Better Bounds for Online Scheduling

Susanne Albers*

Abstract

We study a classical problem in online scheduling. A sequence of jobs must be scheduled on m identical parallel machines. As each job arrives, its processing time is known. The goal is to minimize the makespan. Bartal, Fiat, Karloff and Vohra [3] gave a deterministic online algorithm that is 1.986-competitive. Karger, Phillips and Torng [11] generalized the algorithm and proved an upper bound of 1.945. The best lower bound currently known on the competitive ratio that can be achieved by deterministic online algorithms it equal to 1.837. In this paper we present an improved deterministic online scheduling algorithm that is 1.923-competitive, for all $m \geq 2$. The algorithm is based on a new scheduling strategy, i.e., it is not a generalization of the approach by Bartal *et al.* Also, the algorithm has a simple structure. Furthermore, we develop a better lower bound. We prove that, for general m, no deterministic online scheduling algorithm can be better than 1.852-competitive.

1 Introduction

We study a classical problem in online scheduling. A sequence of jobs must be scheduled on m identical parallel machines. Whenever a job arrives, its processing time is known in advance, and the job must be scheduled immediately on one of the machines, without knowledge of any future jobs. Preemption of jobs is not allowed. The goal is to minimize the *makespan*, i.e., the completion time of the last job that finishes.

Algorithms for this scheduling problem are used in multiprocessor scheduling. Moreover, the problem is important because it is the root of many problem variants where, for instance, preemption is allowed, precedence constraints exist among jobs, or machines run at different speeds. The problem was first investigated by Graham [10]. In fact, Graham studied the *offline* version of the problem, when all jobs are known in advance. The problem of computing an optimal offline schedule for a given job sequence in NP-hard [9]. Graham gave a fast scheduling heuristic that achieves a good approximation ratio. He developed the well-known *List* algorithm that takes the given jobs one by one and always schedules them on the least loaded machine. Clearly, *List* is also an online algorithm.

Following [13], we call a deterministic online scheduling algorithm A *c-competitive* if, for all job sequences $\sigma = J_1, J_2, \ldots, J_n$,

$$A(\sigma) \leq c \cdot OPT(\sigma),$$

where $A(\sigma)$ is the makespan of the schedule produced by A and $OPT(\sigma)$ is the makespan of an optimal schedule for σ.

*Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany. E-mail: albers@mpi-sb.mpg.de. Work supported in part by the EU ESPRIT LTR Project N. 20244 (ALCOM-IT).

Graham's *List* algorithm is $(2-\frac{1}{m})$-competitive. Galambos and Woeginger presented an algorithm that is $(2-\frac{1}{m}-\epsilon_m)$-competitive, where $\epsilon_m > 0$, but ϵ_m tends to 0 as m goes to infinity. It was unknown for a long time whether there is an algorithm that achieves a competitive ratio of c, $c < 2$, for general m. Bartal, Fiat, Karloff and Vohra [3] then gave an algorithm that is 1.986-competitive, for all $m \geq 70$. Karger, Phillips and Torng [11] generalized the algorithm and proved a competitive ratio of 1.945, for all m. This has been the best upper bound known so far for general m. For the special case $m = 4$, Chen, van Vliet and Woeginger [6] developed an algorithm that is 1.733-competitive. With respect to lower bounds, Faigle, Kern and Turan [7] showed that no deterministic online algorithm can have a competitive ratio smaller than $(2-\frac{1}{m})$ for $m = 2$ and $m = 3$. Thus, for these values of m, *List* is optimal. Faigle *et al.* [7] also proved that no deterministic online algorithm can be better than 1.707-competitive, for any $m \geq 4$. The best lower bound known so far for general m is due to Bartal, Karloff and Rabani [4] who showed that no deterministic online algorithm can have a competitive ratio smaller than 1.837, for $m \geq 3454$. For more work on related online scheduling problems see, for instance, [1, 2, 5, 12, 14].

In this paper we present an improved deterministic online algorithm for the scheduling problem defined above. The algorithm is 1.923-competitive, for all $m \geq 2$. Our algorithm is based on a new scheduling strategy, i.e., it is not a generalization of the approach by Bartal *et al.* [3]. Moreover, the algorithm has a simple structure. At any time, the algorithm maintains a set S_1 of $\lfloor \frac{m}{2} \rfloor$ machines with a low load and a set S_2 of $\lceil \frac{m}{2} \rceil$ machines with a high load. Every job is either scheduled on the least loaded machine in S_1 or on the least loaded machine in S_2. The decision, which of the two machines to choose, depends on the ratio of the load on machines in S_1 to the load on machines in S_2. A description of the algorithm is given in Section 2. A detailed analysis follows in Section 3. We also develop a better lower bound for online scheduling. In Section 4 we show that if a deterministic online scheduling algorithm is c-competitive for all $m \geq 80$, then $c \geq 1.852$.

2 The new scheduling algorithm

For the description of the algorithm we need some definitions. Let the *load* of a machine be the sum of the processing times of the jobs already assigned to it. At any time, the algorithm maintains a list of the machines sorted in non-decreasing order by current load. Let M_k^t denote the machine with the k-th smallest load, $1 \leq k \leq m$, after exactly t jobs have been scheduled. In particular, M_1^t is the machine with the smallest load and M_m^t is the machine with the largest load. We denote by l_i^t the load of machine M_i^t, $1 \leq i \leq m$. Note that the load l_m^t of the most loaded machine is always equal to the current makespan.

As previous algorithms [3, 11], our new scheduling strategy tries to prevent schedules in which the load on all machines is about the same. If all machines have the same load, with all previous jobs being very small, an adversary can present an additional large job and force a competitive ratio of $(2-\frac{1}{m})$. This is the worst-case scenario for *List*.

Our new algorithm, called *M2*, always tries to maintain i machines with a low load and $m-i$ machine with a high load, where $i = \lfloor \frac{m}{2} \rfloor$. The goal is to always have a schedule in which the total load L_l on the i lightly loaded machines is at most α times the total load L_h on the $m-i$

heavily loaded machines, for some α to be specified later. A schedule satisfying $L_l \leq \alpha L_h$ is always prepared to handle a large incoming job and can easily maintain a competitive ratio of c, where c is 1.923.

Algorithm *M2* always schedules a new job J_t with processing time p_t on the least loaded machine as long as $L_l \leq \alpha L_h$ is satisfied after the assignment. Note that during this assignment, the load L_l on the lightly loaded machines does not necessarily increase by p_t because the least loaded machine might become one of the machines M_j^t, $i < j \leq m$. If an assignment of J_t to the least loaded machine results in $L_l > \alpha L_h$, then *M2* considers scheduling J_t on the machine with the $(i+1)$-st smallest load. However, if this assignment increases the makespan and the new makespan exceeds $c \cdot (L_l + L_h)/m$, then J_t is finally scheduled on the least loaded machine, ignoring the violation of $L_l \leq \alpha L_h$. Note that $L_l + L_h$ is the sum of the processing times of all jobs that have arrived so far, and thus $(L_l + L_h)/m$ is a lower bound on the optimum makespan.

Algorithm M2: Set $c = 1.923$, $i = \lfloor \frac{m}{2} \rfloor$ and $j = 0.29m$. Set $\alpha = \frac{(c-1)i - j/2}{(c-1)(m-i)}$. Every new job J_t is scheduled as follows. Let L_l be the sum of the loads on machines $M_1^t, \ldots, M_i^t$ if J_t is scheduled on the least loaded machine. Similarly, let L_h be the sum of the loads on machines $M_{i+1}^t, \ldots, M_m^t$ if J_t is scheduled on the least loaded machine. Let l_m^t be the makespan, i.e. the load of the most loaded machine, if J_t is scheduled on the machine with the $(i+1)$-st smallest load. Recall that l_m^{t-1} is the makespan before the assignment of J_t.

Schedule J_t on the least loaded machine if one of the following conditions holds.

(a) $L_l \leq \alpha L_h$

(b) $l_m^t > l_m^{t-1}$ and $l_m^t > c \cdot \frac{L_l + L_h}{m}$

Otherwise schedule J_t on the machine with the $(i+1)$-st smallest load.

Theorem 1 *Algorithm M2 is 1.923-competitive, for all $m \geq 2$.*

Before analyzing the algorithm in the next section, we discuss the choice of α. First observe that $0 < \alpha < 1$, for $m \geq 2$. The inequality $0 < \alpha$ holds because $c - 1 > 1/2$ and $i > j$; thus $(c-1)i - j/2 > 0$. Inequality $\alpha < 1$ holds because $(c-1)i \leq (c-1)(m-i)$ and $j/2 > 0$. In fact, for even m, $\alpha = \frac{(c-1) - j/m}{c-1} \approx 0.686$ and, for odd m, α tends to this value as m goes to infinity. Always setting $\alpha = 0.686$ in the algorithm *M2* asymptotically results in the same competitive ratio of 1.923. Choosing $\alpha = \frac{(c-1)i - j/2}{(c-1)(m-i)}$ has two advantages. (1) We can prove a competitiveness of 1.923 for even small m. (2) In the analysis we can do symbolic calculations where a fixed $\alpha = 0.686$ would require numeric calculations.

3 Analysis of the algorithm

We present a detailed proof of Theorem 1. The analysis presented by Graham [10] for the *List* algorithm, combined with the observation that algorithm *M2* only schedules a job on the machine with the $(i+1)$-st smallest load if the resulting makespan does not exceed 1.923 times the optimum makespan, shows that *M2* is c-competitive, where $c = \max\{(2 - \frac{1}{m}), 1.923\}$, for all $m \geq 2$. This gives the desired bound for small m. For $m \geq 8$, the following analysis applies.

Consider an arbitrary job sequence $\sigma = J_1, J_2, \ldots, J_n$. Let p_t be the processing time of J_t, $1 \leq t \leq n$. We will show that *M2* schedules every job J_t, $1 \leq t \leq n$, such that

$$M2(\sigma_t) \leq 1.923 \cdot OPT(\sigma_t),$$

where $M2(\sigma_t)$ is the makespan of the schedule produced by *M2* on the subsequence $\sigma_t = J_1, J_2, \ldots, J_t$ and $OPT(\sigma_t)$ is the makespan of an optimal schedule for σ_t.

3.1 The basic analysis

Suppose that *M2* has already scheduled the first $t-1$ jobs and that a competitive ratio of $c = 1.923$ was maintained at all times. Let

$$L = \sum_{s=1}^{t} p_s.$$

L is the sum of the loads on all machines after J_t is assigned.

If the makespan does not change during the assignment of J_t, then by induction hypothesis there is nothing to show. Also, if the makespan changes but is bounded from above by $c\frac{L}{m}$, then we are done because $\frac{L}{m}$ is a lower bound on the optimum makespan for σ_t.

Thus we concentrate on the case that during the assignment of J_t, the makespan increases and exceeds $c\frac{L}{m}$. Condition (b) in algorithm *M2* implies that J_t is scheduled on the least loaded machine. Let $l_1 = l_1^{t-1}$ be the load of the least loaded machine immediately before J_t is assigned.

First we consider the case that $l_1 \leq (c-1)\frac{L}{m} = 0.923\frac{L}{m}$. We have

$$M2(\sigma_t) \;=\; l_1 + p_t \;\leq\; (c-1)\tfrac{L}{m} + p_t.$$

If $p_t \leq \frac{L}{m}$, then

$$M2(\sigma_t) \;\leq\; (c-1)\tfrac{L}{m} + \tfrac{L}{m} \;\leq\; c\tfrac{L}{m} \;\leq\; c \cdot OPT(\sigma_t).$$

If $p_t = (1+\delta)\frac{L}{m}$, for some positive δ, then

$$\begin{aligned} M2(\sigma_t) &= l_1 + p_t \\ &\leq (c-1)\tfrac{L}{m} + (1+\delta)\tfrac{L}{m} \\ &\leq c \cdot (1+\delta)\tfrac{L}{m} \;=\; c \cdot p_t \\ &\leq c \cdot \max_{1 \leq s \leq t} p_s \;\leq\; c \cdot OPT(\sigma_t). \end{aligned}$$

Here we use the fact that $\max_{1 \leq s \leq t} p_s$ is also lower bound on the optimum makespan.

In the remainder of this proof we will study the situation that the load on the least loaded machine is greater than $(c-1)\frac{L}{m}$, i.e., $l_1 = (c-1+\epsilon)\frac{L}{m}$ for some positive ϵ. Since l_1 cannot be greater than $\frac{L}{m}$, we have $0 < \epsilon \leq 2 - c = 0.077$. Note that all machines must have a load of at least $(c-1+\epsilon)\frac{L}{m}$. Since J_t is assigned to the least loaded machine and the makespan after the assignment is greater than $c\frac{L}{m}$, we have $p_t > c\frac{L}{m} - l_1 \geq (1-\epsilon)\frac{L}{m} \geq (\frac{1}{2}+\epsilon)\frac{L}{m}$. Our goal is to show that the sequence $\sigma_{t-1} = J_1, J_2, \ldots, J_{t-1}$ contains m jobs, each with a processing time of at least $(\frac{1}{2}+\epsilon)\frac{L}{m}$. Then there are $m+1$ jobs with a processing time of at least $(\frac{1}{2}+\epsilon)\frac{L}{m}$, two of which must be scheduled on the same machine in an optimal schedule. Thus

$$OPT(\sigma_t) \;\geq\; (1+2\epsilon)\tfrac{L}{m}.$$

If $p_t \le \frac{L}{m}$, then

$$\begin{aligned} M2(\sigma_t) &= l_1 + p_t \;\le\; (c-1+\epsilon)\frac{L}{m} + \frac{L}{m} \\ &\le c(1+\epsilon)\frac{L}{m} \;\le\; c \cdot OPT(\sigma_t). \end{aligned}$$

If $p_t = (1+\delta)\frac{L}{m}$ for some positive δ, then

$$\begin{aligned} M2(\sigma_t) &= (c-1+\epsilon)\frac{L}{m} + p_t \;\le\; (c+\epsilon+\delta)\frac{L}{m} \\ &\le c \cdot \max\{(1+2\epsilon),\ (1+\delta)\}\frac{L}{m} \\ &\le c \cdot OPT(\sigma_t). \end{aligned}$$

In each case, Theorem 1 is proved. It remains to show that the sequence $\sigma_{t-1} = J_1, J_2, \ldots, J_{t-1}$ does contain m jobs, each with a processing time of at least $(\frac{1}{2}+\epsilon)\frac{L}{m}$.

3.2 Identifying large jobs

We have to analyze jobs in the sequence σ_{t-1}. Let *time* s, $1 \le s \le t$, denote the point of time immediately after J_s is scheduled. (Time 0 is the point of time before any jobs are scheduled.) For any time s, $1 \le s \le t$, let L_s be the total load on the m machines, i.e.,

$$L_s = \sum_{r=1}^{s} p_r.$$

Note that $L_t = L$.

Definition 1 At any time s, $1 \le s \le t$, the schedule constructed by *M2* is called *balanced* if the total load on the i lightly loaded machines $M_1^s, \ldots, M_i^s$ is at most α times the total load on the $m-i$ heavily loaded machines $M_{i+1}^s, \ldots, M_m^s$.

In the following, when refering to machines $M_1^s, \ldots, M_m^s$, we will often drop s when the meaning is clear.

Lemma 1 *At time $t-1$, i.e. immediately before J_t is scheduled, M2's schedule is not balanced.*

Proof: Immediately before J_t is scheduled, the total load on the machines $M_1, \ldots, M_i$ is at least

$$\overline{L}_l \;=\; i(c-1+\epsilon)\frac{L}{m} \;=\; (i(c-1) - \tfrac{j}{2})\frac{L}{m} + \frac{j}{2}\frac{L}{m} + i\epsilon\frac{L}{m}.$$

If *M2*'s schedule was balanced, then the total load on machines $M_{i+1}, \ldots, M_m$ would be at least $\frac{1}{\alpha}\overline{L}_l$. Thus the total load before the assignment of J_t would be at least

$$\begin{aligned} L_{t-1} &\ge (1+\tfrac{1}{\alpha})\overline{L}_l \\ &= i(c-1+\epsilon)\frac{L}{m} + (m-i)(c-1)\frac{L}{m} + \frac{1}{\alpha}\frac{j}{2}\frac{L}{m} + \frac{1}{\alpha}\epsilon\frac{L}{m} \\ &= (c-1)L + \frac{j}{2\alpha}\frac{L}{m} + \epsilon(i+\tfrac{1}{\alpha})\frac{L}{m} \\ &> L + (c-2)L + \frac{j}{2\alpha}\frac{L}{m} \\ &> L \end{aligned}$$

because $\alpha = \frac{(c-1)i - j/2}{(c-1)(m-i)} \le \frac{(c-1)-j/m}{c-1} \le \frac{7}{10}$ and thus $(c-2)L + \frac{j}{2\alpha}\frac{L}{m} > 0$. We have a contradiction. □

3.2.1 Analyzing load

Definition 2 At any time s, $1 \leq s \leq t$, a machine is called *full* if its load is at least $(c-1+\epsilon)\frac{L}{m}$.

Recall that at time $t-1$, all machines have a load of at least $(c-1+\epsilon)\frac{L}{m}$ and, thus, are full.

For $k = 1, \ldots, m$, let t_k be the most recent time when exactly k machines were full. Note that

$$t_1 \ < \ t_2 \ < \ \ldots \ < \ t_m = t-1.$$

Of particular interest to us will be the time $t_{m-\lfloor j \rfloor}$ when exactly $m - \lfloor j \rfloor$ machines were full. Let t', $t_{m-\lfloor j \rfloor} \leq t' < t-1$, be the most recent time when *M2*'s schedule was balanced. If *M2*'s schedule was not balanced during the time interval $[t_{m-\lfloor j \rfloor}, t-1]$, then let $t' = t_{m-\lfloor j \rfloor}$. Let f be the number of machines that are full at time t'.

Our goal is to show that at time t', the total load on the non-full machines $M_1, \ldots, M_{m-f}$ in *M2*'s schedule is at most $(c-1.5)(m-f)\frac{L}{m}$. We will show this using the following lemmas. Let

$$X \ = \ (c-1)L - \tfrac{j}{2}\tfrac{L}{m}$$
$$Y \ = \ (c-1)^2 L - \tfrac{cj}{2}\tfrac{L}{m}.$$

Lemma 2 *If at time t', the total load on the non-full machines $M_1, \ldots, M_{m-f}$ in M2's schedule were greater than $(c-1.5)(m-f)\frac{L}{m}$, then the total load at time t would satisfy $L \ > \ X + \frac{Y}{c}((1-\frac{c}{m})^{-(\lfloor j \rfloor+1)} - 1)$.*

Lemma 3 $\quad X + \frac{Y}{c}((1-\frac{c}{m})^{-(\lfloor j \rfloor+1)} - 1) \ \geq \ L$

Since Lemma 2 is more involved, we start with a proof of Lemma 3.

Proof of Lemma 3: We have

$$(1-\frac{c}{m})^{-(\lfloor j \rfloor+1)} \ \geq \ (1-\frac{c}{m})^{-j} \ \geq \ e^{cj/m}.$$

The first inequality follows because $\lfloor j \rfloor + 1 \geq j$. Thus,

$$\begin{aligned} X + \tfrac{Y}{c}((1-\tfrac{c}{m})^{-(\lfloor j \rfloor+1)} - 1) \ &\geq \ (1-\tfrac{1}{c})L + (\tfrac{(c-1)^2}{c}L - \tfrac{j}{2m}L) \cdot e^{cj/m} \\ &= \ (1-\tfrac{1}{1.923})L + (\tfrac{0.923^2 - 1.923 \cdot 0.145}{1.923}L) \cdot e^{0.29 \cdot 1.923}. \end{aligned}$$

Evaluating the last expression gives that it is at least $1L$. $\Box$

Proof of Lemma 2: In order to prove the lemma, we have to keep track of the load on the m machines during the entire time interval $[t', t]$. For $k = f, \ldots, m+1$, let

$$Z_k \ = \ X + \tfrac{Y}{c}((1-\tfrac{c}{m})^{-(k-m+\lfloor j \rfloor)} - 1).$$

We will show by induction on k that for $k = f, \ldots, m$,

$$L_{t_k} - \Phi_{t_k} \ > \ Z_k, \tag{1}$$

where Φ is a non-negative potential that we will define in a moment. Using the inequality $L_{t_m} - \Phi_{t_m} > Z_m$, we will then prove $L > Z_{m+1}$.

We first explain the purpose of the potential. We want to show that during the time interval $(t', t-1]$, every time another machine becomes full, a job J with a large processing time p must be scheduled. Since *M2*'s schedule is not balanced, *M2* would prefer to assign J to machine M_{i+1}. However, this is impossible because p is too large, and thus *M2* has to schedule J on the least loaded machine, causing another machine to become full. In some cases, when M_{i+1} has a high load, we will not be able to argue that J's processing time is greater than a certain value. In these cases we will pay some "missing processing time" out of the potential. This way we can ensure that J's *amortized processing time* is greater than the desired value. J's amortized processing time is the actual processing time plus the change in potential.

Formally, the potential Φ is defined as follows. At time t', we *color* some of the load in *M2*'s schedule. More precisely, on each of the machines $M_1, \ldots, M_{i+m-f}$ we color the load that is above level $(c-1)\frac{L}{m}$. We can imagine that we draw a horizontal line at level $(c-1)\frac{L}{m}$ across *M2*'s schedule and color the load on machines $M_1, \ldots, M_{i+m-f}$ that is above this line. Note that this way, a job might be partially colored. During time interval $(t', t]$, the colored load is updated as follows.

1. Whenever *M2* schedules a job that causes one more machine to become full, we choose the least loaded machine with colored load among $M_{i+1}, \ldots, M_m$ and remove the color from its load.

2. Whenever a job is assigned to a machine with colored load, we color that job.

3. After the final job J_t is scheduled, the color is removed from all machines.

At any time, let $I = \{k | M_k \text{ has colored load}\}$ and let c_k, $k \in I$, be the amount of the load that is colored on M_k. Define

$$\Phi = \sum_{k \in I} c_k.$$

During the interval $(t', t-1]$, the following invariants hold.

(I1) Whenever *M2* schedules a job that causes one more machine to become full, there is a machine in $\{M_{i+1}, \ldots, M_m\}$ with colored load.

(I2) If a machine has colored load, then all its load above level $(c-1)\frac{L}{m}$ is colored.

(I3) At any time, if machine M_{i+1} has load $(c-1+\delta)\frac{L}{m}$ for some positive δ, then $c_k \geq \delta$ for all $k \in I$ with $k \geq i+1$.

(I4) At any time, there exists a machine among $M_1, \ldots, M_i$ with colored load at least $\epsilon\frac{L}{m}$.

Invariant (I1) holds because at time t' there are $m-f$ machines in $\{M_{i+1}, \ldots, M_m\}$ with colored load, exactly $m-f$ more jobs are scheduled in $(t', t-1]$ that cause a machine to become full and every time this happens, by update rule 1, the number of machines in $\{M_{i+1}, \ldots, M_m\}$ with colored load is reduced by exactly 1. Invariant (I2) follows from update rule 2. (I3) is immediate from (I2). Invariant (I4) holds because initially, at time t', we color load on the i lightly loaded machines that are full and these machines remain in the set of lightly loaded machines during $(t', t-1]$.

Base of the induction. In order to prove inequality (1) for $k = f$, we have to evaluate $L_{t'}$, the total load on the m machines at time t'. We will show that

$$L_{t'} - \Phi_{t'} \ > \ Z_f. \tag{2}$$

This implies that inequality (1) holds for $k = f$ because between time t' and time t_f the number of full machines remains the same and whenever the load on the m machines increases by p, the potential increases by at most p, see update rule 2.

If *M2*'s schedule is not balanced at time t', i.e., $t' = t_{m-\lfloor j \rfloor}$ and $f = m - \lfloor j \rfloor$, then

$$Z_f \ = \ X \le (c-1)L - \tfrac{\lfloor j \rfloor}{2}\tfrac{L}{m}. \tag{3}$$

By assumption, at time t', the load on the non-full machines is greater than $(c-1.5)(m-f)\frac{L}{m} = (c-1.5)\lfloor j \rfloor \frac{L}{m} = (c-1)\lfloor j \rfloor \frac{L}{m} - \frac{\lfloor j \rfloor}{2}\frac{L}{m}$. The load on the full machines is at least $(c-1)f\frac{L}{m} + \Phi_{t'} = (c-1)(m - \lfloor j \rfloor)\frac{L}{m} + \Phi_{t'}$. We obtain

$$L_{t'} - \Phi_{t'} \ > \ (c-1)L - \tfrac{\lfloor j \rfloor}{2}\tfrac{L}{m}. \tag{4}$$

Inequalities (3) and (4) give the desired bound.

We study the case that *M2*'s schedule is balanced at time t'. The total load on the non-full machines is greater than $(c-1.5)(m-f)\frac{L}{m}$. The load on each machine $M_{m-f+1}, \ldots, M_i$ is at least $(c-1+\epsilon)\frac{L}{m}$. Thus the total load on the i lightly loaded machines $M_1, \ldots, M_i$ is greater than

$$\begin{aligned}
\overline{L}_l \ &= \ (c-1)(i-m+f)\tfrac{L}{m} + (c-1.5)(m-f)\tfrac{L}{m} \\
&= \ (c-1)i\tfrac{L}{m} - \tfrac{1}{2}(m-f)\tfrac{L}{m} \\
&\ge \ (c-1)i\tfrac{L}{m} - \tfrac{j}{2}\tfrac{L}{m} + \tfrac{1}{2}(f - m + \lfloor j \rfloor)\tfrac{L}{m}.
\end{aligned}$$

Since the schedule is balanced, the total load $\overline{L}_h$ on the heavily loaded machines $M_{i+1}, \ldots, M_m$ is at least $\frac{1}{\alpha}$ times the above expression, i.e.,

$$\overline{L}_h \ > \ (c-1)(m-i)\tfrac{L}{m} + \tfrac{1}{2\alpha}(f - m + \lfloor j \rfloor)\tfrac{L}{m}.$$

The load $(c-1)(m-i)\frac{L}{m}$ can fill the machines $M_{i+1}, \ldots, M_m$ up to a level of $(c-1)\frac{L}{m}$. Of the additional load $\frac{1}{2\alpha}(f - m + \lfloor j \rfloor)\frac{L}{m}$, at least a fraction of $\frac{(m-i)-(m-f)}{m-i} = \frac{f-i}{m-i}$ is located on machines $M_{i+m-f+1}, \ldots, M_m$ and does not go into the potential. Thus,

$$L_{t'} - \Phi_{t'} \ > \ X + \tfrac{1}{2}(f - m + \lfloor j \rfloor)(1 + \tfrac{1}{\alpha}\tfrac{f-i}{m-i})\tfrac{L}{m}.$$

We have to show that

$$\tfrac{1}{2}(f - m + \lfloor j \rfloor)(1 + \tfrac{1}{\alpha}\tfrac{f-i}{m-i})\tfrac{L}{m} \ \ \ge \ \ \tfrac{Y}{c}((1 - \tfrac{c}{m})^{-(f-m+\lfloor j \rfloor)} - 1) \tag{5}$$

holds for every $f \in \{m - \lfloor j \rfloor, \ldots, m\}$.

We define functions

$$g(x) = \tfrac{1}{2}(x - m + \lfloor j \rfloor)(1 + \tfrac{1}{\alpha}\tfrac{x-i}{m-i})\tfrac{L}{m}$$
$$h(x) = \tfrac{Y}{c}((1 - \tfrac{c}{m})^{-(x-m+\lfloor j \rfloor)} - 1).$$

Obviously, $g(m - \lfloor j \rfloor) = h(m - \lfloor j \rfloor)$. We will show that $g(m + j - \lfloor j \rfloor) > h(m + j - \lfloor j \rfloor)$. Since $g(x)$ is a polynomial of degree 2 and $h(x)$ is an exponentially increasing function, $g(x) > h(x)$ must hold for all $x \in (m - \lfloor j \rfloor, m]$. (If $g(x_0) \le h(x_0)$ were true for some $x_0 \in (m - \lfloor j \rfloor, m]$, then $g(x_0) < h(x_0)$ for all $x > x_0$.)

Recall that, as mentioned in the proof of Lemma 1, $\alpha \le \frac{7}{10}$. Also, evaluating Y with its actual parameters gives $Y \le 0.574L$. If $x = m + j - \lfloor j \rfloor$, then $g(x) \ge \frac{1}{2}0.29m(1 + \frac{10}{7})\frac{L}{m} \ge 0.35L$. Also, $h(x) = \frac{Y}{c}((1 - \frac{c}{m})^{-0.29m} - 1)$ is decreasing in m and evaluates to less than $0.34L$ for all $m \ge 4$. The proof of inequality (5) is complete.

Induction step. We show that if inequality (1) holds for $k-1$, then it also holds for k. Let s_k be the earliest point of time when exactly k machines are full. We have $t_{k-1} < s_k \le t_k$. For all $s \in [t_{k-1}, s_k - 1]$,

$$L_s - \Phi_s \ > \ Z_{k-1}.$$

This is because of the induction hypothesis and the fact that if the load on the m machines increases by p between time t_{k-1} and time $s_k - 1$, then the potential increases by at most p.

Let L_{s_k-1} be the total load on the m machines at time $s_k - 1$ and let $l = l_{i+1}^{s_k-1}$ be the load on machine M_{i+1} at time $s_k - 1$. Suppose $l = (c - 1 + \delta)\frac{L}{m}$ for some $\delta \ge 0$. The job J_{s_k} that causes the k-th machine to become full is scheduled on the least loaded machine. Since *M2*'s schedule is not balanced at time s_k, *M2* would prefer to schedule J_{s_k} on machine M_{i+1}. Since this is not possible, condition (b) in algorithm *M2* implies that the processing time p_{s_k} of J_{s_k} must satisfy

$$l + p_{s_k} \ > \ \frac{c(L_{s_k-1} + p_{s_k})}{m},$$

which is equivalent to

$$p_{s_k} \ > \ (\tfrac{c}{m}L_{s_k-1} - l)/(1 - \tfrac{c}{m}).$$

Consider the change in potential during the assignment of J_{s_k}. Update rule 1 and invariants (I1)–(I3) imply that the potential drops by at least $\delta\frac{L}{m}$.

$$\begin{aligned}
p_{s_k} - \Delta\Phi \ &> \ (\tfrac{c}{m}L_{s_k-1} - l)/(1 - \tfrac{c}{m}) + \delta\tfrac{L}{m} \\
&\ge \ (\tfrac{c}{m}(Z_{k-1} + \Phi_{s_k-1}) - l)/(1 - \tfrac{c}{m}) + \delta\tfrac{L}{m} \\
&\ge \ (\tfrac{c}{m}Z_{k-1} + \tfrac{c\delta}{m}\tfrac{L}{m} - (c - 1 + \delta)\tfrac{L}{m})/(1 - \tfrac{c}{m}) + \delta\tfrac{L}{m} \\
&\ge \ (\tfrac{c}{m}Z_{k-1} - (c-1)\tfrac{L}{m})/(1 - \tfrac{c}{m}) \\
&= \ \tfrac{Y}{m}(1 - \tfrac{c}{m})^{-(k-m+\lfloor j \rfloor)}.
\end{aligned}$$

The second inequality follows because of the induction hypothesis. The third inequality holds because at time $s_k - 1$, there is at least one machine in $\{M_{i+1}, \ldots, M_m\}$ with colored load, i.e., $\Phi_{s_k-1} \ge \delta\frac{L}{m}$. Thus,

$$\begin{aligned}
L_{s_k} - \Phi_{s_k} \ &\ge \ L_{s_k-1} - \Phi_{s_k-1} + p_{s_k} - \Delta\Phi \\
&> \ Z_{k-1} + \tfrac{Y}{m}(1 - \tfrac{c}{m})^{-(k-m+\lfloor j \rfloor)}
\end{aligned}$$

$$\begin{aligned} &= X + \tfrac{Y}{c}((1-\tfrac{c}{m})^{-(k-m+\lfloor j \rfloor)} - 1) \\ &= Z_k. \end{aligned}$$

The induction step is complete because during time interval $(s_k, t_k]$ the inequality is maintained.

We finally have to prove

$$L > Z_{m+1}. \tag{6}$$

Our inductive proof shows $L_{t-1} - \Phi_{t-1} > Z_m$. Job J_t is scheduled on the least loaded machine and by assumption $l_1 + p_t > \frac{c(L_{t-1}+p_t)}{m}$, where $l_1 = (c-1+\epsilon)\frac{L}{m}$ is the load of the least loaded machine at time $t-1$, i.e., immediately before J_t is scheduled. Recall that at time t we remove the color from the load in *M2*'s schedule. Invariant (I4) implies that the potential at time t must decrease by at least $\epsilon\frac{L}{m}$. Calculations identical to that in the inductive step show inequality (6). □

We summarize again the results of Lemmas 2 and 3.

Lemma 4 *At time t', the total load on the non-full machines $M_1, \ldots, M_{m-f}$ is at most $(c-1.5)(m-f)\frac{L}{m}$.*

3.2.2 Tracing the assignment of large jobs

We now identify jobs with a processing time of at least $(\frac{1}{2}+\epsilon)\frac{L}{m}$.

Lemma 5 *a) During the time interval $(t_{m-i}, t']$, $f-m+i$ jobs, each of size at least $(\frac{1}{2}+\epsilon)\frac{L}{m}$, are scheduled.*

b) At time t_{m-i}, each of the machines $M_1, \ldots, M_{f-m+i+1}$ has a load of at most $(c-1.5)\frac{L}{m}$. The total load on the machines $M_{i-m+f+1}, \ldots, M_i$ is at most $(c-1.5)(m-f)\frac{L}{m}$.

Proof: *Part a)* At time t_{m-i}, $m-i$ machines are full. At time t', f machines are full, where $f \geq m - \lfloor j \rfloor$. Consider the $f-m+i$ steps in $(t_{m-i}, t']$ at which the number of full machines increases. Since at least $m-i$ machines are full, the number of full machines can only increase if a job is scheduled on the least loaded machine. By Lemma 4, at time t', the total load on the $m-f$ least loaded machines is at most $(c-1.5)(m-f)\frac{L}{m}$. This implies that at time t', the load on the least loaded machine is at most $(c-1.5)\frac{L}{m}$. Thus, at any of the $f-m+i$ steps in $(t_{m-i}, t']$ at which the number of full machines increases, the load on the least loaded machine is at most $(c-1.5)\frac{L}{m}$. Hence jobs of size at least $(c-1+\epsilon)\frac{L}{m} - (c-1.5)\frac{L}{m} = (\frac{1}{2}+\epsilon)\frac{L}{m}$ are introduced.

Part b) The machine with the $(f-m+i+1)$-st smallest load at time t_{m-i} becomes the least loaded machine no later than time t', when f machines are full. Thus, if at time t_{m-i}, machine $M_{f-m+i+1}$ (or any machine M_k with $k \leq f-m+i$) had a load greater than $(c-1.5)\frac{L}{m}$, then the load of the least loaded machine at time t' would be greater than $(c-1.5)\frac{L}{m}$. Lemma 4 implies that this is impossible. Similarly, if at time t_{m-i}, machines $M_{i-m+f+1}, \ldots, M_i$ had a total load of at least $(c-1.5)(m-f)\frac{L}{m}$, then the total load on $M_1, \ldots, M_{m-f}$ at time t' would be at least $(c-1.5)(m-f)\frac{L}{m}$. Again, Lemma 4 gives the desired statement. □

Lemma 6 *During the time interval $(t', t-1]$, $m-f$ jobs of size at least $(\frac{1}{2}+\epsilon)\frac{L}{m}$ are scheduled.*

Proof: By the definition of t', *M2*'s schedule is not balanced during $[t'+1, t-1]$. Let $s \in [t'+1, t-1]$ be any of the $m-f$ time steps in $[t'+1, t-1]$ at which the number of full machines has just increased. If the load on the least loaded machine is at most $(c-1.5)\frac{L}{m}$ when J_s is scheduled, then $p_s \geq (c-1+\epsilon)\frac{L}{m} - (c-1.5)\frac{L}{m} = (\frac{1}{2}+\epsilon)\frac{L}{m}$.

Suppose that immediately before the assignment of J_s, the least loaded machine has a load greater than $(c-1.5)\frac{L}{m}$. Let l_{i+1}^{s-1} be the load of machine M_{i+1} and suppose $l_{i+1}^{s-1} = (c-1+\epsilon+\delta)\frac{L}{m}$ for some non-negative δ. By the definition of t', at least $m - \lfloor j \rfloor$ machines are full at any time in $[t', t-1]$. Thus,

$$\begin{aligned}
L_{s-1} &\geq (m-\lfloor j\rfloor)(c-1+\epsilon)\tfrac{L}{m} + \lfloor j\rfloor(c-1.5)\tfrac{L}{m} + (m-i)\delta\tfrac{L}{m} \\
&\geq (c-1)L - \tfrac{1}{2}\lfloor j\rfloor + (m-\lfloor j\rfloor)\epsilon\tfrac{L}{m} + \tfrac{\delta}{2}L \\
&\geq (c-1)L - \tfrac{j}{2}\tfrac{L}{m} + (m-j)\epsilon\tfrac{L}{m} + \tfrac{\delta}{2}L.
\end{aligned}$$

The second inequality follows because $m - i \geq \frac{1}{2}m$. Since *M2*'s schedule is not balanced, *M2* would prefer to schedule J_s on machine M_{i+1} but cannot because

$$l_{i+1}^{s-1} + p_s \; > \; c(L_{s-1}+p_s)/m.$$

Hence,

$$\begin{aligned}
p_s &\geq (\tfrac{c}{m}L_{s-1} - l_{i+1}^{s-1})/(1-\tfrac{c}{m}) \\
&\geq \tfrac{c}{m}L_{s-1} - l_{i+1}^{s-1} \\
&\geq c(c-1-\tfrac{j}{2m})\tfrac{L}{m} - (c-1)\tfrac{L}{m} + c(1-\tfrac{j}{m})\epsilon\tfrac{L}{m} - \epsilon\tfrac{L}{m} + \tfrac{c\delta}{2}\tfrac{L}{m} - \delta\tfrac{L}{m}.
\end{aligned}$$

The load $l_{i+1}^{s-1} = (c-1+\epsilon+\delta)\frac{L}{m}$ cannot be greater than $(3-c-\epsilon)\frac{L}{m}$ since otherwise we would have, at time $t-1$, $m-i$ machines each with a load greater than $(3-c-\epsilon)\frac{L}{m}$ and i machines each with a load of at least $(c-1+\epsilon)\frac{L}{m}$, resulting in a total load greater than L. Thus, $\delta \leq 4-2c-2\epsilon$ and

$$\begin{aligned}
p_s &\geq ((c-1)^2 - \tfrac{cj}{2m})\tfrac{L}{m} + (c-1-\tfrac{cj}{m})\epsilon\tfrac{L}{m} + 2(\tfrac{c}{2}-1)(2-c-\epsilon)\tfrac{L}{m} \\
&\geq ((c-1)^2 - (c-2)^2 - \tfrac{cj}{2m})\tfrac{L}{m} + (1-\tfrac{cj}{m})\epsilon\tfrac{L}{m} \\
&\geq 0.567\tfrac{L}{m} + 0.442\epsilon\tfrac{L}{m} \\
&\geq (\tfrac{1}{2}+\epsilon)\tfrac{L}{m}
\end{aligned}$$

for all $\epsilon \leq 0.12$. Recall that our ϵ is at most $2-c = 0.077$. □

We now consider the time $t_{m-i-\lfloor j\rfloor}$ when exactly $m-i-\lfloor j\rfloor$ machines are full. Let t'', $t_{m-i-\lfloor j\rfloor} \leq t'' \leq t_{m-i}$, be the earliest point of time at which the machine with the $(i+1)$-st smallest load has a load greater than $(c-1.5)\frac{L}{m}$.

Lemma 7 *During the time interval $(t'', t_{m-i}]$, every job is scheduled on the least loaded machine.*

Proof: We first show that at any time $s \in [t'', t_{m-i}]$, *M2*'s schedule is balanced. Lemma 5 part b) implies that at time s the total load on the lightly loaded machines $M_1, \ldots, M_i$ is at most $\overline{L}_l = i(c-1.5)\frac{L}{m}$. By the definition of t'', the total load on the heavily loaded machines $M_{i+1}, \ldots, M_m$ at time s is at least

$$\begin{aligned}\overline{L}_h &= (m-i-\lfloor j\rfloor)(c-1)\tfrac{L}{m} + \lfloor j\rfloor(c-1.5)\tfrac{L}{m} \\ &= (m-i)(c-1)\tfrac{L}{m} - \tfrac{\lfloor j\rfloor}{2}\tfrac{L}{m}.\end{aligned}$$

We show that at time s, the total load on the lightly loaded machines is at most α times the load on the heavily loaded machines. This holds if $\overline{L}_l \leq \alpha\overline{L}_h$, i.e., if

$$i(c-1.5)\tfrac{L}{m} \;\leq\; \tfrac{i(c-1)-j/2}{(c-1)(m-i)}\left((m-i)(c-1)\tfrac{L}{m} - \tfrac{\lfloor j\rfloor}{2}\tfrac{L}{m}\right),$$

which is equivalent to

$$(c-1)(i\lfloor j\rfloor + (m-i)j) - j\tfrac{\lfloor j\rfloor}{2} \;\leq\; (c-1)i(m-i).$$

This in turn holds if

$$jm - \tfrac{j^2}{2(c-1)} \;\leq\; i(m-i).$$

The left-hand side is at most $0.245m^2$, and the right-hand side is $\frac{1}{4}m^2$ for even m at and least $0.246m^2$ for odd $m \geq 9$. Thus, at time s, *M2*'s schedule is balanced.

Now consider job J_{s+1} scheduled at time $s+1$. Let l_1^s be the load on the least loaded machine at time s. We have $l_1^s \leq (c-1.5)\frac{L}{m}$. Let p be a processing time such that $l_1^s + p = (c-1.5)\frac{L}{m}$. The property stated in part b) of Lemma 5 must also hold at time s because the load on the lightly loaded machines $M_1, \ldots, M_i$ can only be smaller. Thus, if J_{s+1} has a processing time of at most p, scheduling J_{s+1} on the least loaded machine would result in a total load of at most $i(c-1.5)\frac{L}{m}$ on machines $M_1, \ldots, M_i$. Since the total load on machines $M_{i+1}, \ldots, M_m$ is at least $(m-i)(c-1)\frac{L}{m} - \frac{\lfloor j\rfloor}{2}\frac{L}{m}$, the calculations of the preceding paragraph show that *M2*'s schedule must be balanced after the assignment.

Suppose that J_{s+1} has a processing time $p_{s+1} > p$ and that scheduling J_{s+1} on the least loaded machine results in a load of $i(c-1.5)\frac{L}{m} + \delta\frac{L}{m}$ on machines $M_1, \ldots, M_i$, for some $\delta > 0$. This implies that at time $s+1$, the load on any of the machines $M_{i+1}, \ldots, M_{i+\lfloor j\rfloor}$ must be at least $(c-1+\delta)\frac{L}{m}$. With the above definitions of $\overline{L}_l$ and $\overline{L}_h$, we conclude that after the assignment of J_{s+1} to the least loaded machine, the total load on $M_1, \ldots, M_i$ is at most $\overline{L}_l + \delta\frac{L}{m}$ and the total load on $M_{i+1}, \ldots, M_m$ is at least $\overline{L}_h + \lfloor j\rfloor\delta\frac{L}{m}$. Since, for $m \geq 8$, we have $\lfloor j\rfloor \geq 2$ and $\alpha \geq \frac{1}{2}$, *M2*'s schedule must be balanced. □

Lemma 8 *At time $t''-1$, the load on machine M_{i+1} is at most $(c-1.5)\frac{L}{m}$.*

Proof: If $t'' > t_{m-i-\lfloor j\rfloor}$, then the lemma follows from the definition of t''. We show that t'' cannot be equal to $t_{m-i-\lfloor j\rfloor}$. Recall that $f \geq m - \lfloor j\rfloor$. Thus, Lemma 5 part b) implies that at time t_{m-i}, machine $M_{i-\lfloor j\rfloor+1}$ has a load of at most $(c-1.5)\frac{L}{m}$. If $t'' = t_{m-i-\lfloor j\rfloor}$, then there are $\lfloor j\rfloor$ steps in $(t'', t_{m-i}]$ at which the number of full machines increases. By Lemma 7, at all these steps, the jobs are assigned to the least loaded machine. Thus at time t'', the load of machine M_{i+1} cannot be greater than the load of machine $M_{i-\lfloor j\rfloor+1}$ at time t_{m-i}. This means that M_{i+1} has a load of at most $(c-1.5)\frac{L}{m}$ at time t'', contradicting the choice of t''. □

Lemma 9 *During time interval* $(0, t_{m-i}]$, $m - i$ *jobs of size at least* $(\frac{1}{2} + \epsilon)\frac{L}{m}$ *are scheduled.*

Proof: Let k be the number of machines that are full at time t''. Consider the k steps in $(0, t'']$ at which the number of full machines increases. At any of these steps, before the assignment of the job, the load on M_1 and M_{i+1} is at most $(c-1.5)\frac{L}{m}$ each, see Lemma 8. Thus jobs of size at least $(c-1+\epsilon)\frac{L}{m} - (c-1.5)\frac{L}{m} \geq (\frac{1}{2}+\epsilon)\frac{L}{m}$ must be scheduled. At the $m-i-k$ steps in $(t'', t_{m-i}]$ at which the number of full machines increases, jobs are scheduled on the least loaded machine (Lemma 7). The least loaded machine has a load of at most $(c-1.5)\frac{L}{m}$ and we conclude again that jobs of size at least $(\frac{1}{2}+\epsilon)\frac{L}{m}$ must be scheduled. $\square$

Lemma 5 part a) as well as Lemmas 6 and 9 imply the following statement.

Lemma 10 *During time interval* $(0, t-1]$, m *jobs of size at least* $(\frac{1}{2} + \epsilon)\frac{L}{m}$ *are scheduled.*

By the discussion immediately preceding Section 3.2, the proof of Theorem 1 is complete.

4 The lower bound

We develop an improved lower bound for deterministic scheduling algorithms.

Theorem 2 *Let A be a deterministic online scheduling algorithm. If A is c-competitive for all $m \geq 80$, then $c \geq 1.852$.*

Proof: We will construct a job sequence σ such that $A(\sigma) \geq 1.852 \cdot OPT(\sigma)$. The job sequence consists of several rounds. We assume that m is a multiple of 40.

Round 1:

Subround 1.1: m jobs with a processing time of $x_1 = 0.01$.

Subround 1.2: m jobs with a processing time of $x_2 = 0.06$.

Round 2:

Subround 2.1: $\frac{19}{20}m$ jobs with a processing time of $y_1 = 0.282$.

Subround 2.2: $\frac{1}{20}m$ jobs with a processing time of $y_2 = 0.4$.

Round 3:

Subround 3.1: $\frac{1}{2}m$ jobs with a processing time of $z_1 = 0.5$.

Subround 3.2: $\frac{1}{4}m$ jobs with a processing time of $z_2 = 1 - y_2 = 0.6$.

Subround 3.3: $\frac{3}{40}m$ jobs with a processing time of $z_3 = 1 - y_1 = 0.718$.

Subround 3.4: $\frac{3}{40}m$ jobs with a processing time of $z_4 = 0.84$.

Subround 3.5: $\frac{1}{10}m + 1$ jobs with a processing time of $z_5 = 1$.

Note that in the third round, $m + 1$ jobs have to be scheduled.

In the following, when analyzing the various subrounds, we will often compare the makespan produced by an online algorithm A in a subround to the optimum makespan at the end of the subround. It is clear that the optimum makespan during the subround can only be smaller.

Analysis of Round 1: Clearly, in order to maintain 1.852-competitiveness, online algorithm A must schedule the m jobs in Subround 1.1 on different machines. Also, A must schedule the m jobs in Subround 1.2 on different machines. Otherwise, A's makespan would be at least $x_1 + 2x_2 = 0.13$. Since the optimum makespan during the subround is always at most $x_1 + x_2 = 0.07$ and $\frac{0.13}{0.07} > 1.857$, A would not be 1.852-competitive. At the end of the first round, A has a load of $l_1 = x_1 + x_2 = 0.07$ on each of its machines.

Analysis of Round 2: At the end of Subround 2.1, the optimum makespan is at most $x_2+y_1 = 0.342$. On each of $\frac{19}{20}m$ machines, OPT schedules an x_2-job and a y_1-job. The remaining $\frac{1}{20}m$ machines have an x_2-job and 20 jobs of size x_1. If A does not schedule the jobs in Subround 2.1 on different machines, then its makespan is at least $x_1 + x_2 + 2y_1 = 0.634 > 1.853(x_2 + y_1)$. The optimum makespan after Subround 2.2 is $y_1 + 2x_2 = 0.402$. In an optimal schedule, $\frac{1}{20}m$ machines have a y_2-job, $\frac{1}{2}m$ machines have a y_1-job and two x_2-jobs. The remaining machines have a y_1-job and at most three x_1-jobs. Online algorithm A must schedule the jobs of Subround 2.2 on different machines and these machines may not contain any y_1-job since otherwise A's makespan is at least $x_1 + x_2 + y_1 + y_2 = 0.752 > 1.87(y_1 + 2x_2)$. At the end of Round 2, the least loaded machine in A's schedule has a load of $l_2 = x_1 + x_2 + y_1 = 0.352$.

Analysis of Round 3: *Subround 3.1*: After Subround 3.1, the optimum makespan is $y_1+y_2 = 0.682$. In an optimal schedule, $\frac{1}{20}m$ machines contain a y_1 and a y_2. $\frac{1}{2}m$ machines contain a z_1, two x_1 and two x_2. $\frac{9}{20}m$ machines contain two y_1. Algorithm A must schedule all z_1-jobs on different machines. Otherwise its makespan would be at least $l_2 + 2z_1 = 1.352 > 1.98(y_1 + y_2)$.

Subround 3.2: At the end of the subround, the optimum makespan is $y_1 + z_1 = 0.782$. In OPT's schedule, $\frac{1}{2}m$ machines have a y_1 and a z_1. $\frac{1}{20}m$ machines have a y_1 and a y_2. $\frac{1}{5}m$ machines have two y_1, three x_1 and three x_2. $\frac{1}{4}m$ machines have a z_2 and some of them have two additional x_1 and x_2. Algorithm A must schedule each z_2-job on a machine not containing any z_1 or z_2. Otherwise its makespan would be at least $l_2 + z_1 + z_2 = 1.452$, which is greater than $1.856(y_1 + z_1)$.

Subround 3.3: The optimum makespan after the subround is $3y_1 = 0.846$. In an optimal schedule $\frac{1}{2}m$ machines have a y_1, a z_1 and an x_2. $\frac{1}{4}m$ machines have a z_2, two x_2 and four x_1. $\frac{3}{40}m$ machines have a z_3. $\frac{3}{20}m$ machines three y_1. $\frac{1}{40}m$ machines two y_2. As before, A may not schedule any z_3-job on a machine containing a z_1, a z_2 or a z_3 because this would result in a makespan of at least $l_2 + z_1 + z_3 = 1.57 > 1.855(3y_1)$.

Subround 3.4: The optimum makespan is $y_2 + z_1 = 0.9$. In OPT's schedule, all the z-jobs are scheduled on different machines. $\frac{1}{20}m$ machines having a z_1 also contain a y_2. $(\frac{1}{2} - \frac{1}{20})m$ machines containing a z_1 also have a y_1, an x_2 and up to three x_1. The $\frac{1}{4}m$ machines having a z_2 also have a y_1. Machines having a z_3 also have three x_2. Machines having a z_4 also have an x_2. At this point, OPT is left with $\frac{1}{10}m$ machines on which it has to schedule $\frac{1}{4}m$ jobs with a processing time of x_2 and $\frac{1}{4}m$ jobs with a processing time of y_1. This can be done by scheduling (a) $\frac{1}{40}m$ machines with ten x_2 and one y_2 each and (b) $\frac{3}{40}m$ machines with three y_1. As usual, A may not schedule a z_4-job on a machine having already any z-job; otherwise its makespan is at least $l_2 + z_1 + z_4 = 1.692 = 1.88(y_2 + z_1)$.

Subround 3.5: The online algorithm A must schedule one of the z_5-jobs on a machines already containing another z-job, because a total of $m + 1$ jobs have to be scheduled in Round 3. This gives a makespan of at least $x_1 + x_2 + y_1 + z_1 + z_5 = 1.852$. We will show that OPT can schedule all the jobs with a makespan of 1 if $m \geq 80$. An optimal schedule is as follows. $\frac{1}{10}m$ machines have a z_5. $\frac{1}{4}m$ machines have two z_1. $\frac{3}{40}m$ machines have a z_4, two x_1 and two x_2. $\frac{3}{40}m$ machines have a z_3 and a y_1. $\frac{1}{5}m$ machines have a z_2, one y_1, two x_1 and one x_2. $\frac{1}{20}m$ machines have a z_2 and a y_2. $\frac{9}{40}m$ machines have three y_1, two x_1 and two x_2. OPT has $\frac{1}{40}m$ machines left on which it has to schedule one z_5 and $\frac{1}{5}m$ jobs of size x_2. This can be done if at least two machines are left, i.e. if $m \geq 80$. OPT can use one machine for the z_5-job and the remaining machines for the x_2-jobs. □

5 Open problems

An interesting problem is to formulate and analyze a generalization of the algorithm *M2* that, at any time, is allowed to schedule a new job on any of the m machines. In such an algorithm, the ratio of the load on the i-th smallest machine to the load on the $(i+1)$-st smallest machine has to be bounded by some α_i, $1 \leq i \leq m-1$. The problem is to specify α_i's and a proper scheduling rule that is able to maintain these values. A first step in this direction is to maintain three set S_1, S_2 and S_3 of $m/3$ machines with a low, medium and high load, respectively.

More generally, with respect to the scheduling problem studied here, a fundamental open problem is to develop randomized online algorithms that achieve a competitive ratio smaller than the deterministic lower bound, for all m.

References

[1] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin and O. Waarts. On-line machine scheduling with applications to load balancing and virtual circuit routing. In *Proc. 25th ACM Symposium on Theory of Computing*, pages 623–631, 1993.

[2] B. Awerbuch, Y. Azar, E.F. Grove, M.Y. Kao, P. Krishnan and J.S. Vitter. Load balancing in the L_p norm. In *Proc. 36th IEEE Annual Symposium on Foundations of Computer Science*, pages 383–391, 1995.

[3] Y. Bartal, A. Fiat, H. Karloff and R. Vohra. New algorithms for an ancient scheduling problem. *Journal of Computer and System Sciences*, 51:359–366, 1995.

[4] Y. Bartal, H. Karloff and Y. Rabani. A better lower bound for on-line scheduling. *Information Processing Letters*, 50:113–116, 1994.

[5] Y. Bartal, S. Leonardi, A. Marchetti-Speccamela, J. Sgall and L. Stougie. Multiprocessor scheduling with rejection. In *Proc. 7th ACM-SIAM Symposium on Discrete Algorithms*, pages 95–104, 1996.

[6] B. Chen, A. van Vliet and G. Woeginger. New lower and upper bounds for on-line scheduling. *Operations Research Letters*, 16:221-230, 1994.

[7] U. Faigle, W. Kern and G. Turan. On the performance of on-line algorithms for particular problems. *Acta Cybernetica*, 9:107–119, 1989.

[8] G. Galambos and G. Woeginger. An on-line scheduling heuristic with better worst case ratio than Graham's list scheduling. *SIAM Journal on Computing*, 22:349–355, 1993.

[9] M.R. Garay and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[10] R.L. Graham. Bounds for certain multi-processing anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

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

[12] R. Motwani, S. Phillips and E. Torng. Non-clearvoyant scheduling. In *Proc. 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 422–431, 1993.

[13] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, 1985.

[14] D. Shmoys, J. Wein and D.P. Williamson. Scheduling parallel machines on-line. In *Proc. 32nd IEEE Annual Symposium on Foundations of Computer Science*, pages 131–140, 1991.