Unrelated parallel machine scheduling with resource dependent processing times

Unrelated parallel machine scheduling with resource dependent processing times[†]

Alexander Grigoriev[*] Maxim Sviridenko[**] Marc Uetz[*]

Abstract. We consider machine scheduling problems where jobs have to be processed on unrelated parallel machines in order to minimize the schedule makespan. The processing time of any job is dependent on the usage of a scarce resource that can be distributed over the jobs in process. The more of that resource is allocated to a job, the smaller its processing time. This model is a natural variant of a generalized assignment problem studied previously by Shmoys and Tardos, the main difference lying in the fact that we assume that the resource is renewable, and not a total budget constraint. Using linear programming based rounding techniques, combined with a simple greedy algorithm, we derive the first constant-factor approximation algorithm for this problem. Our approach can be adapted to yield an approximation algorithm for the problem with dedicated machines, too. Moreover, we prove a non-approximability result for the problem with dedicated machines, and we derive a lower bound for the employed analysis.

1 Introduction and related work

Unrelated parallel machine scheduling to minimize the makespan, denoted $R||C_{\max}$ in the three-field notation of Graham et al. [5], is one of the classical problems in scheduling. Given are n jobs that have to be scheduled on m parallel machines, and the processing time of job j on machine i is p_{ij}. The goal to minimize the latest job completion, the makespan $C_{\max}$. If the number of machines m is not fixed, the best approximation algorithm known to date is a 2-approximation by Lenstra, Shmoys and Tardos [9]. In the same paper, it is shown that this problem cannot be approximated within a factor smaller than 3/2, unless P=NP.

Shmoys and Tardos [10] consider the same problem with the additional feature of costs λ_{ij} if job j is processed on machine i. They show that, if a

[†]This research was done while the second author was visiting Maastricht University, partially supported by METEOR, the Maastricht Research School of Economics of Technology and Organizations.

[*]Maastricht University, Faculty of Economics and Business Administration, Quantitative Economics, P.O. Box 616, NL-6200 MD Maastricht, The Netherlands, Email: {a.grigoriev, m.uetz}@ke.unimaas.nl.

[**]IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, U.S.A., Email: sviri@us.ibm.com.

schedule with total cost Λ and makespan T exists, a schedule with total cost Λ and makespan at most $2T$ can be found in polynomial time. The proof relies on rounding the solution of an LP relaxation. They obtain the same result even for a more general version of the problem, namely when the processing time p_{ij} of any job-machine pair is not fixed, but may be reduced linearly, in turn for a linear increase of the associated cost λ_{ij} [10]. Note that, in both versions of the problem studied in [10], the costs λ_{ij} are *non-renewable* resources, such as a monetary budget, with a global budget Λ.

In this paper, we consider a different variant of the problem. Namely, the processing times p_{ij} of any job-machine pair can be reduced by utilizing a *renewable* resource, such as additional workers, that can be distributed over the jobs. In other words, a maximum number of k units of a resource 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 assumptions on costs and processing times in [10], the only assumption we require is that the processing times p_{ijs}, which now depend also on the number s of allocated resources, are non-increasing for each job-machine pair. That is, we assume that $p_{ij0} \geq \cdots \geq p_{ijk}$ for all jobs j and all machines i.

To the best of our knowledge, this problem has not been addressed in the literature before, at least not in this generality. What comes close to our work is a manuscript by Grigoriev, Kellerer, and Strusevich [6], where a restricted variant of the problem is addressed. They consider dedicated, parallel machines, thus each job is dedicated beforehand to be processed on a given machine. Moreover, their model is restricted to a binary resource, that is, the availability of the additional resource is k=1, and any job may be processed either with or without using that resource. Finally, the number of machines m in their paper is considered fixed, and not part of the input. For that restricted problem, they derive a $(3 + \varepsilon)$–approximation, and for the problem with $m = 2$ machines, they prove weak NP-hardness and the existence of a fully polynomial time approximation scheme [6].

When we restrict even further, and assume that the decision on the allocation of resources to jobs is fixed, we are back at scheduling under resource constraints as introduced by Blazewicz, Lenstra, and Rinnooy Kan [1]. More recently, such problems with dedicated machines have been discussed by Kellerer and Strusevich [7, 8]. We refer to these papers for various complexity results, and note that NP-hardness of the problem with dedicated machines and a binary resource was established by Kellerer and Strusevich [7][1].

2 Results and methodology

We derive the first constant-factor approximation algorithms for the problem at hand. Our approach is based upon an integer linear program that defines a relaxation of the problem, extending the one used by Grigoriev, Kellerer, and Strusevich [6]. The main idea is the utilization of an aggregate version of the

[1] To be more precise, they show weak NP-hardness for the case where the number of machines is fixed, and strong NP-hardness for an arbitrary number of machines [7].

resource constraints, yielding a formulation that does not require time-indexed variables. We then consider the linear programming relaxation of this integer program. In a first step, the solution of this LP relaxation is rounded into a (still fractional) solution for another linear program. We then show that this in fact defines an instance (and solution) of the linear programming relaxation used by Shmoys and Tardos [10] for the generalized assignment problem. In a second step, we thus apply their rounding procedure to obtain an approximate integral solution for the original integer programming relaxation. From this solution, we extract both the machine assignments and the resource allocations for the jobs. We then use Grahams list scheduling [3] to generate a feasible schedule. Using the LP lower bounds, we prove that this schedule is not more than a factor 6.83 away from the optimum. For the special case of dedicated machines, our approach simplifies, because the machine assignments are fixed beforehand, thus the second rounding step is not required. For that case, we prove a performance bound of 5.83. For both problems, we furthermore provide an instance showing that the employed LP based analysis cannot yield anything better than a 2 approximation.

Concerning lower bounds on the approximability of the unrelated parallel machine scheduling problem at hand, note that it is a generalization of the classical unrelated parallel machine scheduling problem $R||C_{\max}$. Therefore, utilizing the result of Lenstra, Shmoys, and Tardos [9] for unrelated parallel machine scheduling, it cannot be approximated better than 3/2, unless P=NP.

Restricting to the special case of dedicated parallel machines, strong NP-hardness follows from the results of Kellerer and Strusevich [7], hence the problem cannot admit an FPTAS, unless P=NP. This is true even for the case of a binary resource, that is, if $k = 1$. Using a gap-reduction from Partition, we furthermore show that the problem with dedicated machines cannot be approximated better than 3/2, unless P=NP. This non-approximability result is true under the assumption that the functions p_{js}, $s = 0, \ldots, k$, describing the resource-dependent processing time of jobs j, can be encoded succinctly. In particular, this includes the case where each job has no more than $O(\log k)$ different (resource dependent) processing times, and, a fortiori, the case when resource allocations are fixed beforehand.

3 Problem definition

Let $V = \{1, \ldots, n\}$ be a set of jobs. Jobs must be processed non-preemptively on a set of m unrelated parallel machines, and 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 $s = 0, 1, \ldots, k$ of units of an additional resource, for instance additional workers, that may speed up its processing. If s resources are allocated to a job j, the processing time of that job on machine i is p_{ijs}. The only assumption on the processing times in 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). 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 jobs in process. Moreover, the amount of resources assigned to any job must be the same along its processing. In other words, if s resources are allocated to some job j, and S_j and C_j denote start and completion time of job j, respectively, only $k - s$ of the resources are available for other jobs between S_j and C_j.

4 IP relaxation and LP-based rounding

Let x_{ijs} denote binary variables, indicating that an amount of s resources is used for processing job j on machine i. Then the following integer linear program, referred to as (IP) has a feasible solution if there is a feasible schedule of length C for our original problem.

$$\sum_{i=1}^{m}\sum_{s=0}^{k} x_{ijs} = 1 \ , \qquad \forall\, j \in V \tag{1}$$

$$\sum_{j\in V}\sum_{s=0}^{k} x_{ijs}\, p_{ijs} \le C \ , \qquad \forall\, i = 1,\ldots,m\,, \tag{2}$$

$$\sum_{j\in V}\sum_{i=1}^{m}\sum_{s=0}^{k} x_{ijs}\, s\, p_{ijs} \le k\, C \ , \tag{3}$$

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

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

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 that fact that the total processing on each machine is a lower bound on the makespan. Inequalities (3) represent the aggregated resource constraints: 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 (IP), but they will be used in rounding a solution for the linear relaxation of (IP).

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

$$x_{ijs} \ge 0\,, \qquad j \in V\,,\ s = 0,\ldots,k\,,\ i = 1,\ldots,m$$

also has a solution of value at most C if there is a feasible schedule for the original scheduling problem with makespan C. We refer to this linear programming relaxation as (LP), and note that it can be solved in polynomial time, because it has a polynomial number of variables and constraints. By using binary search, we can find in polynomial time the smallest value C^{LP} such that (LP) has a feasible solution x^{LP}. Then C^{LP} is a lower bound on the makespan of any feasible schedule.

Rounding the LP solution. Given a pair $(C^{\mathrm{LP}}, x^{\mathrm{LP}})$ we next define an integer solution x^* from x^{LP} by the following, 2-phase rounding procedure. In the first rounding phase, we transform a fractional solution x^{LP} to another fractional solution $\bar{x}$, in such a way that for every machine-job pair (i, j) there is exactly one index s (amount of resource) such that $\bar{x}_{ijs}$ is nonzero. Intuitively, we decide for every machine-job pair on the amount of resources it may consume. By doing this, we effectively get rid of the index s. This new fractional solution in fact defines a fractional solution for an LP relaxation for the generalized assignment problem discussed by Shmoys and Tardos [10]. Therefore, we will be able to use their rounding procedure as our second rounding phase, and thus we eventually obtain an integral solution x^* from x^{LP}.

First, let us choose an arbitrary ε such that

$$0 \leq \varepsilon \leq 1\,.$$

Then, for every machine i and job j individually, define

$$\tilde{y}_{ij} = \sum_{s=0}^{k} x^{\mathrm{LP}}_{ijs} \tag{5}$$

as the total fractional value allocated by the LP solution x^{LP} to the machine-job pair (i, j). Then let index $t^{ij} \in \{0, \ldots, k\}$ be chosen minimal with the property that

$$\sum_{s=0}^{t^{ij}} x^{\mathrm{LP}}_{ijs} \geq (1-\varepsilon)\,\tilde{y}_{ij}\,.$$

Then, for every machine i and job j define index

$$s^{ij} = \arg\,\min_{s \geq t^{ij}}\; s\,p_{ijs}\,. \tag{6}$$

We consider a fractional solution $\bar{x}$ defined by

$$\bar{x}_{ijs} = \begin{cases} \tilde{y}_{ij} & s = s^{ij}\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{7}$$

By definition, this solution fulfills (1). Moreover, we claim that it is an approximate solution for inequalities (2) and (3) in the following sense.

Lemma 4.1. *Let $(C^{\mathrm{LP}}, x^{\mathrm{LP}})$ be an optimal fractional solution for the linear programming relaxation* (LP)*, and let $\bar{x} = (\bar{x}_{ijs})$ be the fractional solution obtained by the above described rounding procedure. Then*

$$\sum_{j \in V} \sum_{s=0}^{k} \bar{x}_{ijs}\,p_{ijs} \leq \frac{1}{1-\varepsilon}\,C^{\mathrm{LP}}\;, \qquad i = 1, \ldots, m\,, \tag{8}$$

$$\sum_{j \in V} \sum_{i=1}^{m} \sum_{s=0}^{k} \bar{x}_{ijs}\,s\,p_{ijs} \leq \frac{k}{\varepsilon}\,C^{\mathrm{LP}}\;. \tag{9}$$

Proof. The proof of both claims is based on proving the statement for every machine-job pair. Validity of (8) can be seen as follows. We know that

$$(1-\varepsilon)\,\bar{x}_{ijs^{ij}} = (1-\varepsilon)\,\tilde{y}_{ij} \leq \sum_{s\leq t^{ij}} x^{\mathrm{LP}}_{ijs}$$

by definition of t^{ij}. By the fact that $p_{ijt^{ij}} \leq p_{ijs}$ for all $s \leq t^{ij}$, we therefore have

$$(1-\varepsilon)\,\bar{x}_{ijs^{ij}} p_{ijt^{ij}} \leq \sum_{s\leq t^{ij}} x^{\mathrm{LP}}_{ijs}\, p_{ijs} \leq \sum_{s=0}^{k} x^{\mathrm{LP}}_{ijs}\, p_{ijs}$$

for every machine i and job $j \in V$. Again due to monotonicity, $p_{ijs^{ij}} \leq p_{ijt^{ij}}$ for all $j \in V$ and $i = 1, \ldots, m$, and we obtain

$$\begin{aligned}\sum_{s=0}^{k} \bar{x}_{ijs}\, p_{ijs} \;&=\; \bar{x}_{ijs^{ij}} p_{ijs^{ij}} \;\leq\; \bar{x}_{ijs^{ij}} p_{ijt^{ij}} \\ &\leq\; \frac{1}{1-\varepsilon} \sum_{s=0}^{k} x^{\mathrm{LP}}_{ijs}\, p_{ijs}\end{aligned}$$

for all jobs $j \in V$ and machines $i = 1, \ldots, m$. Summing over $j \in V$, inequalities (8) follow for any machine i.

To see (9), first observe that $\varepsilon\,\bar{x}_{ijs^{ij}} = \varepsilon\,\tilde{y}_{ij} \leq \sum_{s\geq t^{ij}} x^{\mathrm{LP}}_{ijs}$ by definition of t^{ij}. Therefore,

$$\varepsilon\,\bar{x}_{ijs^{ij}} s^{ij} p_{ijs^{ij}} \leq \sum_{s\geq t^{ij}} x^{\mathrm{LP}}_{ijs}\, s\, p_{ijs} \leq \sum_{s=0}^{k} x^{\mathrm{LP}}_{ijs}\, s\, p_{ijs}$$

for every machine i and job $j \in V$, where the first inequality follows because s^{ij} was chosen among all $s \geq t^{ij}$ such as to minimize $s\, p_{ijs}$. Hence, we obtain

$$\sum_{s=0}^{k} \bar{x}_{ijs}\, s\, p_{ijs} = \bar{x}_{ijs^{ij}} s^{ij}\, p_{ijs^{ij}} \leq \frac{1}{\varepsilon} \sum_{s=0}^{k} x^{\mathrm{LP}}_{ijs}\, s\, p_{ijs}\,,$$

for all jobs $j \in V$ and machines $i = 1, \ldots, m$. Summing over $j \in V$ and all machines $i = 1, \ldots, m$ yields (9). □

Next, we want to use the rounding procedure by Shmoys and Tardos in order to end up with an integer solution.

Lemma 4.2 (Shmoys & Tardos [10, Theorem 2.1]). *Given a feasible fractional solution $\tilde{y} = (\tilde{y}_{ij})$ to the linear program*

$$\sum_{i=1}^{m} y_{ij} = 1 \;, \qquad \forall\; j \in V \tag{10}$$

$$\sum_{j\in V} y_{ij}\tau_{ij} \leq T \;, \qquad \forall\; i = 1, \ldots, m\,, \tag{11}$$

$$\sum_{j\in V} \sum_{i=1}^{m} y_{ij}\lambda_{ij} \leq \Lambda \;, \tag{12}$$

$$y_{ij} \geq 0 \;, \qquad \forall\; i, j. \tag{13}$$

with nonnegative parameters $T, \Lambda, \tau = (\tau_{ij})$, and $\lambda = (\lambda_{ij})$, there is a polynomial time algorithm which finds an integral solution to the following integer program

$$\begin{aligned}
&\sum_{i=1}^{m} y_{ij} = 1 \ , && \forall\ j \in V \\
&\sum_{j \in V} y_{ij}\tau_{ij} \leq T + \tau_{max} \ , && \forall\ i = 1, \ldots, m\,, \\
&\sum_{j \in V} \sum_{i=1}^{m} y_{ij}\lambda_{ij} \leq \Lambda \ , \\
&y_{ij} \in \{0, 1\} \ , && \forall\ i, j.
\end{aligned}$$

where $\tau_{\max} = \max_{i,j}\{\tau_{ij} \mid \tilde{y}_{ij} > 0\}$. □

The fractional solution $\tilde{y}$ defined in (5), however, is nothing but a feasible fractional solution for linear program (10)–(13), namely with parameters $T = 1/(1-\varepsilon)\, C^{\mathrm{LP}}$, $\Lambda = k/\varepsilon\, C^{\mathrm{LP}}$, $\tau_{ij} = p_{ijs^{ij}}$, and $\lambda_{ij} = s^{ij} p_{ijs^{ij}}$ for all job-machine pairs (i, j). Therefore, combining Lemma 4.1, the above result of Shmoys and Tardos, and the fact that

$$\tau_{\max} = \max_{i,j} p_{ijs^{ij}} \leq \max_{i,j,s}\{p_{ijs} \mid x^{\mathrm{LP}}_{ijs} > 0\} \leq C^{\mathrm{LP}}$$

by inequalities (4), we conclude the following.

Lemma 4.3. *We can find a feasible solution $x^* = (x^*_{ijs})$ for the following linear integer program in polynomial time.*

$$\sum_{i=1}^{m} \sum_{s=0}^{k} x_{ijs} = 1 \ , \qquad \forall\ j \in V, \tag{14}$$

$$\sum_{j \in V} \sum_{s=0}^{k} x_{ijs}\, p_{ijs} \leq \left(1 + \frac{1}{1-\varepsilon}\right) C^{\mathrm{LP}} \ , \qquad \forall\ i\,, \tag{15}$$

$$\sum_{j \in V} \sum_{i=1}^{m} \sum_{s=0}^{k} x_{ijs}\, s\, p_{ijs} \leq \frac{k}{\varepsilon}\, C^{\mathrm{LP}} \ , \tag{16}$$

$$x_{ijs} \in \{0, 1\} \ , \qquad \forall\ i, j, s\,. \tag{17}$$

5 LP based greedy algorithm

Our approach to obtain a constant factor approximation for the scheduling problem is now the following. We first use the rounded 0/1-solution from the previous section in order to decide both, on the amount of resources allocated to every individual job j, and on the machine where this job must be executed. More precisely, job j must be processed on machine i and use s additional resources iff $x^*_{ijs} = 1$, where x^* is the feasible integral solution of (14)–(17) obtained after the 2-phase rounding. Then the jobs are scheduled according to the greedy list scheduling algorithm of Graham [3], in arbitrary order.

Algorithm LP-Greedy. With the resource allocations and machine assignments as determined by the LP based rounding, do until all jobs are scheduled: Starting at time 0, iterate over completion times of jobs, and schedule as many jobs as allowed, obeying the machine assignments and the resource constraints.

6 Analysis of greedy algorithm

We have the following performance bound.

Theorem 6.1. *Algorithm* LP-Greedy *is a* $(4+2\sqrt{2})$*–approximation algorithm for unrelated parallel machine scheduling with resource-dependent processing times, where* $4+2\sqrt{2} \approx 6.83$.

The fact that the algorithm requires only polynomial time follows directly from the fact that both, solving and rounding the LP relaxation, as well as the list scheduling, can be implemented in polynomial time.

To verify the performance bound, we need some additional notation. Consider some schedule S produced by algorithm LP-Greedy, and denote by C^{LPG} the corresponding makespan. Denote by C^{OPT} the makespan of an optimal solution. For schedule S, let $t(\beta)$ the earliest point in time after which only *big jobs* are processed, big jobs j being defined as jobs that have a resource consumption larger than $k/2$. Moreover, let $\beta = C^{\mathrm{LPG}} - t(\beta)$ be the length of the period in which only big jobs are processed (note that possibly $\beta = 0$).

Next, we fix a machine, say machine i, on which some job completes at time $t(\beta)$ which is not a big job. Due to the definition of $t(\beta)$, such a machine must exist, because otherwise all machines were idle right before $t(\beta)$, contradicting the definition of the greedy algorithm. Note that, between time 0 and $t(\beta)$, periods may exist where machine i is idle. Denote by α the total length of busy periods on machine i between 0 and $t(\beta)$, and by γ the total length of idle periods on machine i between 0 and $t(\beta)$. We then have that

$$C^{\mathrm{LPG}} \leq \alpha + \beta + \gamma\,. \tag{18}$$

Due to (15), we get

$$\alpha \leq \sum_{j \in V} \sum_{s=0}^{k} x^*_{ijs}\, p_{ijs} \leq \left(1 + \frac{1}{1-\varepsilon}\right) C^{\mathrm{LP}}\ . \tag{19}$$

The next step is an upper bound on γ, the total length of idle periods on machine i.

Lemma 6.1.

$$\gamma \leq \frac{2}{\varepsilon}\, C^{\mathrm{LP}} - \beta\,.$$

Proof. First, observe that the total resource consumption of the schedule is at least

$$\beta\,\frac{k}{2} + \gamma\,\frac{k}{2}\,.$$

This because, on the one hand, all jobs after $t(\beta)$ are big jobs and require at least $k/2$ resources, by definition of $t(\beta)$. On the other hand, during all idle periods on machine i between 0 and $t(\beta)$, at least $k/2$ of the resources must be in use as well. Assuming the contrary, there was an idle period on machine i with at least $k/2$ free resources. But after that idle period, due to the selection of $t(\beta)$ and machine i, some job is processed on machine i which is not a big job. This job could have been processed earlier during the idle period, contradicting the definition of the greedy algorithm.

Next, recall that $(k/\varepsilon)\, C^{\mathrm{LP}}$ is an upper bound on the total resource consumption of the jobs, due to (16). Hence, we obtain

$$\frac{k}{\varepsilon}\, C^{\mathrm{LP}} \geq \beta\, \frac{k}{2} + \gamma\, \frac{k}{2}\,.$$

Rearrangement yields the claimed bound on γ. □

Now we are ready to prove the performance bound of Theorem 6.1.

Proof of Theorem 6.1. First, use (18) together with (19) and Lemma 6.1 to obtain

$$\begin{aligned} C^{\mathrm{LPG}} &\leq \left(1 + \frac{1}{1-\varepsilon}\right) C^{\mathrm{LP}} + \frac{2}{\varepsilon}\, C^{\mathrm{LP}} \\ &\leq \left(1 + \frac{1}{1-\varepsilon} + \frac{2}{\varepsilon}\right) C^{\mathrm{OPT}}\,. \end{aligned}$$

Solving for the best possible value for ε gives $\varepsilon = 2 - \sqrt{2} \approx 0.5858$, which yields the claimed performance bound of $4 + 2\sqrt{2} \approx 6.83$. □

7 Dedicated machines

As a special case of the unrelated machine scheduling model considered so far, let us assume that the jobs are assigned to machines *beforehand*. That is, the set of jobs V is partitioned into m subsets $V_1, \ldots, V_m$ a priori, V_i being the jobs that must be processed on machine i.

By letting all but one machine assignment result in very large processing times, this is obviously a special case of the unrelated machine scheduling model. Hence, our above analysis also yields a 6.83–approximation for this model. However, noting that the machine index i can be eliminated from the linear

program, we can use the following LP relaxation instead.

$$\text{min.} \quad C \tag{20}$$

$$\text{s. t.} \quad \sum_{s=0}^{k} x_{js} = 1 \ , \qquad \forall\, j \in V\,, \tag{21}$$

$$\sum_{j\in V_i} \sum_{s=0}^{k} x_{js}\, p_{js} \le C \ , \qquad \forall\, i = 1,\ldots,m\,, \tag{22}$$

$$\sum_{j\in V} \sum_{s=0}^{k} x_{js}\, s\, p_{js} \le k\, C \ , \tag{23}$$

$$x_{js} \ge 0 \ , \qquad \forall j, s\,. \tag{24}$$

This way, the accordingly adapted first phase rounding of (7) already yields an integral solution. More precisely, given a fractional solution $(C^{\mathrm{LP}}, x^{\mathrm{LP}})$ for the above LP relaxation, we choose index t^j minimal with the property that

$$\sum_{s=0}^{t^{ij}} x_{js}^{\mathrm{LP}} \ge 1 - \varepsilon\,,$$

and define index $s^j = \arg\min_{s\ge t^j} s p_{js}$. Then the solution $\bar{x}$, defined by $\bar{x}_{js} = 1$ if $s = s^j$ and $\bar{x}_{js} = 0$ otherwise, is already integral. Hence, the second phase rounding is not required at all, and instead of using the bounds (15) and (16), we now have an integral solution $\bar{x} = (\bar{x}_{js})$ which fulfills the constraints

$$\sum_{j\in V_i} \sum_{s=0}^{k} \bar{x}_{js}\, p_{js} \le \frac{1}{1-\varepsilon}\, C \ , \qquad \forall\, i = 1,\ldots,m\,,$$

$$\sum_{j\in V} \sum_{s=0}^{k} \bar{x}_{js}\, s\, p_{js} \le \frac{k}{\varepsilon}\, C \ .$$

Validity of these bounds is proved exactly as in Lemma 4.1. This eventually gives a slightly improved performance bound for the dedicated machine model.

Theorem 7.1. *Algorithm* LP-Greedy *is a* $(3 + 2\sqrt{2})$*–approximation algorithm for dedicated parallel machine scheduling with resource-dependent processing times, where* $3 + 2\sqrt{2} \approx 5.83$.

8 Lower bounds on approximation

The problem with unrelated machines cannot be approximated within a factor smaller than 3/2, because this lower bound even holds for the problem without an additional resource [9]. We next show that the same non-approximability result holds for the problem with dedicated machines.

Theorem 8.1. *In general, there is no polynomial time approximation algorithm for dedicated parallel machine scheduling with resource-dependent processing times that has a performance guarantee less than* 3/2*, unless* P=NP.

Proof. The proof relies on a gap-reduction from the NP-complete problem PARTITION [2]: Given n integers a_j, with $\sum_{j=1}^{n} a_j = 2k$, it is NP-complete to decide if there exists a subset $W \subseteq \{1, \ldots, n\}$ with $\sum_{j \in W} a_j = k$. Now, given an instance of PARTITION, let us define an instance of the dedicated machine scheduling problem as follows. Each a_j gives rise to one job j with an individual machine. Hence, we have n jobs and $m = n$ dedicated machines. There are k units available of the additional resource. Any job j has a processing time defined by

$$p_{js} = \begin{cases} 3 & \text{if } s < a_j \\ 1 & \text{if } s \geq a_j \,. \end{cases}$$

Since the functions p_{js}, $s = 0, \ldots, k$, can be encoded in $\mathrm{O}(\log k)$ for all jobs j, this indeed defines a polynomial transformation. We claim that there exists a feasible schedule with makespan $C_{\max} < 3$ if and only if there exists a solution for the PARTITION problem. Otherwise, the makespan is at least 3. To this end, observe that in any solution with makespan $C_{\max} < 3$, we may assume that each job j consumes exactly a_j units of the resource: If it was less than a_j for some jobs j, the makespan would be at least 3; if it was more than a_j for some job j, letting the resource allocation equal a_j does not violate feasibility, while maintaining the same processing time. Now, if and only if there is a solution, say W, for the PARTITION problem, there exists a resource feasible schedule with makespan 2, namely where jobs $j \in W$ start at time 0, and all jobs $j \notin W$ start at time 1. Hence, if there was an approximation algorithm with performance guarantee less than 3/2, it would yield a solution with makespan $C_{\max} < 3$ if and only if there is a solution for the given instance of PARTITION, which cannot be unless P=NP. □

Remark. Notice that the above reduction only yields non-approximability for problems where the resource-dependent processing times are encoded succinctly. More precisely, in the proof we explicitly require that the encoding of the functions p_{js}, $s = 0, \ldots, k$, is possible in $\mathrm{O}(\log k)$ for all jobs j. Otherwise, the above transformation would not be polynomial. In fact, we did *not* make this assumption in the LP based approximation algorithm of Section 7, where we use kn variables in the formulation of the LP relaxation. However, notice that the above proof yields the same non-approximability result for the problem with dedicated machines where the resource consumption of jobs is fixed beforehand. This because we can just define a_j as the (fixed) resource consumption of job j, and $p_j = 1$ as its (fixed) processing time; the remaining argument being the same.

Corollary 8.1. *There is no polynomial time approximation algorithm for dedicated parallel machine scheduling with a single resource that has a performance guarantee less than* 3/2*, unless* P=NP*.*

Finally, let us briefly address the quality of the LP lower bound used in the analysis. The following instance shows that our LP-based analysis cannot yield anything better than a 2-approximation, for both versions of the problem.

Example 1. *Consider the problem with $m = 2$ dedicated machines and k units of the additional resource, where k is odd. There are 2 jobs, each to be processed on its own machine, with the following resource-dependent processing times*

$$p_{js} = \begin{cases} 2k+1 & \text{if } s < \frac{k}{2} \\ k & \text{if } s > \frac{k}{2} \end{cases}$$

for both jobs j. □

In this example, we have the following, feasible solution for the LP-relaxation (20)–(24): $x_{js} = 1$ if $s = \lceil k/2 \rceil$, and $x_{js} = 0$ otherwise, for both jobs j. This setting of variables would yield $C \geq k$ by (22), and $kC \geq 2k\lceil k/2 \rceil = k(k+1)$ by (23). Therefore, $C = (k+1)$ is a feasible solution. In other words, we know that

$$C^{\mathrm{LP}} \leq k + 1\,,$$

but in the optimal solution, $C^{\mathrm{OPT}} = 2k$. Hence, the gap between C^{LP} and C^{OPT} can be as large as $2 - \varepsilon$, for any $\varepsilon > 0$. The bad quality of the LP lower bound is a consequence of the fact that we only use an aggregate formulation of the resource constraints in (23) (or (3), respectively), whereas any schedule has to respect the resource constraint at any time.

References

[1] J. Blazewicz, J. K. Lenstra and A. H. G. Rinnooy Kan, Scheduling subject to resource constraints: Classification and complexity, Discrete Applied Mathematics, **5** (1983), pp. 11–24.

[2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completenes*, W. H. Freeman, New York, 1979.

[3] R. L. Graham, *Bounds for certain multiprocessing anomalies*, Bell System Technical Journal, **45** (1966), pp. 1563–1581. See also [4].

[4] R. L. Graham, *Bounds on multiprocessing timing anomalies*, SIAM Journal on Applied Mathematics, **17** (1969), pp. 416–429.

[5] 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), pp. 287–326.

[6] A. Grigoriev, H. Kellerer and V. A. Strusevich, Scheduling parallel dedicated machines with the speeding-up resource, manuscript (2003). Extended abstract in: 6th Workshop on Models and Algorithms for Planning and Scheduling Problems, Aussois, France, 2003, pp. 131–132.

[7] H. Kellerer and V. A. Strusevich, Scheduling parallel dedicated machines under a single non-shared resource, European Journal of Operational Research, **147** (2003), pp. 345–364.

[8] H. Kellerer and V. A. Strusevich, Scheduling problems for parallel dedicated machines under multiple resource constraints, Discrete Applied Mathematics, **133** (2004), pp. 45–68.

[9] J. K. Lenstra, D. B. Shmoys and E. Tardos, Approximation algorithms for scheduling unrelated parallel machines, Mathematical Programming, Series A, **46** (1990), pp. 259–271.

[10] D. B. Shmoys and E. Tardos, An approximation algorithm for the generalized assignment problem, Mathematical Programming, Series A, **62** (1993), pp. 461–474.