J Sched (2011) 14: 89–101
DOI 10.1007/s10951-010-0177-x

Competitive ratio of List Scheduling on uniform machines and randomized heuristics

Antoine Musitelli · Jean-Marc Nicoletti

Published online: 4 May 2010

Abstract We study online scheduling on m uniform machines, where $m-1$ of them have a reference speed 1 and the last one a speed s with $0 \leq s \leq 1$. The competitive ratio of the well-known List Scheduling (LS) algorithm is determined. For some values of s and $m = 3$, LS is proven to be the best deterministic algorithm. We describe a randomized heuristic for m machines. Finally, for the case $m = 3$, we develop and analyze the competitive ratio of a randomized algorithm which outperforms LS for any s.

Keywords Analysis of algorithms · Online algorithms · Competitive ratio · Randomized scheduling

1 Introduction

A uniform machine system consists of m, $m \geq 1$ machines $(M_1, M_2, \ldots, M_m)$. A processing speed s_j, $s_j \geq 0$ is associated with each machine. The machines are *uniform* in that a machine M_j carries out s_j units of processing in one unit of time. An input to this problem consists of a sequence $\sigma = (\sigma_1, \ldots, \sigma_n)$ of n, $n \geq 1$ jobs of different *sizes* (*running or processing times*), each of which has to be assigned to a machine. Sometimes, to simplify notation, we will identify a job with its size. The job sequence σ is given *online*, i.e., the jobs arrive one by one. Neither the total number of jobs that need to be scheduled nor the size of the jobs is previously known. The processing time of job σ_i becomes known only when σ_{i-1} has already been scheduled. As soon as job σ_i appears, it must be assigned to one of the machines. We consider the *nonpreemptive* version of this problem (once a job is started, it must be processed without interruption until completion on the same machine). A *schedule* is an assignment of some job sequence to the machines. The *completion time* of a machine in some fixed schedule equals the total processing time of the jobs that are assigned to this machine.

The makespan of an algorithm H for a job sequence σ is denoted by $\mathrm{H}(\sigma)$ and we denote by $\mathrm{OPT}(\sigma)$ the makespan of the optimal schedule in the offline case. The performance of an online algorithm H on a job sequence σ is measured by the *approximation ratio*

$$\frac{\mathrm{H}(\sigma)}{\mathrm{OPT}(\sigma)}.$$

We call a deterministic online algorithm H *c-competitive* if, for each job sequence σ,

$$\mathrm{H}(\sigma) \leq c\mathrm{OPT}(\sigma).$$

The quality of an online algorithm H is measured by its *competitive ratio*

$$R^{\mathrm{H}}(m) = \sup_{\sigma \in X}\left(\frac{\mathrm{H}(\sigma)}{\mathrm{OPT}(\sigma)}\right),$$

where X is the set of all instances of the problem. List Scheduling (LS), which always assigns the current job to the machine that will complete it first, is a simple example of a nonpreemptive online algorithm.

A *randomized algorithm* is one which somehow bases its decision making on the outcome of random coin flips. Randomization has proven to be a useful tool in the design of algorithms, both online and offline. For randomized algorithms, since $\mathrm{H}(\sigma)$ is a random variable we use the ex-

A. Musitelli
Dipartimento di Matematica pura ed applicata, Università degli Studi di Padova, Via Trieste 63, 35121 Padova, Italy
e-mail: musitell_math@libero.it

J.-M. Nicoletti (✉)
ROSO/IMA, EPFL, Lausanne 1015, Switzerland
e-mail: jean-marc.nicoletti@epfl.ch

pected makespan $\mathbf{E}(\mathrm{H}(\sigma))$ in the definitions of approximation ratio, c-competitiveness, and competitive ratio (this corresponds to what is called the oblivious adversary). A *random schedule* is a set of assignments of some job sequence to the machines, where each assignment has a certain probability. Frequently, when working with randomized algorithms we will employ the term makespan when we mean the expected makespan.

In this paper, we consider the particular instance of the above problem when $s_j = 1$ for $j = 1, \ldots, m-1$ and $s_m = s \leq 1$. The machines running at speed 1 are said to be *fast* and are denoted by $F_1, \ldots, F_{m-1}$, while a machine having a speed s is *slow* and is denoted by S. When $s = 0$ and machine S has a job of size t, we assume that the completion time of this job on S is $\frac{t}{0} = \infty$. Since our model depends on the parameter s, for an online algorithm H, we denote by $R^{\mathrm{H}}(m, s)$ its competitive ratio. This model is also interesting in the sense that for a fixed $m \geq 3$, we analyze the system with m and $m-1$ identical machines together, by using one machine whose speed can vary continuously from 0 to 1.

The paper is organized as follows. Section 2 gives a brief history of the problem and describes the results. Section 3 analyzes the LS algorithm. In Sect. 4, a randomized algorithm called (m, s, α)-Linear Invariant is presented, where α is some parameter. We prove an upper bound for the competitive ratio of Linear Invariant for $m = 2, \ldots, 7$ and some number α. Finally, we study in more detail the case of $m = 3$ machines in Sect. 5. We provide a randomized algorithm called TwoGroups that outperforms LS for any s $(0 \leq s \leq 1)$.

2 Previous works and results

In the classical model with m identical machines, the LS algorithm is known to have a competitive ratio equal to $2 - \frac{1}{m}$. Moreover, it is the best online deterministic algorithm for $m = 2$ and $m = 3$. See, for example, (Hochbaum 1997). For $m = 2$ identical machines, a randomized algorithm achieving a competitive ratio of $\frac{4}{3}$ is presented by Bartal et al. (1995) and is proven to be the best online randomized algorithm in this model. A generalization of this algorithm called Linear Invariant has been proposed by Seiden (2000) for $m \geq 2$ identical machines. For small values of m, it is the best known randomized algorithm. Seiden (2000) proved some upper bounds on the competitive ratio of Linear Invariant for different values of m and some suitable parameter α, see Table 1. Linear Invariant competes better than any deterministic algorithm for $m = 3, 4, 5$ and better than the best known deterministic algorithms for $m = 6, 7$.

The model with two uniform machines (i.e., one machine with a reference speed 1 and the other one with a speed $s \geq 1$) has been extensively studied by Epstein et al. (2001).

Table 1 Upper bound (U.B.) on the competitive ratio of Linear Invariant for different values of m and α

m	α	U.B.
2	2	$\frac{4}{3}$
3	1.806865	1.55665
4	2.040258	1.65888
5	2.123240	1.73376
6	2.113960	1.78295
7	2.103110	1.81681

In this case, the problem is symmetric to the case of one machine with speed 1 and another of speed $s \leq 1$. By assuming $0 \leq s \leq 1$, they proved that LS is the best deterministic algorithm in this model and its competitive ratio is $c(2, s) = \min\{1 + s, \frac{2+s}{1+s}\}$.

Our model with $m - 1$ $(m \geq 2)$ uniform machines each with a processing speed of 1, and one machine with a speed s, $1 \leq s \leq 2$, has been studied by Cho and Sahni (1980). They showed that

$$R^{\mathrm{LS}}(m, s) \leq 1 + \frac{m-1}{m+s-1} \min\{2, s\} \leq 3 - \frac{4}{m+1},$$

and $R^{\mathrm{LS}}(m, 2) = 3 - \frac{4}{m+1}$. However, for fixed values of s $(1 < s < 2)$ the competitive ratio of LS is unknown.

Li and Shi (1998) proved that the competitive ratio of LS cannot be improved for $m = 2$ and $m = 3$ by any heuristic, and presented an algorithm which is $(\frac{3m-1}{m+1} - \epsilon_m)$-competitive, where ϵ_m is a positive number close to 0. Asymptotically their algorithm did not improve the heuristic LS since ϵ_m may tend to zero as m tends to infinity. For any heuristic H, they proved that $\sup_{s \geq 1} R^{\mathrm{H}}(m, s) \geq 2$. In the master thesis Kokash (2004) presents an online algorithm which is 2-competitive for an arbitrary $s > 1$ and $m \geq 3$. (See also Cheng et al. 2006.)

In this paper, we prove that in our model of m machines with $0 \leq s \leq 1$, $R^{\mathrm{LS}}(m, s)$ is equal to

$$c(m, s) = \min\left\{\frac{2m-3+s}{m-1}, \frac{2m-2+s}{m-1+s}\right\}.$$

We observe that $c(3, s) = \frac{3+s}{2}$ for $0 \leq s \leq s_{\max}(3)$ and $c(3, s) = \frac{4+s}{2+s}$ for $s_{\max}(3) \leq s \leq 1$, where $s_{\max}(3) = \frac{-3+\sqrt{17}}{2}$. For $0 \leq s \leq s_1 = \sqrt{2} - 1$ and $m = 3$, we show that LS is the best deterministic algorithm. See Fig. 1.

Furthermore, we describe a randomized heuristic called (m, s, α)-Linear Invariant generalizing the Linear Invariant algorithm. In our model of m machines with one of speed s $(0 \leq s \leq 1)$, the Linear Invariant algorithm used as if all machines were identical is proved to be $\frac{c}{s}$-competitive for m, α and c (U.B.) as in Table 1 (see Fig. 1 with $m = 3$ and $c = 1.55665$).

We also develop and analyze a randomized algorithm called TwoGroups on $m = 3$ machines. Its competitive ratio is proved to be $\frac{2}{3}(s + 2)$, for $0 \leq s \leq s_2$, and $\frac{2(s+3)}{2s+3}$ for

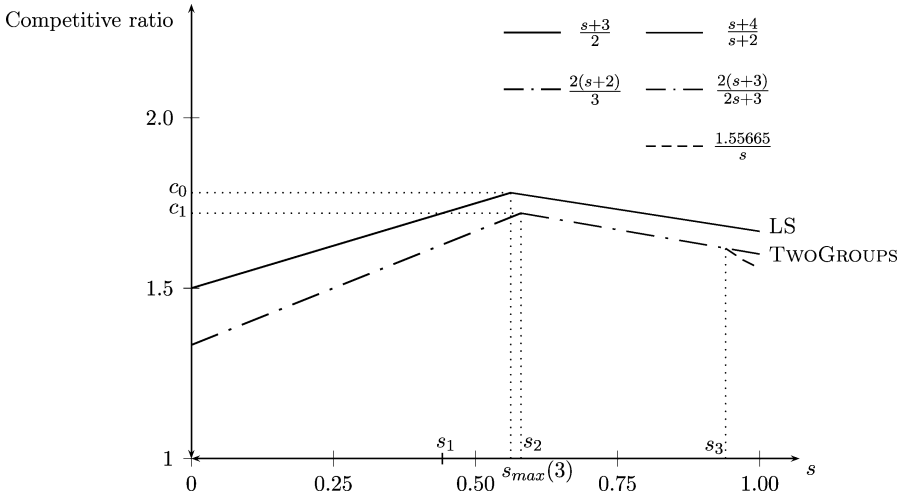

Fig. 1 Competitive ratio of LS and TwoGroups and an upper bound of $\frac{1.55665}{s}$ for the competitive ratio of Linear Invariant, where $m = 3$, $s_{\max}(3) = \frac{-3+\sqrt{17}}{2}$, $s_1 = \sqrt{2} - 1$, $s_2 = -1 + \frac{\sqrt{10}}{2}$, $s_3 = 0.96823$, $c_0 = \frac{3+\sqrt{17}}{4}$, $c_1 = \frac{2+\sqrt{10}}{3}$, and $\alpha = 1.806865$

$s_2 < s \leq 1$, where $s_2 = -1 + \frac{\sqrt{10}}{2}$. For $s = 0$, the TwoGroups algorithm is $\frac{4}{3}$-competitive, hence optimal by a theorem in Bartal et al. (1995). Moreover, for all $0 \leq s \leq 1$, it significantly outperforms LS. We observe that for $s_3 = 0.96823 \leq s \leq 1$, Linear Invariant (with $\alpha = 1.806865$) competes better than TwoGroups. See Fig. 1.

3 Analysis of LS

In this section, we treat in detail the greedy LS approximation algorithm. We also prove that LS is the best deterministic algorithm for case $m = 3$.

Let us give a formal definition of the LS algorithm. Suppose that LS has already scheduled a sequence of jobs and the total charges of jobs on machines $F_1, \ldots, F_{m-1}, S$ are $t_1, \ldots, t_{m-1}, t_m$, respectively. Suppose that a new job of size t arrives. If $\min_{1 \leq i \leq m-1} t_i + t < \frac{t_m + t}{s}$, then LS assigns this job on a fast machine with index $i_0 = \operatorname{argmin}_{1 \leq i \leq m-1} t_i$, otherwise LS puts it on machine S.

To study the worst case of an algorithm, we can imagine that some adversary tries to choose a job sequence for which the makespan computed by the algorithm is as large as possible. Testing the performance of an algorithm requires an estimate of its approximation ratio for a given job sequence. Since we do not know the exact value of the optimal makespan for each job sequence, we have to use some lower bounds on the optimal makespan. We show two of them.

Let σ be a job sequence. For each job σ_i, we denote by t_i its processing time. An optimal makespan cannot exceed the average machine load, that is,

$$\sum_{\sigma_i \in \sigma} \frac{t_i}{m - 1 + s} \leq \mathrm{OPT}(\sigma). \tag{1}$$

Furthermore, the optimal makespan is at least the running time of the largest job scheduled on one of the machines F_j $(1 \leq j \leq m-1)$:

$$\mathrm{OPT}(\sigma) \geq \max_{\sigma_i \in \sigma} t_i. \tag{2}$$

In the following proposition, we show that the competitive ratio of LS is smaller than or equal to

$$c(m, s) = \min\left\{\frac{2m - 3 + s}{m - 1}, \frac{2m - 2 + s}{m - 1 + s}\right\},$$

by using the lower bounds (1) and (2) for the optimal makespan. The proof is a direct generalization of case $m = 2$ studied in Epstein et al. (2001).

Proposition 1 *For any $0 \leq s \leq 1$, $m \geq 2$, LS is $c(m, s)$-competitive.*

Proof Let σ be a job sequence. To simplify notation, when a job σ_i is scheduled on machine F_j, we write $\sigma_i \in F_j$. Denote by σ_k the last job to complete processing. We may assume this is the last job of the sequence. If not, we can dispense with the remaining jobs and not reduce the approximation ratio. First we show that the LS algorithm is $\frac{2m-2+s}{m-1+s}$-competitive. We have

$$\begin{aligned}
\mathrm{LS}(\sigma) &\leq t_k + \sum_{\sigma_i \in F_1, i \neq k} t_i \qquad (3)\\
&= \sum_{\sigma_i \in \sigma} t_i - \sum_{j=2}^{m-1} \left(t_k + \sum_{\sigma_i \in F_j, i \neq k} t_i \right)\\
&\quad - \left(t_k + \sum_{\sigma_i \in S, i \neq k} t_i \right) + (m-1) t_k\\
&\leq (m - 1 + s)\mathrm{OPT}(\sigma) - (m-2)\mathrm{LS}(\sigma)\\
&\quad - s\mathrm{LS}(\sigma) + (m-1) t_k. \qquad (4)
\end{aligned}$$

Indeed, by definition of the algorithm, the inequality (3) is true. The inequality (4) follows from the lower bound (1) on the optimal makespan and the definition of the algorithm. Thus, using (2) and (4) we obtain that

$$\begin{aligned}&(m-1+s)\mathrm{LS}(\sigma)\\&\quad\le (m-1+s)\mathrm{OPT}(\sigma)+(m-1)\mathrm{OPT}(\sigma);\\&\mathrm{LS}(\sigma)\le\frac{2m-2+s}{m-1+s}\mathrm{OPT}(\sigma).\end{aligned}$$

Furthermore, LS is $\frac{2m-3+s}{m-1}$-competitive. To prove that, we define an algorithm called LSF. This algorithm schedules every job on machines $F_1,\dots,F_{m-1}$ like LS, ignoring machine S. Let $\sigma_{k'}$ be the last job of σ scheduled by LSF to complete processing. We may assume that $\sigma_{k'}$ is the last job of the sequence. Using the lower bound (1) and the definition of LSF, we have

$$\begin{aligned}\mathrm{LSF}(\sigma)&\le t_{k'}+\sum_{\sigma_i\in F_1,\,i\ne k'}t_i\\&=\sum_{\sigma_i\in\sigma}t_i-\sum_{j=2}^{m-1}\Bigg(t_{k'}+\sum_{\sigma_i\in F_j,\,i\ne k'}t_i\Bigg)+(m-2)t_{k'}\\&\le(m-1+s)\mathrm{OPT}(\sigma)-(m-2)\mathrm{LSF}(\sigma)\\&\quad+(m-2)t_{k'}.\end{aligned}$$

Again the explanations are the same as above. The preceding inequality together with the lower bound (2) yields

$$\mathrm{LSF}(\sigma)\le\frac{2m-3+s}{m-1}\mathrm{OPT}(\sigma).$$

So LSF is $\frac{2m-3+s}{m-1}$-competitive.

It remains to show that the competitive ratio of LS does not exceed that of LSF. Let σ^F be the subsequence of σ containing σ_k and the jobs assigned on the fast machines $F_1,\dots,F_{m-1}$ by LS. Using the definitions of LS and LSF, we have that $\mathrm{LS}(\sigma)\le\mathrm{LSF}(\sigma^F)$. Since $\mathrm{OPT}(\sigma)\ge\mathrm{OPT}(\sigma^F)$, it follows that $\frac{\mathrm{LS}(\sigma)}{\mathrm{OPT}(\sigma)}\le\frac{\mathrm{LSF}(\sigma^F)}{\mathrm{OPT}(\sigma^F)}$, which completes the proof. □

We note that $c(m,s)$ depends on the functions $s\mapsto\frac{2m-3+s}{m-1}$ and $s\mapsto\frac{2m-2+s}{m-1+s}$. Let

$$s_{\max}(m)=\operatorname*{argmax}_{0\le s\le 1}c(m,s).$$

A simple computation and function analysis yield that $c(m,s)=\frac{2m-3+s}{m-1}$ for $0\le s\le s_{\max}(m)$, $c(m,s)=\frac{2m-2+s}{m-1+s}$ for $s_{\max}(m)\le s\le 1$, and

$$s_{\max}(m)=\frac{-(2m-3)+\sqrt{(2m-3)^2+4(m-1)}}{2}.\tag{5}$$

See Fig. 1 for an illustration with $m=3$. The following lemma can easily be proven.

Lemma 1 *The function $s_{\max}$ of the variable m is decreasing over the domain $[2,\infty)$ and $\lim_{m\to\infty}s_{\max}(m)=\frac{1}{2}$.*

Now let us see the proof of the lower bound for the competitive ratio of LS. We shall require the following key lemma for proving the next proposition.

Lemma 2 *Let $m\ge 2$ and $0\le s\le s_{\max}(m)$. For any $0\le u\le m-1$, the inequality $\frac{u}{s}\ge s-1+2u$ holds.*

Proof If $s=0$, then the statement of the lemma is obvious.

Now suppose $0<s\le s_{\max}(m)$. Since $s>0$, the inequality $\frac{u}{s}\ge s-1+2u$ is equivalent to $s^2+s(2u-1)-u\le 0$. Observe that the left-hand side of the preceding inequality is a polynomial of degree 2 in s. One root of this polynomial is provided by the function $g(u):=\frac{-(2u-1)+\sqrt{(2u-1)^2+4u}}{2}$. So, for proving the lemma, it suffices to show that $0<s\le g(u)$ for any $0\le u\le m-1$. It can easily be checked that the function g is decreasing over the set of nonnegative reals. Therefore, $g(u)\ge g(m-1)=s_{\max}(m)$ for all $0\le u\le m-1$. Since $s_{\max}(m)\ge s$, this implies the lemma. □

We state a first lower bound to the competitive ratio of LS, in the model of m machines with $0\le s\le s_{\max}(m)$. In the proof of the next proposition, we define a job sequence σ having a last job of size $m-1$. Our goal is to force the situation when, after the assignment of σ without the last job, every fast machine has a load $m-2+s$, the slow machine is empty, and the optimal schedule of the sequence σ has a makespan $m-1$.

Proposition 2 *For any $m\ge 2$ and $0\le s\le s_{\max}(m)$, the competitive ratio of* LS *is at least $\frac{2m-3+s}{m-1}$.*

Proof In the following proof, we define a sequence σ consisting of $k+3$ lays of $m-1$ jobs of the same size t_j $(0\le j\le k+2)$ and one last large job of size t_{k+3}, where k is a nonnegative integer. We choose the numbers $t_0,\dots,t_{k+3}$ so that LS attributes one job of size t_j to each fast machine for all $0\le j\le k+2$ and the last job of size t_{k+3} is put on some fast machine, yielding a makespan of $\frac{2m-3+s-\epsilon}{m-1}$, where ϵ is arbitrarily small. For simplicity, we may have some jobs of zero length.

Let $m\ge 2$ and $0\le s\le s_{\max}(m)$. Let k be the highest nonnegative integer such that $1\le\frac{m-1}{2^k}\le 2$ and $\delta=(m-2)-\sum_{i=1}^{k}\frac{m-1}{2^i}$. (In the case where $m=2$, $k=0$ and so $\delta=0$.) Notice that $\delta=m-2-(m-1-\frac{m-1}{2^k})=-1+\frac{m-1}{2^k}$ and since $1\le\frac{m-1}{2^k}\le 2$, we have $0\le\delta\le 1$. We distinguish the three following cases.

(a) $0 \le s \le \frac{1}{2}$. We define $t_0 = 0$, $t_1 = \min\{s, \delta\}$, and $t_2 = \max\{s, \delta\}$.
(b) $s > \frac{1}{2}$ and $\delta \le \frac{1}{2}$. We define $t_0 = 0$, $t_1 = s - \frac{1}{2}$, and $t_2 = \delta + \frac{1}{2}$.
(c) $s > \frac{1}{2}$ and $\delta > \frac{1}{2}$. We define $t_0 = s - \frac{1}{2}$, $t_1 = -\frac{1}{2} + \frac{m-1}{2^{k+1}}$, and $t_2 = \frac{m-1}{2^{k+1}}$.

Denote $t_j = \frac{m-1}{2^{k+3-j}}$ for $j = 3, \ldots, k+3$ and $j_0 = \operatorname{argmin}_j\{t_j > 0\}$. For $j = 0, \ldots, k+2$ and $i = 1, \ldots, m-1$, $\sigma_{j(m-1)+i}$ has a processing time equal to $t_{j_0} - \epsilon$, if $j = j_0$, for some $0 < \epsilon < t_{j_0}$, and t_j otherwise. The job $\sigma_{(k+3)(m-1)+1}$ is of length $t_{k+3} = m - 1$. We state the following claim.

Claim *The following inequalities hold*:

$$t_{j_0} - \epsilon < \frac{t_{j_0} - \epsilon}{s}, \quad \textit{and}$$

$$\sum_{i=0}^{j} t_i - \epsilon < \frac{t_j}{s}, \quad \textit{for } j = j_0 + 1, \ldots, k+3.$$

Proof of the claim Since $s < 1$, it is clear that $t_{j_0} - \epsilon < \frac{t_{j_0} - \epsilon}{s}$. For any $j \le 2$, we deal with cases (a), (b), and (c) separately.

(a) If $t_1 \ne 0$, then $t_0 + t_1 + t_2 - \epsilon < 2t_2$. Since $s \le \frac{1}{2}$, it follows that $t_0 + t_1 + t_2 - \epsilon < \frac{t_2}{s}$.
(b) We have $t_1 + t_2 = s + \delta$. By Lemma 1, $s \le s_{\max}(2) = \frac{-1+\sqrt{5}}{2}$ and so $s(t_0 + t_1 + t_2) \le s\delta + s^2 \le (\frac{-1+\sqrt{5}}{2})\delta + (\frac{-1+\sqrt{5}}{2})^2 \le \delta + \frac{1}{2} = t_2$. Hence, $t_0 + t_1 + t_2 - \epsilon < \frac{t_2}{s}$.
(c) Since $\frac{1}{2} < \delta \le 1$ and $\delta = -1 + \frac{m-1}{2^k}$, it follows that $\frac{3}{2} \le \frac{m-1}{2^k} \le 2$. This implies

$$\frac{3}{4} \le t_2 \le 1 \quad \text{and} \quad \frac{1}{4} \le t_1 \le \frac{1}{2}. \tag{6}$$

The inequality $t_0 + t_1 < \frac{t_1}{s}$ is equivalent to $t_1 > \frac{s(s-\frac{1}{2})}{1-s}$, by replacing t_0 by $s - \frac{1}{2}$. Let $f(s') = \frac{s'(s'-\frac{1}{2})}{1-s'}$ for any $s' \in]\frac{1}{2}; \frac{-1+\sqrt{5}}{2}]$. The function f of the variable s' is increasing over the set $]\frac{1}{2}; \frac{-1+\sqrt{5}}{2}]$ and $\frac{1}{2} < s \le s_{\max}(2) = \frac{-1+\sqrt{5}}{2}$. Since $f(\frac{-1+\sqrt{5}}{2}) = \frac{7-3\sqrt{5}}{2(3-\sqrt{5})} < \frac{1}{4}$ and $t_1 \ge \frac{1}{4}$ by (6), it follows that $t_1 > f(s)$ and so $t_0 + t_1 - \epsilon < \frac{t_1}{s}$.

Secondly, since $t_0 + t_1 = s - 1 + t_2$, the inequality $t_0 + t_1 + t_2 \le \frac{t_2}{s}$ is equivalent to $t_2 \le \frac{s(1-s)}{2s-1}$. Let $g(s') = \frac{s'(1-s')}{2s'-1}$ for any $s' \in \mathbb{R}$. The function g of the variable s' is decreasing and as previously $s \le \frac{-1+\sqrt{5}}{2}$. Since $g(\frac{-1+\sqrt{5}}{2}) = 1$ and $t_2 \le 1$ by (6), we deduce that $t_2 \le g(s)$. Therefore, $t_0 + t_1 + t_2 - \epsilon < \frac{t_2}{s}$.

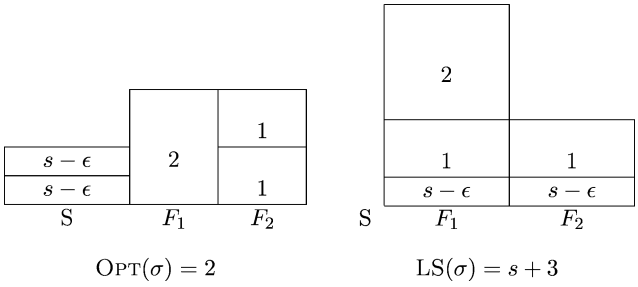

Fig. 2 The optimal schedule of the sequence $\sigma = (s-\epsilon, s-\epsilon, 1, 1, 2)$ and the schedule of σ output from LS in case $m = 3$ and $0 < s \le \frac{1}{2}$

Now observe that $t_0 + t_1 + t_2 = s - 1 + \frac{m-1}{2^k}$ in cases (a), (b), and (c). So for any $j \ge 3$, we have

$$\begin{aligned}\sum_{i=0}^{j} t_i - \epsilon &= s - 1 + \frac{m-1}{2^k} - \epsilon + \sum_{i=3}^{j} \frac{m-1}{2^{k+3-i}} \\ &= s - 1 - \epsilon + 2t_j \\ &< \frac{t_j}{s},\end{aligned}$$

where the last inequality follows from Lemma 2 and the fact that $\epsilon > 0$. This terminates the proof of the claim. □

Using the claim, the following can easily be proven by induction on j $(0 \le j \le k+2)$. LS with input $(\sigma_1, \ldots, \sigma_{(j+1)(m-1)})$ attributes one job of size t_i to each fast machine for all $0 \le i \le j$. Moreover, the last job of length t_{k+3} in σ is put on some fast machine. Since $t_0 + t_1 + t_2 = s - 1 + \frac{m-1}{2^k}$, we have that $\sum_{i=0}^{k+2} t_i = m - 2 + s$. From these arguments, we deduce that $\text{LS}(\sigma) = 2m - 3 + s - \epsilon$.

Let us show that the optimal makespan of σ is equal to $m - 1$. Let us consider case (a). We put $m - 1$ jobs of size s on machine S and $m - 1$ jobs of size δ on fast machine F_1. Then, dealing with the remaining jobs of lengths $\frac{m-1}{2}, \frac{m-1}{4}, \ldots, \frac{m-1}{2^k}$ in decreasing order, we assign jobs on F_2 until we obtain a load of $m - 1$; then we proceed in the same way for $F_3, \ldots, F_{m-1}$. Moreover, the total load is $\sum \sigma_i = (m-1)\sum_{i=0}^{k+2} t_i + t_{k+3} = (m-1)(m-2+s) + m - 1 = (m-1)^2 + (m-1)s$. So by scheduling any remaining job (if any exists) on F_1, we obtain a completion time of $m - 1$ on F_1. Thus, $\text{OPT}(\sigma) = m - 1$.

In cases (b) and (c), $m - 1$ jobs of size $t_1 = s - \frac{1}{2}$ and one of size $\frac{m-1}{2}$ are put on machine S. (Observe that in case (b)), if $m = 2$ then $t_2 = \frac{m-1}{2}$, and if $m = 3$ then $k = 1$ and $t_3 = \frac{m-1}{2}$.) In case (b) or (c), since $t_2 \le 1$ or $t_1 \le 1$, by (6) we assign $m - 1$ jobs of size t_2 or t_1, respectively, on the same fast machine. The remaining jobs are scheduled as in case (a). Thus, $\text{OPT}(\sigma) = m - 1$.

We conclude that the approximation ratio of LS for the sequence σ is $\frac{2m-3+s-\epsilon}{m-1}$. By letting ϵ tend to 0, it results that $R^{\text{LS}}(m, s) \ge \frac{2m-3+s}{m-1}$. □

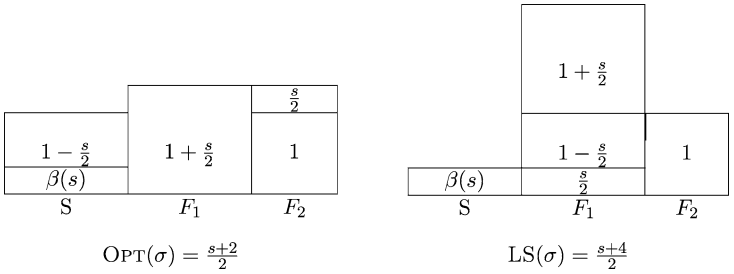

Fig. 3 The optimal schedule of the job sequence $\sigma=(\frac{s}{2},1,1-\frac{s}{2},\beta(s),1+\frac{s}{2})$ and the schedule of σ output from LS in case $m=3$ (and $k=2m-2$)

Proposition 3 *For any $m\geq 2$ and $s_{\max}(m)<s<1$, we have $R^{\text{LS}}(m,s)\geq\frac{2m-2+s}{m-1+s}$.*

Proof For $m=2$, a proof of the inequality $R^{\text{LS}}(2,s)\geq\frac{2+s}{1+s}$ appears in Epstein et al. (2001). We give here a more direct proof. Let $s_{\max}(2)<s<1$ and consider the sequence $\sigma=(1-s-\epsilon,s,s^2+s-1,1+s)$ where $0<\epsilon<1-s$. Note that $s^2+s-1>0$ since $s>s_{\max}(2)$. One can verify that $\text{OPT}(\sigma)=1+s$ and the approximation ratio of LS for this sequence is $\frac{2+s-\epsilon}{1+s}$. Hence, by letting ϵ tend to 0, it follows that $R^{\text{LS}}(2,s)\geq\frac{2+s}{1+s}$.

For $m\geq 3$, we will define a sequence $t=(t_1,\ldots,t_{2m-3})$ such that $t_1\leq t_2\leq\cdots\leq t_{m-2}\leq t_{m-1}=1$, $1\geq t_m\geq t_{m+1}\geq\cdots\geq t_{2m-3}$, and $t_i+t_{m-1+i}=1$ for $i=1,\ldots,m-2$. Then a job sequence σ containing the subsequence t is chosen in such a way that LS schedules t_i and t_{m-1+i} on the same machine for any i $(1\leq i\leq m-2)$. Moreover, $t_1,\ldots,t_{m-1}$ are scheduled on distinct fast machines. A last job of size $1+\frac{s}{m-1}$ in σ yields a makespan of $2+\frac{s}{m-1}$ for σ, while $\text{OPT}(\sigma)$ is proven to be equal to $1+\frac{s}{m-1}$. See Fig. 3 for an example with $m=3$.

Let $m\geq 3$ and $s_{\max}(m)<s<1$. We define $u_j=j\frac{s}{m-1}$ and $v_j=1-u_j$ for $j=1,\ldots,m-2$. Let $j_0=\lfloor\frac{m-1}{2s}\rfloor$. Since $s>\frac{1}{2}$, we observe that $j_0\leq m-2$. Let us denote by $(t_1,\ldots,t_{m-2})$ the sequence obtained by an increasing ordering of the numbers $u_1,\ldots,u_{j_0},v_{j_0+1},\ldots,v_{m-2}$. Let

$$t_{m-1}=1 \quad \text{and}$$

$$t_{m-1+i}=\begin{cases}v_j, & \text{if } t_i=u_j \text{ for some } j,\\ u_j, & \text{if } t_i=v_j \text{ for some } j,\end{cases}$$

for $i=1,\ldots,m-2$. We have that

$$t_i+t_{m-1+i}=1 \quad \text{for } i=1,\ldots,m-2. \tag{7}$$

Let $t=(t_1,\ldots,t_{2m-3})$, $\beta(s)=\frac{s^2}{m-1}+\frac{2m-3}{m-1}s-1$ ($\beta(s)>0$ since $s>s_{\max}(m)$) and consider the LS procedure with input t. Clearly, LS assigns each of the jobs $t_1,\ldots,t_{m-1}$ to fast machines (and no two jobs on the same machine) since $s<1$ and $t_1\leq t_2\leq\cdots\leq t_{m-1}$. It may occur that LS schedules a job t_i $(i\geq m)$ on the slow machine. If this happens, we denote by k the smallest index of a job scheduled on S and let $\sigma=(t_1,\ldots,t_{k-1},\beta(s),t_k,\ldots,t_{2m-3},1+\frac{s}{m-1})$; otherwise, let $k=2m-2$ and $\sigma=(t,\beta(s),1+\frac{s}{m-1})$. By construction $\sigma_k=\beta(s)$ (and $k\geq m$).

Claim 1 LS *with input $(t_1,\ldots,t_{k-1})$ produces a schedule with $k-m+1$ fast machines of load 1 and fast machines of load $t_{k-m+1},t_{k-m+2},\ldots,t_{m-2}$ (when $k<2m-2$), respectively.*

Proof of Claim 1 Let us prove by induction on l that for $l=m,\ldots,k$ LS with input $(t_1,\ldots,t_{l-1})$ produces a schedule with $l-m+1$ fast machines of load 1 and fast machines of load $t_{l-m+1},t_{l-m+2},\ldots,t_{m-2}$, respectively.

If $l=m$, then the proof has been given previously. Suppose that for some $m\leq l<k$, LS with input $(t_1,\ldots,t_{l-1})$ produces a schedule with $l-m+1$ fast machines of load 1 and fast machines of load $t_{l-m+1},t_{l-m+2},\ldots,t_{m-2}$, respectively. Since $t_{l-m+1}\leq t_{l-m+2}\leq\cdots\leq t_{m-2}\leq 1$, from the definition of k and LS it follows that t_l is assigned on a fast machine of load t_{l-m+1}. Using (7), we deduce that the claim is true for $l+1$. This concludes the proof of Claim 1. □

Suppose that $k<2m-2$. By Claim 1, in the schedule obtained by LS with input $(t_1,\ldots,t_{k-1})$, the smallest load of a machine is equal to $t_{k-(m-1)}$; and since t_k is assigned on machine S, by the definition of LS, we have

$$t_{k-(m-1)}+t_k\geq\frac{t_k}{s}. \tag{8}$$

Now, let us consider the LS procedure with input σ. Since $(\sigma_1,\ldots,\sigma_{k-1})=(t_1,\ldots,t_{k-1})$, Claim 1 holds for the sequence $(\sigma_1,\ldots,\sigma_{k-1})$ (instead of $(t_1,\ldots,t_{k-1})$).

Claim 2 *The job σ_k is scheduled on machine S.*

Proof of Claim 2 If $k=2m-2$, then by Claim 1 LS with input $(\sigma_1,\ldots,\sigma_{k-1})$ outputs a schedule all of whose fast machines have a load 1. Since $\beta(s)+1>\frac{\beta(s)}{s}$, the job $\sigma_k=\beta(s)$ is assigned on S. Now suppose $k<2m-2$. By Claim 1, the current smallest load is equal to $t_{k-(m-1)}$. So, for showing that σ_k is put on S by LS, it suffices to prove the inequality

$$t_{k-(m-1)}+\sigma_k>\frac{\sigma_k}{s}. \tag{9}$$

Using (7) and (8), we have that $\frac{t_k}{s} \leq 1$. Thus, $t_{k-(m-1)} + \sigma_k = 1 - t_k + \beta(s) \geq 1 - s + \beta(s)$ (where the equality follows from (7)). So, for proving (9), it suffices to show that $1 - s + \beta(s) > \frac{\beta(s)}{s}$, or equivalently $s^3 + (m-3)s^2 - (2m-3)s + (m-1) > 0$ by using the expression of $\beta(s)$ in the function of s. Let $h(s') = s'^3 + s'^2(m-3) - (2m-3)s' + (m-1)$ for any $s' \in]0;1[$. One can verify that the slope of the function h on its domain $]0;1[$ is negative and so $h(s) > h(1) = 0$ since $0 < s < 1$. This proves inequality (9) and terminates the proof of Claim 2. □

In what follows, by assuming $k < 2m-2$, we show that neither of the jobs $\sigma_{k+1}, \ldots, \sigma_{2m-2}$ is scheduled on machine S whose current load is equal to $\beta(s)$. Notice that each of the jobs $\sigma_{k+1}, \ldots, \sigma_{2m-2}$ has a length u_j, for some $j \geq j_0 + 1$, or v_j for some $j \leq j_0$. Since $u_{j_0+1} \leq u_{j_0+2} \leq \cdots \leq u_{m-2}$ and $v_1 \geq v_2 \geq \cdots \geq v_{j_0}$, it results that

$$\min\{\sigma_{k+1}, \ldots, \sigma_{2m-2}\} \geq \min\{v_{j_0}, u_{j_0+1}\}. \tag{10}$$

Claim 3 *We have that* $\frac{\beta(s)+v_{j_0}}{s} > 1$ *and* $\frac{\beta(s)+u_{j_0+1}}{s} > 1$.

Proof of Claim 3 Using the expressions $\beta(s) = \frac{s^2}{m-1} + \frac{2m-3}{m-1}s - 1$ and $v_{j_0} = 1 - \lfloor \frac{m-1}{2s} \rfloor \frac{s}{m-1}$, the inequality $\frac{\beta(s)+v_{j_0}}{s} > 1$ is equivalent to $s + m - 2 - \lfloor \frac{m-1}{2s} \rfloor > 0$. Since $s > \frac{1}{2}$, $\frac{m-1}{2s} < m-1$ and so $\lfloor \frac{m-1}{2s} \rfloor \leq m-2$, which implies that $s + m - 2 - \lfloor \frac{m-1}{2s} \rfloor > 0$. Therefore, $\frac{\beta(s)+v_{j_0}}{s} > 1$.

Similarly, since $u_{j_0+1} = \lfloor \frac{m-1}{2s} \rfloor \frac{s}{m-1} + \frac{s}{m-1}$, we have that

$$\frac{\beta(s) + u_{j_0+1}}{s} > 1 \iff$$
$$s^2 + \left((m-1) + \left\lfloor \frac{m-1}{2s} \right\rfloor\right)s - (m-1) > 0. \tag{11}$$

Let p be an integer such that $\lfloor \frac{m-1}{2s} \rfloor = m - p$. Clearly, $p \geq 2$ since $s > \frac{1}{2}$. If $p = 2$, then by (11) and since $s > s_{\max}(m)$ it follows that $\frac{\beta(s)+u_{j_0+1}}{s} > 1$.

Now assume $p \geq 3$. We have $\frac{m-1}{2s} < m - p + 1$. Hence, $s > \frac{m-1}{2(m-p+1)}$. The preceding inequality together with the expressions $\lfloor \frac{m-1}{2s} \rfloor = m - p$ and $p \geq 3$ imply that

$$\begin{aligned}
&\left((m-1) + \left\lfloor \frac{m-1}{2s} \right\rfloor\right)s - (m-1) \\
&= (2m - p - 1)s - (m-1) \\
&\geq (2m - p - 1)s + (3-p)s - (m-1) \\
&> 0.
\end{aligned}$$

Using (11), it follows that $\frac{\beta(s)+u_{j_0+1}}{s} > 1$. This concludes the proof of Claim 3. □

Using (10) and Claim 3, in the same way as for proving Claim 1, it can be proven that LS with input $(\sigma_1, \ldots, \sigma_{2m-2})$ produces a schedule with $m-1$ fast machines of load 1 and a slow machine of load $\beta(s)$. The last job of length $1 + \frac{s}{m-1}$ can be scheduled on any machine which gives a makespan equal to $2 + \frac{s}{m-1}$.

Let us describe an optimal schedule as follows. Machine F_j receives the jobs v_j and u_{j+1} for $j = 1, \ldots, m-3$ and F_{m-2} has the jobs $\sigma_{m-1} = 1$ and u_1. Then F_{m-1} obtains the job $\sigma_{2m-1} = 1 + \frac{s}{m-1}$ and the load of machine S is equal to $\sigma_k + v_{m-2} = \frac{s^2}{m-1} + s$ (since $\sigma_k = \beta(s)$). Thus, the optimal makespan is $1 + \frac{s}{m-1}$ and we obtain an approximation ratio of $\frac{2m-2+s}{m-1+s}$. □

Theorem 1 *The competitive ratio of* LS *is* $c(m,s)$ *for all* $0 \leq s \leq 1$ *and* $m \geq 2$.

Proof The proof follows from Propositions 1, 2, and 3. □

We conclude this section by proving that in the model of 3 machines, the competitive ratio of any deterministic algorithm is at least $\frac{3}{2} + \frac{s}{2} = c(3,s)$ for all $0 \leq s \leq s_1 = \sqrt{2} - 1$. Using Theorem 1, this implies that LS is the best deterministic algorithm for all $0 \leq s \leq s_1$ and $m = 3$ machines. (See Fig. 1.)

Proposition 4 *For all* $0 \leq s \leq s_1$ *and* $m = 3$, *the competitive ratio of any deterministic algorithm is at least* $\frac{3}{2} + \frac{s}{2}$.

Proof Let $0 \leq s \leq s_1$ and $\sigma = (\sigma_1, \ldots, \sigma_5)$ be a sequence of five jobs whose processing times are s for σ_1 and σ_2, 1 for σ_3 and σ_4, 2 for σ_5. Let H be a deterministic algorithm.

If H schedules σ_1 on S, we obtain an approximation ratio of $\frac{1}{s} \geq \frac{3}{2} + \frac{s}{2}$ (since $s \leq s_1 = \sqrt{2} - 1$ and $\textsc{Opt}((\sigma_1)) = s$). So we may suppose that σ_1 is scheduled on F_1. If H assigns the second job σ_2 to S, again the approximation ratio is $\frac{1}{s}$, while if it is scheduled on F_1, the approximation ratio equals $2 \geq \frac{3}{2} + \frac{s}{2}$ since $s \leq 1$. So we may assume that σ_2 is on F_2. If the third job σ_3 is scheduled on S, the approximation ratio is again $\frac{1}{s}$. Hence, we may suppose that it is scheduled on F_1. If σ_4 is assigned to S, we obtain an approximation ratio of $\frac{1}{s(s+1)} \geq \frac{3}{2} + \frac{s}{2}$ (since $s \leq s_1$ and $\textsc{Opt}((\sigma_1, \ldots, \sigma_4)) = s + 1$). And if σ_4 is scheduled on F_1, we have an approximation ratio of $\frac{2+s}{1+s} \geq \frac{3}{2} + \frac{s}{2}$ for $s \leq \sqrt{2} - 1 = s_1$. So we may assume that σ_4 is on F_2. Finally, if the last job is scheduled on S, the approximation ratio is $\frac{1}{s}$, so we may suppose that σ_5 is scheduled either on F_1 or F_2. Then the approximation ratio is $\frac{3}{2} + \frac{s}{2}$ (since $\textsc{Opt}(\sigma) = 2$). See Fig. 2 (with $\epsilon = 0$). Thus, we have shown that for any deterministic algorithm, the competitive ratio is at least $\frac{3}{2} + \frac{s}{2}$ for $s \leq s_1$. □

4 Randomized scheduling on m machines

We describe a direct generalization of Seiden's Linear Invariant algorithm (Seiden 2000) that we call (m, s, α)-Linear Invariant for our model of m machines with one of speed s, where α is a parameter. For $s = 1$, (m, s, α)-Linear Invariant is Seiden's Linear Invariant algorithm. We begin by defining notation and then explain how (m, s, α)-Linear Invariant schedules each job.

Given a schedule, we call the machines with the greatest and the smallest completion time the *tall* and *short* machine, respectively. If a new job σ_i arrives, we define the σ_i-*second shortest* and σ_i-*short* machine as the machines whose completion time of the current load and job σ_i are the second shortest and shortest one, respectively. The (m, s, α)-Linear Invariant algorithm places each new job σ_i on the σ_i-short or σ_i-second shortest machine.

After k jobs are scheduled, there are $K = 2^k$ schedules $\Pi_1, \ldots, \Pi_K$ that the algorithm might have chosen. The algorithm keeps track of each of these schedules, and of the probability with which it has chosen each schedule. In schedule Π_j, we let x_j and z_j be the completion time of the tall and short machine, respectively. We let p_j be the probability of Π_j (we describe how the p_j's are chosen later). We define

$$x = \sum_j p_j x_j, \qquad z = \sum_j p_j z_j.$$

The expected makespan is x.

The (m, s, α)-Linear Invariant algorithm schedules each new job σ_i on the σ_i-short or σ_i-second shortest machines, maintaining the invariant

$$x \geq \alpha z \tag{12}$$

where α is a real constant greater than 1. The intuition behind this invariant is that the algorithm should keep the load on the short machine low in order to "be prepared" for the possibility of a large job arriving.

This is accomplished as follows. When a new job σ_i arrives, let z_{short} and x_{short} be the new values of z and x, respectively, if σ_i is scheduled on the σ_i-short machine in every schedule. Let $z_{2\text{nd}}$ and $x_{2\text{nd}}$ be the new values of z and x if σ_i is scheduled on the σ_i-second shortest machine in every schedule. If $x_{\text{short}} \geq \alpha z_{\text{short}}$, then σ_i is scheduled on the σ_i-short machine in every schedule. Otherwise, note that $z_{2\text{nd}} = z$ and that $x_{2\text{nd}} \geq x$. Therefore, $x_{2\text{nd}} \geq \alpha z_{2\text{nd}}$. Let

$$p = \frac{x_{2\text{nd}} - \alpha z_{2\text{nd}}}{x_{2\text{nd}} - \alpha z_{2\text{nd}} - (x_{\text{short}} - \alpha z_{\text{short}})}.$$

It is not hard to see that $p \in [0, 1]$. We schedule σ_i on the σ_i-short machine with probability p, and on the σ_i-second shortest machine with probability $1 - p$. Thus, the invariant is maintained at each step.

The number of schedules which are remembered can be reduced to k, using a technique due to Bartal et al. (1995). This yields an algorithm which schedules all jobs in $O(k^2 \log m)$ time. Seiden has shown that the number of schedules may be reduced to $O(1)$ at the expense of an increased competitive ratio (Borodin et al. 1995; Sgall 1994). This yields a total running time of $O(k \log m)$.

Note that (m, s, α)-Linear Invariant with $m = 2$, $s = 1$, and $\alpha = 2$ is the 2-machine algorithm of Bartal et al. (1995).

In our model of m uniform machines with one of speed s, we may use the Linear Invariant algorithm as if the slow machine were identical to the fast machines. We deduce the following upper bound on the competitive ratio of this algorithm.

Theorem 2 *The Linear Invariant algorithm is $\frac{c}{s}$-competitive with α and c (U.B.) in Table 1, for $m = 2, \ldots, 7$.*

Proof Let $2 \leq m \leq 7$, α and c (U.B.) be the numbers in some row of Table 1. In the model of m identical machines, the following lower bounds on the optimal (offline) makespan are well known:

- The total amount of processing divided by m.
- The size of any job.
- Twice the size of the smallest job in a set of $m + 1$ jobs.

For a sequence σ, let us denote by *lower_bound*(σ) the maximum number among the three previous lower bounds. Seiden (2000) has proven that

$$\text{LINEAR INVARIANT}(\sigma) \leq c \cdot \textit{lower\_bound}(\sigma).$$

Now let us consider our model of m uniform machines with one of speed s. Let σ be a job sequence. Let us schedule σ as if the slow machine were identical to machines $F_1, \ldots, F_{m-1}$, by using Linear Invariant. Let x be the obtained makespan. From the previous inequality, it follows that

$$x \leq \frac{c}{s}\, \textit{lower\_bound}(\sigma) \leq \frac{c}{s}\, \text{OPT}(\sigma). \qquad \square$$

5 Randomized scheduling on 3 machines

In this section, we will concentrate on the case $m = 3$ machines for our model. An analytical proof of an upper bound on the competitive ratio of $(3, s, \alpha)$-Linear Invariant, for some fixed α and s $(0 \leq s \leq 1)$, seems to be difficult. So, we develop and analyze a randomized algorithm called TwoGroups which is proven to have a competitive ratio of

$$c_2 = \begin{cases} \frac{2}{3}(s+2), & \text{if } 0 \leq s \leq -1 + \frac{\sqrt{10}}{2}, \\ \frac{2(s+3)}{2s+3}, & \text{if } -1 + \frac{\sqrt{10}}{2} < s \leq 1. \end{cases}$$

For values of s close to 1, we may apply Linear Invariant with $\alpha = 1.806865$ (see Table 1), as if the slow machine were identical to the others.

Let us describe the TwoGroups algorithm. The three machines are divided into two groups. The first one contains machine S and the second one is composed of machines F_1 and F_2. Suppose that a job sequence $(\sigma_1, \ldots, \sigma_{k-1})$ has been assigned and a new job σ_k, with processing time t_k, arrives. TwoGroups computes the total load of jobs scheduled on S, say c, and the total load of jobs assigned to $\{F_1, F_2\}$ divided by 2, say d. Let $\alpha(s) = \frac{4s^2+8s-6}{s^2+6s}$.

Procedure TwoGroups
Input: A random schedule of $(\sigma_1, \ldots, \sigma_{k-1})$ and a job σ_k.
Output: A random schedule of $(\sigma_1, \ldots, \sigma_k)$.

For $s > -1 + \frac{\sqrt{10}}{2}$, if $\frac{c}{s} \leq \alpha(s)d$, then the job σ_k is scheduled on machine S.

Otherwise, σ_k is always scheduled on group $\{F_1, F_2\}$, by using $(2, 1, 2)$-Linear Invariant.

For $s > -1 + \frac{\sqrt{10}}{2}$, the TwoGroups algorithm maintains machine S with a large load. This enables us to schedule a big job on $\{F_1, F_2\}$ without increasing the makespan too much. We denote M the makespan of TwoGroups and OPT the optimal makespan (in the offline case) for the job sequence $\sigma := (\sigma_1, \ldots, \sigma_k)$. Let d' be one-half the total load of jobs assigned to group $\{F_1, F_2\}$ and c' the total load of S after scheduling σ. We will use the same lower bounds on the optimal makespan as in the deterministic case.

We recall here the definitions and basic facts about $(2, 1, 2)$-Linear Invariant analyzed in Bartal et al. (1995). We distinguish the machines of group $\{F_1, F_2\}$ according to their load. Indeed, we denote the *tall* and *short* machines of group $\{F_1, F_2\}$ to be the machines with the greatest and smallest loads to date, respectively. We denote by y and z the expected load of the tall and the short machine, respectively, in group $\{F_1, F_2\}$. Then the algorithm from Bartal et al. (1995) maintains the following *invariant*:

$$y \geq 2z. \tag{13}$$

We say that the invariant is *tight* if $y = 2z$. For an example of the scheduling of a job sequence by $(2, 1, 2)$-Linear Invariant, see Fig. 4.

A job σ_i is said to be *big* if it is scheduled on $\{F_1, F_2\}$ and if it is at least as large as the sum of all previous jobs assigned to $\{F_1, F_2\}$. In Bartal et al. (1995), it has been shown that we can divide the times where the invariant is not tight into phases. A job starts a new phase if it is big, and after the job is added, the invariant is not tight. The phase consists of that job and all successive jobs up to but not including the first job which starts a new phase or makes the invariant tight. Moreover, during a phase each job is scheduled

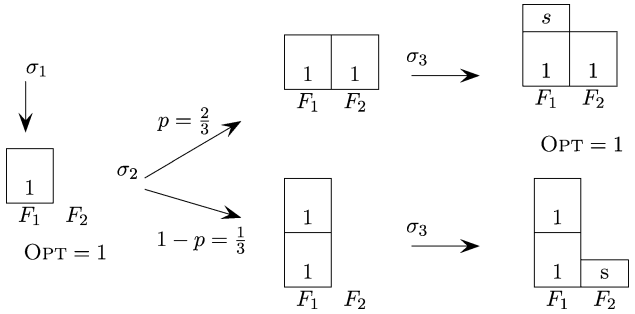

Fig. 4 Job sequence $\sigma = (1, 1, s)$ scheduled by $(2, 1, 2)$-Linear Invariant on $\{F_1, F_2\}$, where p is the probability of the schedule with σ_2 on F_2

deterministically on the smallest machine in each schedule. Finally, it has been shown that after the first job in a phase is scheduled, the expected makespan does not increase during the phase and is at most $\frac{4}{3}$ times the size of the latest big job.

After k jobs have been scheduled on group $\{F_1, F_2\}$, we may assume that there are k current schedules denoted by $\Pi_1, \ldots, \Pi_k$. In a schedule Π_i, we let y_i and z_i be the heights of the tall and short machines, respectively, of group $\{F_1, F_2\}$ and let p_i be the probability of Π_i. Let $y = \sum_{i=1}^{k} p_i y_i$ and $z = \sum_{i=1}^{k} p_i z_i$. If the invariant is tight after the assignment of σ_k, then since $y = 2z$ and $y + z = 2d'$ we have

$$y = \frac{4}{3}d'. \tag{14}$$

Before studying the competitiveness, we state a very simple lemma.

Lemma 3 *Let $f : \mathbb{R} \to \mathbb{R}$ be a convex function, $a, b \in \mathbb{R}$, $a \leq y_i \leq b$ and $0 \leq p_i, p \leq 1$ for $i = 1, \ldots, k$ $(k \in \mathbb{N})$ such that $\sum_{i=1}^{k} p_i = 1$ and $pa + (1-p)b = \sum_{i=1}^{k} p_i y_i$. Then*

$$\sum_{i=1}^{k} p_i f(y_i) \leq pf(a) + (1-p)f(b).$$

Proof For each i, let $0 \leq q_i \leq 1$ such that $y_i = q_i a + (1 - q_i)b$.

Thus, $pa + (1-p)b = \sum_{i=1}^{k} p_i q_i a + \sum_{i=1}^{k} p_i (1 - q_i)b = \sum_{i=1}^{k} p_i q_i a + (1 - \sum_{i=1}^{k} p_i q_i)b$. Since any number between a and b can be written in a unique way as $\lambda a + (1-\lambda)b$ where $0 \leq \lambda \leq 1$, we deduce that $p = \sum_{i=1}^{k} p_i q_i$ and $1 - p = \sum_{i=1}^{k} p_i (1 - q_i)$. By using the convexity of f, it follows that

$$\sum_{i=1}^{k} p_i f(y_i) = \sum_{i=1}^{k} p_i f\big(q_i a + (1 - q_i)b\big)$$

$$\leq \sum_{i=1}^{k} p_i\big(q_i f(a) + (1-q_i) f(b)\big)$$
$$= pf(a) + (1-p) f(b). \qquad \square$$

For the analysis of the competitiveness, we consider a sequence σ of k jobs, c, c', d, d', y, and z as defined above. First we study the case $0 \leq s \leq -1 + \frac{\sqrt{10}}{2}$. Observe that by the definition of the algorithm we do not have to consider machine S. Thus, we only focus on what happens in group $\{F_1, F_2\}$. Using some ideas of Bartal et al. (1995), we prove the upper and lower bounds on the competitive ratio of TwoGroups.

Proposition 5 *For $0 \leq s \leq -1 + \frac{\sqrt{10}}{2}$, the approximation ratio of TwoGroups for σ is upper bounded by $\frac{2}{3}(s+2)$.*

Proof Assume that the sequence σ has been scheduled. Suppose first that the invariant is not tight. Consider the latest big job, denoted by J. Let $\bar{y}$ and $\bar{z}$ be the expected load of the tall and short machine respectively just before the assignment of J. We have the invariant $\bar{y} \geq 2\bar{z}$ and $J \geq \bar{y} + \bar{z}$. So $J \geq 3\bar{z}$. By the definition of $(2,1,2)$-Linear Invariant, J is deterministically assigned to the short machine and the makespan does not change during the phase starting by J. Using (2) and the inequality $\frac{\bar{z}}{J} \leq \frac{1}{3}$, we deduce that $\frac{M}{\mathrm{OPT}} = \frac{J+\bar{z}}{\mathrm{OPT}} \leq \frac{J+\bar{z}}{J} \leq \frac{4}{3} \leq \frac{2}{3}(s+2)$.

Now suppose that the invariant is tight. We have $M = y$. By (14) $M = \frac{4}{3}d'$. From (1), it follows that

$$\frac{M}{\mathrm{OPT}} \leq \frac{\frac{4}{3}d'}{\frac{2d'}{s+2}} = \frac{2}{3}(s+2). \qquad \square$$

Proposition 6 *For $0 \leq s \leq -1 + \frac{\sqrt{10}}{2}$, the competitive ratio of TwoGroups is at least $\frac{2}{3}(s+2)$.*

Proof To show the proposition we consider the following job sequence of three jobs σ_1, σ_2, and σ_3 whose processing times are 1 for σ_1 and σ_2, s for σ_3. See Fig. 4.

The TwoGroups algorithm schedules σ_1 (w.l.o.g.) on F_1. If the second job is scheduled on the small machine, i.e., F_2, then the invariant (13) is not maintained. So in order to obtain a tight invariant we have to tentatively move the job σ_2 to the tall machine F_1. Let p be the probability of the schedule with σ_2 on the less loaded machine. We determine that $p = \frac{2}{3}$. If the third job is deterministically scheduled on the less loaded machine in both schedules, then the invariant (13) remains. So the job σ_3 is permanently assigned to the less loaded machine in each schedule. We compute an expected makespan of $\frac{2}{3}(1+s) + \frac{1}{3}2 = \frac{2}{3}(s+2)$. Since $\mathrm{OPT}((\sigma_1, \sigma_2, \sigma_3)) = 1$, we obtain an approximation ratio of $\frac{2}{3}(s+2)$. $\square$

Theorem 3 *For $0 \leq s \leq -1 + \frac{\sqrt{10}}{2}$, the competitive ratio of TwoGroups is equal to $\frac{2}{3}(s+2)$.*

Proof The proof is a direct consequence of Propositions 5 and 6. $\square$

Now we consider the case $-1 + \frac{\sqrt{10}}{2} < s \leq 1$. We distinguish three cases according to the values that $\frac{c'}{s}$ can take.

Case 1: $\frac{c'}{s} \leq d'$
Case 2: $d' \leq \frac{c'}{s} \leq 2d'$
Case 3: $2d' \leq \frac{c'}{s}$.

For each case, we compute an upper bound on the approximation ratio. We will choose $\alpha(s)$ so that the upper bounds in cases 1 and 3 fit. In case 2, we will obtain an upper bound on the approximation ratio smaller than those of cases 1 and 3.

Lemma 4 *In case 1, the approximation ratio of TwoGroups is upper bounded by $\frac{2}{3}\frac{\alpha(s)s+2s+4}{\alpha(s)s+2}$.*

Proof Since $\frac{c'}{s} \leq d'$, the makespan is given by a machine of group $\{F_1, F_2\}$ in any schedule Π_i. If the invariant in group $\{F_1, F_2\}$ is not tight, we can show as in Proposition 5 that the expected makespan M is upper bounded by $\frac{4}{3}\mathrm{OPT}$.

Now suppose that the invariant is tight. Using (14), we deduce that $M = \frac{4}{3}d'$. W.l.o.g., we may suppose that the job σ_k is on group $\{F_1, F_2\}$. Indeed, if σ_k is assigned to machine S, then since the makespan does not increase and the optimum does not decrease, the inequality $\frac{\mathrm{TWOGROUPS}(\sigma)}{\mathrm{OPT}(\sigma)} \leq \frac{\mathrm{TWOGROUPS}((\sigma_1,\ldots,\sigma_{k-1}))}{\mathrm{OPT}((\sigma_1,\ldots,\sigma_{k-1}))}$ is true. Thus, $2d' = 2d + t_k$. We have

$$\frac{M}{\mathrm{OPT}} \leq \frac{\frac{4}{3}d'}{\max\{t_k, \frac{2d+t_k+c}{s+2}\}} \tag{15}$$
$$\leq \frac{2}{3}\frac{2d+t_k}{\max\{t_k, \frac{2d+t_k+\alpha(s)sd}{s+2}\}}. \tag{16}$$

The inequality (15) uses lower bounds (1) and (2) for the optimum and the expression $M = \frac{4}{3}d'$, while inequality (16) follows from the equality $2d' = 2d + t_k$ and the definition of the algorithm (by assumption job σ_k is assigned to group $\{F_1, F_2\}$ and so $\frac{c}{s} > \alpha(s)d$).

If $t_k \leq \frac{2d+t_k+\alpha(s)sd}{s+2}$, then $\frac{d}{2d+t_k} \geq \frac{s+1}{\alpha(s)s+2s+4}$; using (16) we deduce

$$\frac{M}{\mathrm{OPT}} \leq \frac{2}{3}\frac{2d+t_k}{\frac{2d+t_k+\alpha(s)sd}{s+2}}$$
$$= \frac{2}{3}(s+2)\frac{1}{1+\frac{\alpha(s)sd}{2d+t_k}}$$

$$\leq \frac{2}{3}(s+2)\frac{1}{1+\frac{\alpha(s)s(s+1)}{\alpha(s)s+2s+4}}$$
$$= \frac{2}{3}\frac{\alpha(s)s+2s+4}{\alpha(s)s+2}.$$

Otherwise, $t_k \geq \frac{2d+t_k+\alpha(s)sd}{s+2}$. Then $\frac{d}{t_k} \leq \frac{s+1}{\alpha(s)s+2}$ and using (16) we also obtain the inequality $\frac{M}{\text{OPT}} \leq \frac{2}{3}\frac{\alpha(s)s+2s+4}{\alpha(s)s+2}$. □

Lemma 5 *In case* 2, $\frac{s+2}{s+1}$ *is an upper bound for the approximation ratio of TwoGroups.*

Proof Suppose first that the invariant is tight. Using (14), we have

$$M = \sum_{i=1}^{k} p_i \max\left\{\frac{c'}{s}, y_i\right\}$$
$$= \sum_{i=1}^{k} p_i \left(\frac{y_i + \frac{c'}{s} + |y_i - \frac{c'}{s}|}{2}\right)$$
$$= \frac{2}{3}d' + \frac{c'}{2s} + \frac{1}{2}\sum_{i=1}^{k} p_i \left|y_i - \frac{c'}{s}\right|. \tag{17}$$

The first equality in (17) gives the expected makespan of the algorithm after k jobs have been assigned. This value depends on different schedules for which the makespan can be determined either by machine S or a machine of group $\{F_1, F_2\}$.

Let $f(u) = |u - \frac{c'}{s}|$. Clearly f is a convex function and $d' \leq y_i \leq 2d'$ for $i = 1, \ldots, k$. Since the invariant is tight, we have $\sum_{i=1}^{k} p_i y_i = \frac{4}{3}d'$ and $\frac{4}{3}d' = \frac{2}{3}d' + \frac{1}{3}(2d')$. So we can apply Lemma 3 with $a = d'$, $b = 2d'$, and $p = \frac{2}{3}$. We obtain

$$\sum_{i=1}^{k} p_i f(y_i) \leq \frac{2}{3}f(d') + \frac{1}{3}f(2d'). \tag{18}$$

Thus, by (17) and (18), we deduce

$$M \leq \frac{2}{3}d' + \frac{c'}{2s} + \frac{1}{2}\left[\frac{2}{3}\left(\frac{c'}{s} - d'\right) + \frac{1}{3}\left(2d' - \frac{c'}{s}\right)\right] \tag{19}$$
$$= \frac{2}{3}\left(\frac{c'}{s} + d'\right). \tag{20}$$

Now suppose that the invariant is not tight. Denote by J the size of the last job which is big.

If $\frac{c'}{s} \leq J$, then the makespan is determined by what happens in group $\{F_1, F_2\}$, and as in the proof of Proposition 5, one can prove that it is upper bounded by $\frac{4}{3}\text{OPT}$.

Otherwise, let β, γ and $0 \leq p \leq 1$ be such that βJ is the total load of group $\{F_1, F_2\}$, $\sum_{i=1}^{k} p_i y_i = \gamma J$ and $pJ + (1-p)\beta J = \gamma J$. Note that $\frac{\beta J}{2} = d'$. Since $\frac{c'}{s} > J$ and $\beta J \geq \frac{c'}{s}$ (by assumption of case 2), it follows that $\beta > 1$. We have

$$p = \frac{\beta - \gamma}{\beta - 1}. \tag{21}$$

Before the assignment of J, the total load of group $\{F_1, F_2\}$ was $(\beta - 1)J$. From the invariant (13), the expected load of the smallest machine was at most $\frac{(\beta-1)}{3}J$. The fact that J is scheduled deterministically on the smallest machine implies that $\gamma J \leq \frac{\beta-1}{3}J + J = \frac{\beta+2}{3}J$. Furthermore, γ is clearly greater than 1. So

$$1 \leq \gamma \leq \frac{\beta+2}{3}. \tag{22}$$

Furthermore,

$$J \leq y_i \leq \beta J \quad \forall i \quad \text{and} \quad \sum_{i=1}^{k} p_i y_i = \gamma J.$$

In the same way as we obtained (17), we compute

$$M = \frac{\gamma J}{2} + \frac{c'}{2s} + \frac{1}{2}\sum_{i=1}^{k} p_i \left|y_i - \frac{c'}{s}\right|. \tag{23}$$

Applying Lemma 3 as above, with $a = J$ and $b = \beta J$, from (23) it follows that

$$M \leq \frac{\gamma}{2}J + \frac{c'}{2s} + \frac{1}{2}\left[p\left(\frac{c'}{s} - J\right) + (1-p)\left(\beta J - \frac{c'}{s}\right)\right].$$

If we replace p by its value (21) on the right-hand side of this inequality, we obtain a linear expression in γ denoted $h(\gamma)$:

$$h(\gamma) = \frac{1}{\beta - 1}\left[\frac{c'}{s}(\beta - \gamma) + \beta(\gamma - 1)J\right].$$

In particular,

$$h(1) = \frac{c'}{s} \quad \text{and} \quad h\left(\frac{\beta+2}{3}\right) = \frac{2}{3}\left[\frac{c'}{s} + \frac{\beta J}{2}\right] \geq h(1),$$

where the inequality comes from the hypothesis $\beta J \geq \frac{c'}{s}$. Since h is linear in γ with a positive slope, using (22) we deduce

$$M \leq h(\gamma) \leq h\left(\frac{\beta+2}{3}\right)$$
$$= \frac{2}{3}\left[\frac{c'}{s} + \frac{\beta J}{2}\right] = \frac{2}{3}\left(\frac{c'}{s} + d'\right). \tag{24}$$

Whenever the invariant is tight or not, using the lower bound (1) on the optimal makespan it follows from (20) or

(24) that

$$\frac{M}{\text{OPT}} \le \frac{2}{3}\frac{(s+2)}{s}\frac{c'+sd'}{c'+2d'} = g(c'),$$

where $g(x) = \frac{2}{3}\frac{(s+2)}{s}\frac{x+sd'}{x+2d'}$. We may verify that g is an increasing function, whenever $0 \le s < 2$. Since $\frac{c'}{s} \le 2d'$, we deduce that $\frac{M}{\text{OPT}} \le g(c') \le g(2d's) = \frac{s+2}{s+1}$. □

Lemma 6 *In case* 3, *the approximation ratio of TwoGroups is upper bounded by* $\frac{\alpha(s)s^2+2\alpha(s)s+2}{s(\alpha(s)s+2)}$.

Proof Since $2d' \le \frac{c'}{s}$, w.l.o.g. we may suppose that the job σ_k has been assigned to machine S. So we have $c' = c + t_k$ and $c \le \alpha(s)sd$ by definition of the algorithm. We define the function $\bar{g}(x) := \frac{s+2}{s}\frac{x+t_k}{x+2d+t_k}$. Since the function $\bar{g}$ is increasing in the interval $[0;\infty)$, it follows that $\bar{g}(c) \le \bar{g}(\alpha(s)sd)$. The preceding inequality together with $c \le \alpha(s)sd$ imply that

$$\frac{M}{\text{OPT}} = \frac{\frac{c+t_k}{s}}{\text{OPT}} \le \frac{\frac{c+t_k}{s}}{\max\{t_k, \frac{c+2d+t_k}{s+2}\}} \le \frac{1}{s}\frac{\alpha(s)sd+t_k}{\max\{t_k, \frac{\alpha(s)sd+2d+t_k}{s+2}\}}, \quad (25)$$

where the first inequality comes from lower bounds (1) and (2) for the optimum. Studying the cases where t_k is greater or smaller than $\frac{\alpha(s)sd+2d+t_k}{s+2}$, in the same way as for proving Lemma 4, we deduce from (25) that

$$\frac{M}{\text{OPT}} \le \frac{\alpha(s)s^2+2\alpha(s)s+2}{s(\alpha(s)s+2)},$$

by using the implications $t_k \le \frac{\alpha(s)sd+2d+t_k}{s+2} \Rightarrow \frac{d}{\alpha(s)sd+t_k} \ge \frac{s+1}{(s+2)\alpha(s)s+2}$ and $t_k \ge \frac{\alpha(s)sd+2d+t_k}{s+2} \Rightarrow \frac{d}{t_k} \le \frac{s+1}{\alpha(s)s+2}$. □

Proposition 7 *For* $-1+\frac{\sqrt{10}}{2} < s \le 1$, *the competitive ratio of TwoGroups is at least* $2\frac{s+3}{2s+3}$.

Proof In this proof, c and d will denote the total load of S and one-half the total load of $\{F_1, F_2\}$, respectively, just before the assignment of some job. To show the theorem we consider the job sequence $\sigma = (\sigma_1, \ldots, \sigma_4)$ of four jobs whose processing times are $4s^2+8s-6$ for σ_1, $2(2s+3)$ for σ_2, $2(3-s)$ for σ_3, and finally $2(2s+3)$ for σ_4. Let us observe that for $s > -1+\frac{\sqrt{10}}{2}$, $\alpha(s) = \frac{4s^2+8s-6}{s^2+6s} > 0$.

At the beginning, we have $c = d = 0$. The first job σ_1 is scheduled on machine S (since the condition $\frac{c}{s} \le \alpha(s)d$ is satisfied). Then $c = 4s^2+8s-6$ and $d = 0$. The second job is scheduled on group $\{F_1, F_2\}$ (since $\frac{4s^2+8s-6}{s} > \alpha(s)d = 0$). So $d = 2s+3$. The third job is also assigned to group $\{F_1, F_2\}$ (since $\frac{4s^2+8s-6}{s} > \alpha(s)d \Leftrightarrow s+6 > d = 2s+3$). Then $d = s+6$. Finally σ_4 is scheduled on machine S (since $\frac{4s^2+8s-6}{s} = \alpha(s)d \Leftrightarrow s+6 = d$). Then $\frac{c}{s} = \frac{4s^2+8s-6+2(2s+3)}{s} = 4s+12 > 2d = 2s+12$. Thus, the makespan is determined by machine S. Since $\text{OPT}(\sigma) = 4s+6$, the approximation ratio is $\frac{4s+12}{4s+6} = 2\frac{s+3}{2s+3}$. See Fig. 5. □

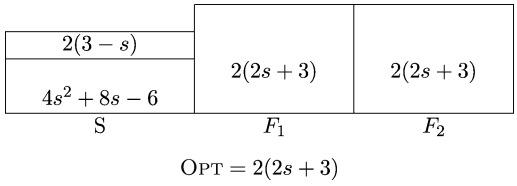

Fig. 5 The optimal schedule for the job sequence $(4s^2+8s-6, 2(2s+3), 2(3-s), 2(2s+3))$

Theorem 4 *For* $-1+\frac{\sqrt{10}}{2} < s \le 1$, *the competitive ratio of TwoGroups is equal to* $\frac{2(s+3)}{2s+3}$.

Proof The lower bound on the competitive ratio of TwoGroups follows from Proposition 7.

To obtain an upper bound to the competitive ratio of TwoGroups, we compare the results from Lemmas 4, 5, and 6. We choose the function $\alpha(s)$ yielding the smallest approximation ratio. This happens when the approximation ratios of cases 1 and 3 are equal. We obtain $\alpha(s) = \frac{4s^2+8s-6}{s^2+6s}$. Note that for $s \ge 0$, $\alpha(s) \ge 0$ if and only if $-1+\frac{\sqrt{10}}{2} \le s$. By replacing the last fraction corresponding to $\alpha(s)$ in $\frac{\alpha(s)s^2+2\alpha(s)s+2}{s(\alpha(s)s+2)}$, we obtain an approximation ratio of $\frac{2(s+3)}{2s+3}$ for cases 1 and 3. We may verify that the approximation ratio $\frac{s+2}{s+1}$ of case 2 is bounded by $\frac{2(s+3)}{2s+3}$ for $s \ge 0$. So TwoGroups is $\frac{2(s+3)}{2s+3}$-competitive, for $-1+\frac{\sqrt{10}}{2} < s \le 1$. □

6 Conclusion

In this paper, we have settled the nonpreemptive deterministic and randomized version of the online scheduling of independent jobs on m uniform machines, $m-1$ of them being identical. We have determined the competitive ratio of the well-known LS algorithm, and we have determined its competitive ratio for any $0 \le s \le 1$ and $m \ge 2$. For $0 \le s \le s_{\max}(m)$, LS has the same competitive ratio as LSF. This means that we do not need the slow machine if its speed lies in this interval; one might indeed have expected the existence of such a threshold value for the slow machine's speed. We also have shown that LS is the best online deterministic algorithm for $0 \le s \le -1+\sqrt{2}$ and $m = 3$ machines.

Furthermore, a randomized heuristic on m machines has been described. In the model of 3 machines, a randomized

algorithm called TwoGroups has been developed and studied. The TwoGroups algorithm has been proven to be more competitive than LS for any s in the interval [0, 1]. The technique for analyzing its competitive ratio certainly could be adapted to deal with other scheduling problems. For s close to 1, we have shown, as expected, that Linear Invariant competes better than TwoGroups.

Acknowledgements The authors thank Gerhard Woeginger and Thomas Liebling for some helpful comments on this paper. They are also very grateful to one referee whose remarks have greatly improved the paper.

References

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

Borodin, A., Irani, S., Raghavan, P., & Schieber, B. (1995). Competitive paging with locality of reference. *Journal of Computer and System Sciences*, *50*(2), 244–258.

Cheng, T. C. E., Ng, C. T., & Kotov, V. (2006). A new algorithm for online uniform-machine scheduling to minimize the makespan. *Information Processing Letters*, *99*(3), 102–105.

Cho, Y., & Sahni, S. (1980). Bounds for list schedules on uniform processors. *SIAM Journal on Computing*, *9*(1), 91–103.

Epstein, L., Noga, J., Seiden, S., Sgall, J., & Woeginger, G. (2001). Randomized on-line scheduling on two uniform machines. *Journal of Scheduling*, *4*(2), 71–92.

Hochbaum, D. S. (1997). *Approximation algorithms for NP-hard problems*. Boston: PWS Publishing Company.

Kokash, N. (2004). *An efficient heuristic for on-line scheduling in system with one fast machine*. Master Thesis.

Li, R., & Shi, L. (1998). An on-line algorithm for some uniform processor scheduling. *SIAM Journal on Computing*, *27*(2), 414–422.

Seiden, S. S. (2000). Randomized online multiprocessor scheduling. *Algorithmica*, *28*(2), 173–216.

Sgall, J. (1994). *On-line scheduling on parallel machines*. Ph.D. Thesis, Carnegie-Mellon University, Pittsburgh, PA.