LP Rounding and an Almost Harmonic Algorithm for Scheduling with Resource Dependent Processing Times*

Alexander Grigoriev[1], Maxim Sviridenko[2], and Marc Uetz[1]

[1] Maastricht University, Quantitative Economics, P.O.Box 616, 6200 MD Maastricht, The Netherlands. Email: {a.grigoriev, m.uetz}@ke.unimaas.nl
[2] IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA. Email: sviri@us.ibm.com

Abstract We consider a scheduling problem on unrelated parallel machines with the objective to minimize the makespan. In addition to its machine dependence, the processing time of any job is dependent on the usage of a scarce renewable resource, e.g. workers. A given amount of that resource can be distributed over the jobs in process at any time. The more of the resource is allocated to a job, the smaller is its processing time. This model generalizes the classical unrelated parallel machine scheduling problem by adding a time-resource tradeoff. It is also a natural variant of a generalized assignment problem studied by Shmoys and Tardos. On the basis of an integer linear programming formulation for (a relaxation of) the problem, we adopt a randomized LP rounding technique from Kumar et al. (FOCS 2005) in order to obtain a deterministic, integral LP solution that is close to optimum. We show how this rounding procedure can be used to derive a deterministic 3.75-approximation algorithm for the scheduling problem. This improves upon previous results, namely a deterministic 6.83-approximation, and a randomized 4-approximation. The improvement is due to the better LP rounding and a new scheduling algorithm that can be viewed as a restricted version of the harmonic algorithm for bin packing.

1 Introduction

Unrelated parallel machine scheduling to minimize the makespan, $R||C_{\max}$ in the three-field notation of Graham et al. [3], is one of the classical problems in combinatorial optimization. Given are n jobs that have to be scheduled on m parallel machines, and the processing time of job j if processed on machine i is p_{ij}. The goal is to minimize the latest job completion, the makespan $C_{\max}$. If the number of machines m is part of the input, the best approximation algorithm to date is a 2-approximation by Lenstra, Shmoys and Tardos [11]. Moreover, the

* This work was done while the second author was visiting Maastricht University, supported by METEOR, the Maastricht Research School of Economics of Technology and Organizations.

problem cannot be approximated within a factor strictly smaller than 3/2, unless P=NP [11].

In this paper, we consider a generalization of the unrelated parallel machine scheduling problem $R||C_{\max}$ by adding a *time-resource* tradeoff. This generalization also involves a scarce renewable resource (e.g., workers) that can be used in order to speed up the processing times of the jobs. This generalization has recently been studied by Grigoriev et al. [4] and Kumar et al. [10]; it can be seen also as a variant of the unrelated machine scheduling problem with budget constraint that was considered by Shmoys and Tardos [13]. More precisely, a maximum number of k units of a resource is available at any time. It may be used to speed up the jobs, and the available amount of k units of that resource must not be exceeded at any time. In contrast to the linearity assumption of the relation of processing times and costs in [13], the only assumption we make in this paper is that the processing times p_{ijs}, which now depend also on the number s of allocated resources, are non-increasing in s for each job-machine pair. That is, we assume that $p_{ij0} \geq p_{ij1} \geq \cdots \geq p_{ijk}$ for all jobs j and all machines $i = 1, \ldots, m$.

As a matter of fact, machine scheduling problems with the additional feature of a *nonrenewable* resource constraint, such as a total budget constraint, have received quite some attention in the literature as *time-cost tradeoff* problems. To give a few references, see [1,7,8,13,14]. Surprisingly, time-resource tradeoff problems with a *renewable* resource constraint, such as personnel, seem to have received less attention, although they are not less appealing from a practical viewpoint. Apart from our previous paper [4], some (restricted) versions of the problem were considered in [5,6,9,12,15].

In this work we improve upon our previous deterministic 6.83-approximation from [4], and upon a randomized 4-approximation by Kumar, Marathe, Parthasarathy, and Srinivasan [10]. The main result of the paper is a deterministic 3.75-approximation algorithm for the unrelated machine scheduling problem with resource dependent processing times. Compared to our previous approach from [4], the improvement is obtained by using two new ingredients. The first ingredient is a more sophisticated LP rounding that can be seen as a derandomized version of a randomized rounding approach by Kumar et al. [10]. The second ingredient is a new scheduling algorithm that resembles (a restricted version of) the well-known harmonic algorithm for bin packing.

In fact, the new rounding procedure can be viewed as an extension of the Shmoys and Tardos rounding theorem for the generalized assignment problem [13]. In this extension we consider the generalized assignment problem on a bipartite *multigraph* instead of a simple bipartite graph. Note that the techniques from [13] do not seem to be extendible to the case of a multigraph.

Combining the greedy scheduling algorithm from our previous paper [4] with the new rounding theorem yields a deterministic 4-approximation algorithm for scheduling with resource dependent processing times; matching the corresponding result of Kumar et al. [10]. But using a more sophisticated linear programming relaxation that still fits the framework for the rounding procedure, com-

bined with (a restricted version of) the harmonic algorithm for the classical bin packing problem, we design an even better 3.75-approximation algorithm. In particular, this considerably improves upon our previous 6.83-approximation [4].

2 Problem definition

Let $V = \{1, \dots, n\}$ be a set of jobs. Jobs must be processed non-preemptively on a set $M = \{1, \dots, m\}$ of unrelated parallel machines. The objective is to find a schedule that minimizes the makespan $C_{\max}$, that is, the time of the last job completion. During its processing, a job j may be assigned an amount $s \in \{0, 1, \dots, k\}$ of an additional resource, for instance additional workers, that may speed up its processing. If s resources are allocated to a job j, and the job is processed on machine i, the processing time of that job is p_{ijs}. The only assumption on the processing times, regarding their dependence on the amount of allocated resources, is monotonicity. That is, we assume that

$$p_{ij0} \geq p_{ij1} \geq \cdots \geq p_{ijk}$$

for every machine-job pair (i, j). Without loss of generality, we also assume that all processing times p_{ijs} are integral. Hence, we can restrict to feasible schedules where the jobs only start (and end) at integral points in time.

The allocation of resources to jobs is restricted as follows. At any time, no more than the available k units of the resource may be allocated to the set of jobs in process. Moreover, since we assume a discrete resource, the amount of resources assigned to any job must be integral, and we require it to be the same along its processing. In other words, if $\ell \leq k$ units of the resource are allocated to some job j, t_j and t'_j denote j's starting and completion time, respectively, only $k - \ell$ of the resources are available for other jobs between t_j and t'_j.

We finally introduce an additional piece of notation. Since we do not assume that the functions p_{ijs} are *strictly* decreasing in s, the only information that is effectively required is the *breakpoints* of p_{ijs}, that is, indices s where $p_{ijs} < p_{ij,s-1}$. Hence, define the 'relevant' indices for job j on machine i as

$$S_{ij} = \{0\} \cup \{s \mid s \leq k,\ p_{ijs} < p_{ij,s-1}\} \subseteq \{0, \dots, k\}\,.$$

Considering this index sets obviously suffices, since in any solution, if s units of the resource are allocated to some job j, we may as well only use s' units, where $s' \leq s$ and $s' \in S_{ij}$, without violating feasibility.

3 LP-based approximations for unrelated parallel machines

Integer programming relaxation. Let x_{ijs} denote binary variables, indicating that an amount of s resources is used for processing job j on machine i. Then consider

the following integer linear program, referred to as (IP).

$$\sum_{i\in M}\sum_{s\in S_{ij}} x_{ijs} = 1 \ , \qquad \forall\, j \in V\,, \tag{1}$$

$$\sum_{j\in V}\sum_{s\in S_{ij}} x_{ijs}\, p_{ijs} \le C \ , \qquad \forall\, i \in M\,, \tag{2}$$

$$\sum_{j\in V}\sum_{i\in M}\sum_{s\in S_{ij}} x_{ijs}\, s\, p_{ijs} \le k\, C \ , \tag{3}$$

$$x_{ijs} = 0 \ , \qquad \text{if } p_{ijs} > C, \tag{4}$$

$$x_{ijs} \in \{0,1\} \ , \qquad \forall\, i,j,s. \tag{5}$$

Here, C represents the schedule makespan. Equalities (1) make sure that every job is assigned to one machine and uses a constant amount of resources during its processing. Inequalities (2) express the fact that the total processing on each machine is a lower bound on the makespan. Inequality (3) represents the aggregated resource constraint: In any feasible schedule, the left-hand side of (3) is the total resource consumption of the schedule. Because no more than k resources may be consumed at any time, the total resource consumption cannot exceed $k\,C$. Finally, constraints (4) make sure that we do not use machine-resource pairs such that the job processing time exceeds the schedule makespan. These constraints are obviously redundant for the integer program (IP), but they will play a role later when rounding a fractional solution for the linear relaxation of (IP). Summarizing the above observations, we have:

Lemma 1 ([4]). *If there is a feasible schedule with makespan C for the unrelated machine scheduling problem with resource dependent processing times, integer linear program* (1)–(5) *has a feasible solution* (C, x).

Linear programming relaxation. The integer linear program (IP) with the 0/1-constraints on x relaxed to

$$x_{ijs} \ge 0\,, \qquad j \in V\,,\ s \in S_{ij}\,,\ i \in M$$

also has a solution for value C if there is a feasible schedule for the original scheduling problem with makespan C. We note that it can be solved in polynomial time, because it has a polynomial number of variables and constraints. Since we assume integrality of data, we are actually only interested in integral values C. Moreover, an upper bound for C is given by $\sum_{j\in V} \min_{i\in M}\{p_{ijk}\}$. Therefore, by using binary search on possible values for C, we can find in polynomial time the smallest integral value C^{LP} such that the linear programming relaxation of (1)–(5) has a feasible solution x^{LP}. We therefore obtain the following.

Lemma 2 ([4]). *The smallest integral value value C^{LP} such that the linear programming relaxation of* (1)–(5) *has a feasible solution is a lower bound on on the makespan of any feasible schedule, and it can be computed in polynomial time.*

Notice that, as long as we insist on constraints (4), we can not just solve a single linear program minimizing C, since constraints (4) depend nonlinearly on C. Moreover, due to the fact that we only search for integral values C, the binary search on C does not entail any additional approximation error.

Rounding the LP solution. Given a feasible solution $(C^{\mathrm{LP}}, x^{\mathrm{LP}})$ for the linear programming relaxation of (1)–(5), the vector x^{LP} may clearly be fractional. We aim at rounding this fractional solution to an integer one without sacrificing too much in terms of violation of the constraints (2) or (3). We present a rounding procedure that is inspired by a recent paper by Kumar et al. [10]. In fact, it can be seen as a deterministic version of the randomized rounding algorithm of [10]. In the following lemma, we replace the total resource consumptions of jobs, $s\,p_{ijs}$, by arbitrary (nonnegative) coefficients c_{ijs}. This will come in handy in Section 4.

Lemma 3. *Let C^{LP} be the minimal integer for which the following linear program has a feasible solution*

$$\sum_{i\in M}\sum_{s\in S_{ij}} x_{ijs} = 1 \ , \qquad \forall\ j \in V, \tag{6}$$

$$\sum_{j\in V}\sum_{s\in S_{ij}} x_{ijs}\, p_{ijs} \le C^{\mathrm{LP}}, \qquad \forall\ i \in M\,, \tag{7}$$

$$\sum_{j\in V}\sum_{i\in M}\sum_{s\in S_{ij}} x_{ijs} c_{ijs} \le kC^{\mathrm{LP}} \ , \tag{8}$$

$$x_{ijs} = 0 \ , \qquad \text{if } p_{ijs} > C, \tag{9}$$

$$x_{ijs} \ge 0 \ , \qquad \forall\ i,j,s\,, \tag{10}$$

and let $(C^{\mathrm{LP}}, x^{\mathrm{LP}})$ be the corresponding feasible solution, then we can find a feasible solution $x^* = (x^*_{ijs})$ for the following integer linear program in polynomial time.

$$\sum_{i\in M}\sum_{s\in S_{ij}} x_{ijs} = 1 \ , \qquad \forall\ j \in V, \tag{11}$$

$$\sum_{j\in V}\sum_{s\in S_{ij}} x_{ijs}\, p_{ijs} \le C^{\mathrm{LP}} + p_{\max}, \qquad \forall\ i \in M\,, \tag{12}$$

$$\sum_{j\in V}\sum_{i\in M}\sum_{s\in S_{ij}} x_{ijs} c_{ijs} \le kC^{\mathrm{LP}} \ , \tag{13}$$

$$x_{ijs} \in \{0,1\} \ , \qquad \forall\ i,j,s\,, \tag{14}$$

where $p_{\max} = \max\{p_{ijs} \mid x^{\mathrm{LP}}_{ijs} > 0\}$ and $c_{ijs} \ge 0$ are arbitrary fixed coefficients.

One option to prove the lemma is to derandomize the corresponding randomized rounding algorithm of [10], using the method of conditional probabilities.

However, for reasons of self-containedness and accessibility, we prefer to present a direct proof here. Notice, however, that the basic elements of the proof are indeed the same as in [10].

Proof (of Lemma 3). The rounding algorithm works in stages. Let x denote the current fractional solution in a given stage. In the first stage, define $x = x^{\mathrm{LP}}$, and notice that x^{LP} fulfills (11)–(13). Subsequently, we alter the current solution x, while maintaining validity of constraints (11) and (13).

In each stage, we consider a bipartite multigraph $G(x) = (V \cup M, E)$, where the set E of edges is defined as follows. For every pair $i \in M$ and $j \in V$, E contains a set of parallel edges, namely one for each fractional value $0 < x_{ijs} < 1$, $s = 0, \ldots, k$. Therefore, we could have up to $k+1$ parallel edges between every machine-job pair (i, j). Notice that the degree of any non-isolated vertex $v \in V$ is at least 2, due to constraint (11). We furthermore eliminate isolated vertices from graph $G(x)$.

We will encode each edge $e \in E$ by the triplet (i, j, s). For every vertex $w \in V \cup M$ let d_w denote its degree in $G(x)$. We define a variable ϵ_{ijs} for every edge $(i, j, s) \in E$ and a set of linear equations:

$$\sum_{(i,j,s)\in E_j} \epsilon_{ijs} = 0, \;\; j \in V\,, \tag{15}$$

$$\sum_{(i,j,s)\in E_i} p_{ijs}\epsilon_{ijs} = 0, \;\; i \in M \text{ and } d_i \geq 2\,. \tag{16}$$

Let c_1 and c_2 be the number of constraints in (15) and (16), respectively. Let $r \leq \min\{c_1 + c_2, |E|\}$ be the rank of that system. Now observe that $c_1 \leq |V| \leq |E|/2$, because of constraint (11). Moreover, $c_2 \leq |E|/2$ by definition. Thus we obtain that either $r \leq c_1 + c_2 \leq |E| - 1$ or $c_1 = c_2 = |E|/2$. In the latter case, constraints (11), the degree condition in (16), and the fact that there are no isolated vertices, imply that there are exactly $|E|$ vertices in $G(x)$. Hence, the degree of each vertex must equal 2 (and graph $G(x)$ is a collection of even cycles).

Consider the first case when $r \leq |E| - 1$. Since the system of linear equations (15)–(16) is underdetermined, by Gaussian elimination we can find a general solution of this system in the form $\epsilon_{ijs} = \sum_{t=1}^{|E|-r} \alpha_{tijs}\delta_t$ in polynomial time. Here, δ_t, $t = 1 \ldots, |E| - r$, are the real valued parameters representing the degrees of freedom of the linear system, and $\alpha_{tijs} \neq 0$ are the corresponding coefficients. Hence, by fixing $\delta_2 = \delta_3 = \cdots = \delta_{|E|-r} = 0$, we obtain a solution $\epsilon_{ijs} = \alpha_{1ijs}\delta_1$. For convenience of notation we just write $\epsilon_{ijs} = \alpha_{ijs}\delta$, and note that δ is an arbitrary parameter.

Next, we can define a new fractional solution of the original linear program by letting

$$\bar{x}_{ijs} = \begin{cases} x_{ijs} + \alpha_{ijs}\delta & \text{if } x_{ijs} \text{ is fractional,} \\ x_{ijs} & \text{otherwise.} \end{cases}$$

Due to constraints (15) and (16) we obtain that constraints (11) are satisfied for all $j \in V$, and constraints (12) are satisfied for all $i \in M$ except those vertices

(machines) $i \in M$ that have $|E_i| = 1$. Finally, since $\sum_{j \in V} \sum_{i \in M} \sum_{s \in S_{ij}} \bar{x}_{ijs} c_{ijs}$ is a linear function of δ we obtain that constraint (13) is satisfied either for positive or for negative δ. Therefore, by choosing δ either maximal or minimal such that $0 \leq \bar{x}_{ijs} \leq 1$ and such that constraint (13) is still satisfied, we obtain a new solution with one more integral variable satisfying constraints (11) and (13).

Repeating the above procedure we either end up with an integral solution x, fulfilling constraints (11) and (13), together with an empty graph $G(x)$, or we end up with some fractional solution x such that the degree of each vertex in $G(x)$ is at most 2 (even exactly 2). This means that at most two fractional jobs are assigned to any machine, and each fractional job is assigned to at most two machines. If that happens, we continue with with a rounding procedure that is akin to the dependent rounding that was proposed by Gandhi et al. [2]. Let us therefore call the following rounding stages *late* stages, and the previous ones *early* stages.

In a late stage, the maximum vertex degree in $G(x)$ is 2. Moreover, since $G(x)$ is bipartite, we can partition $G(x)$ into two matchings M_1 and M_2. Thus we can define a new fractional solution

$$\bar{x}_{ijs} = \begin{cases} x_{ijs} & \text{for } x_{ijs} \text{ integral,} \\ x_{ijs} + \delta & \text{for } (i,j,s) \in M_1, \\ x_{ijs} - \delta & \text{for } (i,j,s) \in M_2, \end{cases}$$

for some δ. Again, since $\sum_{j \in V} \sum_{i \in M} \sum_{s \in S_{ij}} \bar{x}_{ijs} c_{ijs}$ is a linear function of δ we obtain that constraint (13) is satisfied either for positive or for negative δ. Therefore, by choosing δ either maximal or minimal such that $0 \leq \bar{x}_{ijs} \leq 1$ and such that constraint (13) is still satisfied, we obtain a new solution with at least one more integral variable, still satisfying constraint (13). Moreover, since the two edges incident to any vertex $v \in V$ must belong to different matchings M_1 and M_2, the assignment constraint (11) remains valid, too. Notice that the resulting graph $G(\bar{x})$ still has vertex degrees at most 2, since only edges are dropped due to the rounding. Hence, we can iterate the rounding until all variables are integral.

In the end of the rounding algorithm we obtain an integral solution that obviously satisfies constraints (11). Since on every step we were choosing a solution minimizing a linear function corresponding to constraint (13), we obtain that this constraint is satisfied too.

To show that constraints (12) are satisfied for each $i \in M$, we have to show that the left hand side of the original constraint (2) increases by at most $p_{\max} = \max\{p_{ijs} \mid x^{\mathrm{LP}}_{ijs} > 0\}$. We consider the rounding stage when (2) is violated for machine $i \in M$.

On the one hand, this might happen in an early stage when $d_i = 1$. In this case, however, since there is exactly one fractional edge incident to i, we could add at most $p_{\max}$ in any future rounding stages to the total load of machine i. Hence, constraint (12) is fulfilled by machine i.

On the other hand, the violation of the original constraint (2) might happen in a late stage, where all vertices in $G(x)$ have degree at most 2. When $d_i = 1$

we argue as before. So assume $d_i = 2$. Consider machine i together with its two incident edges (i,j,s) and (i,j',s'). Whenever $x_{ijs} + x_{ij's'} \geq 1$ before the rounding, we claim that the total load of machine i increases by at most $p_{\max}$ by any possible further rounding. This because the total remaining increase in the left hand side of (2) for machine i is at most $(1-x_{ijs})p_{ijs}+(1-x_{ij's'})p_{ij's'} \leq p_{\max}$. So assume that $x_{ijs} + x_{ij's'} < 1$. We claim that at most one of the jobs j and j' will finally be assigned to machine i. To see why, consider the stage where one of these variables was rounded to an integer. Recalling that edges (i,j,s) and (i,j',s') must belong to different matchings M_1 and M_2, we may assume that x_{ijs} is rounded up, and $x_{ij's'}$ is rounded down. Clearly, $x_{ijs} + x_{ij's'} < 1$ holds before that rounding stage. Assuming that x_{ijs} is rounded to 1, it must hold that $x_{ij's'} \geq 1 - x_{ijs}$, because otherwise $x_{ij's'}$ would become negative. In other words, $x_{ijs} + x_{ij's'} \geq 1$, a contradiction. Hence, the only way to round one of the variables x_{ijs} or $x_{ij's'}$ to an integer, is to round $x_{ij's'}$ down to 0. Therefore edge $(i,j's')$ disappears, and indeed, at most job j can be assigned to machine i. Clearly, the resulting increase in the left hand side of (2) for machine i is again at most $p_{\max}$. Hence, constraint (12) is fulfilled after the rounding. □

4 Scheduling

We complete the paper by designing a new 3.75–approximation algorithm for the unrelated parallel machine scheduling problem with resource dependent processing times. Notice that this improves considerably upon the 6.83–approximation from [4], and also upon the 4–approximation from [10]. To achieve this result, we apply the same rounding as in Lemma 3 to another integer programming relaxation, and we use a scheduling algorithm that is inspired by the harmonic algorithm for bin packing.

Let $B_{ij} \subseteq S_{ij}$ be the set of breakpoints that lie in the interval $(k/2, k]$, i.e., $B_{ij} = \{ s \in S_{ij} \mid k/2 < s \leq k \}$. If any two jobs are processed using s resources, where $s \in B_{ij}$, these two jobs cannot be processed in parallel. Then consider the following integer linear program.

$$\sum_{i\in M} \sum_{s\in S_{ij}} x_{ijs} = 1 \ , \qquad \forall\, j \in V\,, \tag{17}$$

$$\sum_{j\in V} \sum_{s\in S_{ij}} x_{ijs}\, p_{ijs} \leq C \ , \qquad \forall\, i \in M\,, \tag{18}$$

$$\sum_{j\in V} \sum_{i\in M} \left(1.5 \sum_{s\in S_{ij}} x_{ijs}\, \frac{s}{k}\, p_{ijs} + \ 0.25 \sum_{s\in B_{ij}} x_{ijs} p_{ijs} \right) \leq 1.75C \ , \tag{19}$$

$$x_{ijs} = 0 \ , \qquad \text{if } p_{ijs} > C, \tag{20}$$

$$x_{ijs} \in \{0,1\} \ , \qquad \forall\, i,j,s. \tag{21}$$

Lemma 4. *If there is a feasible schedule with makespan C for the unrelated machine scheduling problem with resource dependent processing times, integer linear program* (17)–(21) *has a feasible solution* $(C, \tilde{x})$.

Proof. To prove the lemma we only have to verify validity of the new total resource constraint (19). For any feasible schedule, two jobs with resource consumption larger then $k/2$ cannot be processed in parallel, so

$$\sum_{j \in V} \sum_{i=1}^{m} \sum_{s \in Bij} \tilde{x}_{ijs} p_{ijs} \leq C. \tag{22}$$

Combining (22) with valid inequality (3) we derive inequality (19). □

As before, by binary search on C while using Lemma 4 instead of Lemma 1, we can find a lower bound C^{LP} on the makespan of an optimal solution for the unrelated machine scheduling problem.

Lemma 5. *Let C^{LP} be the lower bound on the makespan of an optimal solution, and let $(C^{\mathrm{LP}}, x^{\mathrm{LP}})$ be the corresponding feasible solution of the LP-relaxation of* (17)–(21)*, then we can find a feasible solution $x^* = (x^*_{ijs})$ for the following integer linear program in polynomial time.*

$$\sum_{i \in M} \sum_{s \in S_{ij}} x_{ijs} = 1 \ , \qquad \forall\ j \in V, \tag{23}$$

$$\sum_{j \in V} \sum_{s \in S_{ij}} x_{ijs}\, p_{ijs} \leq C^{\mathrm{LP}} + p_{\max}, \qquad \forall\ i \in M\,, \tag{24}$$

$$\sum_{j \in V} \sum_{i \in M} \left(1.5 \sum_{s \in S_{ij}} x_{ijs}\, \frac{s}{k}\, p_{ijs} + 0.25 \sum_{s \in B_{ij}} x_{ijs} p_{ijs} \right) \leq 1.75 C^{\mathrm{LP}} \ , \tag{25}$$

$$x_{ijs} \in \{0, 1\} \ , \qquad \forall\ i, j, s\,, \tag{26}$$

where $p_{\max} = \max\{p_{ijs} \mid x^{\mathrm{LP}}_{ijs} > 0\}$.

Proof. The proof follows from Lemma 3 with

$$c_{ijs} = \begin{cases} \left(\frac{1.5s}{1.75k} + \frac{0.25}{1.75}\right)\, p_{ijs} & \text{for all } s \in B_{ij}\,, \\ \frac{1.5s}{1.75k}\, p_{ijs} & \text{for all } s \in S_{ij} \setminus B_{ij}\,. \end{cases}$$

□

Now, we are ready to present a scheduling algorithm with performance guarantee 3.75. To that end, we first partition the set of jobs J into three groups J_1, J_2, and J_3 according to the amount of resources consumed. Define $J_1 = \{j \mid k/2 < s^* \leq k \ \text{ and } \ x_{ijs^*} = 1\}$, $J_2 = \{j \mid k/3 < s^* \leq k/2 \ \text{ and } \ x_{ijs^*} = 1\}$, and $J_3 = \{j \mid s^* \leq k/3 \ \text{ and } \ x_{ijs^*} = 1\}$.

Algorithm LP-Greedy: Let the resource allocations and machine assignments be determined by the rounded LP solution as in Lemma 5. The algorithm schedules jobs group by group. In the first phase it schedules jobs from J_1 one after another (they cannot be processed in parallel since they consume too much resources). Let C_1 be the completion time of the last job from J_1. In the second phase the algorithm schedules jobs from J_2 starting at time C_1. The algorithm always tries to run two jobs from J_2 in parallel. Let C_2 be the first time when the algorithm fails to do so. This could happen either because J_2 is empty or all remaining jobs must be processed on the same machine, say M_1. In the last case the algorithm places all remaining jobs on M_1 without idle time between them. In the third phase the algorithm greedily schedules jobs from J_3, starting no earlier than time C_2. So if some job from J_3 can be started at the current time C_2, we start processing this job. When no jobs can start at the current time we increment the current time to the next job completion time and repeat until all jobs are scheduled. Let C_3 be the completion time of the last job from the set $J_2 \cup J_3$.

We now estimate the makespan C^{LPG} of the schedule. Consider the machine i with the job that finishes last in the schedule. Let B be the total time when machine i is busy and I be the total time when machine i is idle in the interval $[0, C^{\mathrm{LPG}}]$, i.e., $C^{\mathrm{LPG}} = B + I$. By constraint (24) in Lemma 5, we have $B \leq C^{\mathrm{LP}} + p_{max} \leq 2C^{\mathrm{LP}}$, where the last inequality follows from (20).

To bound the total idle time on machine i we consider two cases. We first consider the case where the job that finishes last belongs to J_1 or J_2. If the last job is from J_1, intervals $[C_1, C_2]$ and $[C_2, C_3]$ have length 0. If the last job is from J_2, there is no idle time on machine i in the interval $[C_2, C_3]$. Thus in both cases, there is no idle time on machine i in the interval $[C_2, C_3]$. Let I_1 be the total idle time on machine i during $[0, C_1]$ and I_2 be the total idle time during $[C_1, C_2]$. Then $I = I_1 + I_2$ is the total idle time on machine i. Since we process one job at a time from J_1 during the time interval $[0, C_1]$ and two jobs at a time from J_2 during $[C_1, C_2]$, the total resource consumption of the schedule during idle times on machine i is at least

$$I_1 \frac{k}{2} + I_2 \frac{2k}{3} .$$

Letting $R_I := I_1/2 + 2I_2/3$, the total resource consumption of the schedule during idle times on machine i is at least $R_I k$, and we therefore get

$$\begin{aligned} I \;=\; I_1 \;+\; I_2 \;&=\; \frac{3}{2} R_I \;+\; \frac{I_1}{4} \\ &\leq \sum_{j \in V} \sum_{i \in M} \left(1.5 \sum_{s \in S_{ij}} \tilde{x}_{ijs} \frac{s}{k} p_{ijs} + 0.25 \sum_{s \in B_{ij}} \tilde{x}_{ijs} p_{ijs} \right) \\ &\leq 1.75 C^{\mathrm{LP}} . \end{aligned} \tag{27}$$

Here, the first inequality holds since $\sum_{j\in V}\sum_{i\in M}\sum_{s\in S_{ij}}\tilde{x}_{ijs}\frac{s}{k}p_{ijs}$ equals the total resource consumption of the schedule divided by k, and since $I_1 \leq C_1 = \sum_{j\in V}\sum_{i\in M}\sum_{s\in B_{ij}}\tilde{x}_{ijs}p_{ijs}$. The second inequality follows from (25).

Similarly, if the last job in the schedule belongs to J_3, let I_1 be the total idle time on machine i during $[0, C_1]$, I_2 be the total idle time on machine i during $[C_1, C_2]$ and I_3 be the total idle time on machine i during $[C_2, C_3]$. Then $I = I_1 + I_2 + I_3$ is the total idle time on machine i. Again, we process one job at a time from J_1 during the time interval $[0, C_1]$, and two jobs at a time from J_2 during $[C_1, C_2]$. Moreover, due to the resource constraint the last job –which is from J_3– could not be scheduled at idle times on machine i during $[C_2, C_3]$, so the total resource consumption of the schedule during idle times on machine i in $[C_2, C_3]$ is at least $2/3\,k$. Hence, the total resource consumption of the schedule during idle times on machine i is at least

$$I_1\frac{k}{2} + (I_2 + I_3)\,\frac{2k}{3}\,.$$

Again, letting $R_I := I_1/2 + 2(I_2 + I_3)/3$, the total resource consumption of the schedule during idle times on machine i is at least $R_I k$, and we get

$$I \;=\; I_1 \;+\; (I_2 + I_3) \;=\; \frac{3}{2}R_I \;+\; \frac{I_1}{4}\,.$$

Exactly as before in (27) we conclude that $I \leq 1.75C^{\mathrm{LP}}$. Therefore, in either of the two cases we have $C^{\mathrm{LPG}} = B + I \leq 2\ C^{\mathrm{LP}} + 1.75\ C^{\mathrm{LP}} = 3.75\ C^{\mathrm{LP}}$, and we have proved the following theorem.

Theorem 1. *Algorithm* LP-Greedy *is a* 3.75*–approximation algorithm for unrelated parallel machine scheduling with resource dependent processing times.*

Acknowledgments

We thank the referees for some helpful remarks.

References

1. Z.-L. Chen, Simultaneous Job Scheduling and Resource Allocation on Parallel Machines, *Annals of Operations Research* **129** (2004), 135–153.
2. R. Gandhi, S. Khuller, S. Parthasarathy, and A. Srinivasan, Dependent Rounding in Bipartite Graphs, in *Proc. 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002, 323–332.
3. R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: A survey, *Annals of Discrete Mathematics* **5** (1979), 287–326.
4. A. Grigoriev, M. Sviridenko and M. Uetz, Unrelated Parallel Machine Scheduling with Resource Dependent Processing Times, Proceedings 11th Conference on Integer Programming and Combinatorial Optimization, M. Jünger and V. Kaibel (eds.), *Lecture Notes in Computer Science* 3509, Springer, 2005, 182–195.

5. A. Grigoriev and M. Uetz, Scheduling Jobs with Linear Speedup, Proceedings 3rd Workshop on Approximation and Online Algorithms, T. Erlebach and P. Persiano (eds.), *Lecture Notes in Computer Science* 3879, Springer, 2006, 203–215.
6. K. Jansen, Scheduling Malleable Parallel Tasks: An Asymptotic Fully Polynomial Time Approximation Scheme, *Algorithmica* **39** (2004), pp. 59-81.
7. K. Jansen and M. Mastrolilli, Approximation schemes for parallel machine scheduling problems with controllable processing times, *Computers and Operations Research* **31** (2004), 1565–1581.
8. J. E. Kelley and M. R. Walker, *Critical path planning and scheduling: An introduction*, Mauchly Associates, Ambler (PA), 1959.
9. H. Kellerer and V. A. Strusevich, Scheduling parallel dedicated machines under a single non-shared resource, *European Journal of Operational Research* **147** (2003), 345–364.
10. V. S. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan, Approximation Algorithms for Scheduling on Multiple Machines, *Proc. 46th Annual IEEE Symposium on Foundations of Computer Science*, 2005, 254–263.
11. J. K. Lenstra, D. B. Shmoys and E. Tardos, Approximation algorithms for scheduling unrelated parallel machines, *Mathematical Programming*, Series A **46** (1990), 259–271.
12. G. Mounie, C. Rapine, and D. Trystram, Efficient Approximation Algorithms for Scheduling Malleable Tasks, *Proc. 11th Annual ACM Symposium on Parallel Algorithms and Architectures*, 1999, 23–32.
13. D. B. Shmoys and E. Tardos, An approximation algorithm for the generalized assignment problem, *Mathematical Programming*, Series A **62** (1993), 461–474.
14. M. Skutella, Approximation algorithms for the discrete time-cost tradeoff problem, *Mathematics of Operations Research* **23** (1998), pp. 909–929.
15. J. Turek, J. L. Wolf, and P. S. Yu, Approximate Algorithms for Scheduling Parallelizable Tasks, *Proc. 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, 1992, 323–332.