Unrelated Parallel Machine 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
{a.grigoriev, m.uetz}@ke.unimaas.nl
[2] IBM T. J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA
sviri@us.ibm.com

Abstract. We consider unrelated parallel machine scheduling problems with the objective to minimize the schedule makespan. In addition to its machine-dependence, the processing time of any job is also dependent on the usage of a scarce renewable resource. An amount of k units of that resource, e.g. workers, can be distributed over the jobs in process, and the more of that resource is allocated to a job, the smaller its processing time. The model generalizes the classical unrelated machine scheduling problem, adding a resource-time tradeoff. It is also a natural variant of a generalized assignment problem studied previously by Shmoys and Tardos, the difference lying in the fact the resource is renewable and not a total budget constraint. We use a two-phased LP rounding technique to assign resources to jobs and jobs to machines. Combined with Graham's list scheduling, we thus prove the existence of a $(4+2\sqrt{2})$-approximation algorithm. We show how our approach can be adapted to scheduling problems with dedicated machines as well, with an improvement of the performance bound to $(3+2\sqrt{2})$. Moreover, we derive a lower bound of 2 for the employed LP-based analysis, and we prove a (3/2)-inapproximability result.

1 Introduction and Related Work

Unrelated parallel machine scheduling to minimize the makespan, $R||C_{\max}$ in the three-field notation of Graham et al. [6], 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 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 not fixed, the best approximation algorithm to date is a

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

M. Jünger and V. Kaibel (Eds.): IPCO 2005, LNCS 3509, pp. 182–195, 2005.

2-approximation by Lenstra, Shmoys and Tardos [10]. Moreover, the problem cannot be approximated within a factor smaller than 3/2, unless P=NP [10].

Shmoys and Tardos [12] 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 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} [12]. Note that, in both versions of the problem studied in [12], 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 of [12]. 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 allocated to 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 assumption on the costs and processing times in [12], 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 $p_{ij0} \geq \cdots \geq p_{ijk}$ for all jobs j and all machines i. The practical motivation to study this problem is evident; to give an example, one may think of production planning where additional (overtime) workers can be allocated to specific tasks within the production in order to reduce the production cycle time.

Related Work. In a manuscript by Grigoriev et al. [7], a restricted version 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, thus the availability of the additional resource is $k = 1$. Any job may be processed either with or without using that resource, with a reduced processing time if the resource is used. Finally, the number of machines m in their paper is considered fixed, and not part of the input. For that problem, they derive a $(3+\varepsilon)$–approximation, and for the problem with $m = 2$ machines, they derive (weak) NP-hardness and a fully polynomial time approximation scheme [7].

Jobs with resource dependent processing times also appear in the literature as *malleable* or *parallelizable tasks*, e.g. in [11, 13]. In these models, jobs can be processed on one or more parallel processors, and they have non-increasing processing times p_{js} in the number s of processors used. Any processor can only handle one job at a time, and the goal is to minimize the schedule makespan. Turek et al. [13] derive a 2–approximation algorithm for this problem. In fact, the model considered in [13] is just a special case of the problem considered in this paper. Interpreting the parallel processors as a generic 'resource' that must be allocated to jobs, the problem of [13] corresponds to the problem considered in this paper,

letting n jobs, with resource dependent processing times, be processed on $m = n$ *identical* parallel machines (instead of unrelated parallel machines). Mounie et al. [11] consider yet another variant, in that the processor allocations must be contiguous (for that problem, [13] includes a 2.7–approximation). Moreover, in [11] it is not only assumed that the processing times p_{js} are non-increasing in s, but also the processing 'areas' $s \cdot p_{js}$ are assumed to be non-decreasing in s. For that problem, a $\sqrt{3}$–approximation is derived in [11].

When we restrict even further, and assume that the decision on the allocation of resources to jobs is fixed beforehand, we are back at (machine) scheduling under resource constraints as introduced by Blazewicz et al. [1]. More recently, such problems with dedicated machines have been discussed by Kellerer and Strusevich [8, 9]. 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 in [8]. More precisely, 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 [8].

From the more practical viewpoint, a resource-time tradeoff problem has also been addressed by Chen [2]. He considers parallel, identical machine scheduling problems. Just like in the paper by Shmoys and Tardos [12], job processing times can be reduced by utilizing a non-renewable, monetary resource. The objective considered in [2] is either the total weighted completion times of jobs, or the weighted number of tardy jobs, incremented by the total resource consumption of the schedule. The paper describes branch and bound algorithms, together with computational results.

Results and Methodology. We derive a constant-factor approximation algorithm for the problem at hand. Our approach is based upon an integer linear program that defines a relaxation of the problem, extending a formulation used by Grigoriev et al. [7]. The main idea is the utilization of an aggregate version of the resource constraints, yielding a formulation that does not require time-indexed variables. More precisely, we use a formulation that takes as input all possible processing times p_{ijs} of jobs. 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 [12] 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 [4] to generate a feasible schedule. Using the LP lower bounds, we prove that this schedule is not more than a factor $(4 + 2\sqrt{2})$ 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 $(3 + 2\sqrt{2})$.

For both problems, unrelated and dedicated machines, we furthermore provide an instance showing that the linear programming based analysis cannot yield anything better than a 2 approximation.

Concerning lower bounds on the approximability, note that the problem at hand is a generalization of the classical unrelated machine scheduling problem $R||C_{\max}$. Therefore it cannot be approximated better than a multiplicative factor of 3/2, unless P=NP [10]. Restricting to the special case of dedicated machines, strong NP-hardness follows from Kellerer and Strusevich [8], hence the problem cannot admit an FPTAS, unless P=NP. This is true even for the case of a binary resource, i.e., if $k = 1$. We furthermore show that, for general k, the problem cannot be approximated better than a multiplicative factor of 3/2, unless P=NP.

2 Problem Definition

Let $V = \{1, \ldots, n\}$ be a set of jobs. Jobs must be processed non-preemptively on a set of m unrelated 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 $s \in \{0, 1, \ldots, k\}$ 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 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 set of jobs in process. Moreover, the amount of resources assigned to any job must be the same along its processing. In other words, if $s \leq k$ resources are allocated to some job j, and x_j denotes its starting time on some machine i, only $k - s$ of the resources are available for other jobs between x_j and $x_j + p_{ijs}$.

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

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

Considering this index set suffices, since in any solution, if s resources are allocated to some job j on machine i, we may as well use $s' = \min\{r \mid r \leq s,\ p_{ijr} = p_{ijs}\} \in S_{ij}$ resources for that job, without violating feasibility.

3 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 the original scheduling problem.

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

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

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

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

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

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. 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 later in rounding a solution for the linear relaxation of (IP). Notice that this integer program may have a feasible solution for some integer value of C, although no feasible schedule with makespan C exists; see Example 1 further below.

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

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

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 relaxation as (LP), and 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. Therefore, by using binary search, we can find in polynomial time the smallest integral 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 [12]. 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

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

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

$$\sum_{s \in S_{ij}, s \leq t^{ij}} x^{\mathrm{LP}}_{ijs} \geq (1 - \varepsilon)\, \tilde{y}_{ij}\,. \tag{6}$$

Then, for every machine i and job j define index $s^{ij} \geq t^{ij}$ as the minimizer of $s \cdot p_{ijs}$, for $s \geq t^{ij}$,

$$s^{ij} = \arg\min_{s \geq t^{ij}} \; s \cdot p_{ijs}\,. \tag{7}$$

By definition, it follows that $s^{ij} \in S_{ij}$. We now consider a fractional solution $\bar{x}$ defined by

$$\bar{x}_{ijs} = \begin{cases} \tilde{y}_{ij} & s = s^{ij}\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{8}$$

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 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*

$$\sum_{j \in V} \sum_{s \in S_{ij}} \bar{x}_{ijs}\, p_{ijs} \leq \frac{1}{1 - \varepsilon}\, C^{\mathrm{LP}}\,, \qquad i = 1, \ldots, m\,, \tag{9}$$

$$\sum_{j \in V} \sum_{i=1}^{m} \sum_{s \in S_{ij}} \bar{x}_{ijs}\, s\, p_{ijs} \leq \frac{k}{\varepsilon}\, C^{\mathrm{LP}}\,. \tag{10}$$

Proof. The proof of both claims is based on proving the statement for every machine-job pair. Validity of (9) can be seen as follows. We know that

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

by definition of t^{ij} in (6). By the fact that $p_{ijt^{ij}} \leq p_{ijs}$ for all $s \leq t^{ij}$, we therefore have

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

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

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

for all jobs $j \in V$ and machines $i = 1, \ldots, m$. Summing over $j \in V$, and using (2), inequalities (9) follow for any machine i.

To see (10), first observe that $\varepsilon\,\bar{x}_{ijs^{ij}} = \varepsilon\,\tilde{y}_{ij} \leq \sum_{s\in S_{ij}, s\geq t^{ij}} x^{\mathrm{LP}}_{ijs}$ by definition of t^{ij}, since t^{ij} is the *minimal* index with property (6). Therefore,

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

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

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

for all jobs $j \in V$ and machines $i = 1, \ldots, m$. Summing over $j \in V$ and all machines $i = 1, \ldots, m$, and using (3), eventually yields (10). □

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

Lemma 2 (Shmoys & Tardos [12–Theorem 2.1]). *Given a feasible fractional solution $\tilde{y} = (\tilde{y}_{ij})$ to the linear program*

$$\sum_{i=1}^{m} y_{ij} = 1 \ , \qquad \forall\ j \in V \tag{11}$$

$$\sum_{j \in V} y_{ij}\tau_{ij} \le T \ , \qquad \forall\ i = 1, \ldots, m\,, \tag{12}$$

$$\sum_{j \in V} \sum_{i=1}^{m} y_{ij}\lambda_{ij} \le \Lambda \ , \tag{13}$$

$$y_{ij} \ge 0 \ , \qquad \forall\ i, j. \tag{14}$$

with nonnegative parameters $T, \Lambda, \tau = (\tau_{ij})$, and $\lambda = (\lambda_{ij})$, there is a polynomial time algorithm which computes an integral solution $\bar{y}$ to (11), (13), (14), *and*

$$\sum_{j \in V} \bar{y}_{ij}\tau_{ij} \le T + \tau_{max} \ , \qquad \forall\ i = 1, \ldots, m\,, \tag{15}$$

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 (11)–(14), 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 1, the above result of Shmoys and Tardos, and the fact that

$$\tau_{\max} = \max_{i,j} p_{ijs^{ij}} \le \max_{i,j,s}\{p_{ijs} \mid x^{\mathrm{LP}}_{ijs} > 0\} \le C^{\mathrm{LP}} \tag{16}$$

by constraints (4), we can show the following.

Lemma 3. *Let $(C^{\mathrm{LP}}, x^{\mathrm{LP}})$ be an optimal fractional solution for the linear programming relaxation* (LP)*, then we can find a feasible solution $x^* = (x^*_{ijs})$ for the following integer linear program in polynomial time.*

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

$$\sum_{j \in V} \sum_{s \in S_{ij}} x_{ijs}\, p_{ijs} \le \left(1 + \frac{1}{1-\varepsilon}\right) C^{\mathrm{LP}} \ , \qquad \forall\ i\,, \tag{18}$$

$$\sum_{j \in V} \sum_{i=1}^{m} \sum_{s \in S_{ij}} x_{ijs}\, s\, p_{ijs} \le \frac{k}{\varepsilon}\, C^{\mathrm{LP}} \ , \tag{19}$$

$$x_{ijs} \in \{0, 1\} \ , \qquad \forall\ i, j, s\,. \tag{20}$$

Proof. We briefly summarize the previously described steps. Using the fractional solution $x^{\mathrm{LP}} = (x^{\mathrm{LP}}_{ijs})$, define $\tilde{y} = (\tilde{y}_{ij})$ as in (5), and apply the rounding defined

by (8). This yields a fractional solution $\bar{x} = (\bar{x}_{ijs})$ that is nonzero only for one resource index $s = s^{ij}$, for any pair of i and j, as defined in (7). Interpreting $\tilde{y}$ as fractional solution for the generalized assignment problem (11)–(14), use Lemma 2 to round it to an integral solution $\bar{y} = (\bar{y}_{ij})$. Now define the integral solution x^* by

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

With the help of Lemmas 1 and 2, and utilizing (16), it is now straightforward to verify that x^* fulfills (17)–(20). □

4 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 (17)–(20) obtained after the 2-phase rounding. Then the jobs are scheduled according to the greedy list scheduling algorithm of Graham [4], 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.

Theorem 1. *Algorithm* LP-Greedy *is a* $(4 + 2\sqrt{2})$*–approximation algorithm for unrelated parallel machine scheduling with resource dependent processing times.*

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 first need some additional notation. Consider some schedule $\mathcal{S}$ 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 $\mathcal{S}$, let $t(\beta)$ denote the earliest point in time after which only *big jobs* are processed, big jobs 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

$$C^{\mathrm{LPG}} = \alpha + \beta + \gamma\,. \tag{21}$$

Due to (18), we get that for machine i

$$\alpha \leq \sum_{j \in V} \sum_{s \in S_{ij}} x^*_{ijs}\, p_{ijs} \leq \left(1 + \frac{1}{1-\varepsilon}\right) C^{\mathrm{LP}}\ . \tag{22}$$

The next step is an upper bound on $\beta + \gamma$, the length of the final period where only big jobs are processed, together with the length of idle periods on machine i.

Lemma 4. *We have that*

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

Proof. First, observe that the total resource consumption of schedule $\mathcal{S}$ 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 (19). Hence, we obtain

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

Dividing by $2/k$ yields the claimed bound on $\beta + \gamma$. □

Now we are ready to prove the performance bound of Theorem 1.

Proof (of Theorem 1). First, use (21) together with (22) and Lemma 4 to obtain

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

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

5 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, and p_{js}, $s = 0, \ldots, k$, denotes the resource dependent processing time of job j.

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 $(4+2\sqrt{2})$–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.

$$\text{min.} \quad C \tag{23}$$

$$\text{s. t.} \quad \sum_{s \in S_j} x_{js} = 1 \ , \qquad \forall\, j \in V\,, \tag{24}$$

$$\sum_{j \in V_i} \sum_{s \in S_j} x_{js}\, p_{js} \le C \ , \qquad \forall\, i = 1, \ldots, m\,, \tag{25}$$

$$\sum_{j \in V} \sum_{s \in S_j} x_{js}\, s\, p_{js} \le k\, C \ , \tag{26}$$

$$x_{js} \ge 0 \ , \qquad \forall j, s\,. \tag{27}$$

Here, according to our previous notation, S_j are the breakpoints of the function p_{js}, hence

$$S_j = \{0\} \cup \{s \mid s \le k,\ p_{js} < p_{j,s-1}\} \subseteq \{0, \ldots, k\}\,.$$

This way, the accordingly adapted first phase rounding of (8) 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 $\sum_{s \in S_j, s \le t^j} x^{\mathrm{LP}}_{js} \ge 1 - \varepsilon$ and define index $s^j = \arg\min_{s \ge t^j} s \cdot p_{js}$. Again, it follows that $s^j \in S_j$. 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, Shmoys and Tardos' rounding is not required, and instead of using the bounds (18) and (19), we now have an integral solution $\bar{x} = (\bar{x}_{js})$ which fulfills the constraints

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

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

Validity of these bounds is proved along the same lines as Lemma 1. This eventually yields an improved performance bound for the dedicated machine model.

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

Lower Bound for the (Integer) Linear Program. We next give an instance to show that the integer linear program we use can be a factor 2 away from the optimal solution. Hence, our LP-based analysis cannot yield anything better than a 2-approximation, for both versions, the dedicated and the unrelated machine case.

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 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. □

We have the following, feasible integer solution for the LP-relaxation (23)–(27): $x_{js} = 1$ 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 (25), and $kC \geq 2k\lceil k/2 \rceil = k(k+1)$ by (26). Therefore, with $C = (k+1)$, there exists a feasible, even integral, solution for (23)–(27). A fortiori, we know that for the linear programming relaxation (LP), $C^{\mathrm{LP}} \leq k+1$. 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 obviously a consequence of the fact that we only use an aggregate formulation of the resource constraints in (26) (or (3), respectively), whereas any schedule has to respect the resource constraint at any time.

6 Lower Bounds on Approximation

The problem with unrelated machines cannot be approximated within a factor smaller than 3/2 as a generalization of the classical unrelated machine scheduling problem [10], as mentioned earlier. We next show that the same inapproximability result holds for the problem with dedicated machines.

Theorem 3. *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 PARTITION [3]: 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 $\sum_{j\in W} a_j = k$. 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

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

Hence, the a_j's are the only breakpoints in the functions p_{js}, and the index set $S_j = \{0, a_j\}$ for all jobs j. In other words, the functions p_{js} can be encoded in $\mathrm{O}(\log a_j)$ for all jobs j, and the transformation is indeed polynomial. 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. □

Finally, it is not difficult to see that the above proof yields the same inapproximability result for the problem with dedicated machines, even if the resource consumption of jobs is fixed beforehand.

Corollary 1. *There is no polynomial time approximation algorithm for resource constrained dedicated machine scheduling that has a performance guarantee less than* 3/2*, unless* P=NP*.*

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. Z.-L. CHEN, Simultaneous Job Scheduling and Resource Allocation on Parallel Machines, *Annals of Operations Research*, **129** (2004), pp. 135-153.
3. M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completenes*, W. H. Freeman, New York, 1979.
4. R. L. GRAHAM, Bounds for certain multiprocessing anomalies, *Bell System Technical Journal*, **45** (1966), pp. 1563–1581. See also [5].
5. R. L. GRAHAM, Bounds on multiprocessing timing anomalies, SIAM *Journal on Applied Mathematics*, **17** (1969), pp. 416–429.
6. 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.
7. A. GRIGORIEV, H. KELLERER AND V. A. STRUSEVICH, Scheduling parallel dedicated machines with the speeding-up resource, manuscript (2003). Extended abstract in: *Proceedings of the 6th Workshop on Models and Algorithms for Planning and Scheduling Problems*, Aussois, France, 2003, pp. 131–132.

8. 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.
9. H. Kellerer and V. A. Strusevich, Scheduling problems for parallel dedicated machines under multiple resource constraints, *Discrete Applied Mathematics*, **133** (2004), pp. 45–68.
10. 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.
11. G. Mounie, C. Rapine, and D. Trystram, Efficient Approximation Algorithms for Scheduling Malleable Tasks, *Proceedings of the 11th Annual ACM Symposium on Parallel Algorithms and Architectures*, 1999, pp. 23–32.
12. D. B. Shmoys and E. Tardos, An approximation algorithm for the generalized assignment problem, *Mathematical Programming*, Series A, **62** (1993), pp. 461–474.
13. J. Turek, J. L. Wolf, and P. S. Yu, Approximate Algorithms for Scheduling Parallelizable Tasks, *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, 1992, pp. 323–332.