Multiprocessor Scheduling with Rejection *

Yair Bartal† Stefano Leonardi‡ Alberto Marchetti-Spaccamela†
Jiří Sgall§ Leen Stougie¶

Abstract

We consider a version of multiprocessor scheduling with the special feature that jobs may be rejected at a certain penalty. An instance of the problem is given by m identical parallel machines and a set of n jobs, each job characterized by a processing time and a penalty. In the on-line version the jobs become available one by one and we have to schedule or reject a job before we have any information about future jobs. The objective is to minimize the makespan of the schedule for accepted jobs plus the sum of the penalties of rejected jobs.

The main result is a $1 + \phi \approx 2.618$ competitive algorithm for the on-line version of the problem, where ϕ is the golden ratio. A matching lower bound shows that this is the best possible algorithm working for all m. For fixed m we give improved bounds, in particular for $m = 2$ we give a $\phi \approx 1.618$ competitive algorithm, which is best possible.

For the off-line problem we present a fully polynomial approximation scheme for fixed m and a polynomial approximation scheme for arbitrary m. Moreover we present an approximation algorithm which runs in time $O(n \log n)$ for arbitrary m and guarantees a $2 - \frac{1}{m}$ approximation ratio.

*A preliminary version of this work appeared in the Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms (1996), pp. 95-103.

†E-mail yairb@math.tau.ac.il. Department of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel. Research supported in part by Ben Gurion Fellowship, the Ministry of Science and Arts.

‡E-mail {leonardi, marchetti}@dis.uniroma1.it. Dipartimento di Informatica Sistemistica, Università di Roma "La Sapienza", via Salaria 113, 00198-Roma, Italia. This work was partly supported by ESPRIT BRA Alcom II under contract No.7141, and by Italian Ministry of Scientific Research Project 40% "Algoritmi, Modelli di Calcolo e Strutture Informative".

§E-mail: sgall@math.cas.cz, http://www.math.cas.cz/~sgall/. Mathematical Institute, AS CR, Žitná 25, 115 67 Praha 1, Czech Republic and Dept. of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Prague. Partially supported by grants A1019602 and A1019901 of GA AV ČR, postdoctoral grant 201/97/P038 of GA ČR, and cooperative research grant INT-9600919/ME-103 from the NSF (USA) and the MŠMT (Czech republic). Part of this work was done at Institute of Computer Science, Hebrew University, Jerusalem, Israel; supported in part by Golda Meir Postgraduate Fellowship.

¶E-mail leen@win.tue.nl. Department of Mathematics, Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, The Netherlands. Supported by the Human Capital Mobility Network DONET of the European Community

1 Introduction

Scheduling jobs on parallel machines is a classical problem that has been widely studied for more than three decades [6, 12]. In this paper we consider a version of the problem that has the special feature that jobs can be rejected at a certain price.

We call this problem *Multiprocessor Scheduling with Rejection* and use the abbreviation MSR. Given are m identical machines and n jobs. Each job is characterized by its *processing time* and its *penalty*. A job can either be rejected, in which case its penalty is paid, or scheduled on one of the machines, in which case its processing time contributes to the completion time of that machine. The processing time is the same for all the machines, as they are identical. Preemption is not allowed, i.e., each job is assigned to a single machine, and once started is processed without interruption. The objective is to minimize the *sum of the makespan and the penalties* of all rejected jobs. Makespan (the length of the schedule) is defined as the maximum completion time taken over all machines.

In the on-line version of MSR jobs become available one by one, and the decision to either reject a job or to schedule it on one of the machines has to be made before any information about next jobs is disclosed. In particular, there may be no other jobs. On-line algorithms are evaluated by the *competitive ratio*; an on-line algorithm is c-competitive if for each input the cost of the solution produced by the algorithm is at most c times the cost of an optimal solution (cf. [14]).

The main goal of an on-line MSR algorithm is to choose the correct balance between the penalties of the jobs rejected and the increase in the makespan for the accepted jobs. At the beginning it might have to reject some jobs, if the penalty for their rejection is small compared to their processing time. However, at a certain point it would have been better to schedule some of the previously rejected jobs since the increase in the makespan due to scheduling those jobs in parallel is less than the total penalty incurred. In this scenario the on-line MSR problem can be seen as a non-trivial generalization of the well-known Rudolph's *ski rental problem* [11]. (In that problem, a skier has to choose whether to rent skis for the cost of 1 per trip, or to buy them for the cost of c, without knowing the future number of trips. The best possible deterministic strategy is to rent for the first c trips and buy afterwards. In our problem, rejecting jobs is analogous to renting, while scheduling one job is analogous to buying, as it allows to schedule $m-1$ more jobs of no bigger processing time without extra cost.)

Our main result is a best possible, $1+\phi \approx 2.618$ competitive algorithm for the on-line MSR problem, where $\phi = (1+\sqrt{5})/2$ is the golden ratio. We prove that no deterministic algorithm that receives m as input can achieve a better competitive ratio independent of m.

For small values of m we give better upper and lower bounds. In particular for $m = 2$ we obtain a best possible, $\phi \approx 1.618$ competitive algorithm. For $m = 3$ we obtain 2-competitive algorithms and show a lower bound of 1.839.

Our results should be compared with the current knowledge about on-line algorithms for the classical multiprocessor scheduling problem. In that problem, each job has to be scheduled, hence it is equivalent to a special case of our problem where each penalty is larger than the corresponding processing time. Graham's *list scheduling* algorithm schedules each job on the currently least loaded machine and is $2 - \frac{1}{m}$ competitive [7]. It is known that for $m > 3$ list scheduling is not optimal [5], and in fact there exist $2 - \varepsilon$ competitive algorithms for small constant $\varepsilon > 0$ [2, 10, 1]. The best POSSIBLE competitive ratio is known to be between 1.85 and 1.92 (see [1]), but its precise value is unknown. In contrast, for the more general on-line MSR problem we do find the optimal competitive ratio. More surprisingly, our algorithms achieving the optimal competitive ratio schedule the accepted jobs using list scheduling, which is inferior when rejections are not allowed!

Next we consider the off-line MSR problem. We present an approximation algorithm with a $2 - \frac{1}{m}$ worst-case approximation ratio running in time $O(n \log n)$ for arbitrary m. We also present a *fully polynomial approximation scheme* for MSR for any fixed m and a *polynomial approximation scheme* for arbitrary m, i.e., where m is part of the input.

More explicitly, the approximation schemes give algorithms with running time either polynomial in n and $1/\epsilon$ but exponential in m, or polynomial in n and m, but exponential in $1/\epsilon$, where ϵ is the maximal error allowed. This implies that for the more general problem with possible rejection of jobs we have algorithms that are essentially as good as those known for the classical problem without rejection. In fact, our algorithms are based on the techniques used for the problem without rejection, namely on the fully polynomial approximation scheme for fixed m [9] (based on a dynamic programming formulation of the problem) and the polynomial approximation scheme for arbitrary m [8].

Obviously, the MSR problem on a single machine is easily solved exactly by scheduling every job whose processing time does not exceed its penalty, and for $m \geq 2$ it is NP-hard to find the optimal solution, similarly as in the classical case without rejections.

The on-line algorithms and lower bounds are presented in Sections 3 and 4. Section 5 contains the results about the off-line problem.

2 Notation

An instance of the MSR problem consists of a *number of machines* m and a *set of jobs* J, $|J| = n$. We abuse the notation and denote the j-th job in the input sequence by j. Each job $j \in J$ is characterized by a pair (p_j, w_j), where p_j is its *processing time* and w_j is its *penalty*.

For a set of jobs $X \subseteq J$, $W(X) = \sum_{j \in X} w_j$ is the total penalty of jobs in X, and $M(X) = \sum_{j \in X} p_j/m$ is the sum of the loads of the jobs in X, where the *load* of a job j is defined by p_j/m. The set $B = \{j \mid w_j \leq p_j/m\}$ contains jobs with penalty less than or equal to their load.

Given a solution produced by an on-line or approximation algorithm, R denotes the set of all rejected jobs, A denotes the set of all accepted job, T denotes the largest processing time of all accepted jobs. For their analogs in the optimal solution we use R^{OPT}, A^{OPT}, T^{OPT}, respectively. Z^{OPT} denotes the total cost of the optimal solution for a given instance of the problem, and Z^H is the cost achieved by algorithm H. An on-line algorithm ON is c-competitive if $Z^{ON} \leq c \cdot Z^{OPT}$

for every input instance.

The golden ratio is denoted by $\phi = (\sqrt{5}+1)/2 \approx 1.618$. We will often use the property of the golden ratio that $\phi - 1 = 1/\phi$.

Using list scheduling, the makespan of a schedule is bounded from above by the processing time of the job that finishes last plus the sum of the loads of all other scheduled jobs [7]. We denote this bound by $C^{LS}(X)$ for a set X of scheduled jobs. If ℓ is the job in X that finishes last, then

$$C^{LS}(X) = M(X - \{\ell\}) + p_\ell \leq M(X) + (1 - \frac{1}{m})T. \tag{1}$$

3 On-line scheduling with rejections

In the first part of this section we present an on-line MSR algorithm which works for arbitrary m and achieves the best possible competitive ratio in that case. The corresponding lower bound is given in Section 4.2. For fixed $m \geq 3$ this algorithm gives the best competitive ratio we are able to achieve, however we are not able to prove a matching lower bound. In the second part we present a different algorithm which is best possible for the case of two machines. The corresponding lower bound is given in Section 4.1.

3.1 Arbitrary number of machines

Our algorithm uses two simple rules. First, all jobs in the set B are rejected, which seems advantageous since their penalty is smaller than their load. The second rule is inspired by the relation of MSR to the ski-rental problem and states that a job is rejected unless its penalty added to the total penalty of the hitherto rejected jobs would be higher than some prescribed fraction of its processing time. This fraction parameterizes the algorithm, we denote it by α.

Algorithm RTP(α) (REJECT-TOTAL-PENALTY(α))

(i) If a job from B becomes available, reject it.

(ii) Let W be the total penalty of all jobs from $J - B$ rejected so far. If a job $j = (p_j, w_j) \notin B$ becomes available, reject it if $W + w_j \leq \alpha p_j$, otherwise accept it and schedule it on a least loaded machine.

In Theorem 1 we will prove that for given m, the algorithm is c-competitive if c and $\alpha > 0$ satisfy

$$\begin{aligned} c &\geq 1 + (1 - \frac{1}{m})\frac{1}{\alpha} \\ c &\geq 2 + \alpha - \frac{2}{m}. \end{aligned} \tag{2}$$

To obtain a best possible algorithm for arbitrary m, we use $\alpha = \phi - 1 \approx 0.618$. Then $c = 1 + \phi$ satisfies the inequalities above. For a fixed m, the best c is obtained if equality is attained in both cases. For $m = 2$ this leads to $\alpha = \sqrt{2}/2 \approx 0.707$ and $c = 1 + \sqrt{2}/2 \approx 1.707$, and for $m = 3$ we get $\alpha = 2/3$ and $c = 2$. For general m we obtain

$$\begin{aligned}\alpha &= \frac{-(1-\frac{2}{m}) + \sqrt{5 - \frac{8}{m} + \frac{4}{m^2}}}{2},\\ c &= 1 + \frac{(1-\frac{2}{m}) + \sqrt{5 - \frac{8}{m} + \frac{4}{m^2}}}{2}.\end{aligned}$$

Theorem 1 *The algorithm* **RTP**(α) *for m machines is c-competitive if c and α satisfy (2).*

Proof. First we notice that since our algorithm uses list scheduling for the accepted jobs, its makespan is bounded by $C^{LS}(A) = (1 - \frac{1}{m})T + M(A)$ (cf. (1)). Hence,

$$Z^{ON} \leq (1 - \frac{1}{m})T + M(A) + W(R).$$

For any set $S \subseteq R$, the right-hand side of this inequality can be rewritten as a sum of two terms:

$$Z^{ON} \leq (M(A) + W(R - S) + M(S)) + ((1 - \frac{1}{m})T + W(S) - M(S)). \tag{3}$$

Now, we fix an off-line optimal solution. We use the above inequality for the set $S = (R - B) \cap A^{OPT}$, the set of all jobs rejected by the algorithm in Step (ii) and accepted in the optimal solution. First we bound the first term in (3). Notice that

$$M(A) = M(A \cap A^{OPT}) + M(A \cap R^{OPT}) \leq M(A \cap A^{OPT}) + W(A \cap R^{OPT}), \tag{4}$$

since no job of the set B is accepted by the algorithm, and thus the load of each job accepted by the algorithm is smaller than its penalty. Next we notice that $S \subseteq A^{OPT}$, implying that

$$M(S) = M(A^{OPT} \cap S). \tag{5}$$

Since R^{OPT} and A^{OPT} is a partition of the set of all jobs, and $B \subseteq R$, we obtain

$$\begin{aligned} R - S &= [(R \cap R^{OPT}) \cup (R \cap A^{OPT})] - [(R - B) \cap A^{OPT}] \\ &= (R \cap R^{OPT}) \cup (B \cap A^{OPT}). \end{aligned} \tag{6}$$

From (6) and the definition of B we have

$$W(R - S) = W(B \cap A^{OPT}) + W(R \cap R^{OPT}) \leq M(B \cap A^{OPT}) + W(R \cap R^{OPT}). \tag{7}$$

Inequalities (4), (5), and (7) together imply that

$$M(A) + W(R - S) + M(S) \leq M(A^{OPT}) + W(R^{OPT}) \leq Z^{OPT}.$$

To finish the proof, it is now sufficient to show

$$(1 - \frac{1}{m})T + W(S) - M(S) \leq (1 + \alpha - \frac{2}{m})T^{OPT} + (1 - \frac{1}{m})\frac{1}{\alpha}W(R^{OPT}), \tag{8}$$

and notice that under our conditions (2) on c this is at most

$$(c-1)T^{OPT} + (c-1)W(R^{OPT}) \leq (c-1)Z^{OPT}.$$

All jobs in S are scheduled in the optimal solution and hence have processing time at most T^{OPT}. The algorithm never rejects such a job if this would increase the penalty above αT^{OPT}, and hence

$$W(S) \leq \alpha T^{OPT}. \tag{9}$$

For any job j that was rejected by Step (ii) of the algorithm we have $w_j \leq \alpha p_j$. Summing over all jobs in S we obtain $W(S) \leq \alpha m M(S)$, and hence

$$W(S) - M(S) \leq (1 - \frac{1}{\alpha m})W(S) \leq (1 - \frac{1}{\alpha m})\alpha T^{OPT} = (\alpha - \frac{1}{m})T^{OPT}. \tag{10}$$

Thus, if $T \leq T^{OPT}$, (8) follows. If $T > T^{OPT}$, let W be the penalty incurred by the jobs rejected in Step (ii) of the algorithm until it schedules the first job with processing time T, job j say, having penalty w_j. By the condition in Step (ii) of the algorithm, $\alpha T \leq W + w_j$. On the other hand, $W + w_j \leq W(S) + W(R^{OPT})$, as all jobs rejected in Step (ii) are in $S \cup R^{OPT}$, and also the job with processing time T is in R^{OPT}, since $T > T^{OPT}$. Thus,

$$(1 - \frac{1}{m})T \leq (1 - \frac{1}{m})\frac{1}{\alpha}(W(S) + W(R^{OPT})) \leq (1 - \frac{1}{m})T^{OPT} + (1 - \frac{1}{m})\frac{1}{\alpha}W(R^{OPT}), \tag{11}$$

using (9). Adding (10) to (11) we obtain (8), which finishes the proof. □

Choosing $\alpha = \phi - 1$ and $c = \phi + 1$, both inequalities in (2) are satisfied for any m, which yields our main result. For arbitrarily large m these values are the best possible.

Theorem 2 *The algorithm* **RTP**$(\phi - 1)$ *is* $(1 + \phi)$*-competitive.*

For any choice of m and α the bounds on c given by the inequalities (2) give a tight analysis of the algorithm **RTP**(α), as shown by the following two examples. First, consider the sequence of two jobs $(1 - \frac{1}{\alpha m}, \alpha - \frac{1}{m})$ and $(1 - \varepsilon, \frac{1}{m})$ with $\epsilon > 0$ arbitrarily small. **RTP**(α) rejects the first job and accepts the second job, while in the optimal solution both jobs are rejected. The competitive ratio attained on this sequence is $(1 - \varepsilon + (\alpha - \frac{1}{m}))/\alpha$, which for any $\alpha > 0$ and m can be made arbitrarily close to the first inequality of (2). Second, consider the sequence formed by one job $(1, \alpha)$, $m - 2$ jobs $(1, \frac{1}{m})$, and one job $(1, 1)$. **RTP**(α) rejects the first $m - 1$ jobs and accepts job $(1, 1)$, while the optimal solution accepts all jobs. The competitive ratio is $2 + \alpha - \frac{2}{m}$, leading to the second inequality of (2).

3.2 Two machines

To obtain a best possible, ϕ-competitive algorithm for two machines we use another approach. We simply reject all jobs with penalty at most α times their processing time, where α is again a parameter of the algorithm. Again the optimal value is $\alpha = \phi - 1 \approx 0.618$.

Algorithm RP(α) (REJECT-PENALTY(α))

If a job $j = (p_j, w_j)$ becomes available, reject it if $w_j \leq \alpha p_j$, otherwise accept it and schedule it on a least loaded machine.

Theorem 3 *The algorithm* $\mathbf{RP}(\phi - 1)$ *is* ϕ*-competitive for two machines.*

Proof. If the algorithm does not schedule any job, then

$$Z^{ON} = W(J) \leq 2(\phi - 1)M(A^{OPT}) + W(R^{OPT}) \leq 2(\phi - 1)Z^{OPT} \leq \phi Z^{OPT},$$

and the theorem is proved.

Otherwise denote by ℓ a job that is finished last by the on-line algorithm. Since the algorithm uses list scheduling, the makespan is bounded by $C^{LS}(A) = M(A - \{\ell\}) + p_\ell$, and therefore we have

$$Z^{ON} \leq W(R) + M(A - \{\ell\}) + p_\ell. \tag{12}$$

Notice that

$$W(R) = W(R \cap R^{OPT}) + W(R \cap A^{OPT}) \leq W(R \cap R^{OPT}) + 2(\phi - 1)M(R \cap A^{OPT}), \tag{13}$$

by direct application of the rejection rule of algorithm $\mathbf{RP}(\phi - 1)$.

For any job that is accepted by the algorithm the rejection rule of $\mathbf{RP}(\phi - 1)$ implies that its load is not greater than its penalty. Therefore,

$$\begin{aligned} M(A - \{\ell\}) &= M((A - \{\ell\}) \cap A^{OPT}) + M((A - \{\ell\}) \cap R^{OPT}) \\ &\leq M((A - \{\ell\}) \cap A^{OPT}) + W((A - \{\ell\}) \cap R^{OPT}). \end{aligned} \tag{14}$$

Invoking (13) and (14) in (12), yields

$$Z^{ON} \leq W(R^{OPT} - \{\ell\}) + 2(\phi - 1)M(A^{OPT} - \{\ell\}) + p_\ell. \tag{15}$$

We distinguish two cases. In the first case the optimal solution rejects job ℓ. Since ℓ is scheduled by the algorithm, we have $p_\ell \leq \phi w_\ell$, and therefore

$$Z^{ON} \leq \phi W(R^{OPT}) + 2(\phi - 1)M(A^{OPT}) \leq \phi Z^{OPT}.$$

In the second case ℓ is accepted in the optimal solution. Then, we use the identity $p_\ell = 2(\phi - 1)M(\{\ell\}) + (1 - (\phi - 1))p_\ell$ in (15) to obtain

$$\begin{aligned} Z^{ON} &\leq (2 - \phi)p_\ell + W(R^{OPT}) + 2(\phi - 1)M(A^{OPT}) \\ &\leq (2 - \phi)Z^{OPT} + 2(\phi - 1)Z^{OPT} = \phi Z^{OPT}, \end{aligned}$$

which completes the proof. □

The same approach can be used for larger m as well. However, for $m > 3$ this is worse than the previous algorithm. An interesting situation arises for $m = 3$. Choosing $\alpha = 1/2$ we obtain a 2-competitive algorithm, which matches the competitive ratio of the algorithm **RTP**$(2/3)$ for $m = 3$ in the previous subsection. Whereas **RP**$(1/2)$ rejects all jobs with penalty up to $1/2$ of their processing time, **RTP**$(2/3)$ rejects all jobs with penalty up to $1/3$ of their processing time and also jobs with larger penalty as long as the total penalty payed (by the jobs with smaller or equal processing times) remains at most $2/3$ times the processing time. We can combine these two approaches and show that for any $1/3 \leq \alpha \leq 1/2$, the algorithm that rejects each job with penalty at most α times its PROCESSING TIME and also if the total penalty is up to $1 - \alpha$ times its processing time, is 2-competitive, too. However, no such combined algorithm is better.

4 Lower bounds for on-line algorithms

In the first part of this section we give the lower bound for a small number of machines. In particular it shows that the algorithm presented in Section 3.2 is best possible for $m = 2$. In the second part we exhibit the lower bound for algorithms working for all m.

4.1 Small number of machines

Assume that there exists a c-competitive on-line algorithm for m machines. We prove that c satisfies $c \geq \rho$, where ρ is the solution of the following equation:

$$\rho^{m-1} + \rho^{m-2} + \ldots + 1 = \rho^m. \tag{16}$$

For $m = 2$ we get $\rho = \phi$, and hence prove that the algorithm **RP**$(\phi - 1)$ is best possible. For $m = 3$ we get $\rho \approx 1.839$, and so on. Notice that for arbitrary m this only proves that the competitive ratio is at least 2.

Theorem 4 *For any c-competitive algorithm for MSR on m machines, it holds that $c \geq \rho$, where ρ satisfies equation (16).*

Proof. Given m, let ρ be the solution of equation (16). Consider an adversary providing a sequence of jobs, all with processing time 1. The first job given has penalty $w_1 = 1/\rho$. If the on-line algorithm accepts this job the sequence stops and the algorithm is ρ-competitive. Otherwise, a second job is given by the adversary with penalty $w_2 = 1/\rho^2$. Again, accepting this job by the on-line algorithm makes the sequence stop and the competitive ratio is ρ. Rejection makes the sequence continue with a third job. This process is repeated for at most $m - 1$ jobs with penalties $w_j = 1/\rho^j$ for $1 \leq j = m - 1$. If the on-line algorithm accepts any job in this sequence, job k say, the adversary stops the sequence at that job, yielding a competitive ratio of the on-line algorithm on this sequence of k jobs of

$$\frac{Z^{ON}}{Z^{OPT}} = \frac{1+\sum_{j=1}^{k-1}\frac{1}{\rho^j}}{\sum_{j=1}^{k}\frac{1}{\rho^j}} = \rho,$$

since for any such $k \leq m-1$ in the optimal solution all jobs are rejected.

Otherwise, if none of the first $m-1$ jobs are accepted by the on-line algorithm, another job is presented with penalty $w_m = 1$. In the optimal solution all m jobs are accepted and scheduled in parallel, giving cost 1. The on-line cost is equal to the sum of the penalties of the first $m-1$ jobs plus 1, independent of whether the last job is accepted or rejected. Thus,

$$\frac{Z^{ON}}{Z^{OPT}} = 1 + \sum_{j=1}^{m-1} \frac{1}{\rho^j}.$$

By equation (16), this is exactly ρ, and the theorem follows. □

Corollary 5 *For two machines, no on-line algorithm has competitive ratio less than ϕ.*

4.2 Arbitrary number of machines

Now we prove the lower bound on algorithms working for arbitrary m. The sequence of jobs starts as in the previous section, but additional ideas are necessary.

Theorem 6 *There exists no on-line algorithm that is β-competitive for some constant $\beta < 1+\phi$ and all m.*

Proof. All jobs in the proof have processing time 1. All logarithms are base 2. For contradiction, we assume that the on-line algorithm is β-competitive for a constant $\beta < 1+\phi$, and m is a sufficiently large power of two. Let $a_i = (\log m)^{i+1}$, and let k be the largest integer such that $\log m + \sum_{i=1}^{k} a_i < m$. Calculation gives $k = \lfloor \log m / \log\log m \rfloor - 1$.

Consider again an adversary that intends to provide the following sequence of at most m jobs (all with processing time 1):

$$\begin{array}{rll}
1 & \text{job with penalty} & 1/(1+\phi) \\
1 & \text{job with penalty} & 1/(1+\phi)^2 \\
 & \vdots & \\
1 & \text{job with penalty} & 1/(1+\phi)^{\log m} \\
a_1 & \text{jobs with penalty} & 1/a_1 \\
 & \vdots & \\
a_k & \text{jobs with penalty} & 1/a_k.
\end{array}$$

As in the proof of Theorem 4 we argue that if the on-line algorithm accepts one of the first $\log m$ jobs, the adversary stops the sequence and the competitive ratio is $1+\phi$. Therefore, any β-competitive algorithm has to reject the first $\log m$ jobs. Now, let b_i be the number of jobs with penalty $1/a_i$ that are rejected by the β-competitive algorithm. The penalty the algorithm pays on those jobs is b_i/a_i. Since there are less than m jobs, the optimal cost is at most 1. Thus the total penalty incurred by the on-line algorithm has to be at most β, and in particular there has to exist $\ell \le k$ such that $b_\ell/a_\ell \le \beta/k < 3/k$. Fix such ℓ.

Now consider the following modified sequence of at most $2m$ jobs (again all with processing time 1):

$$\begin{array}{cll} 1 & \text{job with penalty} & 1/(1+\phi) \\ 1 & \text{job with penalty} & 1/(1+\phi)^2 \\ & \vdots & \\ 1 & \text{job with penalty} & 1/(1+\phi)^{\log m} \\ a_1 & \text{jobs with penalty} & 1/a_1 \\ & \vdots & \\ a_\ell & \text{jobs with penalty} & 1/a_\ell \\ M & \text{jobs with penalty} & 6, \end{array}$$

where $M = m + 1 - \sum_{i=1}^{\ell}(a_i - b_i)$.

The sequence is identical up to the jobs with penalty $1/a_\ell$, and hence the on-line algorithm behaves identically on this initial subsequence. In particular, it also rejects all first $\log m$ jobs paying a penalty of at least $\sum_{j=1}^{\log m}(1+\phi)^{-j} = (1-(1+\phi)^{-\log m})/\phi \ge \phi - 1 - 1/m$ for them. Then it also rejects b_i jobs with penalty $1/a_i$, for $i \le \ell$, paying penalty $\sum_{i=1}^{\ell} b_i/a_i$ for them.

The on-line algorithm has to accept all jobs with penalty 6, since the adversary will present at most $2m$ jobs, and hence scheduling them all would lead to a cost of at most 2. By summing the numbers, it follows that the on-line algorithm schedules exactly $m+1$ jobs. Thus, its makespan is at least 2, and its total cost is at least $1 + \phi - 1/m$.

To finish the proof, it is sufficient to present a solution with cost $1 + o(1)$. Consider the solution that rejects

$$\begin{array}{rll} 1 + \log m & \text{jobs with penalty} & 1/a_1, \\ b_1 & \text{jobs with penalty} & 1/a_2, \\ b_2 & \text{jobs with penalty} & 1/a_3, \\ & \vdots & \\ b_{\ell-2} & \text{jobs with penalty} & 1/a_{\ell-1}, \\ b_{\ell-1} + b_\ell & \text{jobs with penalty} & 1/a_\ell, \end{array}$$

and schedules all remaining jobs optimally. First we verify that this description is legal, i.e. there are always sufficiently many jobs with given penalty. By definition, $b_i \le a_i \le a_{i+1}$. For sufficiently large m, we have $1+\log m < a_1$, and due to our choice of ℓ, we also have $b_{\ell-1}+b_\ell \le a_{\ell-1}+3a_\ell/k \le a_\ell$.

In the presented schedule one more job is rejected than in the solution produced by the on-line algorithm, and hence there are only m jobs to be scheduled. Thus, the makespan is 1. The penalty paid is

$$\frac{1+\log m}{a_1}+\sum_{i=1}^{\ell-1}\frac{b_i}{a_{i+1}}+\frac{b_\ell}{a_\ell}=\frac{1+\log m}{(\log m)^2}+\frac{1}{\log m}\sum_{i=1}^{\ell-1}\frac{b_i}{a_i}+\frac{b_\ell}{a_\ell}$$

The sum in the second term is less than the penalty paid by the on-line algorithm, and hence this term is bounded by $O(1/\log m)$. The last term is bounded due to our choice of ℓ, namely it is $O(1/k) = O(\log\log m/\log m)$. Thus, the total penalty paid is $O(\log\log m/\log m) = o(1)$, and the total cost is $1 + o(1)$. □

5 Off-line scheduling with rejection

5.1 An approximation algorithm for arbitrary number of machines

In this section we give a $(2-\frac{1}{m})$-approximation algorithm for MSR on m machines. Our lower bounds imply that such a ratio cannot be achieved by an on-line algorithm. The algorithm rejects all jobs in the set $B = \{j \mid w_j \le p_j/m\}$. From all other jobs it accepts some number of jobs with the smallest processing time, and chooses the best among such solutions.

Algorithm APPROX

(i) Sort all jobs in $J - B$ according to their processing times in non-decreasing order.

(ii) Let S_i, $0 \le i \le |J - B|$, be the solution that schedules the first i jobs from $J - B$ using list scheduling and rejects all other jobs. Choose the solution S_i with the smallest cost.

Note that Step (ii) of the algorithm takes time $O(n \log m)$ (or $O(n)$ in case $m \ge n$), as we can build the schedules incrementally and the bookkeeping of penalties for rejected jobs is simple. Thus, the whole algorithm runs in time $O(n \log n)$, independent of m. A performance analysis leads to the following worst-case ratio.

Theorem 7 *The algorithm APPROX achieves $Z^H \le (2-\frac{1}{m})Z^{OPT}$, where Z^H is the cost of the solution found by the algorithm.*

Proof. We assume that the jobs from $J - B$ are ordered $1, 2, \ldots, |J - B|$, according to the ordering given by Step (i) of the algorithm. If the optimal solution rejects all jobs from $J - B$, by the definition of B it is optimal to reject all jobs from B as well. Thus the solution S_0 that rejects all jobs is optimal and $Z^H = Z^{OPT}$.

Otherwise let ℓ be the last job from $J - B$ accepted in the optimal solution. Consider the solution S_ℓ, which schedules all jobs up to ℓ. Let $A = \{1, \ldots, \ell\}$ be the set of all jobs scheduled in S_ℓ. Job ℓ has the largest running time of all scheduled jobs, and since we use list scheduling, the makespan of S_ℓ is at most

$$C^{LS}(A) = M(A) + (1-\frac{1}{m})p_\ell \le M(A) + (1-\frac{1}{m})Z^{OPT}.$$

Since the cost of the algorithm is at most the cost of S_ℓ, we have

$$\begin{aligned} Z^H &\leq W(J-A) + M(A) + (1-\frac{1}{m})Z^{OPT} \\ &= W(A^{OPT} \cap (J-A)) + W(R^{OPT} \cap (J-A)) + \\ &\quad M(R^{OPT} \cap A) + M(A^{OPT} \cap A) + (1-\frac{1}{m})Z^{OPT}. \end{aligned}$$

By the choice of ℓ, $A^{OPT} \cap (J-A) \subseteq B$, and thus $W(A^{OPT} \cap (J-A)) \leq M(A^{OPT} \cap (J-A))$. Moreover, since A does not contain any job of B, $M(R^{OPT} \cap A) \leq W(R^{OPT} \cap A)$. These observations inserted in the above inequality yield

$$Z^H \leq W(R^{OPT}) + M(A^{OPT}) + (1-\frac{1}{m})Z^{OPT} \leq (2-\frac{1}{m})Z^{OPT}$$

□

That the ratio is tight is shown by the following instance with m jobs (and m machines): $p_1 = \ldots = p_m = 1$, $w_1 = 1-\epsilon$, and $w_2 = \ldots = w_m = \frac{1}{m}(1-\epsilon)$. The heuristic will reject all jobs resulting in $Z^H = (1+\frac{m-1}{m})(1-\epsilon)$. In the optimal solution all jobs are accepted, hence $Z^{OPT} = 1$. Therefore, Z^H/Z^{OPT} can be made arbitrarily close to $2-\frac{1}{m}$.

This example also shows that any heuristic that rejects all jobs in the set B, has a worst-case ratio no better than $2-\frac{1}{m}$, since there is no scheduling at all involved in it. Thus, the only way in which an improvement might be obtained is by accepting possibly also jobs in the set B.

5.2 A fully polynomial approximation scheme fixed m

For the off-line MSR problem there exists a fully polynomial approximation scheme for fixed m. The proof uses a rounding technique based on dynamic programming as was developed in [9], for the classical makespan problem.

Lemma 8 *The MSR problem with integer processing times and penalties can be solved in time polynomial in n and $(Z^{OPT})^m$.*

Proof. We use dynamic programming. Let M_i represent the current load of machine i, $i = 1, \ldots, m$. We compute for each $M_1, \ldots, M_m \leq Z^{OPT}$ the minimal value of total penalty to be paid that can be achieved with these loads. We denote this value after the first j jobs are rejected or scheduled by $W_j(M_1, \ldots, M_m)$ and define it to be ∞ whenever $M_i < 0$ for some i. At the same time we compute the minimal cost of a schedule that can be achieved with given loads $M_1, \ldots, M_m$, denoted $Z(M_1, \ldots, M_m)$. For $M_1, \ldots, M_m \geq 0$ these values can be computed recursively as follows:

$$\begin{aligned} W_0(M_1, \ldots, M_m) &= 0, \\ W_j(M_1, \ldots, M_m) &= \min\{ w_j + W_{j-1}(M_1, \ldots, M_m), \\ &\quad \min_i W_{j-1}(M_1, \ldots, M_{i-1}, M_i - p_j, M_{i+1}, \ldots, M_m)\}, \\ Z(M_1, \ldots, M_m) &= W_n(M_1, \ldots, M_m) + \max_i M_i. \end{aligned}$$

We compute the values in the order of increasing $\max_i M_i$. As soon as $\max_i M_i$ reaches the cost of the current optimal solution, which is the smallest value of Z computed so far, we stop, as we know it is a global optimum. $\square$

Theorem 9 *For any $\varepsilon \geq 0$, there exists an ε-approximation algorithm for the MSR problem that runs in time polynomial in the size of the input instance, n^m and $1/\varepsilon^m$.*

Proof. Given an instance I of the MSR problem with n jobs and m machines, we first use the approximation algorithm from Section 5.1 to obtain the cost Z^H. Now we define an instance I' by rounding the processing times and the penalties of the jobs in I. Namely the processing time p'_j and the penalty w'_j of job j in I' are $p'_j = \lfloor p_j/k \rfloor$ and $w'_j = \lfloor w_j/k \rfloor$ where $k = \varepsilon Z^H/2n$. We obtain the optimal solution of I' by the dynamic programming algorithm presented in the proof of Lemma 8, and derive an approximate solution for I by scheduling the respective jobs on the same machines as in the optimal solution for I'.

The cost $Z^{A(k)}$ of the approximate solution deviates from the optimal solution for I by at most $nk = \varepsilon Z^H/2$. Therefore, by applying the lower bound $Z^{OPT} \geq Z^H/2$ we obtain

$$\frac{|Z^{A(k)} - Z^{OPT}|}{Z^{OPT}} \leq \frac{2nk}{Z^H} = \varepsilon.$$

By Lemma 8 it follows that the running time of the approximation algorithm is polynomial in n and $(Z^{OPT}(I'))^m$. The theorem follows since $Z^{OPT}(I') \leq Z^{OPT}(I)/k \leq 2Z^H/k$ and hence $Z^{OPT}(I') \leq 4n/\varepsilon$. $\square$

5.3 A polynomial approximation scheme for arbitrary m

For arbitrary m we will design a polynomial approximation scheme (PAS) based on the PAS for the makespan problem in [8].

Given an instance with n jobs, m machines, and $\epsilon > 0$, we are to find an ϵ-approximate solution. As an upper bound U on the solution value we use the outcome Z^H of the heuristic presented in Section 5.1. Notice that all jobs with $p_j > U$ will be rejected. Thus, all jobs that are possibly scheduled have processing times in the interval $[0, U]$.

From Theorem 7 we have a lower bound on the optimal solution that we denote by $L = Z^H/2 = U/2$. We define the set $S = \{j \mid p_j \in [0, \epsilon L/3]\}$, a set of jobs with relatively small processing times. Let $D = \{j \mid j \notin S\}$. The remaining interval $(\epsilon L/3, U]$ is partitioned into $s \leq 18\lceil 1/\epsilon^2 \rceil$ subintervals $(l_1, l_2]$, $(l_2, l_3]$, ..., $(l_s, l_{s+1}]$ of length $\epsilon^2 L/9$ each, with $l_1 = \epsilon L/3$ and $l_{s+1} \geq U$. Let D_i be the set of jobs with processing time in the interval $(l_i, l_{i+1}]$, and let the jobs in each such set be ordered so that the penalties are non-increasing. As before define the set $B = \{j \mid w_j \leq p_j/m\}$.

First we will describe how for any subset Δ of D we generate an approximate solution with value $Z^{H(\epsilon)}(\Delta)$. For any such a set Δ we determine a schedule for all the jobs in Δ with an $\epsilon/3$-approximate makespan using the PAS in [8]. All other jobs in D, i.e., all jobs in $D - \Delta$, are rejected. Jobs in the set S that have $w_j \geq \frac{1}{m}p_j$, i.e., jobs in the set $S - B$, are scheduled in any order according to the list scheduling rule starting from the $\epsilon/3$-approximate schedule determined before. The remaining jobs, $j \in S \cap B$, are considered in any order. Each next job is rejected if its assignment to a least loaded machine would cause an increase of the makespan, otherwise it is assigned to a least loaded machine as indicated by list scheduling.

This procedure is applied to every set $D(y_1, \ldots, y_s) \subseteq D$, where $D(y_1, \ldots, y_s)$ denotes the set that is composed of the first y_i elements in the ordered set D_i, $i = 1, \ldots, s$. In this way an approximate solution $Z^{H(\epsilon)}(D(y_1, \ldots, y_s))$ is found for each set $D(y_1, \ldots, y_s)$. The minimum value over all these sets, $Z^{H(\epsilon)} = \min_{(y_1,\ldots,y_s)} Z^{H(\epsilon)}(D(y_1, \ldots, y_s))$, is taken as the output of our procedure.

Theorem 10 *For any $\epsilon > 0$ the algorithm $H(\epsilon)$ described above runs in time polynomial in n and m, and yields*

$$\frac{Z^{H(\epsilon)}}{Z^{OPT}} \leq 1 + \epsilon.$$

Proof. The proof consists of two steps. First, consider the set $A^{OPT} \cap D$ of jobs in D that are accepted in the optimal solution. Applying the heuristic procedure described above to this set of jobs yields the approximate solution $Z^{H(\epsilon)}(A^{OPT} \cap D)$. We will prove that

$$\frac{Z^{H(\epsilon)}(A^{OPT} \cap D)}{Z^{OPT}} \leq 1 + \frac{\epsilon}{3}. \tag{17}$$

In the second step we analyze how much the set $A^{OPT} \cap D$ may differ from $D(y_1, \ldots, y_s)$. Assume that for $i = 1, \ldots, s$, $A^{OPT} \cap D$ consists of y_i^{OPT} jobs from the set D_i. These y_i^{OPT} jobs are not necessarily the first y_i^{OPT} jobs in the ordered set D_i, but we will show that

$$Z^{H(\epsilon)}(D(y_1^{OPT}, \ldots, y_s^{OPT})) \leq Z^{H(\epsilon)}(A^{OPT} \cap D) + \frac{2}{3}\epsilon L. \tag{18}$$

Inequalities (17) and (18) imply that

$$\frac{Z^{H(\epsilon)}(D(y_1^{OPT}, \ldots, y_s^{OPT}))}{Z^{OPT}} \leq 1 + \epsilon.$$

Since, obviously, $Z^{H(\epsilon)} \leq Z^{H(\epsilon)}(D(y_1^{OPT}, \ldots, y_s^{OPT}))$ the theorem follows.

In order to prove inequality (17) two cases are distinguished.

(1) The completion times of the various machines (in the heuristic solution corresponding to $Z^{H(\epsilon)}(A^{OPT} \cap D)$) differ by no more than $\epsilon L/3$. The resulting makespan is the same as the makespan after scheduling the jobs in $A^{OPT} \cap D$ and $S - B$, and due to our assumption it is at most $M(A^{OPT} \cap D) + M(S - B) + \epsilon L/3$. The weight of all rejected jobs is at most $W(S \cap B) + W(D - A^{OPT})$. Thus

$$Z^{H(\epsilon)}(A^{OPT} \cap D) \leq M(A^{OPT} \cap D) + M(S - B) + \frac{\epsilon L}{3} + W(S \cap B) + W(D - A^{OPT}).$$

Using the definition of the set B, we have for the optimal solution

$$Z^{OPT} \geq M(A^{OPT} \cap D) + M(S - B) + W(S \cap B) + W(D - A^{OPT}).$$

From these two inequalities (17) follows immediately.

(2) The completion times of the machines differ by more than $\epsilon L/3$. Since the processing time of each job in S is less than $\epsilon L/3$, we know that no job in the set $S \cap B$ is rejected, and scheduling all jobs in S has not increased the makespan computed for the set $A^{OPT} \cap D$. Let $C^{H(\epsilon)}(A^{OPT} \cap D)$ and $C^{OPT}(A^{OPT} \cap D)$ denote, respectively, the $\epsilon/3$-approximate and the optimal makespan for the jobs in $A^{OPT} \cap D$. In this case

$$Z^{H(\epsilon)}(A^{OPT} \cap D) = C^{H(\epsilon)}(A^{OPT} \cap D) + W(D - A^{OPT}),$$

and

$$Z^{OPT} \geq C^{OPT}(A^{OPT} \cap D) + W(D - A^{OPT}).$$

Moreover, since we have used an $\epsilon/3$-approximate algorithm for scheduling the jobs in $A^{OPT} \cap D$, we have

$$C^{H(\epsilon)}(A^{OPT} \cap D) \leq (1 + \frac{\epsilon}{3})C^{OPT}(A^{OPT} \cap D).$$

Inequality (17) results from the above three inequalities.

In order to prove (18) we need to bound the extra error that might occur due to the fact that $A^{OPT} \cap D \neq D(y_1^{OPT}, \ldots, y_s^{OPT})$. Notice that, for any D_i, $i = 1, \ldots, s$, the difference in processing time between any two jobs in D_i is at most $\epsilon^2 L/9$, and that $D(y_1^{OPT}, \ldots, y_s^{OPT})$ contains the jobs with larger penalties in D_i. The latter implies that the extra error can only be due to the fact that the first y_i^{OPT} jobs in D_i have longer processing times than those in $A^{OPT} \cap D_i$. Since the processing time of a job in D is at least $\epsilon L/3$ and $U \leq 2L$, no more than $6/\epsilon$ jobs from D are scheduled on any machine. Therefore the overall extra contribution to the makespan due to the fact that $A^{OPT} \cap D \neq D(y_1^{OPT}, \ldots, y_s^{OPT})$ can be no more than $(6/\epsilon)(\epsilon^2 L/9) = 2\epsilon L/3$, which implies inequality (18).

This completes the proof of correctness of the approximation.

The running time of the algorithm is dominated by the time required to compute the heuristic $Z^{H(\epsilon)}(D(y_1, \ldots, y_s))$ for each possible set of values $y_1, \ldots, y_s$, such that $0 \leq y_i \leq |D_i|$, $i = 1, \ldots, s$. Since y_i, $i = 1, \ldots, s$, satisfies $1 \leq y_i \leq n$, there are at most $n^s = O(n^{18\lceil 1/\epsilon^2 \rceil})$ possible sets of values $y_1, \ldots, y_s$.

For each of these sets an ϵ-approximate schedule is computed using the algorithm in [8], taking $O((n/\epsilon)^{\lceil 9/\epsilon^2 \rceil})$; attaching the jobs in the set S just adds $O(n^2)$ time to each of these computations. Hence, the overall running time of the algorithm is $O((n^3/\epsilon)^{\lceil 9/\epsilon^2 \rceil})$. This establishes that the algorithm is a polynomial approximation scheme for the problem with arbitrary m. □

6 Open problems and recent developments

Some open problems remain. For the on-line problem tight algorithms for the case of fixed m other than $m = 2$ are still to be established. For the off-line problem perhaps better heuristics may be found by improving the rejection strategy proposed in the algorithm in Section 5.1.

Seiden [13] has proved new results related to our problem. For the variant of deterministic *preemptive* scheduling with rejection he gives a $(4 + \sqrt{10})/3 \approx 2.387$ competitive algorithm for any number of machines, thus showing that allowing preemption can provably be exploited. Interestingly, this yields yet another 2-competitive algorithm for three machines. Also, Seiden notes that our Theorem 4 yields a lower bound for preemptive scheduling as well, and hence yields a lower bound of 2 for general number of machines. For two machines, this shows that our algorithm $\mathbf{RP}(\phi - 1)$ is best possible even among all preemptive algorithms. For 3 machines, an interesting open problem is to establish if preemption allows a better competitive ratio. The best upper bound of 2 and the best lower bound of 1.839 for preemptive algorithms still coincide with those shown in this paper for non-preemptive algorithms.

Seiden [13] also studies randomized scheduling with rejection, preemptive and non-preemptive. He gives algorithms which are better than deterministic for small number of machines, and in particular are 1.5-competitive for 2 machines, both preemptive and non-preemptive; this is best possible for 2 machines. In both cases it is still open if randomized algorithms for any number of machines can be better than their deterministic counterparts.

Epstein and Sgall [4] presented polynomial time approximation schemes for related machines for various objectives, including MSR, thus generalizing the polynomial time approximation scheme given in this paper.

Engels et al [3] study scheduling with rejection where in the objective the makespan is replaced by the sum of the completions times.

Acknowledgements. We thank Giorgio Gallo for having drawn our attention to this scheduling problem. We thank anonymous referees for numerous helpful comments.

References

[1] S. Albers. Better bounds for online scheduling. In *Proc. of the 29th Ann. ACM Symp. on Theory of Computing*, pages 130–139. ACM, 1997.

[2] Y. Bartal, A. Fiat, H. Karloff, and R. Vohra. New algorithms for an ancient scheduling problem. *J. Comput. Syst. Sci.*, 51(3):359–366, 1995.

[3] D. W. Engels, D. R. Karger, S. G. Kolliopoulos, S. Sengupta, R. N. Uma, and J. Wein. Techniques for scheduling with rejection. In *Proc. of the 6th Ann. European Symp. on Algorithms, Lecture Notes in Comput. Sci. 1461*, pages 490–501. Springer-Verlag, 1998.

[4] L. Epstein and J. Sgall. Approximation schemes for scheduling on uniformly related and identical parallel machines. Technical Report KAM-DIMATIA Series 98-414, Charles University, Prague, 1998.

[5] G. Galambos and G. J. Woeginger. An on-line scheduling heuristic with better worst case ratio than Graham's list scheduling. *SIAM J. Comput.*, 22(2):349–355, 1993.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP–completeness.* Freeman, 1979.

[7] R. L. Graham. Bounds for certain multiprocessor anomalies. *Bell System Technical J.*, 45:1563–1581, Nov. 1966.

[8] D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *J. Assoc. Comput. Mach.*, 34:144–162, 1987.

[9] E. Horowitz and S. Sahni. Exact and approximate algorithms for scheduling non-identical processors. *J. Assoc. Comput. Mach.*, 23:317–327, 1976.

[10] D. R. Karger, S. J. Phillips, and E. Torng. A better algorithm for an ancient scheduling problem. *J. of Algorithms*, 20:400–430, 1996.

[11] R. M. Karp. On-line algorithms versus off-line algorithms: How much is it worth to know the future? In J. van Leeuwen, editor, *Proc. of the IFIP 12th World Computer Congress. Volume 1: Algorithms, Software, Architecture*, pages 416–429. Elsevier Science Publishers, Amsterdam, 1992.

[12] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In S. C. Graves, A. H. G. Rinnooy Kan, and P. Zipkin, editors, *Handbooks in Operations Research and Management Science, Vol. 4: Logistics of Production and Inventory*, pages 445–552. North-Holland, 1993.

[13] S. S. Seiden. More multiprocessor scheduling with rejection. Technical Report Woe-16, Department of Mathematics, TU Graz, Austria, 1997.

[14] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Comm. Assoc. Comput. Mach.*, 28(2):202–208, 1985.