Online scheduling with machine cost and a power function objective

T. Balla[1], J. Csirik*[1], Gy. Dósa[2], and D. Kószó[1]

[1]Department of Informatics, University of Szeged, Árpád tér 2, H-6720 Szeged, Hungary
[2]Department of Mathematics, University of Pannonia, Egyetem u. 10, H-8200 Veszprém, Hungary

Abstract: We will consider a power function variant of online scheduling with machine cost. Here, we have a sequence of independent jobs with positive real sizes. Jobs come one by one and we have to assign them irrevocably to a machine without any knowledge about additional jobs that may follow later on. With this problem the algorithm has no machine at first. When a job arrives, we have the option to purchase a new machine and the cost of purchasing a machine is a fixed constant. In previous studies, the objective was to minimize the sum of the makespan and the cost of the purchased machines. In this paper, we minimize the sum of the rth power of loads of the machines ($r \geq 2$ is an integer) and the cost of purchasing the machines. The cost of a new machine is 1. We prove that no online algorithm has a competitive ratio smaller than $2 - \frac{2^{r-1}}{2^r-1}$ for this problem. Moreover, for $r = 2, 3, 4$ we present a $2 - \frac{2^{r-1}}{2^r-1}$-competitive online algorithm with a detailed competitive analysis which applies the bin packing algorithm First Fit as a slave algorithm.

Keywords: scheduling; online algorithms; analysis of algorithms; power function objective

Dedicated to Professor Imre Rudas on the Occasion of his 75th Birthday

1 Introduction

There is a huge literature on online scheduling, and books on scheduling cover this topic. To the best of our knowledge, quadratic cost function in scheduling was introduced by Townsend [1]. Several authors elaborated these questions in the next years: Bagga et al. [2], Gupta et al. [2], Croce et al. [3], Szwarc and Mukhopadhyay [4] and Wei and Wang [5].

For parallel machines Cheng and Liu [6] investigated the quadratic function objective from a stochastic point of view. Ho et al. [7] used an industrial engineering

* Corresponding author, email address: `jcsirik@gmail.com`

problem formulation, using the Normalized Sum of Square for Workload Deviations (NSSWD) to measure the performance.. Walter and Lawrinenko [8] showed that some theoretical results of [7] were not valid. Schwerdfeger and Walter [9] proposed an improved algorithm for minimizing NSSWD on m identical machines. They also showed that NSSWD was equivalent to the sum of squares of loads. Ouazene et al. [10] considered another problem for workload balancing, where nonpreemptive jobs had to be assigned on identical parallel machines.

Chandra and Wong [11] and Avidor et al. [12] furnished an analysis of algorithms for parallel machine scheduling with a quadratic objective function from an approximation point of view. In [11], they analyzed the Longest Processing Time algorithm from the worst case point of view. The algorithm has a preprocessing step which simply sorts the jobs in decreasing order and then it puts the next job on the machine with the smallest load. The authors proved that for $r = 2$ the algorithm was $\frac{25}{24}$-competitive, and provided lower bounds too. They also gave similar results for $r \geq 3$. Avidor et al. [12] provided results for online algorithms as well.

In this paper we will investigate a variant of online scheduling with machine cost (SMC), which was introduced by Imreh and Noga [13]. In this problem, jobs have a positive size, and they come one at a time. When a job arrives, we need to assign it to a machine, without prior information about the rest of the sequence. At first, the online algorithm has no machines, but it has the option to buy a new one when a job arrives. The cost of purchasing a machine is 1. Since the machine costs and job sizes can be rescaled, any other constant cost function is equivalent. By the load of a machine, we will mean the sum of the processing times of all jobs assigned to the machine. The maximum load is often called the makespan. Their objective is to minimize the sum of cost of the machines and the makespan. In [13], it was proved that no online algorithm could achieve a smaller competitive ratio than $4/3$. Moreover, a $(1+\sqrt{5}/2) \approx 1.618$-competitive algorithm was given. Dósa and He [14] presented an improved algorithm with a competitive ratio of $(2\sqrt{6}+3)/5 \approx 1.5798$. Dósa and Tan [15] showed that $\sqrt{2}$ was a lower bound of the SMC problem. A $\frac{2+\sqrt{7}}{3} \approx 1.5486$-competitive algorithm was also introduced. For the preemptive variant, Jiang and He [16] showed that the lower bound of $4/3$ was still valid, and an algorithm whose competitive ratio did not exceed 1.3798 was designed.

Dósa and He [17] considered the scheduling problem with machine cost and rejection penalties and they gave an optimal online algorithm with a competitive ratio of 2. Nagy-György and Imreh [18] presented an algorithm for the general case and they proved that it was $(3+\sqrt{5})/2$ competitive (≈ 2.618). He and Cai [19] studied two different semi-online versions of the problem of machine cost and makespan objective.

For a general cost function some further results are known from Imreh [20] and Akaria and Epstein [21].

A preliminary version of our paper appeared in [22] where the case $r = 2$ was solved.

We will organize the paper in the following way. In Section 2 we will give some definitions and auxiliary computations. In Section 3, we will present a $2-2^{r-1}/(2^r-1)$ general lower bound for our problem. In Section 4, we consider the properties of the optimal offline algorithm for the exponential variant of the problem. Lastly, in Section 5, we present an online algorithm with a detailed competitive analysis that applies the bin packing algorithm First Fit as a slave algorithm. For $r = 2, r = 3, r = 4$ the algorithm has a competitive ratio of $2-\frac{2^{r-1}}{2^r-1}$. For $r \geq 5$, we will give an upper bound on the competitive ratio for this algorithm.

2 Preliminaries

The variant we investigate here will use a power function of the loads in the objective instead of the makespan. To define it more precisely we will introduce some notations.

We will denote by $\mathscr{N}$ the set of natural numbers $\{0,1,2,\ldots\}$, by $\mathscr{N}^+$ the set $\mathscr{N} \setminus \{0\}$. For every $k,n \in \mathscr{N}$, we define $[k,n] = \{i \in \mathscr{N} \mid k \leq i \leq n\}$. We shall abbreviate $[1,k]$ by $[k]$. Moreover, let $\mathscr{R}^+$ be the set of positive real numbers and $\mathscr{R}$ be the set of real numbers ≥ 0. Let B be a finite set. Then $|B|$ denotes the cardinality of B, i.e., the number of elements of B.

Let J_n be an input for some $n \in \mathscr{N}^+$, i.e. a linearly ordered sequence of n jobs. The jobs will be labeled $1,2,...,n$, denoted by $j_1, j_2, ..., j_n$ and presented to the online algorithm in this order (the online algorithm does not know the value of n). We will denote the processing time of job j_k by $p_k > 0$. Sometimes we will call it the item size.

Given optimal offline solutions for all prefixes of the input, we will need the numbers of machines used by these optimal solutions. Let $\mathbf{OPT}_k$ denote the optimal offline solution after the first k jobs have arrived as well as its cost. For an online algorithm $\mathbf{A}$, let us denote the cost of the algorithm after the first k jobs have arrived by $\mathbf{A}_k$. We denote the number of the machines of $\mathbf{OPT}_k$ by m_k^*. For a specific online algorithm, let m_k denote the number of the machines it buys after having the first k jobs.

Let us suppose that we use k machines and these are $m_1, m_2, ..., m_k$. We will denote the *load of a machine* m_i by $\mathrm{ld}(m_i)$ and this is the sum of the processing times of the jobs on this machine. The *machines of* $\mathbf{A}$ *with respect to* J, denoted by $M_{\mathbf{A},J}$, is a linearly ordered set of machines, used by the algorithm to schedule input J. The *total cost of* $\mathbf{A}$ *on* J *with respect to* r is defined by

$$\mathbf{A}(r,J) = \sum_{m \in M_{\mathbf{A},J}} \mathrm{ld}(m)^r + |M_{\mathbf{A},J}|. \tag{1}$$

Moreover, we denote by $\mathscr{K}_r$ the *set of all offline algorithms for this problem with respect to r*. We say $\mathbf{O} \in \mathscr{K}_r$ is *optimal* if $\mathbf{O}(r,J) \leq \mathbf{O}'(r,J)$ for every input J and

$\mathbf{O}' \in \mathscr{K}_r$. Thus, from the definition there may exist more than one optimal offline algorithm in $\mathscr{K}_r$. We will denote by **OPT** one of the optimal offline algorithms. We evaluate the quality of an online algorithm by using a competitive analysis. Now let **A** be an online algorithm and **OPT** be an optimal offline algorithm. Let $C \in \mathscr{R}^+$. Then **A** is *C-competitive* if

$$\mathbf{A}(r,J) \le C \cdot \mathbf{OPT}(r,J)$$

for each input J. Moreover, we say that **A** is *constant competitive* if there exists a $C \in \mathscr{R}^+$ such that **A** is *C-competitive.*

For each $r \in \mathscr{N}^+, r \ge 2$ we define the following constants

$$b_r = \sqrt[r]{\frac{2^{r-1}}{2^{r-1}-1}} \quad , \quad q_r = \frac{b_r}{2}, \quad c_r = 2 - \frac{2^{r-1}}{2^r - 1},$$

$$\text{and } l_r = \frac{r}{\sqrt[r]{(r-1)^{r-1}}}.$$

Lemma 1. $(b_r)^r = 2(q_r)^r + 1$, *i.e.,* $(2q_r)^r = 2(q_r)^r + 1$.

Proof. On the one hand,

$$(b_r)^r = \frac{2^{r-1}}{2^{r-1}-1}.$$

On the other hand,

$$2(q_r)^r + 1 = 2 \cdot \frac{\frac{2^{r-1}}{2^{r-1}-1}}{2^r} + 1 = \frac{\frac{2^r}{2^{r-1}-1}}{2^r} + 1 = \frac{1}{2^{r-1}-1} + 1 = \frac{2^{r-1}}{2^{r-1}-1}.$$

□

Lemma 2. $c_r = \frac{(2q_r)^r+2}{(2q_r)^r+1}$, *i.e.,* $c_r = \frac{(b_r)^r+2}{(b_r)^r+1}$.

Proof. We know that

$$\frac{(2q_r)^r + 2}{(2q_r)^r + 1} = \frac{\frac{2^{r-1}}{2^{r-1}-1} + 2}{\frac{2^{r-1}}{2^{r-1}-1} + 1} = \frac{2^{r-1} + 2(2^{r-1}-1)}{2^{r-1} + (2^{r-1}-1)} = \frac{3 \cdot 2^{r-1} - 2}{2^r - 1},$$

and

$$c_r = 2 - \frac{2^{r-1}}{2^r - 1} = \frac{4 \cdot 2^{r-1} - 2 - 2^{r-1}}{2^r - 1} = \frac{3 \cdot 2^{r-1} - 2}{2^r - 1}.$$

This completes the proof. □

Lemma 3. *Let $a \in \mathscr{R}$ and $b, x \in \mathscr{R}^+$ such that $a < b$. Let $r > 1$ be a positive integer. Then*

$$(b+x)^r - b^r > (a+x)^r - a^r$$

Proof. We have

$$\begin{aligned}(b+x)^r - b^r - ((a+x)^r - a^r) &= \sum_{i=0}^{r-1} \binom{r}{i} x^{r-i} b^i - \sum_{i=0}^{r-1} \binom{r}{i} x^{r-i} a^i \\ &= \sum_{i=0}^{r-1} \binom{r}{i} x^{r-i} (b^i - a^i) > 0.\end{aligned}$$

□

Lemma 4. *Let $x_1, x_2, ..., x_m > 0$ be real values, $x_1 + x_2 + ... + x_m = c$; moreover $c, u, t \in \mathscr{R}^+$ such that $c < u < t$, and let $r > 1$ be an integer. Then*

$$\sum_{i=1}^{m} (t^r - (t - x_i)^r) > u^r - (u - c)^r.$$

Proof. We shall prove the statement by induction. For $m = 1$ the statement looks like

$$t^r - (t-c)^r > u^r - (u-c)^r,$$

which is a trivial consequence of Lemma 3. Now let us suppose that the claim is true for $i = 1, ..., m-1$. Let us denote for the sake of simplicity $c' = c - x_m$. Then we have

$$\sum_{i=1}^{m-1} (t^r - (t - x_i)^r) > u^r - (u - c')^r.$$

Now, let us apply Lemma 3 by substituting x_m instead of x; moreover $t = b + x$ and $u = a + x$. We have

$$t^r - (t - x_m)^r > u^r - (u - x_m)^r.$$

Combining the two inequalities, we get

$$\begin{aligned}\sum_{i=1}^{m} (t^r - (t - x_i)^r) &= \sum_{i=1}^{m-1} (t^r - (t - x_i)^r) + (t^r - (t - x_m)^r) \\ &> u^r - (u - c')^r + u^r - (u - x_m)^r.\end{aligned}$$

We claim that this is bigger than $u^r - (u-c)^r$, i.e. we need to prove that

$$u^r - (u - c')^r + u^r - (u - x_m)^r > u^r - (u - c)^r.$$

By simple calculation, this is the same as

$$u^r - (u - x_m)^r > (u - c + x_m)^r - (u - c)^r,$$

which again follows from Lemma 3 by substituting $u = b + x$, $u - c + x_m = a + x$ and $x = x_m$. □

The next result is an immediate consequence of Lemma 4.

Corollary. *Let $x_1, x_2, ..., x_m, t > 0$ be real values, $x_1 + x_2 + ... + x_m = q_r$; moreover $t > 2q_r$, and let $r > 1$ be an integer. Then*

$$\sum_{i=1}^{m}(t^r - (t - x_i)^r) > (2q_r)^r - (q_r)^r.$$

Based on the value of r, we will consider three different types of jobs. For the job j_i we will call the job

(i) *small* if $p_i \le q_r$,

(ii) *medium* if $q_r < p_i \le 2q_r$, and

(iii) *big* if $2q_r < p_i$.

Let J be an input. The *total size (of J)*, denoted by $P(J)$, and the *total size of all small jobs (of J)*, denoted by $P_s(J)$, are defined as follows:

$$P(J) = \sum_{j \in J} p_j,$$

$$P_s(J) = \sum_{j \text{ is small}, j \in J} p_j.$$

Note that $P_s(J) \le P(J)$. If J is clear from the context, then we will abbreviate the total size $P(J)$ and the total size of all small jobs $P_s(J)$ by P and P_s, respectively. We shall use the same notation for subsets of J.

Let m_i be a machine. Recall that

$$\mathrm{ld}(m_i) = \sum_{j \in m_i} p_j$$

is called the *load of machine m_i* and $\mathrm{job}(m_i)$ is the linearly ordered *set of jobs of machine m_i*.

3 Lower Bound

Lemma 5. *Let* **A** *be an arbitrary algorithm. Let us suppose that* **A** *uses at least two machines when scheduling J. Consider any two machines m_i, m_j. If $ld(m_i) \ge ld(m_j)$ and we reschedule any job j_k with $p_k < ld(m_j)$ from the machine m_j to the machine m_i, then* $\mathbf{A}(r, J)$ *will grow.*

Proof. Let $m_i, m_j \in M_{A,J}, i \neq j$ and $j_k \in J$, $k \in job(m_j)$ and $0 < p_k < ld(m_j) \le$

$ld(m_i)$. We have to prove that

$$(ld(m_i)+p_k)^r+(ld(m_j)-p_k)^r \geq ld(m_i)^r+ld(m_j)^r.$$

This inequality is equivalent to the following:

$$(ld(m_i)+p_k)^r-ld(m_i)^r \geq ld(m_j)^r-(ld(m_j)-p_k)^r,$$

which trivially follows from Lemma 3. □

The next result is a direct consequence of Lemma 5.

Corollary. *Let* **A** *be an arbitrary algorithm. Let us suppose that* **A** *uses at least two machines when scheduling J. Consider any two machines m_i, m_j. If $ld(m_i) \geq ld(m_j)$ and we reschedule any job j_k with $p_k < ld(m_i)$ from the machine m_i to the machine m_j so that after rescheduling the load of the machine m_i will still be larger than the machine m_j,* $\mathbf{A}(r,J)$ *will be smaller.*

Lemma 6. *An online algorithm which never purchases a second machine is not constant competitive.*

Proof. We will prove our statement by contradiction. Let **A** be a C-competitive online algorithm such that $|M_{\mathbf{A},J}|=1$ for each input J. Let J_k be an input having k jobs, each of size 1. Algorithm **A** will use only one machine and so the cost of **A** is

$$\mathbf{A}(r,J_k)=P(J_k)^r+1=k^r+1. \tag{2}$$

To get a contradiction it is enough to see that

$$\mathbf{A}(r,J_k) \leq C \cdot \mathbf{OPT}(r,J_k)$$

cannot be true for each k.

It is not hard to see that the optimal algorithm schedules one job to individual machines. If the optimum were to schedule two jobs to one machine, then the cost of this machine would be

$$2^r+1.$$

Scheduling these two jobs on different machines would have a cost of

$$1^r+1^r+2=4,$$

so we could decrease the cost of the optimum by putting these two jobs on different machines because clearly

$$2^r+1>4,$$

as $r \geq 2$. So we have

$$\mathbf{OPT}(r,J_k)=2\cdot k. \tag{3}$$

Equations (2) and (3) are true for all k, so we should have

$$\mathbf{A}(r,J_k) = k^r + 1 \le C \cdot \mathbf{OPT}(r,J_k) = C \cdot 2 \cdot k$$

for all k, which is a contradiction. $\square$

Remark. The same method of proof works if we use job sizes ε (instead of sizes 1).

Proposition 1. *Let J be a finite sequence of arbitrarily small $\varepsilon \in \mathscr{R}^+$ jobs, having an even k number of jobs. Then* **OPT** *purchases at least two machines if $b_r \le P(J)$.*

Proof. To prove our statement, we have to check whether the cost of having two machines is smaller than having only one. As k is even this means that

$$2 \cdot (\frac{P(J)}{2})^r + 2 \le P(J)^r + 1.$$

This is true iff

$$\frac{2^{r-1}}{2^{r-1}-1} \le P(J)^r,$$

which holds true if $b_r \le P(J)$. $\square$

Theorem 7. *No online algorithm has a competitive ratio smaller than c_r.*

Proof. Let **A** be a constant competitive online algorithm and J be a finite sequence of arbitrarily small $\varepsilon \in \mathscr{R}^+$ jobs. The sequence terminates depending on the situation where **A** purchases the second machine. When at the moment of purchasing the second machine the number of jobs in J is even, we stop. If the number of jobs is odd at this moment then the input will get one more small job. Algorithm **A** has two choices, namely this last job can be scheduled to the first or the second machine. The cost is smaller if it schedules it to the second machine, so we put it there. Clearly, **A** will purchase a second machine since **A** is constant competitive by assumption and Lemma 6. We now have an even number of jobs in our input so by Proposition 1, **OPT** purchases at least two machines if $b_r \le P$. We will only consider in detail the case where the second machine of A has one small job. The other case is similar to this.

Thus, we consider the following two cases with respect to P.

1. If $P < b_r$, then we have

$$\lim_{\varepsilon \to 0^+} \frac{\mathbf{A}(r,J)}{\mathbf{OPT}(r,J)} = \lim_{\varepsilon \to 0^+} \frac{(P-\varepsilon)^r + \varepsilon^r + 2}{P^r + 1} = \frac{P^r + 2}{P^r + 1} \ge c_r,$$

because by Lemma 2

$$\frac{b_r^r + 2}{b_r^r + 1} = c_r.$$

2. If $b_r \leq P$, then we have

$$\begin{aligned}\lim_{\varepsilon\to 0^+}\frac{\mathbf{A}(r,J)}{\mathbf{OPT}(r,J)} &\geq \lim_{\varepsilon\to 0^+}\frac{(P-\varepsilon)^r+\varepsilon^r+2}{2\cdot(\frac{P}{2})^r+2} \\ &= \frac{P^r+2}{2\cdot(\frac{P}{2})^r+2} \geq c_r,\end{aligned}$$

because by Lemma 1 and by Lemma 2

$$\frac{b_r^r+2}{2\cdot(\frac{b_r}{2})^r+2}=\frac{b_r^r+2}{b_r^r+1}=c_r.$$

□

4 Properties of OPT

From now on let **OPT** be an optimal offline algorithm. To prove the competitiveness of our algorithm in Section 5, we consider the following relaxed problem. We permit preemption for every small job, but not for any medium or big job. In this case, *preemption* means that the execution of a job can be divided into non-overlapping time slots, and these parts can be executed by different machines. Moreover, it is easy to see that the total cost of any optimal offline algorithm for the relaxed problem is a lower bound of the total cost of any optimal offline algorithm for the original problem. Formally, let $\mathbf{OPT}_R$ be an optimal offline algorithm for the relaxed problem and **OPT** be an optimal offline algorithm for the original problem. Then $\mathbf{OPT}_R(J) \leq \mathbf{OPT}(J)$ for each input J.

Proposition 2. *We have $l_r \cdot P(J) \leq \mathbf{OPT}_R(r,J)$ for each input J.*

Proof. We will consider the following two cases with respect to J.

1. Let us suppose that we have $|M|$ machines and P is equally distributable among the $|M|$ machines. This is the best possible scheduling and the cost of this scheduling is
$$|M|\cdot(\frac{P}{|M|})^r+|M|. \tag{4}$$
By a simple computation, this is minimal if
$$|M| = P(J)\cdot\sqrt[r]{r-1}.$$
Substituting we get $\mathbf{OPT}_R(r) = l_r \cdot P(J)$. Hence, the statement holds in this case.

We should make the following remark here. The estimate of the lower bound is always true. However, $l_r \cdot P(J)$ is generally not an integer and we may only have an integral number of machines. If this is not integer, then from the derivative it follows that (4) decreases until it reaches a minimum and then it increases. So we have to take the largest integer before the minimum value and the smallest integer after that. We have to compute (4) using these two integers and the smallest value will give a better estimation of the minimum.

2. If P is not equally distributable among the $|M|$ machines, then $l_r \cdot P(J) < \mathbf{OPT}_R(J)$ by Lemma 5. Thus, the statement holds.

□

We will call a machine *overloaded* if its jobs can be distributed into two sets and the total size of each set is greater than q_r.

Lemma 8. $\mathbf{OPT}_R$ *has no overloaded machine.*

Proof. We will prove our statement by contradiction. We suppose that $\mathbf{OPT}_R$ has an overloaded machine. Let m_i be this machine. Since m_i is overloaded, we can distribute the jobs of m_i into two sets S_1 and S_2 such that $q_r < P(S_1), P(S_2)$ and $P(S_1) + P(S_2) = P(m_i)$. In this case, the total cost of m_i is

$$1 + (P(S_1) + P(S_2))^r.$$

However, if $\mathbf{OPT}_R$ were to purchase two new machines instead of m_i and schedule jobs of S_1 to the first and jobs of S_2 to the second new machine, this cost would be

$$2 + P(S_1)^r + P(S_2)^r,$$

which is $< 1 + (P(S_1) + P(S_2))^r$ because

$$\binom{r}{1} \cdot P(S_1)^{r-1} \cdot P(S_2) + ... + \binom{r}{r-1} \cdot P(S_1) \cdot P(S_2)^{r-1} >$$

$$> (2^r - 2) \cdot (\frac{1}{2})^r \cdot \frac{2^{r-1}}{2^{r-1} - 1} = 1.$$

This is a contradiction. □

Corollary. *The following statements hold for* $\mathbf{OPT}_R$.

1. *Any two big or medium jobs are scheduled to two different machines;*
2. *If a machine processes a big or a medium job, then the rest load is at most* q_r*;*
3. *The total load of a machine is at most* $2q_r$ *if the machine processes only small jobs.*

Proof.

1. It immediately follows from Lemma 8 and the definition of the medium job and the big job.

2. By Lemma 8 and the definition of the small job, the medium job, and the big job, it is easy to see that the statement holds.

3. We prove our statement by contradiction. To get a contradiction, we suppose that $\mathbf{OPT}_R$ has a machine m_i such that $\mathrm{ld}(m_i) > 2q_r$ and $p(j) \le q_r$ for each $j \in \mathrm{job}(m_i)$. From the definition of a small job and $\mathrm{ld}(m_i) > 2q_r$ we have $|\mathrm{job}(m_i)| \ge 3$. Moreover, m_i is overloaded since $\mathrm{ld}(m_i) > 2q_r$. We are now dealing with the relaxed optimum where we can use preemption for the small jobs. So we can make two sets of small jobs such that each of them will have a total size $> q_r$ and so by Lemma 8, the schedule of $\mathbf{OPT}_R$ is not optimal. This is a contradiction.

□

Proposition 3. $\mathbf{OPT}_R$ *has at most one machine with a load less than* q_r*.*

Proof. We will prove our statement by contradiction. We will suppose that $\mathbf{OPT}_R$ has two machines with a load less than q_r. Let $m_1, m_2 \in M$ be two distinct machines such that $\mathrm{ld}(m_1), \mathrm{ld}(m_2) < q_r$. Then the total cost of m_1 and m_2 is $\mathrm{ld}(m_1)^r + \mathrm{ld}(m_2)^r + 2$. Following the proof of Lemma 8, it is obviously greater than $(\mathrm{ld}(m_1) + \mathrm{ld}(m_2))^r + 1$, i.e., $\mathbf{OPT}_R$ schedules the load $\mathrm{ld}(m_1) + \mathrm{ld}(m_2)$ to one machine. Hence there is a contradiction. □

5 Algorithm

In the bin packing problem, there are items with positive sizes and unit capacity bins. The task is to pack the items into as few bins as possible with respect to the bin capacity.

FF is one of the best-known bin packing algorithms. It packs the items one by one, and the next item is always packed into the first bin it fits. If the next item does not fit into any bin, **FF** opens a new bin for it and packs the item into this new bin.

Now, we present our algorithm and prove its competitiveness. We will call our algorithm the *Different Machines* algorithm, denoted by **DM**.

5.1 Description

1. If j_k is a small job or a medium job, then **DM** schedules j_k to a smallmedium (*SM*) machine so that the load of a smallmedium machine cannot exceed $2q_r$. Moreover, **DM** applies the **FF** rule to decide which machine will process j_k if there exists more than one smallmedium machine. If there is no smallmedium machine which could process the job, then **DM** purchases a new (smallmedium) machine.

2. If j_k is a big job, then **DM** schedules j_k to a new machine and it will not schedule any other job to this machine.

In our algorithm, we will use two types of machines. The first type is called SM, which can receive only small and medium jobs, and its maximum possible load is $2q_r$. The second type is called B, which can process only big jobs (*big machines*, B). In the proof we will further divide the SM machines into those that process only small jobs called *small machines* (S) and the remaining machines from SM will be called *medium machines* (M). We will suppose that the algorithm uses a small, b medium and c big machines.

To prove the competitiveness of our algorithm, we need the following lemma.

Lemma 9. *Let $\alpha, \beta, \gamma, \delta \in \mathscr{R}^+$ such that*

$$c_r < \frac{\alpha}{\beta},$$

$$\gamma \leq c_r \cdot \delta < \alpha,$$

$$\delta < \beta.$$

Then

$$c_r < \frac{\alpha - \gamma}{\beta - \delta}.$$

Proof. We prove our statement by contradiction. We suppose $\frac{\alpha-\gamma}{\beta-\delta} \leq c_r$. Because of $\gamma \leq c_r \cdot \delta$ we have

$$\frac{\alpha - c_r \cdot \delta}{\beta - \delta} \leq \frac{\alpha - \gamma}{\beta - \delta} \leq c_r,$$

and from the second inequality we have

$$\frac{\alpha}{\beta} \leq c_r,$$

which is a contradiction. □

5.2 Competitiveness

To prove the competitiveness of **DM**, we need the following concepts. Let $I \subseteq J$. Then we denote by $J - I$ the input if we remove every job of I from J. Moreover, we will call J the *minimal counterexample* if $c_r > \mathbf{DM}(J)/\mathbf{OPT}_R(J)$ and there is no $I \subseteq J$ such that I consists of only big jobs, $I \neq \emptyset$, and $c_r \leq \mathbf{DM}(J-I)/\mathbf{OPT}_R(J-I)$.

Lemma 10. *Consider the (relaxed) optimal scheduling of a minimal counterexample J. In this case, there is no big job that uses a machine on its own in the optimal scheduling.*

Proof. Suppose a big job X with size x uses a machine by itself in the optimal scheduling. This job is also by itself in **DM**. Let $J' = J \setminus \{X\}$. It follows from Lemma 9 that

$$\frac{\mathbf{DM}(J')}{\mathbf{OPT}(J')} = \frac{\mathbf{DM}(J) - 1 - x^2}{\mathbf{OPT}(J) - 1 - x^2} > c_r,$$

which contradicts the fact that J is a minimal counterexample. □

Lemma 11. *If we have a big job in a minimal counterexample, then we do not have a small machine in the relaxed optimum.*

Proof. Suppose we have a big job whose size is $2 \cdot q_r + x$. Let the job be scheduled on machine m' in the relaxed optimal scheduling. We have no medium job on this machine because then the machine would be overloaded and this is not possible by Lemma 8. According to Lemma 10 we have some small jobs scheduled on the machine m' in the relaxed optimal scheduling. Let the sum of sizes of the small jobs on this machine be y. Now if we have an additional machine m'' with small jobs, then the load of this machine is $\leq 2 \cdot q_r$ because the machine cannot be overloaded by Lemma 8 in the relaxed optimum. But then we can reschedule some small jobs of size of $y', y' \leq y$ of m' to m'' (until m'' reaches the load of $2 \cdot q_r$) as then the cost will decrease because of Corollary 3. If we have still small jobs on m' we can reschedule the jobs on machine m'' on two machines, each having a load of q_r, and after rescheduling we repeat the above procedure. So we can decrease the relaxed optimum. Finally the machine with the big job will have only this big job and it can be deleted by Lemma 10. □

According to Lemma 11 if we have a big job in the minimal counterexample then the structure of the relaxed optimum is the following. We have b machines with one medium job and c machines with one big job. The machine with the largest big item must have some small items too. If this is not the case then this machine can be deleted by Lemma 10. All machines must have the same load, say $L > 2 \cdot q_r$. If this is not the case we will reschedule some small jobs and decrease the cost.

Lemma 12. *If the minimal counterexample J has a big job, then $P_s(J) \leq q_r$.*

Proof. Let us suppose that we have a big job. Then according to Lemma 10 on the machine where the big job is scheduled, we have small item(s) in the case of the relaxed optimum solution. We may have some medium machines in the relaxed optimum but according to Lemma 11 we do not have a machine with only small items. We know that the load of each machine is the same; let us say $L > 2 \cdot q_r$.

Let us suppose that $P_s(J) > q_r$ and that we have m machines in the relaxed optimum. Let us take $x_1, x_2, ..., x_m$ from the small jobs so that $x_1 + x_2 + ... + x_m = q_r$. Then the remaining load on the machines will be $L - x_i$, $i = 1, ..., m$. By Corollary 2, we get

$$\sum_{i=1}^{m} (L^r - (L - x_i)^r) > (2q_r)^r - (q_r)^r = (q_r)^r + 1,$$

where the last part is valid by Lemma 1. But this means that if we schedule the small jobs on a separate machine, the cost will be smaller, which is a contradiction. □

Lemma 13. *There is no big job at all in the minimal counterexample.*

Proof. Let us suppose we have a big job in the relaxed optimum. Then because of Lemma 12, $P_s(J) \leq q_r$. In the **DM** scheduling the big and the medium jobs are on different machines; here let $m = b + c$. So the **DM** scheduling has at most $m+1$ machines. In the relaxed optimal schedule the number of machines is m and the load of each machine here is L. It is evident that the maximum load for the **DM** packing is at most L. So if we had

$$\begin{aligned} L^r \cdot (m+1) + m + 1 &\le c_r(L^r \cdot m + m) \\ (m+1)(L^r+1) &\le c_r \cdot m(L^r+1) \\ m+1 &\le c_r \cdot m, \end{aligned}$$

then the input would not be a counterexample. But this is true for $m \ge 3$, as then $c_r \ge 4/3$. So we have either $m = 1$ or $m = 2$.

Case 1. If $m = 1$ then in the relaxed optimal schedule we have one machine, only one big job and all the small jobs are scheduled on this machine. With the **DM** schedule we have two machines; the big job is scheduled on one machine and all the small jobs are scheduled on the other machine. Let the size of the big job be $y > 2q_r$ and the sum of the small jobs sizes be $x \le q_r$. Then

$$\mathbf{DM}(J) = x^r + y^r + 2,\ \mathbf{OPT}_R(J) = (x+y)^r + 1.$$

We claim that

$$\frac{\mathbf{DM}(J)}{\mathbf{OPT}_R(J)} = \frac{x^r + y^r + 2}{(x+y)^r + 1} \le c_r = 1 + \frac{2^{r-1}-1}{2^r-1}. \tag{5}$$

This is clearly equivalent to

$$x^r + y^r + 2 \le (x+y)^r + 1 + \frac{2^{r-1}-1}{2^r-1} \cdot ((x+y)^r + 1).$$

Rearranging, we have

$$1 \le (x+y)^r - x^r - y^r + \frac{2^{r-1}-1}{2^r-1} \cdot ((x+y)^r + 1).$$

Because of Lemma 3 we have $(x+y)^r - x^r - y^r > 0$. However, $x + y > 2q_r$ and so

$$\frac{2^{r-1}-1}{2^r-1} \cdot ((x+y)^r + 1) \ge \frac{2^{r-1}-1}{2^r-1} \cdot ((2 \cdot q_r)^r + 1) = \frac{2^{r-1}-1}{2^r-1} \cdot (\frac{2^{r-1}}{2^{r-1}-1} + 1) = 1.$$

We see that (5) holds.

Case 2. If $m = 2$ then we have two machines in the relaxed optimum. We may have two big jobs or one big and one medium job (and they are on different machines). In both cases we must have some small jobs and the sum of small jobs is $\le q_r$ and both machines have a load of L. If we have two big jobs then in the **DM** scheduling the two big jobs are on different machines and the small jobs are on a third machine. If we have one big and one medium job, then the big job is alone in the **DM** scheduling, and the medium item and the small items will be scheduled on two additional machines. Let $x = P_s(J)$. Then

$$\mathbf{DM} \leq x^r + 2L^r + 3,\ \mathbf{OPT}_R(J) = 2L^r + 2.$$

We have to prove that

$$\frac{\mathbf{DM}}{\mathbf{OPT}_R(J)} \leq \frac{x^r + 2L^r + 3}{2L^r + 2} \leq c_r.$$

It is easy to see that if this inequality is valid for the largest possible x (it is q_r and for the smallest possible L (it is $2q_r$) then it is always valid. So it is enough to demonstrate that

$$\frac{(q_r)^r + 2(2q_r)^r + 3}{2(2q_r)^r + 2} \leq c_r.$$

But we know that $(2q_r)^r = 2\,(q_r)^r + 1$ (Lemma 1), and so $(q_r)^r = ((2q_r)^r - 1)/2$. So we should have

$$\frac{((2q_r)^r - 1)/2 + 2(2q_r)^r + 3}{2(2q_r)^r + 2} = \frac{(2q_r)^r - 1 + 4(2q_r)^r + 6}{4(2q_r)^r + 4} = \frac{5(2q_r)^r + 5}{4(2q_r)^r + 4} = 5/4 \leq c_r.$$

Hence there is no big job in a minimal counterexample. □

5.3 Proof of the competitiveness

We know that if we have a minimal counterexample then we do not have a big job in this counterexample. So it is enough to have inputs with small and medium items. For these inputs we will prove that for $r = 2, 3, 4$

$$\frac{\mathbf{DM}(J)}{\mathbf{OPT}_R(J)} \leq c_r. \tag{6}$$

Let us suppose that the **DM** scheduling of J uses m machines. We have the following cases.

m=1 Both **DM** and $\mathbf{OPT}_R$ buy one machine and we have the same schedule. So (6) is true.

m=2 We know that $b_r < P(J) \leq 2 \cdot b_r$ because we have two machines in the **DM** scheduling. Because of the first part of this inequality, $\mathbf{OPT}_R$ has at least two machines. Following the proof of Proposition 2, $\mathbf{OPT}_R$ will use no more than 3 machines if

$$\sqrt[r]{r-1} \cdot P \leq 3.$$

This means that

$$\sqrt[r]{r-1}\cdot 2\cdot b_r \leq 3,$$
$$\sqrt[r]{r-1}\cdot\sqrt[r]{\frac{2^{r-1}}{2^{r-1}-1}} \leq \frac{3}{2},$$
$$(r-1)\cdot\frac{2^{r-1}}{2^{r-1}-1} \leq \left(\frac{3}{2}\right)^r.$$

This is true for $r=2$. If $r>2$ then on the left hand side the ratio is the largest with $r=3$, namely here 4/3, so it is enough to have

$$(r-1)\cdot\frac{4}{3} \leq \left(\frac{3}{2}\right)^r.$$

This is true for $r=3$ and if we increase r by one, the left hand side increases by 4/3, the right side by more. So we have at most three machines in the relaxed optimum.

We will need the following lemma.

Lemma 14. *For $0\leq x\leq b_r$, we have $\left(\frac{b_r+x}{2}\right)^r \geq \left(\frac{b_r}{2}\right)^r + \frac{2^r-1}{2^r}x^r$.*

Proof.

$$\begin{aligned}\left(\frac{b_r+x}{2}\right)^r &= \left(\frac{1}{2}\right)^r\left(\sum_{i=0}^{r}\binom{r}{i}x^{r-i}b_r^i\right) = \left(\frac{1}{2}\right)^r\left(b_r^r+\sum_{i=0}^{r-1}\binom{r}{i}x^{r-i}b_r^i\right)\\ &\geq \left(\frac{1}{2}\right)^r\left(b_r^r+\sum_{i=0}^{r-1}\binom{r}{i}x^r\right) = \left(\frac{1}{2}\right)^r\left(b_r^r+(2^r-1)x^r\right)\\ &= \left(\frac{b_r}{2}\right)^r+\frac{2^r-1}{2^r}x^r.\end{aligned}$$

□

Case 1. We have two machines in the relaxed optimum. The worst case for **DM** is that we have a load of b_r on the first machine and the rest (i.e. x) on the second one ($0\leq x\leq b_r$). It is enough to prove that

$$(b_r)^r+x^r+2 \leq c_r\left(2\left(\frac{b_r+x}{2}\right)^r+2\right).$$

Using Lemma 14 it is enough to have

$$(b_r)^r+x^r+2 \leq c_r\left(2\left(\frac{b_r}{2}\right)^r+2\cdot\frac{2^r-1}{2^r}x^r+2\right).$$

Here the multiplicator of x^r is larger on the right hand side so it is enough to see that

$$(b_r)^r + 2 \le c_r \left(2\left(\frac{b_r}{2}\right)^r + 2\right).$$

Because $2\left(\frac{b_r}{2}\right)^r + 1 = (b_r)^r$, we get

$$\frac{(b_r)^r + 2}{(b_r)^r + 1} \le c_r,$$

which is true as the left hand side is c_r.

Case 2. We have three machines in the relaxed optimum. We have to prove that

$$(b_r)^r + x^r + 2 \le c_r \left(3\left(\frac{b_r + x}{3}\right)^r + 3\right). \tag{7}$$

We know that $c_r \ge 4/3$, so the right hand side is at least 4. This inequality holds if $(b_r)^r + x^r \le 2$.

We claim that this is true if $x \le b_r/2$. In this case the largest possible value of $(b_r)^r + x^r$ is

$$\frac{2^{r-1}}{2^{r-1}-1} + \frac{1}{2^r} \cdot \frac{2^{r-1}}{2^{r-1}-1}.$$

This is the largest if r is the smallest and then it is $\frac{4}{3} + \frac{1}{8} \cdot \frac{4}{3} = \frac{3}{2} < 2$. So the inequality holds if $x \le b_r/2$.

If $\frac{b_r+x}{3} \ge b_r/2$. Using the lower bound on the right hand side of (7) the right hand side will decrease. The right hand side will decrease further if we replace c_r by 4/3. The left hand side will increase if we replace x^r by $(b_r)^r$. So we have to verify that

$$2(b_r)^r + 2 \le \frac{4}{3}\left(3\left(\frac{b_r}{2}\right)^r + 3\right),$$
$$2(b_r)^r + 2 \le 4\left(\frac{b_r}{2}\right)^r + 4.$$

Again using $2\left(\frac{b_r}{2}\right)^r + 1 = (b_r)^r$, we get

$$2(b_r)^r + 2 \le 2(b_r)^r + 2,$$

and this is clearly true. So we have now examined the case where the relaxed optimum has three machines.

For m ≥ 3, we will use Proposition 2 to estimate the optimum. Let the load of the

DM machines be $t_1, t_2, ..., t_m$. Then we have to prove that

$$(t_1)^r + (t_2)^r + ... + (t_m)^r + m \leq c_r \cdot l_r \cdot P(J). \tag{8}$$

Let the smallest load be x. Then clearly the load of all the other machines is $> b_r - x$. We will suppose that $0 \leq x \leq b_r/2$. Let the load of a further machine be t. Then $b_r - x < t \leq b_r$. Let us investigate how to make the two sides of (8) change if we increase t. The derivative of the right hand side (as a function of t) is $l_r \cdot c_r$. On the left hand side the load of the other machines does not change and the number of machines remains the same. So the first derivative of the left hand side is (as a function of t) $r \cdot t^{r-1} > 0$ and the second derivative is $r(r-1) \cdot t^{r-2} > 0$. So t^r is convex and its first derivative is monotone increasing. This means that the critical points in (8) are the possible endvalues for this machine, i.e. if its load is a possible minimum or a possible maximum.

So it is sufficient to take instead of (8) a stronger inequality where the load of one machine is x, and the load of all the others is either $b_r - x$ or b_r. If all the others have a load of b_r, then the worst case for x is either $x = 0$ or $x = b_r$. Here the first possibility is worse and the right hand side does not increase but the left hand side does.

We now know that one machine may have a load of 0 and all the others have a load of b_r. The other possibility is that one machine has a load of $0 \leq x \leq b_r/2$ and all the others have a load of $b_r - x$.

We will now examine the first case, i.e. one machine has a load of 0 and all the others have a load of b_r. Let

$$z = (m-1)(b_r - x)^r + (x)^r.$$

The second derivative is positive so we again get that the two worst cases are at the end of our interval. More precisely, one of them is the worst case. We already had $x = 0$, and the second case is $x = b_r/2$. We will look at the first one in the following.

If $\frac{b_r}{2} \leq x \leq b_r$, then the worst case is either $x = \frac{b_r}{2}$, which leads to Case 1, or $x = b_r$, which leads to Case 2.

We have now the following two cases:

Case 1. Each machine has a load of q_r. Then we have:

$$(b_r/2)^r \cdot m + m \leq l_r \cdot c_r \cdot (b_r/2 \cdot m).$$

r	$(b_r)^r + 1.5$	$l_r \cdot c_r \cdot b_r$
2	3.5000	3.7710
3	2.8333	2.9715
4	2.6429	2.6610
5	2.5667	2.4793
6	2.5323	2.3538
10	2.5020	2.0759
20	2.5000	1.8294

Table 1
The values of $(b_r)^r + 1.5$ and $l_r \cdot c_r \cdot b_r$ for certain values of r

Let us substitute $(b_r/2)^r = ((b_r)^r - 1)/2$ (Lemma 1). We get

$$\begin{gathered}\frac{(b_r)^r - 1}{2} \cdot m + m \leq l_r \cdot c_r \cdot \frac{b_r}{2} \cdot m,\\ ((b_r)^r - 1) \cdot m + 2m \leq l_r \cdot c_r \cdot b_r \cdot m,\\ (b_r)^r - 1 + 2 \leq l_r \cdot c_r \cdot b_r,\\ (b_r)^r + 1 \leq l_r \cdot c_r \cdot b_r.\end{gathered}$$

Multiplying with $m-1$,

$$(b_r)^r \cdot (m-1) + m - 1 \leq l_r \cdot c_r \cdot b_r \cdot (m-1). \tag{9}$$

Case 2. In the second case all the bins are full and one is empty. Here

$$(b_r)^r \cdot (m-1) + m \leq l_r \cdot c_r \cdot b_r \cdot (m-1). \tag{10}$$

This is a stronger condition than that of Case 1. So it is enough to investigate this one. If this is valid then Case 1 will be fulfilled too. Dividing by $m-1$, we get

$$(b_r)^r + 1 + \frac{1}{m-1} \leq l_r \cdot c_r \cdot b_r. \tag{11}$$

This will be fulfilled harder if m is small. For $m = 3$, we have

$$(b_r)^r + 1.5 \leq l_r \cdot c_r \cdot b_r. \tag{12}$$

Substituting for some r values in

$$\frac{2^{r-1}}{2^{r-1}-1} + 1.5 \leq \frac{r}{\sqrt[r]{(r-1)^{r-1}}} \cdot (2 - \frac{2^{r-1}}{2^r - 1}) \cdot \sqrt[r]{\frac{2^{r-1}}{2^{r-1}-1}},$$

we get the values shown in Table 1.

Now we have proved (6) for $r = 2,3,4$. This means that **DM** is the best possible online algorithm for this problem as its competitive ratio equals the lower bound for this problem for $r = 2, 3, 4$. □

It is not hard to see that for $r \geq 5$ the competitive ratio of **DM** is larger than c_r.

A final remark. We can have an upper bound of the competitive ratio of **DM** for $r \geq 5$. To obtain this, we can use (12). If we replace c_r by some $d_r \in \mathscr{R}^+$ with $c_r \leq d_r$ here, we get

$$(b_r)^r + 1.5 \leq l_r \cdot d_r \cdot b_r. \tag{13}$$

From here

$$\frac{2^{r-1}}{2^{r-1}-1} + \frac{3}{2} \leq d_r \cdot \frac{r}{\sqrt[r]{(r-1)^{r-1}}} \cdot \sqrt[r]{\frac{2^{r-1}}{2^{r-1}-1}} \tag{14}$$

$$\left(\frac{2^{r-1}}{2^{r-1}-1} + \frac{3}{2}\right) \cdot \frac{\sqrt[r]{(r-1)^{r-1}}}{r} \cdot \sqrt[r]{\frac{2^{r-1}-1}{2^{r-1}}} \leq d_r \tag{15}$$

and so the left hand side is an upper bound of the competitive ratio for $r \geq 5$. Naturally c_r is a lower bound for the competitive bound for every r, so for $r \geq 5$ we have an interval for the competitive ratio.

Acknowledgment

Research of György Dósa is supported in part by the National Research, Development and Innovation Office –NKFIH– under the grant SNN 129364.

References

[1] W. Townsend. The single machine problem with a quadratic penalty function of completion times: A branch-and-bound solution. *Management Science*, 24(5):530–534, 1978.

[2] P. C. Bagga and K. R. Kalra. Note - a node elimination procedure for townsend's algorithm for solving the single machine quadratic penalty function scheduling function. *Management Science*, 26:633–636, 1980.

[3] F. D. Croce, R. Tadei, P. Baracco, and R. D. Tullio. On minimizing the weighted sum of a quadratic completion time on a single machine. *Proceedings IEEE International Conference on Robotics and Automation*, 3:816–820, 1993.

[4] W. Szwarc and S. K. Mukhopadhyay. Minimizing a quadratic cost function of waiting times in single-machine scheduling. *Journal of the Operational Research Society*, 46(6):753–761, 1995.

[5] C. M. Wei and J. B. Wang. Single machine quadratic penalty function scheduling with deteriorating jobs and group technology. *Applied Mathematical Modelling*, 34:3642–3647, 2010.

[6] T. C. E. Cheng and Z. Liu. Parallel machine scheduling to minimize the sum of quadratic completion times. *IIE Transactions*, 36:11–17, 2004.

[7] J. C. Ho, T. L. B. Cheng, A. J. Ruiz-Torres, and F. J. López. Minimizing the normalized sum of square for workload deviations on m parallel machines. *Computers and Industrial Engineering*, 56:186–192, 2009.

[8] R. Walter and A. Lawrinenko. A note on minimizing the normalized sum of squares for workload deviations on m parallel machines. *Computers and Industrial Engineering*, 75:257–259, 2014.

[9] S. Schwerdfeger and R. Walter. Improved algorithms to minimize workload balancing criteria on identical parallel machines. *Computers and Operations Research*, 93:123–134, 2018.

[10] Y. Ouazene, N. Q. Nguyen, and F. Yalaoui. Workload balancing on identical parallel machines: Theoretical and computational analysis. *Applied Sciences*, 11(8), 2021.

[11] A. K. Chandra and C. K. Wong. Worst-case analysis of a placement algorithm related to storage allocation. *SIAM J. Comput.*, 4(3):249–263, 1975.

[12] A. Avidor, Y. Azar, and J. Sgall. Ancient and new algorithms for load balancing in the l_p norm. *Algorithmica*, 29(3):422–441, 2001.

[13] C. Imreh and J. Noga. *Scheduling with Machine Cost*, volume 1671 of *Lecture Notes in Computer Science*, pages 168–176. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[14] G. Dósa and Y. He. Better online algorithms for scheduling with machine cost. *SIAM Journal on Computing*, 33(5):1035–1051, 2004.

[15] G. Dósa and Z. Tan. New upper and lower bounds for online scheduling with machine cost. *Discrete Optimization*, 7(3):125–135, 2010.

[16] Y. Jiang and Y. He. Preemptive online algorithms for scheduling with machine cost. *Acta Informatica*, 41(6):315–340, 2005.

[17] G. Dósa and Y. He. Scheduling with machine cost and rejection. *Journal of Combinatorial Optimization*, 12(4):337–350, 2006.

[18] J. Nagy-György and C. Imreh. Online scheduling with machine cost and rejection. *Discrete Applied Mathematics*, 155(18):2546–2554, 2007.

[19] Y. He and S. Cai. Semi-online scheduling with machine cost. *Journal of Computer Science and Technology*, 17(6):781–787, 2002.

[20] C. Imreh. Online scheduling with general machine cost functions. *Discrete Applied Mathematics*, 157(9):2070–2077, 2009.

[21] I. Akaria and L. Epstein. An optimal online algorithm for scheduling with general machine cost functions. *J. Sched.*, 23(2):155–162, 2020.

[22] J. Csirik, G. Dósa, and D. Kószó. Online scheduling with machine cost and a quadratic objective function. In A. Chatzigeorgiou, R. Dondi, H. Herodotou, C. A. Kapoutsis, Y. Manolopoulos, G. A. Papadopoulos, and F. Sikora, editors, *SOFSEM 2020: Theory and Practice of Computer Science - 46th International Conference on Current Trends in Theory and Practice of Informatics, SOFSEM 2020, Limassol, Cyprus, January 20-24, 2020, Proceedings*, volume 12011 of *Lecture Notes in Computer Science*, pages 199–210. Springer, 2020.