Online scheduling with general machine cost functions

Cs. Imreh

Department of Informatics, University of Szeged, Árpád tér 2, H-6720 Szeged, Hungary

Received 10 October 2006; received in revised form 30 June 2007; accepted 4 October 2007
Available online 21 February 2008

Abstract

For most scheduling problems the set of machines is fixed initially and remains unchanged for the duration of the problem. Recently online scheduling problems have been investigated with the modification that initially the algorithm possesses no machines, but that at any point additional machines may be purchased. In all of these models the assumption has been made that each machine has unit cost. In this paper we consider the problem with general machine cost functions. Furthermore we also consider a more general version of the problem where the available machines have speed, the algorithm may purchase machines with speed 1 and machines with speed $s$. We define and analyze some algorithms for the solution of these problems and their special cases. Moreover we prove some lower bounds on the possible competitive ratios.

Keywords: Online algorithms; Scheduling; Competitive analysis

1. Introduction

In machine scheduling, we typically have a fixed set of machines. The scheduling algorithm makes no decision regarding the initial set of machines nor is it allowed to change the set of machines later. It is usually assumed that the provided machines can be utilized without cost. It is a natural idea to start an investigation of how scheduling problems change when machine costs are considered.

This problem was first considered in [9]. The differences to the classical online scheduling problem are that (1) no machines are initially provided, (2) when a job is revealed the algorithm has the option to purchase new machines, and (3) the objective is to minimize the sum of the makespan and the cost of the machines. In the paper [9] it is supposed that each machine has unit cost. The problem of scheduling with machine cost is investigated in the online list model where the jobs arrive one-by-one and the decision maker has to schedule the jobs without any information on the further jobs, and also in the time model where each job has a release time and the jobs cannot be started before their release time. In the list model a \((1 + \sqrt{5})/2 = 1.618\)-competitive algorithm is presented, in the time model an 1.693-competitive algorithm is given. Furthermore it has been proven that no online algorithm can achieve smaller competitive ratio than 4/3 in the list model, and no online algorithm can achieve smaller competitive ratio than 1.186 in the time model.
Later the problem in the list model was further analyzed. In [13] a lower bound on the competitive ratio of the possible online randomized algorithms is presented. In [8] some semi-online versions of the problem (known largest job, known total processing time) are investigated. In [5] the list model is further analyzed: the upper bound is improved to 1.5798 and also the semi-online results from [8] are slightly improved. The scheduling problem with machine cost where it is allowed to preempt the jobs is studied in [14].

Further extension of the problem is defined in [12]. In this version the decision maker has to purchase the machines and there is also the possibility to reject the jobs. In the paper a $3 + \sqrt{5})/2$-competitive algorithm is presented for the solution of the problem. The special case where the size of each job is not greater than 1 is also studied, an optimal 2-competitive algorithm is presented for the solution of the problem in [6].

There are only a few works on other online problems where it is allowed to purchase extra resources. In [4] the generalized version of the paging problem is investigated. In that paper the cost of purchasing extra memory is described by an arbitrary cost function. It is determined that which functions allows us to develop the constant competitive algorithm and some particular cost functions are investigated in detail.

All of the papers on scheduling with machine cost investigate the simplest case, in these papers it is supposed that each machine has unit cost. In this paper we consider the problem with general cost function. We suppose that the cost of the machines can be arbitrary, it is described by a nondecreasing machine cost function denoted by $c(m)$. The value $c(m)$ is the cost of purchasing the first $m$ machines, in other words the cost of the $m$th machine is $c(m) - c(m - 1)$. The objective is to minimize the total cost of purchasing the machines. We consider the online problem where the jobs arrive one-by-one and the decision maker has to schedule the jobs without any information on the further jobs. We call this model scheduling with general machine cost. We analyze an algorithm from a class of algorithms defined in [9], and we prove that the competitive ratio of that algorithm is $3 + \sqrt{5})/2 \approx 2.618$ for general cost function. We also consider the special case of the small jobs where the processing time of the jobs cannot be larger than the minimal cost of the machines, in this case we give a 2-competitive algorithm. Furthermore we show that no online algorithm can have smaller competitive ratio than 2. Since this bound is also true in the case where the jobs are small, we obtain that our algorithm in this case has the smallest possible competitive ratio.

In the second part of the paper we study a more general scenario where the machines have speed. We suppose that there are two sets of machines $S_1$ contains machines with speed 1, and $S_2$ contains machines with speed $s > 1$. The algorithm has to purchase the machines. The nondecreasing function $c_1$ describes the cost of the machines from set $S_1$ ($c_1(k)$ is the cost of purchasing the first $k$ machines from set $S_1$) and the nondecreasing function $c_2$ describes the cost of the machines from set $S_2$ ($c_2(k)$ is the cost of purchasing the first $k$ machines from set $S_2$). The objective is to minimize the sum of the purchasing cost of the machines and the makespan. We consider again the online problem. We present an algorithm for this general problem which has competitive ratio 6. It is also shown that no online algorithm exists for this problem with competitive ratio smaller than 2.325. We also consider the special case where there is a fixed number of machines, there are $n$ machines with speed 1 and $m$ machines with speed $s$ and the goal is to minimize the makespan. (We obtain this model with $c_1(k) = 0$ if $k \leq n$, $c_1(k) = \infty$ if $k > n$, $c_2(k) = 0$ if $k \leq m$ and $c_2(k) = \infty$ if $k > n$.) This problem can be considered as the special case of the online scheduling problem on related machines (see [1–3]). Similar problems (makespan minimization with two groups of machines) are studied in the case of unrelated machines in [10] and [11]. Using the proof of the competitive ratio from the general machine cost case we obtain that the competitive ratio of the Greedy algorithm which schedules each job on the machines where it can be completed first is 4 in this case (the competitive ratio of this algorithm is $\Theta (\log n)$ in the general related machines scheduling problem). We present a more sophisticated algorithm which has competitive ratio 3.

2. Preliminaries

Throughout the remainder of the paper we will use the following notations. The jobs will be labelled $j_1, \ldots, j_n$ and presented to the online algorithm in this order. We denote the processing time of job $j_i$ by $p_i$. The total amount of processing time of the first $\ell$ jobs is $P_\ell = \sum_{i=1}^{\ell} p_i$, the maximal processing time of the first $\ell$ jobs is $M_\ell = \max_{1 \leq i \leq \ell} p_i$. Furthermore for an arbitrary set $I$ of jobs let $P(I) = \sum_{j \in I} p_j$, and $M(I) = \max_{j \in I} p_j$.

For the identical machines case the cost of the machines is described by a nondecreasing machine cost function $c(m)$, as it is given in the introduction. In the related machines case where $S_1$ contains machines with speed 1 and $S_2$ contains machines with speed $s > 1$, if a job has processing time $p_i$ then it takes $p_i$ and $p_i/s$ time to process it on the machines of $S_1$ and $S_2$, respectively. The time which is needed to process the jobs assigned to a machine is
called the load of the machine (this is the ratio of the sum of the processing times and the speed of the machine). The cost of the machines is described by two nondecreasing machine cost functions $c_1(m)$ and $c_2(m)$, as it is given in the introduction.

Under either model, we will use $A(\sigma)$ to denote the cost of an algorithm $A$ on a given sequence of jobs $\sigma$. Similarly, we denote the optimal offline cost on a sequence $\sigma$ by $OPT(\sigma)$.

If $A(\sigma) \leq C \cdot OPT(\sigma)$ for every sequence of jobs $\sigma$ then $A$ is called $C$-competitive. The competitive ratio of $A$ is the infimum of all values for which $A$ is $C$-competitive.

We now propose a class of online algorithms for the case of identical machines. This class is defined in [9] for the case where each machine has cost 1. We analyze two particular algorithms from this class. For an increasing sequence $\varphi = (0 = \varphi_1, \varphi_2, \ldots \varphi_i, \ldots)$ we define an online algorithm $A_\varphi$. When job $j_\ell$ is revealed $A_\varphi$ purchases machines (if necessary) so that the current number of machines $i$ satisfies $\varphi_i \leq P_\ell < \varphi_{i+1}$. $A_\varphi$ then assigns job $j_\ell$ to the least loaded machine. This scheduling rule of the jobs is the same as in the classical list scheduling algorithm of Graham [7].

3. Identical machines

3.1. Algorithm for the general case

In this part we consider an online algorithm from the class described in Section 2. Let $\varphi = \frac{1+\sqrt{5}}{2} \approx 1.618$ and $A = A_\varphi$ for $\varphi = (0, c(2)\varphi, 2c(3)\varphi, \ldots, (i-1)c(i)\varphi, \ldots)$. Since $c$ is monoton increasing function, $\varphi$ is an increasing sequence.

**Theorem 1.** The competitive ratio of $A$ is $1 + \varphi \approx 2.618$.

**Proof.** We will first prove that $A$ is $1 + \varphi$-competitive. Consider an arbitrary sequence of jobs $\sigma = j_1, \ldots, j_n$ and fix an optimal schedule. Let $m$ be the number of machines used by $A$, $j_\ell$ be the job which is completed at the latest time and $k$ be the number of machines that $A$ owns immediately after $j_\ell$ is released. Since $A$ always assigns a job to the machine with the lightest load $A(\sigma) \leq c(m) + \frac{P_\ell - p_\ell}{k}$. Consider the following cases.

Case A. $k < m$.

Suppose that $k < m$. Then by $P_\ell < kc(k+1)\varphi$ we have $P_\ell/k \leq c(k+1)\varphi \leq c(m)\varphi$. If the optimal algorithm uses at least $m$ machines, then its machine purchasing cost is at least $c(m)$, its makespan is at least $p_\ell$ thus

$$A(\sigma) \leq c(m) + \frac{P_\ell - p_\ell}{k} + p_\ell \leq (1 + \varphi)c(m) + p_\ell \leq (1 + \varphi)OPT(\sigma).$$

If the optimal algorithm uses less than $m$ machines then the optimal makespan is at least $c(m)\varphi$, since the total load is not smaller than $(m-1)c(m)\varphi$. Thus $c(m) \leq OPT(\sigma)/\varphi$. On the other hand $P_\ell/k \leq c(k+1)\varphi \leq c(m)\varphi \leq OPT(\sigma)$ and $p_\ell \leq OPT(\sigma)$, thus we obtain that $A(\sigma) \leq (2 + 1/\varphi)OPT(\sigma) = (1 + \varphi)OPT(\sigma)$.

Case B. $k = m$.

If $k = m$, then $A(\sigma) \leq c(m) + \frac{P_\ell - p_\ell}{m} + p_\ell$. If $OPT$ uses less than $m$ machines, then its makespan is at least $P_\ell/(m-1) \geq c(m)\varphi$. This yields that $c(m) \leq OPT(\sigma)/\varphi$. On the other hand $P_\ell/m \leq OPT(\sigma)$ and $p_\ell \leq OPT(\sigma)$, thus we obtain that $A(\sigma) \leq (2 + 1/\varphi)OPT(\sigma) = (1 + \varphi)OPT(\sigma)$. If the optimal algorithm uses $m$ machines, then its purchasing cost is $c(m)$, and its makespan is at least $\max(p_\ell, P_\ell/m)$. Therefore $A(\sigma) \leq 2OPT(\sigma)$ in this case. Finally, if the optimal algorithm uses more than $m$ machines then its machine purchasing cost is at least $c(m+1) \geq c(m)$ and its makespan is at least $p_\ell$. On the other hand by $P \leq mc(m+1)\varphi$ we have that $P_\ell/m \leq c(m+1)\varphi \leq \varphi OPT(\sigma)$. Therefore, we obtain that $A(\sigma) \leq (1 + \varphi)OPT(\sigma)$ and thus we have proven that $A$ is $1 + \varphi$-competitive.

We now show that $A$ is not $C$-competitive for any $C < 1 + \varphi$. Let $N$ be a large positive integer, $\varepsilon$ be a very small positive number, $K$ be a large positive number and let $c(m) = 0$ if $m < N$, $c(N) = K$, and $c(m) = \infty$ if $m > N$.

Consider the following sequence of jobs. The sequence is started by a long list of $\varepsilon$ size jobs, the total processing time of the jobs is $NK\varphi$. Then the sequence is ended with a job which has processing time $K\varphi$. Then $A$ purchases $N$ machines, its makespan is $2K\varphi$ and its total cost is $(2\varphi + 1)K$. An offline algorithm can purchase $N-1$ machines and use one of them for the large job, and the other machines for the small jobs, thus its makespan and its total cost is $\varphi N + K$. As $N$ tends to $\infty$ the ratio of the costs tends to $\frac{2\varphi + 1}{\varphi} = 1 + \varphi$, and this shows that the competitive ratio of $A$ is at least $1 + \varphi$. \qed
3.2. Small jobs case

In this part we investigate a special case of the problem. In the unit cost models the problem is also investigated under the assumption that the processing time of each job is at most 1. In [5] an optimal 4/3-competitive online algorithm is given for the special case of the original model, in [6] an optimal 2-competitive algorithm is given for the special case of the scheduling problem with machine cost and rejection. The extension of the bound on the job size can be defined as follows. We suppose that the size of each job is bounded by the minimal cost of the machines, which means that \( p_i \leq c(m) - c(m-1) \) holds for each \( i \geq 1 \) and \( m \geq 1 \) (we assume that \( c(0) = 0 \)).

**Theorem 2.** Algorithm \( B \) is 2-competitive in the small jobs case.

**Proof.** We will first prove that \( B \) is 2-competitive. Consider an arbitrary sequence of jobs \( \sigma = j_1, \ldots, j_n \) and fix an optimal schedule. Let \( m \) be the number of machines used by \( B \), \( j_\ell \) be the job which is completed at the latest time and \( k \) be the number of machines that \( B \) owns immediately after \( j_\ell \) is released. Since \( B \) always assigns a job to the machine with the lightest load \( B(\sigma) \leq c(m) + \frac{P_\ell - p_\ell}{k} + p_\ell \). Now we distinguish the following cases.

**Case A.** \( k < m \).

Suppose that \( k < m \). In this case \( P_\ell < kc(k+1) \), thus we have \( P_\ell/k \leq c(k+1) \leq c(m) \), and this yields

\[
B(\sigma) \leq c(m) + \frac{P_\ell - p_\ell}{k} + p_\ell \leq 2c(m) + p_\ell.
\]

If the optimal algorithm purchases at least \( m \) machines, then its purchasing cost is at least \( c(m) \). On the other hand the makespan is at least \( p_\ell \), thus \( B(\sigma) \leq 2OPT(\sigma) \) follows. If the optimal algorithm purchases less than \( m \) machines, then its makespan is at least \( c(m) \) since the total load is not smaller than \( (m-1)c(m) \). On the other hand, by the assumption on the size of the jobs, we obtain that the cost used by the optimal algorithm for purchasing the machines is at least \( p_\ell \), and this yields that \( B(\sigma) \leq 2OPT(\sigma) \).

**Case B.** \( k = m \).

If \( k = m \), then \( B(\sigma) \leq c(m) + \frac{P_\ell - p_\ell}{m} + p_\ell \). If the optimal algorithm uses more than \( m \) machines then its cost is at least \( c(m+1) \). On the other hand \( P_\ell/m \leq c(m+1) \) by the machine purchasing rule and \( c(m) + p_\ell \leq c(m+1) \) by the assumption on the size of the jobs, thus we obtain that \( B(\sigma) \leq 2OPT(\sigma) \). If the optimal algorithm uses \( m \) machines then its purchasing cost is \( c(m) \) and its makespan is at least \( P_\ell/m \), thus its cost is at least \( c(m) + P_\ell/m \). Furthermore \( p_\ell \leq OPT(\sigma) \), thus we obtain that \( B(\sigma) \leq 2OPT(\sigma) \). If the optimal algorithm uses less than \( m \) machines then its makespan is at least \( c(m) \) since the total load is not smaller than \( (m-1)c(m) \). Furthermore the makespan is at least \( P_\ell/(m-1) \geq P_\ell/m \). On the other hand by the assumption on the size of the jobs we obtain that the cost used by the optimal algorithm for purchasing the machines is at least \( p_\ell \), and this yields that \( B(\sigma) \leq 2OPT(\sigma) \).

Concerning algorithm \( B \) it is important to note that its competitive ratio is 3 for the general case. This statement can be proven in the similar way as Theorems 1 and 2.

3.3. Lower bounds

We can prove the following statement.

**Theorem 3.** No online algorithm can have smaller competitive ratio than 2. This lower bound is also true in the small jobs case.

**Proof.** Define a machine cost function as follows. Let

\[
c(m) = \begin{cases} 
1, & \text{if } m = 1, \\
M + m - 1, & \text{if } m \geq 2,
\end{cases}
\]

where \( M \) is a sufficiently large integer. Consider an arbitrary online algorithm and a sequence of jobs where each job has processing time 1: \( p_i = 1 \) for all \( i \). It is easy to see that any algorithm which never purchases a second machine is
not $C$-competitive for any $C$. So suppose that the algorithm purchases a second machine when job $j_\ell$ is released. The sequence of jobs is ended after that job. The cost of the algorithm is $M + 1 + \ell - 1 = M + \ell$.

First suppose that $\ell \leq M$. In this case the optimal offline algorithm purchases only one machine, its cost is $\ell + 1$, therefore the competitive ratio of the algorithm is at least $(M + \ell)/(\ell + 1) \geq 2 - 2/(M + 1)$ in this case.

Now suppose that $\ell > M$. In this case an offline algorithm can purchase $\lceil \sqrt{\ell} \rceil$ machines and produce a schedule which has not greater makespan than $\lceil \sqrt{\ell} \rceil$, therefore the optimal offline cost is at most $M - 1 + 2\lceil \sqrt{\ell} \rceil$. This yields that the competitive ratio is at least $(M + \ell)/(M - 1 + 2\lceil \sqrt{\ell} \rceil) = (1 + \ell/M)/(1 - 1/M + 2\lceil \sqrt{\ell} \rceil/M)$. As $M$ tends to $\infty$, this ratio tends to 2 or it becomes larger than 2 depending on the ratio $\ell/M$.

Since both lower bounds tend to 2 as $M$ tends to $\infty$ we have proven the required lower bound. □

By this lower bound and Theorem 2 we obtain the following result.

**Corollary 4.** Algorithm $B$ achieves the smallest possible competitive ratio in the small jobs case.

4. Related machines

4.1. Greedy algorithm

For the general problem we define the following **Greedy** algorithm. This algorithm uses the value $OPT_\ell$ which is the cost of the optimal offline solution of the input containing the first $\ell$ jobs. Since the offline problem is NP-hard, using $OPT_\ell$ makes the running time of the algorithm exponential. On the other hand it is possible to modify the algorithm to use some approximation of $OPT_\ell$. If we use a $c$-approximation algorithm instead of the optimal algorithm as a subroutine then the same proof shows that the algorithm becomes $4 + 2c$-competitive. The algorithm works as follows. When job $j_\ell$ is revealed Greedy purchases machines (if necessary) so that the current numbers of machines from set $S_1$ and $S_2$ satisfy $c_1(i_1) \leq OPT_\ell < c_1(i_1 + 1)$ and $c_2(i_2) \leq OPT_\ell < c_2(i_2 + 1)$. Greedy then schedules job $j_\ell$ by the LIST scheduling rule: it assigns the job to the machine where the load becomes minimal after the assignment of the job. If there are more machines with this property Greedy uses a faster one. The following statement is valid for the performance of this algorithm.

**Theorem 5.** The competitive ratio of Greedy is 6.

**Proof.** First we show that the algorithm is 6-competitive. Consider an arbitrary sequence of jobs $\sigma = j_1, \ldots, j_n$ and fix an optimal schedule. Let $m_1$ and $m_2$ be the numbers of machines used by Greedy from set $S_1$ and $S_2$ respectively. Let $j_\ell$ be the job which is completed at the latest time and $k_1$ and $k_2$ be the number of machines from set $S_1$ and $S_2$ that Greedy owns immediately after $j_\ell$ is released. Note that $c_1(m_1) \leq OPT(\sigma)$ and $c_2(m_2) \leq OPT(\sigma)$ follow by the rule which determines the number of the purchased machines. Now consider the makespan of the schedule produced by Greedy, denote this makespan by $C(\sigma)$. We now distinguish the following cases.

**Case 1**

Suppose that Greedy schedules $j_\ell$ on a machine from set $S_2$. Consider the jobs which are scheduled on the $k_2$ machines owned by Greedy from set $S_2$ immediately after $j_\ell$ is released in the time interval $[C(\sigma) - p_\ell/s - OPT_\ell, C(\sigma)]$. Denote the set which consists of these jobs and $j_\ell$ by $H$. Using the machine purchasing rule we obtain that $OPT_\ell < c_2(k_2 + 1)$, therefore in the optimal schedule of the first $\ell$ jobs, at most $k_2$ machines are purchased from set $S_2$. This yields that there exists such a job in $H$ which is scheduled on a machine from set $S_1$ in the optimal schedule of the first $\ell$ jobs. Let one such job be $j_x$, and its completion time is $c_x$. Then the inequalities $c_x \geq C(\sigma) - p_\ell/s - OPT_\ell$ and $p_x \leq OPT_x \leq OPT(\sigma)$ are valid.

Denote $x_1$ and $x_2$ the numbers of machines from set $S_1$ and $S_2$ that Greedy owns immediately after $j_x$ is released. It follows from the greedy scheduling rule that each of the $x_1$ machines from $S_1$ has at least $c_x - p_x$ load and each of the $x_2$ machines from $S_2$ has at least $c_x - p_x/s$ load after $j_x$ is released. Therefore $P_x \geq (c_x - p_x)(x_1 + sx_2)$. On the other hand $OPT_x < c_1(x_1 + 1)$ and $OPT_x < c_2(x_2 + 1)$, therefore in the optimal schedule of the first $x$ jobs at most $x_1$ and $x_2$ machines are purchased from the sets $S_1$ and $S_2$ respectively. This yields that $c_x - p_x \leq OPT_x \leq OPT(\sigma)$. Therefore, by $c_x \geq C(\sigma) - p_\ell/s - OPT_\ell$ we obtain that

$$C(\sigma) \leq c_x + p_\ell/s + OPT_\ell \leq OPT(\sigma) + p_x + p_\ell/s + OPT_\ell \leq 4OPT(\sigma).$$
Case 2

Suppose that \textit{Greedy} schedules \( j_\ell \) on a machine from set \( S_1 \). By the greedy scheduling rule it follows that when \( j_\ell \) is released each of the \( k_2 \) machines of \textit{Greedy} from set \( S_2 \) had at least \( C(\sigma) - p_\ell / s \) load. Then consider the jobs which are scheduled on the \( k_2 \) machines owned by \textit{Greedy} immediately after \( j_\ell \) is released from \( S_2 \) in the time interval \([C(\sigma) - p_\ell / s - OPT_\ell, C(\sigma)]\). Denote the set which consists of these jobs and \( j_\ell \) by \( H \). Then \( H \) contains a job which is scheduled on a machine from set \( S_1 \) in the optimal schedule of the first \( \ell \) jobs. Using this observation we can prove that \( C(\sigma) \leq 4OPT(\sigma) \) in the same way as in Case 1.

We have proven that \( C(\sigma) \leq 4OPT(\sigma) \), thus we obtain that \( \text{Greedy}(\sigma) = c_1(m_1) + c_2(m_2) + C(\sigma) \leq 6OPT(\sigma) \) which proves that the algorithm is 6-competitive.

Now we prove that this bound is tight, the algorithm is not \( C \)-competitive for any \( c < 6 \). Let \( m > 4, s > 2(m - 1)(m - 2) \), and let \( n \) be large enough to satisfy the property \( \frac{n + ms}{n - ms} \leq \frac{m - 1}{m - 1} \). Let \( \varepsilon \) be a very small positive number and \( K \) be a larger positive number. Let \( c_1(k) = 0 \) if \( k \leq n \) and \( c_1(k) = \frac{m}{m - 1} K + \varepsilon \) if \( k > n \). Let \( c_2(k) = 0 \), if \( k \leq m \) and \( c_2(k) = \frac{m}{m - 1} K + \varepsilon \) if \( k > m \).

Consider the following sequence of jobs. The sequence is started by a long list of small \( \varepsilon \) size jobs, that list has the total processing time \( K(n + ms) \). Then the next \( ms \) jobs have processing time \( K \). Then \((m - 1)(m - 2)m\) jobs follow each of them has processing time \( Ks/(m - 1)(m - 2) \). Next, one large job with processing time \( Ks \) follows and the sequence is ended with one job of size \( \frac{m}{m - 1} K + \varepsilon \). \textit{Greedy} starts with \( n \) machines from set \( S_1 \) and \( m \) machines from set \( S_2 \). The first set of jobs is scheduled by \textit{Greedy} uniformly. The load of each machine is \( K \) after this set. The second part of the list can be finished on the \( m \) machines of set \( S_2 \) with completion time \( 2K \), which would be the completion time on the machines from set \( S_1 \). Therefore \textit{Greedy} schedules them uniformly on the machines of \( S_2 \), and after this set the load is \( 2K \) on these machines. \textit{Greedy} also schedules the third part uniformly on the \( m \) machines from set \( S_2 \). Then, the load of each machines becomes \( 3K \) (by the assumption \( s > 2(m - 1)(m - 2) \)) scheduling some of the jobs on the machines of \( S_1 \) would give larger completion time there). Next, the job of size \( Ks \) is scheduled on one of the machines from set \( S_2 \) and the load of this machine becomes \( 4K \). An offline algorithm can schedule the first two parts on the machines from set \( S_1 \), the third and fourth parts on the machines of \( S_2 \). In the slowest group \( ms \) machines are used for the large \( K \) size jobs, the remaining \( n - ms \) jobs can schedule the \( \varepsilon \) size jobs with maximal completion time \( \frac{n + ms}{n - ms} K \leq \frac{m}{m - 1} K \). In the faster group one machine is used for the job of size \( Ks \) and the others schedule the third class uniformly and the maximal completion time is \( \frac{m}{m - 1} K \). Therefore the optimal cost is not more than \( \frac{m}{m - 1} K \).

Thus \textit{Greedy} does not purchase new machine for the first four parts of the list. On the other hand after, the last job the optimal makespan is at least \( \frac{m}{m - 1} K + \varepsilon \), thus \textit{Greedy} purchases new machines from both sets and schedules the last job on the faster new machine. Therefore the cost of \textit{Greedy} on this input is \( 4K + 2(\frac{m}{m - 1} K + \varepsilon) \). On the other hand the optimal offline algorithm uses \( n \) machines from set \( S_1 \), \( m \) machines from set \( S_2 \) and it schedules the first two parts on the slower machines, the last three parts on faster machines, and its makespan (and the total cost) is \( \frac{m}{m - 2} K \). The ratio of the two costs tends to \( 6 \) as \( m \) grows, therefore we have proven that the competitive ratio of \textit{Greedy} is \( 6 \). \( \square \)

4.2. Fixed number of machines

In this part we consider the online makespan minimization problem where \( n \) machines with speed 1 and \( m \) machines with speed \( s \) are given. We obtain this model from the general problem with \( c_1(k) = 0 \) if \( k \leq n \), \( c_1(k) = \infty \) if \( k > n \), \( c_2(k) = 0 \) if \( k \leq m \) and \( c_2(k) = 0 \) if \( k > m \). This problem is also the special case of the online scheduling problem on related machines (see [1,2]). For the online scheduling problem on related machines the \textit{Greedy} algorithm which schedules each job on the machine where it can be completed first is investigated in [1] and [3], it has been proven that the competitive ratio of the algorithm is \( \Theta(\log m) \), where \( m \) denotes the number of machines. By the proof of Theorem 5 we obtain easily the following result.

**Corollary 6.** The competitive ratio of \textit{Greedy} is 4 when the numbers of machines are fixed.

Now we present a more sophisticated algorithm. The algorithm uses similar ideas as the algorithms for the similar unrelated machines problem [11]. It is greedy in the sense that it tries to use the set where the total speed of the machines is larger, but it handles the large jobs carefully. We call this algorithm \textit{Modified Greedy} (\textit{MG} in short). The algorithm is defined as follows.
Algorithm MG

• If $ms \geq n$ then schedule each job on the machines of set $S_2$ using the LIST scheduling algorithm.
• If $ms < n$ then use the following algorithm
  - 1. Initialization. Let $R := \emptyset$.
  - 2. When job $j$ arrives, let $r = \max\{P(R \cup \{j\})/(ms), M(R \cup \{j\})/s\}$ (this is a lower bound on the makespan of the optimal schedule of the jobs in $R \cup \{j\}$ on the machines of set $S_2$). If $p_j \geq r$, then
    * (a) Assign $j$ to $S_2$,
    * (b) Set $R = R \cup \{j\}$.
  - 3. Otherwise, assign job $j$ to $S_1$
  - 4. Schedule the job by LIST on the machines of the set where it is assigned to.

The competitive ratio of the algorithm is determined by the following theorem.

**Theorem 7.** The competitive ratio of MG is 3.

**Proof.** First we show that the algorithm is 3-competitive. Consider an arbitrary sequence of jobs $\sigma$ and fix an optimal schedule. First investigate the case when $ms \geq n$. Then MG schedules each job on the machines of set $S_2$. Since it uses the List algorithm for scheduling the jobs thus its makespan is not more than $P(\sigma)/(ms) + M(\sigma)/s$. On the other hand $M(\sigma)/s \leq OPT(\sigma)$ and by $ms \geq n$ it follows that $P(\sigma)/(ms) \leq 2OPT(\sigma)$, thus we obtain that $MG(\sigma) \leq 3OPT(\sigma)$.

Now suppose that $ms < n$. Let $X$ and $Y$ be the sets of jobs scheduled by MG on the machines of $S_1$ and $S_2$ respectively. Now distinguish the following cases.

**Case 1**

Suppose that the makespan is achieved on a machine of $S_2$. Then $MG(\sigma) \leq P(Y)/(ms) + M(Y)/s$. Let $Y_1$ and $Y_2$ be the sets of the jobs from set $Y$ which are scheduled by the optimal solution on the sets $S_1$ and $S_2$ respectively. Let $j_k$ be the last job from $Y_1$. Then $p_k \leq OPT(\sigma)$. On the other hand at the time when it was assigned to set $S_2$ by MG, we had $p_k \geq r \geq P(Y_1)/(ms)$. Thus we obtained that $OPT(\sigma) \geq P(Y_1)/(ms)$. Furthermore, it follows by the definitions that $OPT(\sigma) \geq P(Y)/(ms)$ and $OPT(\sigma) \geq M(Y)/s$. Therefore we obtain that $MG(\sigma) \leq 3OPT(\sigma)$.

**Case 2**

Suppose that the makespan is achieved on a machine of $S_1$. Let $j_k$ be the job which achieves the makespan. Then $MG(\sigma) \leq P(X \setminus \{k\})/n + p_k$. First let us observe that by the assumption $n > ms$ it follows that $P(X \setminus \{k\})/n \leq 2OPT(\sigma)$. If the optimal solution schedules $p_k$ on the machines of $S_1$ then $p_k \leq OPT(\sigma)$ and $MG(\sigma) \leq 3OPT(\sigma)$ immediately follows. Now suppose that the optimal solution schedules $p_k$ on a machine from $S_2$. Then at the time when $p_k$ was assigned to set $S_1$ by MG we had $p_k \leq \max\{P(R \cup \{k\})/(ms), M(R \cup \{k\})/s\}$. If $p_k \leq M(R \cup \{k\})/s$ then $p_k \leq OPT(\sigma)$ and we obtain that $MG(\sigma) \leq 3OPT(\sigma)$. Thus we can suppose that $p_k \leq P(R \cup \{k\})/(ms) \leq P(Y \cup \{k\})/(ms)$. On the other hand using the assumption that $p_k$ is scheduled in the optimal solution on the machines of $S_2$ it can be proven that $P(Y \cup \{k\})/(ms) \leq 2OPT(\sigma)$ in the same way as we have proven the similar statement in Case 1. Therefore we have obtained that $MG(\sigma) \leq P(X \setminus \{k\})/n + p_k \leq P(X \setminus \{k\})/n + P(Y \cup \{k\})/(ms)$ and it has been shown that $P(X \setminus \{k\})/n \leq 2OPT(\sigma)$ and $P(Y \cup \{k\})/(ms) \leq 2OPT(\sigma)$. On the other hand $P(\sigma) \leq OPT(\sigma)(n + ms)$ thus we obtain that $\min\{P(X \setminus \{k\})/n, P(Y \cup \{k\})/(ms)\} \leq OPT(\sigma)$. Using this bound it follows that $P(X \setminus \{k\})/n + P(Y \cup \{k\})/(ms) \leq 3OPT(\sigma)$, which proves that $MG(\sigma) \leq 3OPT(\sigma)$. Since we have investigated all of the possible cases we have proven that MG is 3-competitive.

Now we show that the competitive ratio of the algorithm is at least 3. Let $n = ms$ and consider the following list of jobs. It is started with a long list of small $\epsilon$ size jobs, the total processing time of the jobs is $2Kn$ and it is ended with one large job which has processing time $K$. Then MG schedules each job on the machines of $S_2$, its makespan is $3K$. An offline algorithm can schedule $nK$ amount of jobs on the slow machines, the remaining small jobs on $m - 1$ faster machines, and the last job on the last faster machine. Therefore its makespan is $\frac{m}{m-1}K$, and this shows that the competitive ratio of MG cannot be smaller than 3. \[ \Box \]

4.3. Lower bound

**Theorem 8.** No online algorithm can have smaller competitive ratio for the general problem than $1 + c \approx 2.325$ where $c$ is the solution of the equation $x^3 - x - 1 = 0$. 

Proof. Let $c_1(1) = 0$ and $c_1(m) = K$ if $m \geq 2$, let $c_2(m) = (1 + c)K$ for each $m$ and let $s > K^2$. Suppose that we have an algorithm which has smaller competitive ratio than $1 + c$. Consider a sequence which is started by jobs of size $\varepsilon$. If an algorithm never buys a second machine it is not constant competitive. Suppose that the algorithm purchases a machine from set $S_2$ after some small jobs, then its cost is at least $(1 + c)K$. On the other hand using the slower machines, the purchasing cost is $K$ and the makespan is $\varepsilon$, thus the competitive ratio of the algorithm is at least $1 + c$. Therefore, we can suppose that after some small jobs the algorithm purchases the second slow machine. Let $L$ denote the total amount of small jobs which the algorithm has when it buys the second machine. If $L < K/c$, then the optimal offline algorithm uses only one machine with makespan $L + \varepsilon$ and the cost of the online algorithm is $K + L > (1 + c)L$ which shows that the competitive ratio cannot be smaller than $1 + c$ in this case. Therefore, we can suppose that $L \geq K/c$ and it follows that the makespan of the online algorithm is at least $K/c$. Then the last job arrives which has size $s$. If the algorithm does not get a faster machine for this job, its makespan becomes $K^2$ and its competitive ratio tends to $\infty$ as $K$ grows. Therefore the online algorithm must buy the first fast machine. This yields that its cost for purchasing the machines is $(1 + c)K$, its makespan is at least $K/c$. The optimal algorithm uses only fast machines and its total cost is $cK + 1$. As $K$ tends to $\infty$ the lower bound on the ratio of the two costs tends to $(c + 1 + 1/c)/c = 1 + c$, and this proves the theorem. □

References