Semi-online scheduling with decreasing job sizes

Steve Seiden* Jiří Sgall† Gerhard Woeginger‡

October 27, 1998

Abstract

We investigate the problem of semi-online scheduling jobs on m identical parallel machines where the jobs arrive in order of decreasing sizes. We present a complete solution for the preemptive variant of semi-online scheduling with decreasing job sizes. We give matching lower and upper bounds on the competitive ratio for any fixed number m of machines; these bounds tend to $\frac{1}{2}(1+\sqrt{3}) \approx 1.36603$, as the number of machines goes to infinity. Our results are also best possible for randomized algorithms. For the non-preemptive variant of semi-online scheduling with decreasing job sizes, a result of Graham [SIAM J. Appl. Math. 17(1969), 263–269] yields a $(\frac{4}{3}-\frac{1}{3m})$ competitive deterministic non-preemptive algorithm. For $m=2$ machines, we prove that this algorithm is best possible (it is $\frac{7}{6}$ competitive). For $m=3$ machines we give a lower

*sseiden@acm.org, Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany. This research was done at TU Graz, supported by the START program Y43-MAT of the Austrian Ministry of Science.

†sgall@math.cas.cz, Mathematical Inst., AS CR, Žitná 25, CZ-11567 Praha 1, Czech Republic and Dept. of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Praha. Partially supported by grant A1019602 of GA AV ČR, postdoctoral grant 201/97/P038 of GA ČR, and cooperative research grant INT-9600919/ME-103 from the NSF and MŠMT CR.

‡gwoegi@opt.math.tu-graz.ac.at, Inst. für Mathematik, TU Graz, Steyrergasse 30, A-8010 Graz, Austria. This research has been supported by the START program Y43-MAT of the Austrian Ministry of Science.

bound of $(1+\sqrt{37})/6 \approx 1.18046$. Finally, we present a randomized non-preemptive $\frac{8}{7}$-competitive algorithm for $m=2$ machines and prove that this is optimal.

Keywords: Analysis of algorithms, online algorithms, competitive ratio, scheduling.

1 Introduction

Consider the following scheduling problem: We are confronted with a sequence of jobs with processing times $p_1, p_2, \ldots, p_n$ that must be assigned to m machines. The *load* of a machine is the sum of the processing times of the jobs assigned to that machine. Our goal is to minimize the *makespan*, i.e., the maximum machine load. If the problem is *online*, then each job must be assigned without knowledge of successive jobs. If the problem is *semi-online with decreasing job sizes*, then we know that $p_i \geq p_{i+1}$ for all $i \geq 1$. Hence, in this variant we do have some partial knowledge on the future jobs, which makes the problem easier to solve than standard online scheduling problems.

The quality of an online or semi-online algorithm A is measured by its *competitive ratio*, defined as the smallest number c such that for every list of jobs L, $A(L) \leq c \cdot \mathrm{opt}(L)$, where $A(L)$ denotes the makespan of a schedule produced by algorithm A for scheduling the list L of jobs, and $\mathrm{opt}(L)$ denotes the corresponding makespan of some optimal schedule.

We study semi-online scheduling with decreasing job sizes in the three main variants: preemptive, non-preemptive deterministic, and non-preemptive randomized scheduling. Note that for $m=1$ the trivial deterministic non-preemptive algorithm that schedules all jobs on the single machine without introducing any idle time is optimal. Thus we assume $m \geq 2$ for the rest of the paper.

(1) In the preemptive variant, a job may be preempted, i.e., split into pieces that may be spread over several machines and/or assigned to non-consecutive time slots. However pieces of the same job must not overlap in time, even if scheduled on different machines. In other words, if a job is assigned to time slots $(s_1, t_1]$, $(s_2, t_2]$, …, $(s_k, t_k]$ on one or several machines, then any pair of these time slots must

be disjoint. For the preemptive variant, we give matching lower and upper bounds on the competitive ratio for any fixed number m of machines; these bounds increase with m and tend to $\frac{1}{2}(1+\sqrt{3}) \approx 1.36603$, as the number of machines goes to infinity. Our results are also best possible for randomized algorithms.

(2) For the non-preemptive deterministic variant, we give a lower bound of $\frac{7}{6}$ for $m = 2$ and a lower bound of $\frac{1}{6}(1+\sqrt{37})$ for $m = 3$. Graham [11] has shown that for decreasing job sizes, the greedy algorithm LIST is $(\frac{4}{3} - \frac{1}{3m})$-competitive. Hence, for $m = 2$ machines we have matching bounds. For $m \geq 3$, the problem remains open. Quite surprisingly, the non-preemptive variant allows better competitive ratios than the preemptive variant does. In fact even LIST achieves a better competitive ratio (than possible with preemption) for every $m \geq 2$ as well as in the limit. In the light of the corresponding results for on-line scheduling (see below) this was certainly unexpected to us, and to our knowledge, it is also the first occurrence of this phenomenon.

(3) Finally for the non-preemptive randomized variant, for $m = 2$ machines, we present a randomized $\frac{8}{7}$-competitive algorithm and prove that it is optimal. The cases $m \geq 3$ remain open.

Related work. Other results on semi-online problems are obtained by Azar & Regev [2] who consider semi-online scheduling where the optimum makespan is known in advance, and by Kellerer, Kotov, Speranza & Tuza [13] who consider several semi-online versions of the partition problem, which corresponds to scheduling on 2 machines. Liu, Sidney & van Vliet [14] investigate the problem where a schedule must be created with knowledge only of the number and relative sizes of jobs.

In the standard online problem (where the job sizes are not necessarily decreasing) all the three main variants, preemptive, non-preemptive deterministic, and non-preemptive randomized, have been investigated in the literature. The preemptive variant was solved completely by Chen, van Vliet & Woeginger [7] who designed an online algorithm for preemptive scheduling on m machines with competitive ratio

$$\beta(m) \;=\; \frac{m^m}{m^m-(m-1)^m} \quad\to\quad \frac{e}{e-1} \approx 1.58198. \qquad (1)$$

They also gave a matching lower bound construction which even holds for randomized online algorithms. It is a general phenomenon that for preemptive online scheduling problems randomization does not help, and as we noted above, the same is true also for the semi-online variant studied in this paper.

For the non-preemptive deterministic variant, Graham [10] defined a simple greedy algorithm LIST which places each job on the machine which is currently least loaded. He showed that LIST is $(2-\frac{1}{m})$-competitive on m machines; this analysis is tight. Since Graham's seminal work, many researchers [3, 4, 5, 6, 8, 9, 12, 16, 17, 18]. have investigated this problem. The best possible competitive ratio for a large number of machines is now known to lie in the interval $[1.852, 1.923]$; cf. Albers [1].

Only little is known about non-preemptive randomized online scheduling. For $m = 2$ machines, Bartal, Fiat, Karloff & Vohra [3] give a $\frac{4}{3}$-competitive randomized online algorithm. Chen, van Vliet & Woeginger [5] and independently Sgall [19] show an online lower bound of $\beta(m)$ for any $m \geq 1$. Note that this bound exactly matches the bound of Bartal *et al.* for $m = 2$. Seiden [16, 17] presents randomized online algorithms which outperform their deterministic counterparts for $m = 3, \ldots, 7$.

2 Preemptive scheduling

Throughout this section, we will assume without loss of generality that the first (and hence largest) job has size 1. First, to develop the intuition behind the optimal algorithm, we describe its action on the sequence of jobs of size 1; this is also the sequence that will be used in the lower bound argument. Suppose we want to achieve competitive ratio c. We schedule the first m jobs during the time slots before time c. Then the $(m+i)$-th job is scheduled so that first ci/m of it is scheduled after time c and the rest before time c; thus the largest loads of machines will be $c(1+i/m)$, $c(1+(i-1)/m)$, $\ldots$,

$c(1+1/m)$, and the remaining loads are at most c. For $k = \lfloor m/c \rfloor$, after the $(m+k)$-th job, the new jobs are scheduled only during the slots after c in the similar manner, on at most $k+1$ machines. We need to choose the right value of k and the corresponding value of c so that this scheme works.

For $0 \le k \le m$, we define values

$$\begin{aligned} \gamma_{m,k} &= \frac{2m^2 + 2mk}{2m^2 + k^2 + k}, \\ \gamma_m &= \max_{k=0,\ldots,m} \gamma_{m,k}. \end{aligned}$$

For any $m \ge 2$, let $k(m)$ be the largest integer satisfying $\gamma_m = \gamma_{m,k(m)}$.

Lemma 2.1 *For every $m \ge 2$ the following conditions hold:*

(i) $1 \le \gamma_m \le (1+\sqrt{3})/2 \approx 1.36603$. Moreover, as $m \to \infty$, $\gamma_m \to (1+\sqrt{3})/2$.

(ii) $1 \le k(m) = \lfloor m/\gamma_m \rfloor < m$.

(iii) $\gamma_m < \gamma_{m+1}$.

Proof. (i) Obviously $1 \le \gamma_{m,k} \le (2+x)/(2+x^2)$, where $x = k/m$. This rational function is maximized for $x = \sqrt{3}-1$ and has maximal value $(1+\sqrt{3})/2$. Moreover, for m large and $k = \lfloor m(\sqrt{3}-1) \rfloor$, the value of $\gamma_{m,k}$ converges to $\frac{1}{2}(1+\sqrt{3})$.

(ii) If $k+1 \le m/\gamma_{m,k}$ then

$$\begin{aligned} \gamma_{m,k+1} &= \frac{(2m^2 + 2mk) + 2m}{(2m^2 + k^2 + k) + 2(k+1)} \\ &\ge \min\left\{ \frac{2m^2 + 2mk}{2m^2 + k^2 + k}, \frac{m}{k+1} \right\} = \gamma_{m,k}, \end{aligned}$$

and thus $k \ne k(m)$. Similarly we can verify that, if $k > m/\gamma_{m,k}$, then $\gamma_{m,k-1} > \gamma_{m,k}$ and $k \ne k(m)$. This implies that $k(m) = \lfloor m/\gamma_m \rfloor$. We have $\gamma_{m,0} = 1 \le \gamma_{m,1}$ and thus $k(m) \ge 1$. For $m \ge 2$ we have $\gamma_{m,m} > 1$, thus $k(m) = \lfloor m/\gamma_m \rfloor \le \lfloor m/\gamma_{m,m} \rfloor < m$.

(iii) Let $k = k(m)$, and hence $\gamma_m = \gamma_{m,k}$. We distinguish two cases.

First, assume $k \geq (\sqrt{3}-1)m$. In this case we prove that $\gamma_{m,k} < \gamma_{m+1,k}$, which implies $\gamma_m > \gamma_{m+1}$. Plugging in the definitions of $\gamma_{m,k}$ and $\gamma_{m+1,k}$ and simplifying shows that the $\gamma_{m,k} < \gamma_{m+1,k}$ is equivalent to $2m^2 - 2mk < (k+1)^2$. However, the assumption implies that $2m^2 - 2mk \leq k^2 < (k+1)^2$, and we are done.

Second, assume $k < (\sqrt{3}-1)m$. We prove that $\gamma_{m,k} < \gamma_{m+1,k+1}$, which also implies $\gamma_m < \gamma_{m+1}$. Plugging in the definitions of $\gamma_{m,k}$ and $\gamma_{m+1,k+1}$ and simplifying shows that $\gamma_{m,k} < \gamma_{m+1,k+1}$ is equivalent to

$$(k-m)(k^2 + 2km - 2m^2) + k(3k - m + 2) > 0.$$

Our assumption implies that the first term is positive. From (ii) and (i) we obtain that $k = \lfloor m/\gamma_m \rfloor \geq \lfloor m/2 \rfloor \geq m/2 - 1$. This implies that the second term is positive as well, and the proof is complete. ■

Theorem 2.2 *For every $m \geq 2$, there exists a γ_m competitive deterministic algorithm for semi-online preemptive scheduling on m machines with decreasing job sizes.*

Proof. Let $c = \gamma_m$, and let $k = k(m)$ be the largest integer for which $\gamma_m = \gamma_{m,k}$. Consider the following algorithm: Suppose that at the current moment, jobs of total time T are already scheduled and that a job of size t arrives. We schedule it as follows, without introducing any idle time between consecutive job pieces.

1. If $T + t \leq m$, we schedule the job so that it is scheduled before time c, on the first machine available. More precisely, if the first machine with the current load less than c has load at most $c - t$, we schedule the whole job on it. If not, we schedule on it a portion of the job, so that the load becomes c, and the rest on the next machine.

2. If $T \geq m + i - 1$ and $T + t \leq m + i$ for some integer i, $1 \leq i \leq k$, schedule on each of the first i machines tc/m of the job, and schedule the remainder of the job before time c, as in Step 1.

3. If $T \leq m + i$ and $T + t > m + i$ for some integer i, $0 \leq i \leq k$, schedule on each of the first i machines tc/m of the job, on machine $i + 1$, $(T + t - m - i)c/m$ of the job (or the whole

remainder, if it is smaller—this can happen only if $i = k$), and schedule the remainder of the job before time c, as in Step 1.

4. If $T \geq m + k$, schedule on each of the first k machines tc/m of the job, and the remainder on the machine $k + 1$.

At the moment where the algorithm has assigned a total load of $m + i$, $i = 0, \ldots, k$, the machines $1, 2, \ldots, i + 1$ have loads exactly $c(1 + i/m), c(1 + (i - 1)/m), \ldots, c$, respectively. This follows from adding the loads of all contributions. The only non-trivial part is to verify that the load of machine $i + 1$ is (not less than) c. For $i < k$, this is true already after total size m (from the fact that $k = \lfloor m/c \rfloor$). For $i = k$, it follows from the calculation in the next paragraph.

Before time $m + k$, the total size of portions of jobs scheduled before time c is

$$m + k - \frac{k(k + 1)c}{2m} = cm,$$

using the definition of k and $c = \gamma_k$. This implies the claim in the previous paragraph. Also, this implies that it is actually possible to fit all these portions before time c. They are scheduled so that no two parts of the same job run at the same time on different machines, since $c \geq 1$.

After total load m, the portion of a job scheduled on a single machine after time c is at most c/m. Since the loads of any two of these machines differ by at least c/m at all times, these pieces do not overlap. For a job scheduled at a total load $T < m + i$ but finished after $m + i$, it is also necessary to argue that its portions assigned to machines i and $i + 1$ are not in parallel: machine i starts at load $c(1 + (T + 1 - m - i)/m)$, while the machine $i + 1$ starts at c and finishes at $c(1 + (T + t - m - i)/m)$, which is less since $t \leq 1$. The last fact we need to verify is that the remainders of jobs scheduled on machine $k + 1$ in Step 4 are not scheduled in parallel with other portions of the same job. The remainder of a job of size t is at most $t - ktc/m < tc/m$, because $k = \lfloor m/c \rfloor$. Thus the loads of machines k and $k + 1$ differ by at least c/m after time $m + k$.

The competitive ratio of the algorithm is implied by the claims about the loads of the algorithm. ∎

We designed the algorithm so that by definition it does not introduce any idle time between two consecutive job pieces on the same machine. This is not a random coincidence, by Observation 4.1 in [7], which applies in our case as well, we may always modify any preemptive algorithm so that it never introduces idle times.

Theorem 2.3 *No (deterministic or randomized) algorithm for semi-online preemptive scheduling on $m \geq 2$ machines with decreasing job sizes can have a competitive ratio better than γ_m.*

Proof. Fix $k \leq m$ and a randomized algorithm. Suppose that the algorithm is c-competitive. We will prove that on a sequence of $m+k$ unit jobs the competitive ratio is at least $\gamma_{m,k}$. This implies $c \geq \gamma_m$.

Let the randomized algorithm run on a sequence of $m + k$ unit jobs. Let $\mathbf{X}_i$ be the last time when at least i jobs are running (after scheduling all $m + k$ jobs); note that $\mathbf{X}_i$ is a random variable, since the algorithm is randomized. We claim that for $0 \leq i \leq k$,

$$\mathrm{E}[\mathbf{X}_{k+1-i}] \;\leq\; \frac{c(m+i)}{m}. \tag{2}$$

As long as $k + 1 - i$ jobs are running, at least one of the first $m + i$ jobs is running. Therefore, the average makespan of the algorithm on the first $m + i$ jobs is at least $\mathrm{E}[\mathbf{X}_{k+1-i}]$. Since the optimal schedule for these $m + i$ jobs has makespan $(m + i)/m$, inequality (2) follows from the c-competitiveness of the algorithm.

The total size of jobs scheduled by the algorithm is, for every random choice, at most

$$\sum_{i=1}^{m} \mathbf{X}_i \;\leq\; \sum_{i=1}^{k} \mathbf{X}_i + (m-k)\mathbf{X}_{k+1}.$$

This size has to be at least $m + k$, and by using (2) we obtain

$$m + k \;\leq\; \sum_{i=1}^{k} \mathbf{X}_i + (m-k)\mathbf{X}_{k+1} \;\leq\; c\left(m + \frac{k(k+1)}{2m}\right). \tag{3}$$

This is equivalent to $c \geq \gamma_{m,k}$. The lower bound follows. ■

Corollary 2.4 *There exists a deterministic algorithm for semi-online preemptive scheduling with decreasing job sizes whose competitive ratio is* $\frac{1}{2}(1+\sqrt{3}) \approx 1.36603$. *No randomized algorithm can achieve a better competitive ratio.* ■

3 Deterministic scheduling

Theorem 3.1 *Any deterministic algorithm for semi-online non-preemptive scheduling with decreasing job sizes has competitive ratio at least* $7/6$ *for* $m = 2$, *and competitive ratio at least* $\frac{1}{6}(1+\sqrt{37})$ *for* $m = 3$. *Moreover, for* $m = 2$ *there exists a matching* $7/6$*-competitive deterministic algorithm (due to Graham [11]).*

Proof. The case $m = 2$ follows as in Graham [11]: Consider an instance with 2 jobs of size 3 followed by 3 jobs of size 2. If a deterministic algorithm puts the 2 jobs of size 3 on the same machine, no other jobs may arrive; hence, its competitive ratio is at least 2. If it puts the 2 jobs of size 3 on different machines, it is $\frac{7}{6}$-competitive on Graham's sequence. Graham [11] shows that the competitive ratio of List is $\frac{4}{3} - \frac{1}{3m}$, this gives a tight upper bound of $7/6$ for $m = 2$.

Next, consider the case $m = 3$. We show a lower bound of

$$c \;=\; \frac{1}{6}(1+\sqrt{37}) \;\approx\; 1.18046.$$

Let $x = (7-3c)/6 \approx 0.57643$. Consider the following job list: x, x, $1-x$, $1-x$, $\frac{1}{3}$, $\frac{1}{3}$ and $\frac{1}{3}$. The first three jobs must go on three pairwise distinct machines (otherwise the competitive ratio is at least $1/x = 6/(7-3c) > 1.73479 > c$). Consider the fourth job. If it is placed on the same machine as one of the first two jobs, then the algorithm's makespan is 1, whereas the optimal offline makespan is $2(1-x)$; note that $1/(2-2x) = 3/(3c-1) = c$. If the fourth job is placed on the same machine as the third job, then the algorithm's final makespan is at least $7/3-2x = c$. The optimal makespan for the entire list is 1. Summarizing, any online algorithm has competitive ratio at least c. ■

4 Randomized scheduling

In this section, we discuss randomized semi-online non-preemptive scheduling with decreasing job sizes on $m = 2$ machines. We give matching lower and upper bounds of $8/7$ for this problem.

Theorem 4.1 *No randomized algorithm for semi-online non-preemptive scheduling with decreasing job sizes on $m = 2$ machines can have a competitive ratio better than $8/7$.*

Proof. We consider a problem instance with 2 jobs of size 3 followed by 3 jobs of size 2, as in Graham [11]. If the algorithm places the first two jobs on the same machine with probability more than $\frac{1}{7}$, then the competitive ratio is at least $(\frac{1}{7} \cdot 6 + \frac{6}{7} \cdot 3)/3 = \frac{8}{7}$ and we are done. If not, the adversary gives the algorithm the remaining three jobs. The optimal solution places the 2 size 3 jobs together and the 3 size 2 jobs together, yielding a makespan of 6. The online algorithm can do this with probability at most $\frac{1}{7}$. If the online algorithm has placed the 2 size 3 jobs on separate machines, the best makespan it can achieve is 7. Therefore, the competitive ratio is at least $(\frac{1}{7} \cdot 6 + \frac{6}{7} \cdot 7)/6 = \frac{8}{7}$. ■

Theorem 4.2 *There exists a randomized $8/7$ competitive algorithm for semi-online non-preemptive scheduling with decreasing job sizes on $m = 2$ machines.*

Proof. We call our machines A and B. The algorithm places the first two jobs on machine A with probability $\frac{1}{7}$. With probability $\frac{6}{7}$, it places the first on A and the second on B. Thereafter it schedules each job on the machine with least load, i.e., it behaves as LIST [11].

Our randomized algorithm is a distribution over two deterministic algorithms. We analyze it by considering it to be a single deterministic algorithm which creates two schedules. We show that the weighted average of the makespans of the two schedules is at most $\frac{8}{7}$ times the makespan of the optimal offline schedule. In the first schedule, the first two jobs are assigned to machine A, while in the second, they are assigned to machines A and B, respectively. The weights for the schedules are $\frac{1}{7}$ and $\frac{6}{7}$, respectively.

For the purposes of analysis we assume that there are $n \geq 6$ jobs. This assumption is valid since any shorter job list may be extended with size 0 jobs. Define $P_i = \sum_{j=1}^{i} p_j$. Let a_j^i and b_j^i be the loads on machines A and B in schedule i after the first j jobs have be scheduled, respectively. Let $x_i^j = \max\{a_i^j, b_i^j\}$. Let X_i^j be the machine to which job i is assigned in schedule j. Let opt_i be the optimal offline makespan for the first i jobs. We show that

$$\frac{1}{7}x_n^1 + \frac{6}{7}x_n^2 - \frac{8}{7}\mathrm{opt}_n \tag{4}$$

is at most 0.

By definition of the algorithm we have

$$\begin{array}{lllllllll} a_1^1 & = & p_1, & b_1^1 & = & 0, & x_1^1 & = & a_1^1, \\ a_1^2 & = & p_1 + p_2, & b_1^2 & = & 0, & & & \\ a_2^1 & = & p_1, & b_2^1 & = & 0. & & & \end{array} \tag{5}$$

For $3 \leq i \leq 6$ and $j = 1$ and for $2 \leq i \leq 6$ and $j = 2$ we have the restrictions

$$b_i^j = b_{i-1}^j + p_i, \;\; a_i^j = a_{i-1}^j, \;\; x_{i-1}^j = a_{i-1}^j \geq b_{i-1}^j \quad \text{if } X_i^j = A;$$

$$a_i^j = a_{i-1}^j + p_i, \;\; b_i^j = b_{i-1}^j, \;\; x_{i-1}^j = b_{i-1}^j \geq a_{i-1}^j \quad \text{otherwise.} \tag{6}$$

We assume without loss of generality that $p_1 = 1$. We have the constraints

$$P_n \geq \sum_{j=1}^{6} p_j, \tag{7}$$

$$p_1 = 1 \geq p_2 \geq p_3 \geq p_4 \geq p_5 \geq p_6, \tag{8}$$

$$\mathrm{opt}_n \geq p_1, \tag{9}$$

$$\mathrm{opt}_n \geq \frac{P_n}{2}. \tag{10}$$

The optimal algorithm schedules at least three of the first five jobs on the same machine, thus we have

$$\mathrm{opt}_n \geq p_3 + p_4 + p_5. \tag{11}$$

The greedy algorithm is optimal for 4 jobs with decreasing processing times. We therefore have

$$\mathrm{opt}_n \geq x_4^2. \tag{12}$$

Consider how the first five jobs are scheduled. In schedule 1, the first 2 jobs go on A, while the next 3 go on B. I.e.

$$X_1^1 = A, \quad X_2^1 = A, \quad X_3^1 = B, \quad X_4^1 = B, \quad X_5^1 = B.$$

There are 2 possible states which we might be in after the first 5 jobs. Either $X_6^1 = A$ or $X_6^1 = B$. In schedule 2, the first job goes on A, while jobs 2 and 3 go on B. I.e.

$$X_1^2 = A, \qquad X_2^2 = B, \qquad X_3^2 = B.$$

The values of X_4^2, X_5^2 and X_6^2 might each have been A or B, leading to a total of 8 possibilities. Note that

$$X_6^1 = A, \qquad X_4^2 = B, \qquad X_5^2 = B, \qquad X_6^2 = B$$

and

$$X_6^1 = A, \qquad X_4^2 = B, \qquad X_5^2 = B, \qquad X_6^2 = A$$

are impossible: $X_6^1 = A$ implies that $p_3 + p_4 + p_5 \geq p_1 + p_2$ and therefore $p_3 + p_4 \geq p_1$. But in both situations, schedule 2 implies that $p_1 \geq p_2 + p_3 + p_4 \geq 2p_3 + p_4$, a contradiction. Now consider the situation in schedule j for the scheduling of jobs $6, \ldots, n$. Define

$$Q = P_n - P_6, \qquad y_5^j = P_5 - x_5^j = \min\{a_5^j, b_5^j\}, \tag{13}$$

for $j = 1, 2$. There are three possibilities:

1. $y_5^j + p_6 + Q \leq x_5^j$, in which case we have $x_n^j = x_5^j$.

2. $y_5^j + p_6 + Q > x_5^j$ and $y_5^j + Q \leq x_5^j$, in which case we have $x_n^j \leq y_5^j + Q + p_6$.

3. $y_5^j + Q > x_5^j$, in which case we have $x_n^j \leq (P_n - p_6)/2 + p_6$.

Using the above breakdown, there are $14 \times 3 \times 3 = 126$ possible sub-cases. Each of these sub-cases can be described by a linear program with objective function (4), constraints (5-13) and variables $p_1, \ldots, p_6$, P_n, Q, $a_1^j, \ldots, a_6^j$, $b_1^j, \ldots, b_6^j$, $x_1^j, \ldots, x_5^j$, y_5^j, x_n^j, opt_n for $j = 1, 2$. All of these 126 linear programs have objective value at most 0, as we have verified by using a computer. We conclude that the algorithm indeed is $\frac{8}{7}$-competitive. ∎

5 Conclusions and open problems

Several interesting points are:

1. The optimal randomized two machine semi-online algorithm is *barely random* [15], a distribution over a constant number of deterministic algorithms. Each job is scheduled in $O(1)$ time, and the algorithm uses $O(1)$ space. For classical online scheduling, on the other hand, the known optimal algorithm uses an unbounded number of random choices [3]. In fact, it also uses $\Omega(i)$ time to schedule the ith job and $\Omega(n)$ space overall.

2. In the analysis of the randomized two machine algorithm, we use strong bounds on the optimal offline makespan. To show the $\frac{4}{3}$-competitiveness of their algorithm, Bartal *et al.* use only two simple bounds: The processing time of the largest job, and half the total processing time of all jobs.

3. Interestingly, for semi-online scheduling, we have shown preemptive lower bounds which are greater than known deterministic non-preemptive upper bounds. For classical online scheduling, preemptive algorithms strictly outperform non-preemptive ones [1, 5].

4. Our upper bound for preemptive semi-online scheduling does not really require that the job sizes are decreasing: Actually, even the knowledge of the size of the largest job would be sufficient.

Some interesting open problems include:

1. What is the best possible competitive ratio for deterministic semi-online scheduling on $m \geq 3$ machines?

2. What is the best possible competitive ratio for randomized semi-online scheduling on $m \geq 3$ machines? This problem is probably even harder than the first one. For the standard online problem, little is known about algorithms for randomized scheduling [16, 17].

References

[1] ALBERS, S. Better bounds for online scheduling. In *Proceedings of the 29th ACM Symposium on Theory of Computing* (1997), 130–139.

[2] AZAR, Y., AND REGEV, O. Online bin stretching. In *Proceedings of RANDOM* (1998), 71–82.

[3] BARTAL, Y., FIAT, A., KARLOFF, H., AND VOHRA, R. New algorithms for an ancient scheduling problem. *Journal of Computer and System Sciences 51* (1995), 359–366.

[4] BARTAL, Y., KARLOFF, H., AND RABANI, Y. A better lower bound for on-line scheduling. *Information Processing Letters 50* (1994), 113–116.

[5] CHEN, B., VAN VLIET, A., AND WOEGINGER, G. A lower bound for randomized on-line scheduling algorithms. *Information Processing Letters 51* (1994), 219–222.

[6] CHEN, B., VAN VLIET, A., AND WOEGINGER, G. New lower and upper bounds for on-line scheduling. *Operations Research Letters 16* (1994), 221–230.

[7] CHEN, B., VAN VLIET, A., AND WOEGINGER, G. An optimal algorithm for preemptive on-line scheduling. *Operations Research Letters 18* (1995), 127–131.

[8] FAIGLE, U., KERN, W., AND TURÀN, G. On the performance of on-line algorithms for partition problems. *Acta Cybernetica 9* (1989), 107–19.

[9] GALAMBOS, G., AND WOEGINGER, G. An online scheduling heuristic with better worst case ratio than Graham's list scheduling. *SIAM Journal on Computing 22* (1993), 349–355.

[10] GRAHAM, R. L. Bounds for certain multiprocessing anomalies. *Bell Systems Technical Journal 45* (1966), 1563–1581.

[11] GRAHAM, R. L. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics 17* (1969), 263–269.

[12] KARGER, D., PHILLIPS, S., AND TORNG, E. A better algorithm for an ancient scheduling problem. *Journal of Algorithms 20* (1996), 400–430.

[13] KELLERER, H., KOTOV, V., SPERANZA, M., AND TUZA, Z. Semi online algorithms for the partition problem. *Operations Research Letters 21* (1997), 235–242.

[14] LIU, W. P., SIDNEY, J. B., AND VAN VLIET, A. Ordinal algorithms for parallel machine scheduling. *Operations Research Letters 18* (1996), 223–232.

[15] REINGOLD, N., WESTBROOK, J., AND SLEATOR, D. Randomized competitive algorithms for the list update problem. *Algorithmica 11* (1994), 15–32.

[16] SEIDEN, S. S. Randomized algorithms for that ancient scheduling problem. In *Proceedings of the 5th Workshop on Algorithms and Data Structures* (1997), 210–223.

[17] SEIDEN, S. S. Randomized Online Multiprocessor Scheduling. *Algorithmica*. To appear.

[18] SGALL, J. *On-Line Scheduling on Parallel Machines*. PhD thesis, Carnegie-Mellon University, 1994.

[19] SGALL, J. A lower bound for randomized on-line multiprocessor scheduling. *Information Processing Letters 63* (1997), 51–55.

[20] SGALL, J. *On-Line Scheduling*. In *Online Algorithms: The State of the Art*, eds. A. Fiat and G. J. Woeginger, LNCS 1442, Springer Verlag (1998), 196–231.