

Han Hoogeveen · Martin Skutella · Gerhard J. Woeginger

Preemptive scheduling with rejection

Received: October 30, 2000 / Accepted: September 26, 2001

Published online: September 5, 2002 – © Springer-Verlag 2002

Abstract. We consider the problem of preemptively scheduling a set of n jobs on m (identical, uniformly related, or unrelated) parallel machines. The scheduler may reject a subset of the jobs and thereby incur job-dependent penalties for each rejected job, and he must construct a schedule for the remaining jobs so as to optimize the preemptive makespan on the m machines plus the sum of the penalties of the jobs rejected.

We provide a complete classification of these scheduling problems with respect to complexity and approximability. Our main results are on the variant with an arbitrary number of unrelated machines. This variant is APX-hard, and we design a 1.58-approximation algorithm for it. All other considered variants are weakly NP-hard, and we provide fully polynomial time approximation schemes for them. Finally, we argue that our results for unrelated machines can be carried over to the corresponding preemptive open shop scheduling problem with rejection.

Key words. scheduling – preemption – approximation algorithm – worst case ratio – computational complexity – in-approximability

1. Introduction

Consider a system with $m \geq 2$ (identical, uniformly related, or unrelated) parallel machines M_1, \dots, M_m and n jobs J_1, \dots, J_n . Job J_j ($j = 1, \dots, n$) has a *rejection penalty* e_j and a processing time p_{ij} on machine M_i ($i = 1, \dots, m$). In the case of identical machines, the processing times are machine independent, i.e., $p_{ij} \equiv p_j$. In the case of uniformly related machines, the i th machine M_i runs at speed s_i , and $p_{ij} = p_j/s_i$. In the case of unrelated machines, the processing times p_{ij} are arbitrarily structured. In the standard three-field scheduling notation (see e.g. Lawler, Lenstra, Rinnooy Kan & Shmoys [6]) identical machines are denoted by the letter P , uniformly related machines by Q , and unrelated machines by R .

We consider the following optimization problem in such systems: For each job J_j , we must decide whether to accept that job or whether to reject it. The accepted jobs are to be scheduled on the m machines. Preemption is allowed, i.e., a job may be arbitrarily in-

H. Hoogeveen: Utrecht University, Department of Computer Science, P.O.Box 80089, 3508TB Utrecht, The Netherlands, e-mail: slam@cs.uu.nl

M. Skutella: Technische Universität Berlin, Fakultät II — Mathematik und Naturwissenschaften, Institut für Mathematik, MA 6-1, Straße des 17. Juni 136, 10623 Berlin, Germany, e-mail: skutella@math.tu-berlin.de

Supported in part by the EU Thematic Network APPOL, Approximation and Online Algorithms, IST-1999-14084

G. J. Woeginger: University of Twente, Department of Mathematics, P.O.Box 217, 7500AE Enschede, The Netherlands, e-mail: g.j.woeginger@math.utwente.nl

Supported by the START program Y43-MAT of the Austrian Ministry of Science.

interrupted and resumed later on. Every machine can process at most one job at a time, and every job may be processed on at most one machine at a time. For the accepted jobs, we pay the makespan of the constructed schedule, i.e., the maximum job completion time in the schedule. For the rejected jobs, we pay the corresponding rejection penalties. In other words, the objective value is the preemptive makespan of the accepted jobs plus the total penalty of the rejected jobs. We denote this objective function by an entry “ $\text{Rej} + C_{\max}$ ” in the third field of the three-field scheduling notation. For example, $P5 | pmtn | \text{Rej} + C_{\max}$ denotes this problem on five identical machines; $Qm | pmtn | \text{Rej} + C_{\max}$ denotes the problem on uniformly related machines where the number of machines is a fixed constant m that is not part of the input; $R | pmtn | \text{Rej} + C_{\max}$ denotes the problem on unrelated machines where the number of machines is part of the input.

We also consider preemptive open shop scheduling with rejection. There are again n jobs J_1, \dots, J_n and m parallel machines M_1, \dots, M_m . Each job J_j consists of m operations O_{1j}, \dots, O_{mj} ; operation O_{ij} can only be processed on machine M_i and has processing time p_{ij} . No machine may process more than one operation at a time, and no two operations of the same job may be processed at the same time. Apart from this, the operations of a job may be processed in an arbitrary preemptive fashion. Again, we must decide, for each job J_j , whether to accept or reject it. This problem is denoted by $O | pmtn | \text{Rej} + C_{\max}$ and $Om | pmtn | \text{Rej} + C_{\max}$, respectively.

Related scheduling problems with rejection have been studied by Bartal, Leonardi, Marchetti-Spaccamela, Sgall & Stougie [2] for non-preemptive makespan on identical machines, by Engels, Karger, Kolliopoulos, Sengupta, Uma & Wein [4] for total weighted job completion time on a single machine, and by Sengupta [8] for lateness and tardiness criteria.

Complexity. Whereas classical preemptive makespan minimization (the problem where all jobs must be accepted) is polynomially solvable even on an arbitrary number of unrelated machines and also for open shop scheduling [6], preemptive makespan minimization with rejection is hard even in the case of two identical machines. A complete complexity classification is given in Table 1. In Section 4, we will prove weak NP-hardness of $P2 | pmtn | \text{Rej} + C_{\max}$ and $O2 | pmtn | \text{Rej} + C_{\max}$ and strong NP-hardness of $R | pmtn | \text{Rej} + C_{\max}$ and $O | pmtn | \text{Rej} + C_{\max}$. These results induce all negative results stated in Table 1. In Section 2 we discuss a dynamic program which leads to a pseudo-polynomial time algorithm for $Om | pmtn | \text{Rej} + C_{\max}$. Moreover, the results in Section 3 on uniformly related machines and the results in Section 2 on unrelated machines yield the existence of pseudo-polynomial time algorithms for $Q | pmtn | \text{Rej} + C_{\max}$ and $Rm | pmtn | \text{Rej} + C_{\max}$. Perhaps surprisingly, we did not manage to find ‘simple’ pseudo-polynomial time algorithms for these two problems. Instead, we took a detour and

Table 1. The complexity landscape of preemptive makespan with rejection

	Identical	Uniformly related	Unrelated/Open shop
m not part of input	weakly NP-hard, pseudo-polynomial	weakly NP-hard, pseudo-polynomial	weakly NP-hard, pseudo-polynomial
m part of input	weakly NP-hard, pseudo-polynomial	weakly NP-hard, pseudo-polynomial	strongly NP-hard

Table 2. The approximability landscape of preemptive makespan with rejection

	Identical	Uniformly related	Unrelated / Open shop
m not part of input	FPTAS	FPTAS	FPTAS
m part of input	FPTAS	FPTAS	1.58-approximation, APX-hard

constructed a fully polynomial time approximation scheme (FPTAS); the existence of the FPTAS then implies the existence of a pseudo-polynomial time algorithm. Anyway, these two positive results together with the open shop result mentioned above induce all other positive results stated in Table 1.

Approximability. Apart from their complexity, we are also interested in how close one can approach an optimum solution to these NP-hard scheduling problems in polynomial time. An α -approximation algorithm runs in polynomial time and constructs schedules whose values are within a factor of $\alpha > 1$ of the optimum solution value. The number α is called *performance guarantee* or *performance ratio* of the approximation algorithm. A family of polynomial time approximation algorithms with performance guarantee $1 + \varepsilon$ for all fixed $\varepsilon > 0$ is called a *polynomial time approximation scheme* (PTAS). If the running times of the approximation algorithms are even bounded by a polynomial in the input size and $\frac{1}{\varepsilon}$, then these algorithms build a *fully polynomial time approximation scheme* (FPTAS). It is known that unless $P=NP$, a strongly NP-hard optimization problem cannot possess an FPTAS (see e.g. Garey & Johnson [5]). Moreover, an APX-hard optimization problem (see e.g. [1] for a precise definition of APX-hardness) cannot possess a PTAS, unless $P=NP$.

Our approximability classification is given in Table 2. In Section 3 we will derive an FPTAS for the problem $Q | pmtn | Rej + C_{\max}$, and in Section 2 we derive another FPTAS for $Rm | pmtn | Rej + C_{\max}$ and $Om | pmtn | Rej + C_{\max}$. These results induce all FPTAS-entries in Table 2. The variants $R | pmtn | Rej + C_{\max}$ and $O | pmtn | Rej + C_{\max}$ with an arbitrary number of machines are APX-hard, even for the case of uniform rejection penalties (cf. Section 4). In Section 2, we construct a polynomial time $e/(e-1)$ -approximation algorithm for $R | pmtn | Rej + C_{\max}$ and $O | pmtn | Rej + C_{\max}$; note that $e/(e-1) \approx 1.58$. This approximation algorithm is based on a linear programming relaxation of the problems and the analysis of the performance guarantee also implies a bound of $e/(e-1)$ on the quality of this relaxation. Moreover, we present a class of instances showing that the ratio between the true optimum and the LP lower bound can be arbitrarily close to $e/(e-1)$.

Organization of the paper. Section 2 contains the positive results on unrelated machines and on open shops, and Section 3 contains the positive results on uniformly related machines. All negative results (NP-hardness and APX-hardness) are proved in Section 4.

2. Unrelated machines and open shops

In this section we derive a polynomial time $e/(e-1)$ -approximation algorithm for problem $R | pmtn | Rej + C_{\max}$ and an FPTAS for problem $Rm | pmtn | Rej + C_{\max}$.

Moreover, we argue that these results can be carried over to the open shop problems $O | pmtn | \text{Rej} + C_{\max}$ and $Om | pmtn | \text{Rej} + C_{\max}$, respectively.

Consider the following mixed integer linear programming formulation (1) of the problem $R | pmtn | \text{Rej} + C_{\max}$. For job J_j , the binary variable y_j decides whether J_j is rejected ($y_j = 0$) or accepted ($y_j = 1$). The variables x_{ij} describe which percentage of job J_j should be processed on machine M_i . The variable T denotes the optimal preemptive makespan for the accepted jobs.

$$\begin{aligned}
 \min \quad & T + \sum_{j=1}^n (1 - y_j) e_j \\
 \text{s.t.} \quad & \sum_{j=1}^n x_{ij} p_{ij} \leq T \quad \text{for } i = 1, \dots, m \\
 & \sum_{i=1}^m x_{ij} p_{ij} \leq T \quad \text{for } j = 1, \dots, n \\
 & \sum_{i=1}^m x_{ij} = y_j \quad \text{for } j = 1, \dots, n \\
 & x_{ij} \geq 0 \quad \text{for } i = 1, \dots, m \text{ and } j = 1, \dots, n \\
 & y_j \in \{0, 1\} \quad \text{for } j = 1, \dots, n
 \end{aligned} \tag{1}$$

The first set of restrictions states that for every machine the total assigned processing time is at most T . The second set of restrictions states that the total processing time of every job cannot exceed T . The third set of restrictions connects the binary decision variables y_j with the continuous variables x_{ij} . If we want to schedule every job J_j on the m machines according to the values x_{ij} , then we essentially are dealing with a preemptive open shop problem; it is well-known [6] that the smallest number T fulfilling the first two sets of constraints in (1) yields the optimal preemptive makespan. To summarize, every feasible solution of (1) corresponds to a feasible schedule with objective value $T + \sum_{j=1}^n (1 - y_j) e_j$.

Now we replace the integrality conditions $y_j \in \{0, 1\}$ in (1) by $0 \leq y_j \leq 1$. This yields the linear programming relaxation LPR which can be solved to optimality in polynomial time. Let x_{ij}^* , y_j^* , and T^* constitute an optimal solution to LPR. From this solution, we compute a *rounded* solution \tilde{x}_{ij} , \tilde{y}_j , and \tilde{T} for (1) in the following way: We randomly choose a threshold α from the uniform distribution over $[1/e, 1]$. If $y_j^* \leq \alpha$, then we set $\tilde{y}_j := 0$, and otherwise we set $\tilde{y}_j := 1$. Similar *dependent randomized rounding* procedures have already proven useful in other contexts (see e.g. Bertsimas, Teo & Vohra [3]).

For j with $\tilde{y}_j = 0$, we set all variables $\tilde{x}_{ij} = 0$. For j with $\tilde{y}_j = 1$, we set all variables $\tilde{x}_{ij} := x_{ij}^*/y_j^*$. Finally, we set

$$\tilde{T} := \max \left\{ \max_{1 \leq i \leq m} \sum_{j=1}^n \tilde{x}_{ij} p_{ij}, \max_{1 \leq j \leq n} \sum_{i=1}^m \tilde{x}_{ij} p_{ij} \right\}. \tag{2}$$

It can be verified that the values \tilde{x}_{ij} , \tilde{y}_j , and \tilde{T} constitute a feasible solution of (1): All variables \tilde{y}_j are binary. For j with $\tilde{y}_j = 0$, the variables \tilde{x}_{ij} add up to 0. For j with $\tilde{y}_j = 1$, the variables \tilde{x}_{ij} add up to $\sum_i x_{ij}^*/y_j^* = 1$. Finally, in (2) the value of \tilde{T} is fixed to fulfill the first and the second set of restrictions.

Now let us analyze the quality of the rounded solution. For any fixed value of α , \tilde{x}_{ij} is less than a factor of $1/\alpha$ above x_{ij}^* , and hence by linearity also \tilde{T} is less than a factor of $1/\alpha$ above T^* . Therefore, the expected multiplicative increase in the makespan is at most a factor of

$$\frac{e}{e-1} \int_{1/e}^1 1/\alpha \, d\alpha = \frac{e}{e-1} .$$

In the LPR solution, the contribution of job J_j to the total penalty is $(1 - y_j^*)e_j$. The expected contribution of J_j to the penalty in the rounded solution is

$$e_j \cdot \Pr[y_j^* \leq \alpha] = e_j \int_{\max\{1/e, y_j^*\}}^1 \frac{e}{e-1} d\alpha \leq e_j \int_{y_j^*}^1 \frac{e}{e-1} d\alpha = \frac{e}{e-1} (1 - y_j^*) e_j .$$

All in all, the expected objective value for the rounded solution is at most a factor of $e/(e - 1) \approx 1.58$ above the optimal objective value of LPR. Hence, our procedure yields a *randomized* polynomial time $e/(e - 1)$ -approximation algorithm. Since the only critical values for the threshold parameter α are the values y_j^* ($j = 1, \dots, n$), it is straightforward to derandomize this algorithm in polynomial time.

Theorem 1. *The problem $R \mid pmtn \mid \text{Rej} + C_{\max}$ possesses a deterministic polynomial time $e/(e - 1)$ -approximation algorithm. \square*

Since in our analysis the value of the computed schedule was compared to the lower bound given by the value of an optimum solution to the linear programming relaxation LPR, the bound $e/(e - 1)$ also holds for the quality of LPR. Moreover, we can show that this result is tight.

Corollary 1. *The integrality gap of the linear programming relaxation LPR is at most $e/(e - 1)$; this bound is tight, even for the special case of identical parallel machines and uniform rejection penalties $e_j \equiv 1$.*

Proof. It remains to show that the given bound is tight. For each positive integer q we construct an instance with $(q + 1)^q$ identical machines and the same number of jobs with uniform rejection penalties; the processing time of the j th job is set to $p_j = j$. Then, the total rejection penalty in any reasonable schedule with makespan T is equal to $(q + 1)^q - T$; in particular, the value of an optimal schedule is $(q + 1)^q$. However, consider the following feasible solution to LPR:

$$x_{jj} = y_j = \begin{cases} 1 & \text{if } j \leq q^q, \\ q^q/j & \text{if } j > q^q, \end{cases} \quad \text{and} \quad T = q^q .$$

The value of this solution is equal to

$$T + \sum_{j=1}^{(q+1)^q} (1 - y_j) = q^q + \sum_{j=q^q+1}^{(q+1)^q} (1 - q^q/j) = (q + 1)^q - q^q \sum_{j=q^q+1}^{(q+1)^q} \frac{1}{j} .$$

The ratio of this value and the value of an optimal schedule is

$$1 - \left(\frac{q}{q+1}\right)^q \sum_{j=q^q+1}^{(q+1)^q} \frac{1}{j} . \tag{3}$$

Since

$$\int_{q^q+1}^{(q+1)^q+1} \frac{1}{z} dz \leq \sum_{j=q^q+1}^{(q+1)^q} \frac{1}{j} \leq \int_{q^q}^{(q+1)^q} \frac{1}{z} dz$$

and the terms on the left and on the right hand side converge to 1 when q goes to infinity, the same holds for the sum in between. Thus, for large q , the ratio (3) tends to $(e - 1)/e$. This completes the proof. \square

The mixed integer linear program (1) can easily be adapted to the open shop scheduling problem $O | pmtn | \text{Rej} + C_{\max}$. Remove the third and fourth constraints of (1) and replace all variables x_{ij} by y_j . Then, an appropriate adaption of the rounding algorithm discussed above is also an $e/(e - 1)$ -approximation algorithm for $O | pmtn | \text{Rej} + C_{\max}$ and Corollary 1 holds as well.

Theorem 2. *The problem $O | pmtn | \text{Rej} + C_{\max}$ possesses a deterministic polynomial time $e/(e - 1)$ -approximation algorithm.* \square

Let us turn to problem $Rm | pmtn | \text{Rej} + C_{\max}$. The crucial fact for deriving positive results on this problem is the following discretization lemma.

Lemma 1. *Let δ be a real number with $0 < \delta \leq 1/m$, such that $1/\delta$ is integer. Then, the mixed integer linear program (1) possesses a feasible solution, in which the values x_{ij} all are integer multiples of δ^3 and whose objective value is at most a factor of $1 + \delta$ above the optimal objective value of (1).*

Proof. Consider an optimal solution x_{ij}^*, y_j^* , and T^* of the mixed integer linear program (1). Another feasible solution \tilde{x}_{ij} and \tilde{y}_j for (1) is constructed job-wise in the following way. For job J_j , let $\ell(j)$ denote a machine index that maximizes $x_{\ell(j),j}^*$, i.e., an index with $x_{\ell(j),j}^* \geq x_{ij}^*$ for all $1 \leq i \leq m$. Then for $i \neq \ell(j)$, \tilde{x}_{ij} is the value x_{ij}^* rounded down to the next multiple of δ^3 . Moreover, we set $\tilde{y}_j = y_j^*$ and $\tilde{x}_{\ell(j),j} = \tilde{y}_j - \sum_{i \neq \ell(j)} \tilde{x}_{ij}$. Finally, \tilde{T} is computed according to (2). It is straightforward to verify that \tilde{x}_{ij} , \tilde{y}_j , and \tilde{T} is feasible for (1). By construction, the values \tilde{x}_{ij} all are integer multiples of δ^3 for $i \neq \ell(j)$. Moreover, this is also true for $\tilde{x}_{\ell(j),j}$ since $\tilde{y}_j \in \{0, 1\}$ and $1/\delta$ is integer.

We claim that for all $j = 1, \dots, n$ and $i = 1, \dots, m$, the inequality $\tilde{x}_{ij} \leq (1 + \delta)x_{ij}^*$ is fulfilled. If $y_j^* = 0$, this inequality trivially holds since $\tilde{y}_j = \tilde{x}_{ij} = 0$ for $i = 1, \dots, m$ then. Otherwise, if $i \neq \ell(j)$, the inequality holds since $x_{ij}^* - \delta^3 < \tilde{x}_{ij} \leq x_{ij}^*$. Moreover, for $i = \ell(j)$ we have

$$\tilde{x}_{\ell(j),j} = \tilde{y}_j - \sum_{i \neq \ell(j)} \tilde{x}_{ij} < y_j^* - \sum_{i \neq \ell(j)} (x_{ij}^* - \delta^3) < x_{\ell(j),j}^* + m\delta^3 \leq (1 + \delta)x_{\ell(j),j}^* .$$

The first inequality follows from the definition of the \tilde{x}_{ij} with $i \neq \ell(j)$. The second inequality is straightforward. The last inequality is equivalent to $m\delta^2 \leq x_{\ell(j),j}^*$; this is true since $\delta \leq 1/m$ and $x_{\ell(j),j}^* \geq y_j^*/m = 1/m$. Summarizing, the claimed inequalities are indeed fulfilled. Since $\tilde{y}_j \equiv y_j$, the objective value in (1) increases at most by a factor of $1 + \delta$. \square

In the following, we call a feasible solution of (1) where all values x_{ij} are integer multiples of δ^3 as in Lemma 1 a δ -discrete feasible solution. Moreover, we assume without loss of generality that all processing times p_{ij} and rejection penalties e_j are integral. Our next goal is to show that the best δ -discrete feasible solution can be computed in pseudo-polynomial time by a dynamic programming approach. A state of the dynamic program encodes a partial schedule for the first k jobs ($1 \leq k \leq n$). Every state has $m + 2$ components. The first m components store the loads of the m machines in the partial schedule. Component $m + 1$ stores the length of the longest job scheduled so far (i.e., the maximum time that any job needs in the schedule). Component $m + 2$ stores the total penalty of all jobs from J_1, \dots, J_k that have been rejected so far. The state space S_0 is initialized with the all-zero vector. When job J_k is treated, every state \vec{s} from the state space S_{k-1} is updated and yields several new states.

- First, job J_k may be rejected. The corresponding new state results from adding the penalty e_k to the last component of \vec{s} .
- Otherwise, job J_k is accepted. We try all $O(1/\delta^{3m})$ possibilities for the m pieces x_{1j}, \dots, x_{mj} that are integer multiples of δ^3 and that add up to 1. For each appropriate combination the i th ($i = 1, \dots, m$) component of \vec{s} is increased by $x_{ij} p_{ij}$. The new $(m + 1)$ th component is the maximum of the old $(m + 1)$ th component and $\sum_{i=1}^m x_{ij} p_{ij}$.

Finally, after treating the last job J_n we compute the objective values for all states in S_n and output the best one; the objective value equals the maximum of the first $m + 1$ components plus the last component. The running time of this dynamic program is polynomial in n , $1/\delta$, and in the size of the state spaces. Component i ($i = 1, \dots, m$) indicates the load of machine i , which is measured in units of δ^3 ; hence, the number of possible states for component i is $O(\sum_{j=1}^n p_{ij}/\delta^3)$. Similarly, the number of possible states for component $(m + 1)$ is $O(\sum_{i=1}^m p_{ij}/\delta^3)$. Finally, the number of possible states for component $m + 2$ is $O(\sum_{j=1}^n e_j)$. Clearly, this yields a pseudo-polynomial running time.

Lemma 2. *For any instance of $Rm \mid pmtn \mid \text{Rej} + C_{\max}$ and for any δ with $0 < \delta \leq 1/m$ and $1/\delta$ integer, the best δ -discrete schedule can be computed in pseudo-polynomial time. \square*

By applying standard methods, this dynamic programming formulation can be transformed into a fully polynomial time approximation scheme; in fact, the dynamic program belongs to the class of so-called *ex-benevolent* dynamic programs (Woeginger [9]), and therefore automatically leads to an FPTAS for computing the best δ -discrete feasible solution. Finally, let us turn back to the general problem $Rm \mid pmtn \mid \text{Rej} + C_{\max}$. For a given $\varepsilon > 0$, we set $\delta = \min\{1/m, 1/\lceil 3/\varepsilon \rceil\}$ and then compute in fully polynomial time a $(1 + \varepsilon/3)$ -approximation for the best δ -discrete feasible solution. It is easily verified that this yields a $(1 + \varepsilon)$ -approximation of the optimal objective value; hence there is an FPTAS for $Rm \mid pmtn \mid \text{Rej} + C_{\max}$.

It is known that every sufficiently well-behaved optimization problem with an FPTAS is solvable in pseudo-polynomial time (see e.g. Theorem 6.8 in Garey & Johnson [5]). Here, well-behaved means that all solution values are positive integers and that the value

of an optimal solution is polynomially bounded in the size of a unary encoding of the input. While the latter condition is satisfied for $Rm \mid pmtn \mid \text{Rej} + C_{\max}$, the first condition is in general violated. However, it follows from the theory of linear inequalities that the denominator of the value of a fractional extreme solution to the mixed integer linear program (1) is polynomially bounded in the size of a unary encoding of the input (see e.g. Schrijver [7, Theorem 10.1]). Thus, there is a pseudo-polynomially bounded $\varepsilon > 0$ such that any $(1 + \varepsilon)$ -approximative extreme solution to (1) is optimal. The FPTAS discussed above can be used to compute such a solution in pseudo-polynomial time.

Theorem 3. *The problem $Rm \mid pmtn \mid \text{Rej} + C_{\max}$ has an FPTAS, and it is solvable in pseudo-polynomial time.* □

Compared to $Rm \mid pmtn \mid \text{Rej} + C_{\max}$, the situation for the open shop scheduling problem $Om \mid pmtn \mid \text{Rej} + C_{\max}$ is less complicated. Since each operation of a job can only be processed on one machine, there is no need for a discretization lemma. Instead, we can directly solve the problem via dynamic programming in pseudo-polynomial time. The dynamic program is a simplified version of the one discussed above; in particular, it uses the same states. However, in contrast to the case of unrelated machines, if a job J_k is accepted, there is only one possibility of adding it to one of the current partial solutions \vec{s} : for each operation O_{ik} , add its processing time p_{ik} to the i th component of state \vec{s} . Again, this dynamic programming formulation can be transformed into a fully polynomial time approximation scheme by applying standard methods.

Theorem 4. *The problem $Om \mid pmtn \mid \text{Rej} + C_{\max}$ is solvable in pseudo-polynomial time, and it has an FPTAS.* □

3. Uniformly related machines

In this section we will construct an FPTAS and a pseudo-polynomial time algorithm for $Q \mid pmtn \mid \text{Rej} + C_{\max}$. Our line of approach is quite similar to that for $Rm \mid pmtn \mid \text{Rej} + C_{\max}$ in Section 2 which also gave an FPTAS and a pseudo-polynomial time algorithm.

Now consider an instance of $Q \mid pmtn \mid \text{Rej} + C_{\max}$ with m machines and n jobs. Without loss of generality we assume that $m = n$ holds: If $m > n$, then the $m - n$ slowest machines will not be used in any reasonable schedule and may be removed from the instance. If $m < n$, then we introduce $n - m$ dummy machines of speed 0; these dummy machines will not be used in any reasonable schedule. Let $s_1 \geq s_2 \geq \dots \geq s_n$ denote the speeds of the machines (so that processing of a job piece of length L on machine M_i takes L/s_i time). For $i \leq n$ let $S_i = \sum_{k=1}^i s_k$ denote the total speed of the i fastest machines.

Let $a_1 \geq a_2 \geq \dots \geq a_q$ denote the lengths of the q accepted jobs in some schedule. For $i \leq q$ let $A_i = \sum_{k=1}^i a_k$ denote the total length of the i longest accepted jobs. It is well-known [6] that for $m = n$ machines the optimal preemptive makespan for the accepted jobs equals

$$\max_{1 \leq i \leq q} A_i / S_i . \tag{4}$$

This leads to the following dynamic programming formulation of $Q \mid pmtn \mid \text{Rej} + C_{\max}$. Without loss of generality we assume that $p_1 \geq p_2 \geq \dots \geq p_n$, i.e., that the jobs

are ordered by non-increasing processing times. Every state of the dynamic program consists of four values $v_1, v_2, v_3,$ and v_4 and encodes a schedule for a prefix J_1, \dots, J_k of the job sequence. Value v_1 stores the total penalty of the jobs rejected so far, value v_2 stores the total processing time of the jobs accepted so far, value v_3 stores the number of accepted jobs, and value v_4 stores the maximum value A_i/S_i over $1 \leq i \leq v_3$. How do we update a state $[v_1, v_2, v_3, v_4]$ for $J_1, \dots, J_k,$ if also job J_{k+1} has to be considered?

- If job J_{k+1} is rejected, we replace v_1 by $v_1 + e_{k+1}$ and leave everything else unchanged. This yields the state $[v_1 + e_{k+1}, v_2, v_3, v_4]$.
- If job J_{k+1} is accepted, we define $v_2^{new} := v_2 + p_{k+1}$ and $v_3^{new} := v_3 + 1$. Moreover, v_4^{new} becomes the maximum of the old component v_4 and v_2^{new} divided by $S_{v_3^{new}}$. This yields the state $[v_1, v_2^{new}, v_3^{new}, v_4^{new}]$.

We handle job by job in this way, until we end up with a state space for J_1, \dots, J_n . Then we extract from every state $[v_1, v_2, v_3, v_4]$ its objective value $v_1 + v_4$. The state with the best objective value gives the solution of $Q | pmtn | \text{Rej} + C_{\max}$. The time complexity of this dynamic programming formulation mainly depends on the number of states. Since every component in every state is a number whose size is bounded by the input size, the total number of states is pseudo-polynomial. Moreover, we can prove that this dynamic program belongs to the class of *benevolent* dynamic programming formulations [9]. Hence, it can be transformed into an FPTAS by trimming the state space appropriately. Finally, the same arguments as in Section 2 yield the existence of an exact pseudo-polynomial time algorithm.

Theorem 5. *The problem $Q | pmtn | \text{Rej} + C_{\max}$ has an FPTAS, and it is solvable in pseudo-polynomial time. \square*

4. Negative results

In this section we prove the following negative results: the NP-hardness of the problems $P2 | pmtn | \text{Rej} + C_{\max}$ and $O2 | pmtn | \text{Rej} + C_{\max}$, and the APX-hardness of $R | pmtn | \text{Rej} + C_{\max}$ and $O | pmtn | \text{Rej} + C_{\max}$. The strong NP-hardness of the problems $R | pmtn | \text{Rej} + C_{\max}$ and $O | pmtn | \text{Rej} + C_{\max}$ follows along the same lines: our L -reductions (from the APX-hard maximum bounded 3-dimensional matching problem) at the same time constitute Turing-reductions (from the strongly NP-hard 3-dimensional matching problem). Moreover, we note that our L -reduction to the problem $R | pmtn | \text{Rej} + C_{\max}$ also implies APX-hardness and strong NP-hardness for the *non-preemptive* problem variant $R || \text{Rej} + C_{\max}$.

Theorem 6. *The problems $P2 | pmtn | \text{Rej} + C_{\max}$ and $O2 | pmtn | \text{Rej} + C_{\max}$ are NP-hard in the ordinary sense.*

The following proof for the problem $P2 | pmtn | \text{Rej} + C_{\max}$ can easily be modified to yield the hardness result for the open shop problem $O2 | pmtn | \text{Rej} + C_{\max}$.

Proof. The proof is a straightforward reduction from PARTITION. Consider an instance of PARTITION, i.e., n positive integers a_1, \dots, a_n that add up to $2A$. The question is

whether there exists an index set $I \subset \{1, \dots, n\}$ with $\sum_{j \in I} a_j = A$. We introduce $n + 1$ jobs. The jobs J_j with $1 \leq j \leq n$ have penalties a_j and processing times $3a_j$. The job J_{n+1} has penalty $5A$ and processing time $3A$.

We claim that the instance of PARTITION has answer YES if and only if there exists a preemptive schedule with objective value at most $4A$. (Only if): Suppose that there exists an index set I with $\sum_{j \in I} a_j = A$. Process all jobs J_j with $j \in I$ on machine M_1 . Process job J_{n+1} on machine M_2 . Reject all remaining jobs. The resulting schedule has makespan $3A$ and total penalty A ; hence, its objective value equals $4A$. (If): Suppose that there exists a schedule with objective value at most $4A$. Then job J_{n+1} has been accepted, and hence the makespan is at least $3A$. Denote by X the total penalty of the rejected jobs. Since the makespan is $\geq 3A$ and the objective value is at most $4A$, we must have $X \leq A$. The total processing time of the accepted jobs is equal to $3(2A - X)$ (for the jobs $1 \leq j \leq n$) plus $3A$ (for job J_{n+1}). The preemptive makespan on two machines is at least the total scheduled processing time divided by 2. Hence, the objective value of this schedule is at least

$$X + \frac{1}{2}(9A - 3X) = \frac{1}{2}(9A - X) .$$

This must be no more than $4A$, which implies that $X \geq A$. Hence, we conclude that $X = A$, which implies that PARTITION has answer YES. \square

Now we turn to problem $R | pmtn | \text{Rej} + C_{\max}$. The APX-hardness proof is done for the special case of uniform rejection penalties $e_j \equiv 1$ and so-called *restricted assignment*, where the processing times of jobs are not machine-dependent but each job may only be processed on a subset of machines, i.e., $p_{ij} \in \{p_j, \infty\}$. We provide an L -reduction from the APX-hard maximum bounded 3-dimensional matching problem.

Maximum bounded 3-dimensional matching (Max-3DM-B)

Input: Three sets $A = \{a_1, a_2, \dots, a_q\}$, $B = \{b_1, b_2, \dots, b_q\}$ and $C = \{c_1, c_2, \dots, c_q\}$.

A subset T of $A \times B \times C$ of cardinality s , such that any element of A , B and C occurs in exactly one, two, or three triples in T . Note that this implies that $q \leq s \leq 3q$.

Goal: Find a subset T' of T of maximum cardinality such that no two triples of T' agree in any coordinate.

Measure: The cardinality of T' .

Without loss of generality, we restrict ourselves to instances of MAX-3DM-B where the value q and the value of an optimal solution both are even. Notice that an arbitrary instance can easily be modified to fulfill these requirements by taking two disjoint copies of the instance. The following simple observation will be useful.

Lemma 3. *For any instance I of MAX-3DM-B we have $\text{OPT}(I) \geq \frac{1}{7}s$.*

Proof. Select an arbitrary triple t from T . Remove t together with all triples that agree with t in some coordinate from T . Repeat this process until T becomes empty. Since every element occurs in at most 3 triples, at most 7 triples are removed from T in every step. Hence, there are at least $\frac{1}{7}s$ steps and at least $\frac{1}{7}s$ selected triples. Since the selected triples do not agree in any coordinate, they form a feasible 3-dimensional matching. \square

Let $I = (q, T)$ be an instance of MAX-3DM-B. We construct an instance $R(I)$ of the scheduling problem $R | pmtn, e_j \equiv 1, p_{ij} \in \{p_j, \infty\} | \text{Rej} + C_{\max}$ with $s + 22q$ jobs and $s + 17q$ machines, where all penalties e_j are 1 and the processing time of job J_j on machine i is either p_j or infinite (i.e., a job can only be processed on a subset of machines). The *core* of the instance consists of $s + 7q$ jobs and $s + 2q$ machines. There are further $15q$ *non-core machines* and $15q$ *non-core jobs*. The non-core jobs are matched to the non-core machines. The processing time of each non-core job is $15q$ on its matching non-core machine, and it is infinite on all other (core and non-core) machines. Processing of a core job on a non-core machine also takes infinite time (and thus is impossible).

Now we continue our description of the core of the instance. There are s machines, which correspond to the triples in T , and therefore are called the *triple machines*. Moreover, there are $2q$ so-called *element machines*. As to the jobs, each a_j, b_j , and c_j element corresponds to an *element job* with processing time $5q$. An element job can be processed on any element machine; moreover, each triple machine can process the element jobs of the elements occurring in the corresponding triple. Each triple machine has its own matching dummy job; processing this dummy job takes $15q$ units of time, and no other dummy job can be processed on the machine. Each element machine has two matching dummy jobs with processing times $5q$ and $10q$, respectively; again, no other dummy job can be processed on an element machine.

As we will see later, the sole purpose of adding the $15q$ non-core machines with corresponding non-core jobs is to enforce that in the optimal schedule $C_{\max} \geq 15q$. The following lemma gives the basic intuition of how the reduction works.

Lemma 4. *If the optimal solution to an instance I of MAX-3DM-B consists of k triples, then there is a solution to the instance $R(I)$ of the scheduling problem with objective value $16q + (q - k)/2$.*

Proof. Without loss of generality, we assume that the first k triples in T constitute an optimal solution of I . We construct the following solution with makespan $15q$ to instance $R(I)$. The first k triple machines process the element jobs belonging to their triples; the dummy jobs corresponding to the first k triple machines are rejected. The remaining $3(q - k)$ element jobs are grouped into $3(q - k)/2$ pairs which are then processed on an arbitrary subset of $3(q - k)/2$ element machines; the corresponding $3(q - k)/2$ dummy jobs of size $10q$ are rejected. This yields a schedule with $C_{\max} = 15q$ and $k + 3(q - k)/2$ rejected jobs. Hence, the objective value is equal to $16q + (q - k)/2$. \square

The following lemma shows that the schedule constructed in the proof of Lemma 4 in fact is optimal.

Lemma 5. *Let I be an instance of MAX-3DM-B and $0 \leq k \leq q$. Given a solution σ to the scheduling instance $R(I)$ with objective value $c(\sigma) < 16q + (q - k)/2$, one can construct in polynomial time a solution $S(\sigma)$ to I consisting of at least $k + 1$ triples.*

Proof. If the makespan of the given schedule is less than $15q$, then at least $17q + s$ dummy jobs (one for each machine) must have been rejected. Thus, the objective value is at least $17q + s$ which is a contradiction to $c(\sigma) < 16q + (q - k)/2$; this yields $C_{\max} = 15q + \Delta$ for some $\Delta \geq 0$.

If all dummy jobs of length $10q$ are rejected, then the capacity of the $2q$ element machines suffices to process all element jobs and all dummy jobs of length $5q$ within the interval $[0, 15q]$. Thus, if an element job or a dummy job of length $5q$ has been rejected in the given schedule and if it cannot be added to any element machine without increasing the makespan, then there must be at least one dummy job of length $10q$ which was not rejected. Interchanging the two jobs does not deteriorate the value of the schedule. Thus, we can modify the given schedule such that no element job and no dummy job of length $5q$ is rejected. We denote the number of rejected jobs in the resulting schedule by R ; notice that the makespan of this schedule is still bounded by $15q + \Delta$ and $R + \Delta < q + (q - k)/2$.

We consider the triple machines iteratively one after another and construct a solution of instance I ; at the same time, we also modify the current schedule accordingly: If a triple machine processes (fractions of) element jobs for more than $10q$ time units, then we add the corresponding triple to the solution of I . Since the load of the triple machine is at most $15q + \Delta < 17q$, its dummy job must have been rejected; we move all fractions of the three corresponding element jobs to the triple machine increasing its load to $15q$. We denote the cardinality of the resulting solution of instance I by k' . It remains to show that $k' > k$.

We bound the total amount of time that is used in the resulting schedule by triple machines and by element machines for processing element jobs:

- Any triple machine which corresponds to one of the k' chosen triples spends $15q$ time units for processing element jobs.
- Any other machine which does not process all its dummy jobs spends at most $10q + \Delta$ time units for processing element jobs; there are $(R - k')$ such machines.
- Each of the $s + 2q - R$ remaining machines spends at most Δ time units for processing element jobs.

Summarizing, the total processing time of all element jobs is at most

$$\begin{aligned} 15qk' + (10q + \Delta)(R - k') + \Delta(s + 2q - R) &\leq 5qk' + 10qR + 5q\Delta \\ &\leq 5qk' + 10q(R + \Delta) \\ &< 5q(k' - k) + 5q \cdot 3q . \end{aligned}$$

The last inequality follows from $R + \Delta < q + (q - k)/2$. Since the total processing time of all $3q$ element jobs is $3q \cdot 5q$, we get $k' > k$ which concludes the proof. \square

Lemmas 4 and 5 together yield the following result.

Corollary 2. *If an optimal solution to the instance I of MAX-3DM-B consists of k triples, then the value of an optimum solution to the instance $R(I)$ of the scheduling problem is equal to $16q + (q - k)/2$.* \square

We can now state the main result of this section (Since the notion of preemption is not used in the proof of Lemmas 4 and 5, we can use the very same L -reduction to establish APX-hardness of the nonpreemptive problem $R | e_j \equiv 1, p_{ij} \in \{p_j, \infty\} | \text{Rej} + C_{\max}$).

Theorem 7. *The problem $R | p_{mtn}, e_j \equiv 1, p_{ij} \in \{p_j, \infty\} | \text{Rej} + C_{\max}$ is APX-hard.*

Proof. Our L -reduction now looks as follows. Given an instance I of MAX-3DM-B, we construct the instance $R(I)$ of the problem $R \mid pmtn, e_j \equiv 1, p_{ij} \in \{p_j, \infty\} \mid \text{Rej} + C_{\max}$ as described above. The transformation S that maps a given solution for $R(I)$ to a feasible solution of I is given in the proof of Lemma 5. Clearly, R and S can be implemented to run in polynomial time. Moreover, we have for any instance I of MAX-3DM-B that

$$\text{OPT}(R(I)) \leq 17q \leq 17s \leq 119 \text{OPT}(I) ;$$

the first inequality follows from Lemma 4 and the last inequality from Lemma 3. Finally, for any feasible schedule σ of $R(I)$, the feasible solution $S(\sigma)$ of instance I fulfills the inequality

$$\text{OPT}(I) - |S(\sigma)| \leq 2(c(\sigma) - \text{OPT}(R(I)))$$

by Lemma 5 and Corollary 2. □

In order to prove the same result for open shop scheduling, one can use a similar L -reduction from MAX-3DM-B, where, however, the role of the machines and the jobs is reversed. We give a short sketch of the reduction and its proof.

Let $I = (q, T)$ be an instance of MAX-3DM-B. We construct an instance $O(I)$ of the scheduling problem $O \mid pmtn, e_j \equiv 1 \mid \text{Rej} + C_{\max}$ with $s + 21q$ jobs and $18q$ machines, where all penalties e_j are 1. The core of the instance consists of $s + 6q$ jobs and $3q$ machines. There are further $15q$ non-core machines and $15q$ non-core jobs. The non-core jobs are matched to the non-core machines. Each non-core job has one operation with processing time $15q$ on its matching non-core machine; the processing time of all other operations of a non-core job is 0.

In the core of the instance $O(I)$, there are $3q$ machines, one for each element of I . For each such machine, there are two matching dummy jobs which both have an operation with processing time $5q$ on this machine; all other operations of the dummy jobs have processing time 0. Finally, there are s jobs, which correspond to the triples in T , and therefore are called the *triple jobs*. Each triple job has three operations with processing time $5q$; those operations are matched to the element machines corresponding to the elements of the respective triple; again, all other operations of a triple job have processing time 0.

It is a straightforward observation that there always exists an optimal schedule for instance $O(I)$ which rejects no dummy job and has makespan $15q$; in particular, the triples corresponding to the accepted triple jobs must form a feasible solution to I . Moreover, any given schedule can easily be modified to fulfill these conditions without increasing its objective value. This yields the following lemma.

Lemma 6. *If an optimal solution to the instance I of MAX-3DM-B consists of k triples, then the value of an optimum solution to the instance $O(I)$ of the open shop scheduling problem is equal to $15q + s - k$.* □

The lemma and the discussion above contain the main ingredients for the proof of the following theorem.

Theorem 8. *The problem $O \mid pmtn, e_j \equiv 1 \mid \text{Rej} + C_{\max}$ is APX-hard.* □

References

1. G. AUSIELLO, P. CRESCENZI, G. GAMBOSI, V. KANN, A. MARCHETTI-SPACCAMELA, AND M. PROTASI [1999]. *Complexity and Approximation*. Springer, Berlin.
2. Y. BARTAL, S. LEONARDI, A. MARCHETTI-SPACCAMELA, J. SGALL, AND L. STOUGIE [2000]. Multiprocessor scheduling with rejection. *SIAM Journal on Discrete Mathematics* 13, 64–78.
3. D. BERTSIMAS, C. TEO, AND R. VOHRA [1999]. On dependent randomized rounding algorithms. *Operations Research Letters* 24, 105–114.
4. D.W. ENGELS, D.R. KARGER, S.G. KOLLIPOPOULOS, S. SENGUPTA, R.N. UMA, AND J. WEIN [1998]. Techniques for scheduling with rejection. *Proceedings of the 6th European Symposium on Algorithms (ESA'98)*, Springer LNCS 1461, 490–501.
5. M.R. GAREY AND D.S. JOHNSON [1979]. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco.
6. E.L. LAWLER, J.K. LENSTRA, A.H.G. RINNOOY KAN, AND D.B. SHMOYS [1993]. Sequencing and scheduling: Algorithms and complexity. In: S.C. Graves, A.H.G. Rinnooy Kan, and P.H. Zipkin (eds.) *Logistics of Production and Inventory*, Handbooks in Operations Research and Management Science 4, North-Holland, Amsterdam, 445–522.
7. A. SCHRIVVER [1986]. *Theory of Linear and Integer Programming*. John Wiley & Sons, Chichester.
8. S. SENGUPTA [1999]. Algorithms and approximation schemes for minimum lateness and tardiness scheduling with rejection. Manuscript, Laboratory for Computer Science, MIT.
9. G.J. WOEINGER [1999]. When does a dynamic programming formulation guarantee the existence of an FPTAS? *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'99)*, 820–829.