\beta C, C/f_0\}$. If β is chosen as 2, the makespan of any nonpreemptive solution must be at least $2C$, and therefore the gap of the instance tends to 2 when ε tend to zero.

Iterative Procedure. To increase the integrality gap we proceed iteratively as follows. Starting from instance I_0, which will be the base case, we show how to construct instance I_1. An analogous procedure can be used to construct instance I_{n+1} from instance I_n.

Begin by making $1/\varepsilon$ copies of instance I_0, I_0^ℓ for $\ell = 1, \ldots, 1/\varepsilon$, and denote the set of jobs and machines of I_0^ℓ as J_0^ℓ and M_0^ℓ respectively. We impose that jobs in J_0^ℓ can only be processed on machines in M_0^ℓ by setting $p_{ij} = \infty$, for all $j \in J_0^\ell$ and $i \in M_0^k$ such that $k \neq \ell$. Also, denote as $i(1;\ell)$ the copy of machine $i(1)$ belonging to M_0^l. Consider a new job $j(1)$ for which $p_{i(1;\ell)j(1)} = C\beta - C/f_0$ for all $\ell = 1, \ldots, 1/\varepsilon$ (and ∞ otherwise), and define $x_{i(1;\ell)j(1)} = \varepsilon C/p_{i(1;\ell)j(1)}$. This way, the load of each machine $i(1;\ell)$ in the fractional solution is $(1+\varepsilon)C$, and the load corresponding to job $j(1)$ is exactly C. Nevertheless, depending on the value of β, job $j(1)$ may not be completely assigned. A simple calculation shows that for $\beta = (3+\sqrt{5})/2$, job $j(1)$ is completely assigned in the fractional

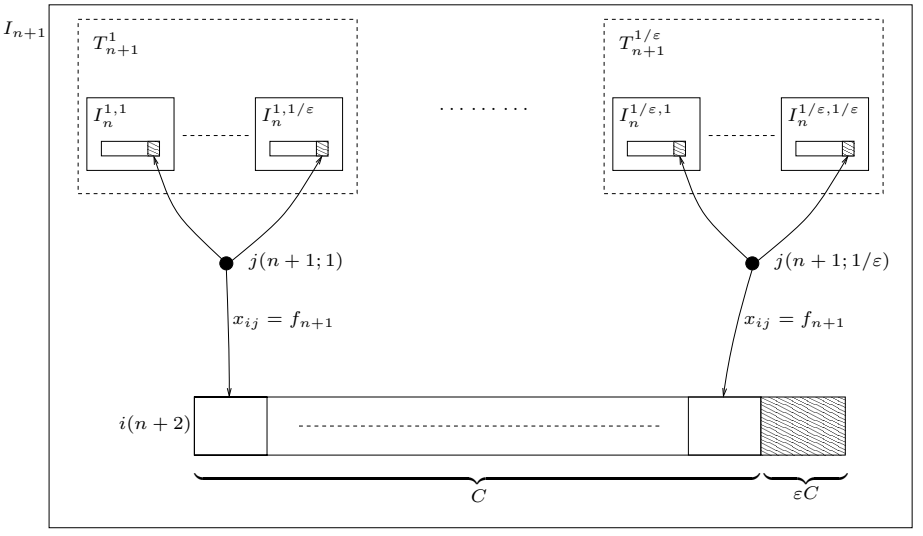

Fig. 1. Construction of instance $I_{n+1}(\beta)$

assignment. Furthermore, as justified before, in any nonpreemptive schedule of makespan less than βC, all jobs in J_0^l must be processed in machine $i(1;\ell)$. Since also job $j(1)$ must be processed in some machine $i(1;\ell)$, the load of that machine must be $\sum_{j\in J_0^\ell} p_{i(1;\ell)j} + p_{i(1;\ell)j(1)} = C/f_0 + C(\beta - 1/f_0) = \beta C$. Then, the gap of the instance already constructed converges to $\beta = (3+\sqrt{5})/2 \approx 2.618$ when ε tend to 0, thus improving the gap of 2 shown before.

On the other hand, for $\beta > (3+\sqrt{5})/2$ (as we would like) there will be some fraction of job $j(1)$, $f_1 := 1 - \sum_{\ell=1}^{1/\varepsilon} x_{i(1;\ell)j(1)} = ((\beta-1)f_0 - 1)/(\beta f_0 - 1)$ that must be processed elsewhere. To overcome this, we do as follows. Let us denote the instance consisting of jobs $\bigcup_{\ell=1}^{1/\varepsilon} J_0^l$ and machines $\bigcup_{\ell=1}^{1/\varepsilon} M_0^\ell$ as T_1, and construct $1/\varepsilon$ copies of instance T_1, T_1^k for $k = 1,\ldots,1/\varepsilon$. Define the processing times of jobs in T_1^ℓ to infinity in all machines of T_1^k, for all $k \neq \ell$, so that jobs of T_1^ℓ can only be processed in machines of T_1^ℓ. Also, consider $1/\varepsilon$ copies of job $j(1)$, and denote them by $j(1;k)$ for $k = 1,\ldots,1/\varepsilon$. As shown before, we can assign a fraction $1 - f_1$ of each job $j(1;k)$ to machines of T_1^k. To assign the remaining fraction f_1, we add an extra machine $i(2)$, with $p_{i(2)j(1;\ell)} := \varepsilon C/f_1$ (and ∞ for all other jobs), so that the fraction f_1 of each job $j(1;\ell)$ takes exactly εC to process in $i(2)$. Then, defining $x_{i(2)j(1;\ell)} = f_1$, the total load of each job $j(1;\ell)$ equals $(1+\varepsilon)C$, while the load of machine $i(2)$ is exactly C. Let us denote the instance we have constructed so far as I_1.

Following an analogous procedure to the one just described, we can construct a sequence of instances and fractional assignments (see Figure 1). Each instance I_n satisfies the following properties:

(i) The fraction of each job $j(n;1),\ldots,j(n,1/\varepsilon)$ assigned to machine $i(n+1)$ is given by $f_n = ((\beta-1)f_{n-1}-1)/(\beta f_{n-1}-1)$.
(ii) Job $j(n+1)$ (or any of its copies) has processing time equal to $C(\beta - 1/f_n)$ on each machine $i(n;\ell)$.
(iii) In any nonpreemptive solution of makespan less than βC, every job $j(n+1;\ell)$ must be processed in machine $i(n+2)$. Therefore the makespan of any nonpreemptive solution is at least $\min\{\beta C, C/f_{n+1}\}$.
(iv) The makespan of the fractional solution constructed is $(1+\varepsilon)C$. In particular the load of machine $i(n+2)$ is C, and therefore a fraction of a job which takes less than εC can still be processed in this machine without increasing the makespan.

To finish the construction procedure, notice that if there is some n^* such that $f_{(n^*-1)} \leq 1/(\beta-1)$, then there is no need to construct the whole instance I_{n^*}, but rather instance T_{n^*} suffices. Indeed, if this is the case job $j(n^*)$ can be totally assigned to machines $i(n^*;\ell)$ on the fractional solution, by defining $x_{i(n^*;\ell)j(n^*)} = \varepsilon$ for all $\ell = 1,\ldots,1/\varepsilon$. This yields a valid assignment since $\sum_{\ell=1}^{1/\varepsilon} p_{i(n^*;\ell)j(n^*)} x_{i(n^*;\ell)j(n^*)} = C(\beta - 1/f_{n^*-1}) \leq C$. Also, by Property (iii), any nonpreemtpive solution of makespan less than βC assigns a load of C/f_{n^*-1} to any machine $i(n^*;\ell)$. Furthermore, job $j(n^*)$ must be processed in some machine $i(n^*;\ell)$, which will have a makespan of $C/f_{n^*-1} + (\beta C - C/f_{n^*-1}) = \beta C$. With this we have sketched the proof of the following lemma.

Lemma 2. *If the procedure finishes, then it returns an instance with a gap of at least* $\beta/(1+\varepsilon)$.

Then, we just need to show that the construction terminates, i.e., that $f_{n^*-1} \leq 1/(\beta-1)$ for some n^*. For that, notice the following.

Lemma 3. *For each* $\beta \in [2,4)$, *if* $f_n > 1/\beta$, *then* $f_{n+1} \leq f_n$.

Lemma 4. *The procedure finishes.*

Proof. If the procedure does not finish, then $f_n > 1/(\beta-1) > 1/\beta$ for all $n \in \mathbb{N}$. Then Lemma (3) implies that $\{f_n\}_{n\in\mathbb{N}}$ is a decreasing sequence. Therefore f_n must converge to some real number $L \geq 1/(\beta-1)$. Thus, Property (i) implies that $L = ((\beta-1)L-1)/(\beta L-1)$, and therefore L is a real root of equation $-\beta x^2 + \beta x - 1$ which is a contradiction if $\beta \in [2,4)$. □

Theorem 2. *The integrality gap of relaxation* $[LL]$ *is 4.*

3 A $(4+\varepsilon)$-Approximation for $R|r_{ij}, pmtn|\sum w_L C_L$

In this section we adapt the rounding technique discussed in the previous chapter to derive a $(4+\varepsilon)$-approximation algorithm for the preemptive version of $R|r_{ij}|\sum w_L C_L$. Our algorithm is based on a time-indexed linear program, whose variables correspond to the fraction of each job processed at each time in each

machine. This kind of linear relaxation was originally introduced by Dyer and Wolsey [8] for $1|r_j|\sum w_jC_j$, and was extended by Schulz and Skutella [22], who used it to obtain a $(3/2+\varepsilon)$-approximation and a $(2+\varepsilon)$-approximation for $R||\sum w_jC_j$ and $R|r_j|\sum w_jC_j$ respectively.

Let us consider a time horizon T, large enough so it upper bounds the greatest completion time of any reasonable schedule, for instance $T = \max_{i\in M,k\in J}\{r_{ik} + \sum_{j\in J} p_{ij}\}$. We divide the time horizon into exponentially-growing time intervals, so that there is only polynomially many of them. For that, let ε be a fixed parameter, and let q be the first integer such as $(1+\varepsilon)^{q-1} \geq T$. Then, we consider the intervals $[0,1], (1,(1+\varepsilon)], ((1+\varepsilon),(1+\varepsilon)^2], \ldots, ((1+\varepsilon)^{q-2},(1+\varepsilon)^{q-1}]$. To simplify the notation, let us define $\tau_0 = 0$, and $\tau_\ell = (1+\varepsilon)^{\ell-1}$, for each $\ell = 1 \ldots q$. With this, the ℓ-th interval corresponds to $(\tau_{\ell-1}, \tau_\ell]$. In what follows we will assume, without loss of generality, that all processing times are positive integers.

Given any preemptive schedule, let $y_{ij\ell}$ the fraction of job j that is processed in machine i in the ℓ-th interval. Then, $p_{ij}y_{ij\ell}$ is the amount of time that job j is processed in machine i in the ℓ-th interval. With this interpretation is easy to see that the following linear program is a relaxation of $R|r_{ij}, pmtn|\sum w_LC_L$:

$$\text{[DW]} \qquad \min \sum_{L\in O} w_L C_L$$

$$\sum_{i\in M}\sum_{\ell=1}^{q} y_{ij\ell} = 1 \qquad \text{for all } j \in J, \quad (5)$$

$$\sum_{j\in J} p_{ij}y_{ij\ell} \leq \tau_\ell - \tau_{\ell-1} \quad \text{for all } \ell = 1,\ldots,q \text{ and } i \in M, \quad (6)$$

$$\sum_{i\in M} p_{ij}y_{ij\ell} \leq \tau_\ell - \tau_{\ell-1} \quad \text{for all } \ell = 1,\ldots,q \text{ and } j \in J, \quad (7)$$

$$\sum_{i\in M}\left(y_{ij1} + \sum_{\ell=2}^{q} \tau_{\ell-1}y_{ij\ell}\right) \leq C_L \qquad \text{for all } L \in O \text{ and } j \in L, \quad (8)$$

$$y_{ij\ell} = 0 \qquad \text{for all } j,i,\ell : r_{ij} > \tau_\ell, \quad (9)$$

$$y_{ij\ell} \geq 0 \qquad \text{for all } i,j,\ell. \quad (10)$$

Let $y^*_{ij\ell}$ and C^*_L be the optimal solution of [DW]. Using the same ideas as in Section 2, we round this solution by taking to zero all variables $y^*_{ij\ell}$ having a coefficient that is too large in (8), and then rescale to obtain a feasible assignment. Then, we use the result in [13], to construct a feasible preemptive schedule inside each interval. More precisely, let $j \in J$, and $L = \arg\min\{C^*_{L'} | j \in L' \in O\}$. For each parameter $\beta > 1$, we define:

$$y'_{ij\ell} = \begin{cases} 0 & \text{if } \tau_{\ell-1} > \beta C^*_L, \\ \frac{y^*_{ij\ell}}{Y_j} & \text{else,} \end{cases} \quad \text{where } Y_j = \sum_{i\in M} \sum_{\ell:\, \tau_{\ell-1}\leq \beta\cdot C^*_L} y^*_{ij\ell}. \quad (11)$$

Lemma 5. *The modified solution y' obtained by applying Equation* (11) *to y^*, satisfies Equation* (5)*. Furthermore, $y'_{ij\ell} = 0$ if $\tau_{\ell-1} > \beta C^*_L$, for all $L \in O$ and $j \in L$, and y' satisfies equations* (6) *and* (7) *when their righthand sides are amplified by a factor of $\beta/(\beta-1)$.*

The proof of the lemma follows the ideas of the rounding in Section 2. Note that since y' only satisfy equations (6) and (7) when their righthand side are amplified, the amount of load assign to each interval may not fit in the available space. Thus, we will have to increase the size of every interval in a factor $\beta/(\beta-1)$. Furthermore, the variables $y'_{ij\ell}$ only assign jobs to intervals that start before βC^*_L in case $j \in L$, allowing us to easily bound the cost of the solution. With the latter observations, we are ready to describe the algorithm.

Algorithm: Greedy Preemptive LP

1. Solve [DW] to optimality and call the solution y^* and $(C^*_L)_{L\in O}$.
2. Define $y'_{ij\ell}$ using Equation (11).
3. Construct a preemptive schedule S as follows.
   (a) For each $\ell = 1, \ldots, q$, define $x_{ij} = y'_{ij\ell}$ and $C_\ell = (\tau_\ell - \tau_{\ell-1})\beta/(\beta-1)$, and apply the algorithm by Lawler and Labetoulle [13] to this fractional solution, to obtain a preemptive schedule (i.e., no job is processed in parallel by two machines) of makespan C_ℓ. Call the preemptive schedule obtained S_ℓ.
   (b) For each job $j \in J$ that is processed by schedule S_ℓ at time $t \in [0, C_\ell]$ in machine $i \in M$, make schedule S process j in machine i at time $\tau_{\ell-1}\beta/(\beta-1) + t$.

Theorem 3. Algorithm: Greedy Preemptive LP *yields a feasible schedule where the completion time of each order $L \in O$ is less than $C^*_L(1+\varepsilon)\beta^2/(\beta-1)$. Moreover, for $\beta = 2$, the algorithm is a $(4+\varepsilon)$-approximation for the preemptive version of $R|r_{ij}|\sum w_L C_L$.*

4 A Constant Factor Approximation for $R|r_{ij}|\sum w_L C_L$

In this section we propose the first constant factor approximation algorithm for the nonpreemptive version of the problem just described, $R|r_{ij}|\sum w_L C_L$, improving the results in [18]. Our algorithm consists on applying the rounding shown in Section 2 to an adaptation of the interval-index linear programming relaxation developed by Hall, Schulz, Shmoys and Wein [10].

Let us consider a large enough time horizon T as in last section. We divide the time horizon into exponentially-growing time intervals, so that there is only polynomially many. For that, let $\alpha > 1$ be a parameter which will determine later and let q be the first integer such as $\alpha^{q-1} \geq T$. With this, consider the intervals $[1,1], (1,\alpha], (\alpha,\alpha^2], \ldots, (\alpha^{q-2}, \alpha^{q-1}]$.

To simplify the notation, let us define $\tau_0 = 1$ and $\tau_\ell = \alpha^{\ell-1}$ for each $\ell = 1, \ldots, q$. With this, the ℓ-th interval corresponds to $(\tau_{\ell-1}, \tau_\ell]$. Note that, for

technical reasons, these definitions slightly differ from the ones on the previous section.

To model the scheduling problem we consider the variables $y_{ij\ell}$, indicating whether job j is finished in the machine i and in the ℓ-th interval. These variables allow us to write the following linear program based on that in [10], which is a relaxation of the scheduling problem even when integrality constraints are imposed,

$$\text{[HSSW]} \qquad \min \sum_{L \in O} w_L C_L$$

$$\sum_{i \in M} \sum_{\ell=1}^{q} y_{ij\ell} = 1 \qquad \text{for all } j \in J, \tag{12}$$

$$\sum_{s=1}^{\ell} \sum_{j \in J} p_{ij} y_{ijs} \leq \tau_\ell \qquad \text{for all } i \in M \text{ and } \ell = 1, \ldots, q, \tag{13}$$

$$\sum_{i \in M} \sum_{\ell=1}^{q} \tau_{\ell-1} y_{ij\ell} \leq C_L \qquad \text{for all } L \in O \text{ and } j \in L, \tag{14}$$

$$y_{ij\ell} = 0 \qquad \text{for all } i, \ell, j : p_{ij} + r_{ij} > \tau_\ell, \tag{15}$$

$$y_{ij\ell} \geq 0 \qquad \text{for all } i, j, \ell. \tag{16}$$

It is clear that [HSSW] is a relaxation of our problem. Indeed, (12) guarantees that each job finishes in some time interval. The left hand side of (13) corresponds to the total load processed on machine i and interval $[0, \tau_\ell]$, and therefore the inequality is valid. The sum in inequality (14) corresponds exactly to $\tau_{\ell-1}$, where ℓ is the interval where job j finishes, so that is at most C_j, and therefore it is upper bounded by C_L if $j \in L$. Also, it is clear that (15) must hold since no job j can finish processing on machine i before $p_{ij} + r_{ij}$.

Let $(y^*_{ij\ell})_{ij\ell}$ and $(C^*_L)_L$ be an optimal solution to [HSSW]. To obtain a feasible schedule we need to round such solution into an integral one. To this end, Hall et. al. [10] used Shmoys and Tardos' result given in Theorem 1. If in [HSSW] all orders are singleton (as in Hall et al's situation), (14) becomes an equality so that one can use Theorem 1 to round a fractional solution to an integral solution of smaller total cost and such that the righthand side of equation (13) is increased to $\tau_\ell + \max\{p_{ij} : y_{ij\ell} > 0\} \leq 2\tau_\ell$, where the last inequality follows from (15). This can be used to derive a constant factor approximation algorithm for the problem. In our setting however, it is not possible to apply Theorem 1 directly, due to the nonlinearity of the objective function. To overcome this difficulty, consider $j \in J$ and $L = \arg\min\{C^*_{L'} | j \in L' \in O\}$, and apply (11) to y^*, thus obtaining a new fractional assignment y'. With this we obtain a solution in which job j is never assigned to an interval starting after βC^*_L. Moreover, the following lemma holds.

Lemma 6. *The modified solution $y'_{ij\ell} \geq 0$ satisfies* (12), (15), *and:*

$$\sum_{s=1}^{\ell} \sum_{j \in J} p_{ij} y'_{ijs} \leq \frac{\beta}{\beta - 1} \tau_\ell \qquad \text{for all } i \in M, \tag{17}$$

$$y'_{ij\ell} = 0 \qquad \text{if } \tau_{\ell-1} > \beta C^*_L, \text{ for all } i, j, \ell, L : j \in L. \tag{18}$$

With the previous lemma on hand we are in position to apply Theorem 1 by interpreting a machine-interval pair (i, ℓ) on [HSSW] as a virtual machine on the theorem. We thus obtain a rounded solution $\hat{y}_{ij\ell} \in \{0, 1\}$ satisfying (12), (15), (18) and

$$\sum_{j \in J} p_{ij} \hat{y}_{ij\ell} \leq \sum_{j \in J} p_{ij} y'_{ij\ell} + \max_{j \in J} \{p_{ij} : y'_{ij\ell} > 0\} \leq \sum_{j \in J} p_{ij} y'_{ij\ell} + \tau_\ell, \tag{19}$$

where the first inequality follows from (4) and the second follows since y' satisfies (15).

To obtain a feasible schedule we do as follows. Define $J_{i\ell} = \{j \in J : \hat{y}_{ij\ell} = 1\}$, and greedily schedule in each machine i all jobs in $\bigcup_{\ell=1}^{q} J_{i\ell}$, starting from those in J_{i1} until we reach J_{iq} (with an arbitrary order inside each set $J_{i\ell}$), respecting the release dates. Let us call the algorithm just described Greedy-LP.

For simplicity, we only show that Greedy-LP is a constant factor approximation algorithm for the case in which all release dates are zero. The case with nontrivial release dates follows from a similar argument.

Theorem 4. *Procedure* Greedy-LP *is a (27/2)-approximation algorithm for* $R||\sum w_L C_L$.

Proof. Let us fix a machine i and take a job $j \in L$ such that $\hat{y}_{ij\ell} = 1$, so that $j \in J_{i\ell}$. Clearly, C_j, the completion time of job j in algorithm Greedy-LP, is at most the total processing time of jobs in $\bigcup_{k=1}^{\ell} J_{ik}$. Then,

$$\begin{aligned} C_j &\leq \sum_{s=1}^{\ell} \sum_{k \in J} p_{ik} \hat{y}_{iks} \leq \sum_{s=1}^{\ell} \left(\sum_{k \in J} p_{ik} y'_{iks} + \tau_s \right) \leq \frac{\beta}{\beta - 1} \tau_\ell + \sum_{s=1}^{\ell} \tau_s \\ &\leq \left(\frac{\beta\alpha}{\beta - 1} + \frac{\alpha^2}{\alpha - 1} \right) \tau_{\ell-1} \leq \beta\alpha \left(\frac{\beta}{\beta - 1} + \frac{\alpha}{\alpha - 1} \right) C^*_L. \end{aligned}$$

The second inequality follows from (19), the third from (17), and the fourth follows from the definition of τ_k. The last inequality follows since, by condition (3), $\hat{y}_{ij\ell} = 1$ implies $y'_{ij\ell} > 0$, so that by (18) we have $\tau_{\ell-1} \leq \beta C^*_L$. Optimizing over the approximation factor, the best possible guarantee given by this method is attained at $\alpha = \beta = 3/2$, and thus we conclude that $C_j \leq 27/2 \cdot C^*_L$ for all $L \in O$ and $j \in L$. □

Theorem 5. Greedy-LP *is a (27/2)-approximation for* $R|r_{ij}|\sum w_L C_L$.

5 Further Results

Beyond the results shown in this paper, we have also considered the problem $P|part|\sum w_L C_L$, where no job can simultaneously belong to more than one order. Following Afrati et al. [1], we were able to develope a PTAS for some restricted versions of this problem, namely, when the number of jobs in each order is constant, the number of machines is constant, or the number of orders is constant. Thus, our algorithm generalizes the known PTAS's in [1,11,24]. The main extra difficulty compared to the case in [1], is that we might have orders that are processed through a long period of time, and their costs are only realized when they are completed. To overcome this issue, and thus be able to apply the dynamic programming ideas of Afrati et al., we simplify the instance and prove that there is a near-optimal solution in which every order is fully processed in a restricted time span. This requires some careful enumeration plus the introduction of artificial release dates. Due to space restrictions this result is left for the full version of this paper (see [27] for details).

References

1. Afrati, F., Bampis, E., Chekuri, C., Karger, D., Kenyon, C., Khanna, S., Milis, I., Queyranne, M., Skutella, M., Stein, C., Sviridenko, M.: Approximation schemes for minimizing average weighted completion time with release dates. In: Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 32–43 (1999)
2. Ambühl, C., Mastrolilli, M.: Single Machine Precedence Constrained Scheduling is a Vertex Cover Problem. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 28–39. Springer, Heidelberg (2006)
3. Ambühl, C., Mastrolilli, M., Svensson, O.: Inapproximability Results for Sparsest Cut, Optimal Linear Arrangement, and Precedence Constrained Scheduling. In: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 329–337 (2007)
4. Canetti, R., Irani, S.: Bounding the Power of Preemption in Randomized Scheduling. SIAM J. Computing 27, 993–1015 (1998)
5. Chekuri, C., Motwani, R.: Precedence constrained scheduling to minimize sum of weighted completion times on a single machine. Discrete Applied Mathematics 98, 29–38 (1999)
6. Chudak, F., Hochbaum, D.S.: A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. Oper. Res. Let. 25, 199–204 (1999)
7. Correa, J.R., Schulz, A.S.: Single Machine Scheduling with Precedence Constraints. Math. Oper. Res. 30, 1005–1021 (2005)
8. Dyer, M.E., Wolsey, L.A.: Formulating the single machine sequencing problem with release dates as a mixed integer program. Discrete Applied Mathematics 26, 255–270 (1999)
9. Graham, R.L.: Bounds for certain multiprocessing anomalies. Bell Systems Technical Journal 45, 1563–1581 (1966)
10. Hall, L.A., Schulz, A.S., Shmoys, D.B., Wein, J.: Scheduling to minimize average completion time: off-line and on-line approximation algorithms. Math. Oper. Res. 22, 513–544 (1997)

11. Hochbaum, D., Shmoys, D.: Using dual approximation algorithm for scheduling problems: Theoretical and practical results. J. ACM 34, 144–162 (1987)
12. Hoogeveen, H., Schuurman, P., Woeginger, G.J.: Non-approximability results for scheduling problems with minsum criteria. INFORMS J. Computing 13, 157–168 (2001)
13. Lawler, E.L., Labetoulle, J.: On Preemptive Scheduling of Unrelated Parallel Processors by Linear Programming. J. ACM 25, 612–619 (1978)
14. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B.: Sequencing and scheduling: Algorithms and complexity. In: Graves, S.C., Rinnooy Kan, A.H.G., Zipkin, P.H. (eds.) Logistics of Production and Inventory, Handbooks in Oper. Res. and Management Science, vol. 4, pp. 445–522. North-Holland, Amsterdam (1993)
15. Lenstra, J.K., Rinnooy Kan, A.H.G.: Complexity of scheduling under precedence constrains. Operations Research 26, 22–35 (1978)
16. Lenstra, J.K., Shmoys, D.B., Tardos, E.: Approximation algorithms for scheduling unrelated parallel machines. Mathematical Programming 46, 259–271 (1990)
17. Leung, J., Li, H., Pinedo, M.: Approximation algorithm for minimizing total weighted completion time of orders on identical parallel machines. Naval Research Logistics 53, 243–260 (2006)
18. Leung, J., Li, H., Pinedo, M., Zhang, J.: Minimizing Total Weighted Completion Time when Scheduling Orders in a Flexible Environment with Uniform Machines. Information Processing Letters 103, 119–129 (2007)
19. Leung, J., Li, H., Pinedo, M.: Scheduling orders for multiple product types to minimize total weighted completion time. Discrete Applied Mathematics 155, 945–970 (2007)
20. Margot, F., Queyranne, M., Wang, Y.: Decompositions, network flows, and a precedence constrained single machine scheduling problem. Operations Research 51, 981–992 (2003)
21. Shachnai, H., Tamir, T.: Multiprocessor Scheduling with Machine Allotment and Parallelism Constraints. Algorithmica 32, 651–678 (2002)
22. Schulz, A., Skutella, M.: Scheduling unrelated machines by randomized rounding. SIAM J. Discrete Math. 15, 450–469 (2002)
23. Shmoys, D.B., Tardos, E.: An approximation algorithm for the generalized assignment problem. Mathematical Programming 62, 561–574 (1993)
24. Skutella, M., Woeginger, G.J.: Minimizing the total weighted completion time on identical parallel machines. Math. Oper. Res. 25, 63–75 (2000)
25. Skutella, M.: Convex quadratic and semidefinite programming relaxations in scheduling. J. ACM 48, 206–242 (2001)
26. Smith, W.E.: Various optimizers for single-stage production. Naval Research Logics Quarterly 3, 59–66 (1956)
27. Verschae, J.: Approximation algorithms for scheduling orders on parallel machines. Mathematical engineering thesis. Universidad de Chile, Santiago, Chile (2008)