# On Randomized Online Scheduling

Susanne Albers[*]
Freiburg University

## ABSTRACT

We study one of the most basic problems in online scheduling. A sequence of jobs has to be scheduled on $m$ identical parallel machines so as to minimize the makespan. Whenever a new job arrives, its processing time is known in advance. The job has to be scheduled immediately on one of the machines without knowledge of any future jobs. In the sixties Graham presented the famous *List* scheduling algorithm which is $(2-\frac{1}{m})$-competitive. In the last ten years deterministic online algorithms with an improved competitiveness have been developed. The first algorithm with a performance guarantee asymptotically smaller than 2 was 1.986-competitive. The competitive ratio was first improved to 1.945 and then to 1.923 and 1.9201. Randomized competitive algorithms that are better than (known) deterministic algorithms were proposed for specific values of $m$, i.e. for $m \in \{2,\ldots,7\}$.

In this paper we present the first randomized online algorithm that performs better than known deterministic algorithms for general $m$. The algorithm is a combination of two deterministic scheduling strategies $A_1$ and $A_2$. Initially, when starting the scheduling process, a scheduler chooses $A_i$, $i \in \{1,2\}$, with probability $\frac{1}{2}$ and then serves the entire job sequence using the chosen algorithm. The new randomized algorithm is 1.916-competitive. We prove that this performance cannot be achieved by a deterministic algorithm based on analysis techniques that have been used in the literature so far: Using know techniques (or generalizations) it is impossible to prove a competitiveness smaller than 1.919 for any deterministic online algorithm. Our results strictly limit the performance that can be achieved with existing techniques.

[*]Institute of Computer Science, Freiburg University, Georges-Köhler-Allee 79, 79110 Freiburg, Germany. E-mail: salbers@informatik.uni-freiburg.de. Work supported by the Deutsche Forschungsgemeinschaft, Project AL 464/3-1, and by the European Community, Projects APPOL and APPOL II.

## 1. INTRODUCTION

We study one of the most basic problems in online scheduling. A job sequence $\sigma = J_1,\ldots,J_n$ has to be scheduled on $m$ identical parallel machines. Whenever a new job $J_t$ arrives, its processing time $p_t$ is known in advance, $1 \le t \le n$. The job has to be scheduled immediately on one of the machines without knowledge of any future jobs. Preemption of jobs is not allowed and any machine may only process one job at a time. The goal is to minimize the *makespan*, which is the completion time of the last job that finishes in the schedule. The problem has been the subject of extensive research, see e.g. [1–13]. Given a job sequence $\sigma$, let $A(\sigma)$ be the makespan produced by an online algorithm $A$ and let $OPT(\sigma)$ be the optimum makespan. Following [14] we call a deterministic online algorithm *c-competitive* if $A(\sigma) \le c \cdot OPT(\sigma)$, for all $\sigma$. A randomized online algorithm is $c$-competitive against any oblivious adversary if the expected makespan satisfies $\mathrm{E}[A(\sigma)] \le c \cdot OPT(\sigma)$, for all $\sigma$.

**Previous results:** Already in the sixties Graham [9] presented the famous *List* scheduling algorithm, which always assigns an incoming job to the least loaded machine, and proved that it is $(2-\frac{1}{m})$-competitive. In the last ten years deterministic online algorithms with an improved competitiveness have been developed. A $(2-\frac{1}{m}-\epsilon_m)$-competitive algorithm was given in [7], where $\epsilon_m$ tends to 0 as $m$ goes to infinity. Bartal *et al.* [2] presented the first algorithm whose performance guarantee is asymptotically smaller than 2. Their algorithm is 1.986-competitive. The competitive ratio was first improved to 1.945 by Karger *et al.* [10], and then to 1.923 and 1.9201, see [1, 6]. Lower bounds on the performance of deterministic online algorithms were given in [1, 3, 5, 8]. The best bound currently known is due to Gormley *et al.* [8], who showed that no deterministic online scheduling algorithm can be better than 1.853-competitive. Since the publication of the paper by Bartal *et al.* [2], there has always been research interest in developing better randomized online algorithms. Bartal *et al.* gave a randomized algorithm for two machines. The algorithm is (4/3)-competitive, and this is the best possible performance for $m = 2$. Chen *et al.* [4] and Sgall [13] proved that no randomized online algorithm can achieve a competitiveness smaller than $1/(1-(1-1/m)^m)$. This expression is equal to 4/3 if $m = 2$ and tends to $e/(e-1)$ as $m \to \infty$. Seiden [11] presented a randomized algorithm whose competitive ratio is smaller than the best known deterministic ratio for $m \in \{3,\ldots,7\}$. The competitiveness is also smaller than the deterministic lower bound for $m = 3,4,5$. The algorithms by Bartal *et al.* and Seiden have to maintain $t$ schedules when $t$ jobs have been scheduled. Seiden [12] modified his algorithm so that it maintains a constant number of schedules. However, the constant is large, i.e. it is equal to 2048, and the competitive ratio is worse, for $m \geq 4$.

**Our contribution:** In this paper we present the first randomized online algorithm that performs better than known deterministic algorithms for general $m$. Our new algorithm, called *Rand*, is a combination of two deterministic algorithms $A_1$ and $A_2$. Initially, when starting the scheduling process, *Rand* chooses $A_i$, $i \in \{1, 2\}$, with probability $\frac{1}{2}$ and then serves the entire job sequence using the chosen algorithm. At most two schedules have to be maintained at any time. Algorithm $A_1$ is a conservative strategy that tries to maintain schedules with a low makespan. On the other hand, $A_2$ is an aggressive strategy that aims at generating schedules with a high makespan. We prove, as the main result of this paper, that the combined algorithm *Rand* is 1.916-competitive.

The improvement over deterministic algorithms may seem small. We show, however, that a competitiveness of 1.916 cannot be proven for a deterministic algorithm based on the analysis techniques that have been used in the literature so far. All the previous analyses of online algorithms only use the following three lower bounds on the optimum makespan. (a) The total amount of processing in a given job sequence divided by $m$. (b) The largest processing time of any job in $\sigma$. (c) Twice the $(m+1)$-st largest processing time in $\sigma$. We show that using only these three lower bounds on the optimum makespan, it is impossible to prove a competitiveness smaller than 1.919 on the performance of any deterministic online algorithm. In fact we prove a stronger statement: Using only the information (1) "The total amount of processing in a job sequence divided by $m$" and (2) "The processing times of the $m+1$ largest jobs in $\sigma$" it is impossible to prove a competitive ratio smaller than 1.919. In addition to (1) and (2) an analysis could consider the processing times of the $(im+1)$-st largest jobs, for $i = 1, \ldots, \lfloor (n-1)/m \rfloor$. We show that this does not help much either; it is impossible to prove a competitiveness smaller than 1.917. These results strictly limit the performance of deterministic algorithms that can be achieved using known techniques.

**Ideas of this paper:** The algorithm *Rand* uses a number of new ideas, the most important feature being that two algorithms $A_1$ and $A_2$ coordinate their scheduling decisions. In each scheduling step, the aggressive algorithm $A_2$ has to take into account its own schedule, the conservative algorithm's schedule as well as *future* schedules that can evolve from the current configuration. The maximum makespan that $A_2$ can afford depends on this configuration.

The previous analyses of randomized online scheduling algorithms for $m \in \{2, \ldots, 7\}$ are heavily based on the fact that if the expected makespan is high, a considerable fraction of the total load in the system resides on the machine with the highest load. This approach does not work for general $m$. Our main contribution in the analysis is that we simultaneously keep track of the schedules maintained by $A_1$ and $A_2$. On job sequences consisting of small and medium size jobs, *Rand*'s expected makespan is small. If *Rand*'s makespan is high, many large jobs must have been scheduled. To identify large jobs, which is essential for proving a competitiveness below $2 - \frac{1}{m}$, we have to analyze the combined load vectors of $A_1$'s and $A_2$'s machines at various points in the scheduling process. Our analysis is quite involved but, unlike some previous analyses, does not rely on computer proofs.

In our lower bound proofs we construct nemesis sequences where the number of different job sizes grows with $m$. Previous lower bound constructions worked with a bounded number of sizes.

**Organization of the paper:** We first present the lower bounds in Section 2 and then concentrate on the development of the randomized online algorithm, which is more involved. A description of the algorithm is given in Section 3. A detailed analysis follows in Section 4.

## 2. LOWER BOUNDS

Several online scheduling algorithms have been proposed in the literature [1, 2, 6, 7, 9, 10, 11]. To evaluate the competitiveness of these algorithms, one has to determine lower bounds on the makespan produced by an optimal offline algorithm. So far, all the analyses are based on the following three lower bounds (a–c) on the optimum makespan. Let $\sigma = J_1, \ldots, J_n$ be a given job sequence and let $p_t$ be the processing time of the $t$-th job, $1 \leq t \leq n$.

(a) $\frac{1}{m} \sum_{t=1}^{n} p_t$
(b) $\max_{1 \leq t \leq n} p_t$
(c) Twice the processing time of the $(m+1)$-st largest job in $\sigma$.

Theorem 1 implies that using only these three lower bounds on the optimum makespan, it is impossible to prove a competitiveness smaller than 1.919 on the performance of any deterministic online algorithm. In fact Theorem 1 is more general. Suppose that we use the following information to derive lower bounds on the optimum makespan.

(1) $\frac{1}{m} \sum_{t=1}^{n} p_t$
(2) The processing times of the $m+1$ largest jobs in $\sigma$.

Information (2) is more general than (b) and (c) as it contains, in particular, the largest and $(m+1)$-st largest processing times and can be used to derive additional lower bounds on the optimum makespan.

THEOREM 1. *Let $A$ be a deterministic online scheduling algorithm. Using only information (1) and (2) to derive lower bounds on the optimum makespan, it is impossible to prove a competitive ratio smaller than 1.919 on $A$'s performance.*

PROOF. We prove the theorem as $m \to \infty$. Given a job sequence $\sigma$, what are the strongest lower bounds on the optimum makespan we can derive from (1) and (2)? In addition to the lower bounds given in (a) and (b), which we will refer to as $B_\sigma^1$ and $B_\sigma^2$ respectively, the strongest additional bound that follows from (2) is $B_\sigma^3$: The sum of the processing times of the $m$-th and the $(m+1)$-st largest jobs. This bound is stronger than (c). To establish the theorem, we will show that there exist job sequences $\sigma$ for which $A$'s makespan is at least $1.919 \max\{B_\sigma^1, B_\sigma^2, B_\sigma^3\}$. An adversary constructs a nemesis job sequence in phases. In each of the first three phases $m$ jobs are presented. Algorithm $A$ will have to schedule the jobs in each phase on different machines. In a fourth phase one large job is generated such that $A$'s makespan for the constructed sequences $\sigma$ is at least $1.919 \cdot \max\{B_\sigma^1, B_\sigma^2, B_\sigma^3\}$.

*Phase 1:* The adversary first presents $m$ jobs with a processing time of $p = 0.05$ each. Algorithm $A$ has to schedule these jobs on different machines since otherwise its competitive ratio would be 2.

*Phase 2:* The adversary first presents $\lfloor 0.975m \rfloor$ jobs, each with a processing time of $q = 0.369$. While these jobs are being scheduled, $B^1 \leq L_1/m$, where $L_1 = mp + \lfloor 0.975m \rfloor q < 0.41m$ is the total processing time of all the jobs after the $q$-jobs have been presented. Moreover, $B^2 = q$ and $B^3 = 2p < B^2$. Here $B^1, B^2$ and $B^3$ denote the current lower bounds for the prefix of $\sigma$ seen so far; we omit the subscript $\sigma$. The online algorithm has to assign the $q$-jobs to different machines. Otherwise its makespan would be $p + 2q = 0.419 + q > 1.919 \max\{B^1, B^2\}$. At this point any machine containing a $q$-job has a load of 0.419.

The adversary then generates $m - \lfloor 0.975m \rfloor = \lceil 0.025m \rceil$ jobs with a processing time of $q' = 0.39$. During the scheduling of these jobs, $B^1 \leq L_2/m$, where $L_2 = L_1 + \lceil 0.025m \rceil q'$ is the total load in the system at the end of the phase, and this value is at least $0.419m$ and upper bounded by $0.42m$ as $m \to \infty$. For the other bounds we have $B^2 = q'$ and $B^3 \leq p + q$. The online algorithm has to schedule the $q'$-jobs on different machines which do not already have a $q$-job. Otherwise the makespan would be $p + q + q' = 0.419 + q' > 1.919 \max\{B^1, B^2, B^3\}$. At the end of this phase each machine in $A$'s schedule was assigned two jobs and has a load of at least 0.419.

*Phase 3:* Define $m_3 = \lfloor (1 + 2 \cdot 0.919)m/1.919 - 2L_2) \rfloor$ and $m'_3 = m - m_3$. As $m \to \infty$ we have $m_3 < 0.64m$. The adversary first creates $m_3$ jobs with a processing time of $r = 0.5$ each. While these jobs are being scheduled $B^3 = 2q = 0.738$ and $B^1 \leq L_3/m$, where $L_3 = L_2 + m_3 r$ is the sum of the processing times when all the $r$-jobs are scheduled. We have $L_3 \leq 1.419m/1.919$. Also $B^2 = 0.5 < B^3$. Algorithm $A$ must schedule the $r$-jobs on different machines since otherwise the makespan would be at least $0.419 + 2r = 1.419 \geq 1.919 \max\{B^1, B^3\}$.

In each of the following $m'_3$ steps the adversary generates a job whose processing time is so large that an assignment of the job to a machine containing an $r$-job would result in a makespan that is at least 1.919 times the current value of $B^1$. Consider the $t$-th step, $1 \leq t \leq m'_3$, and let $l_{t-1}$ be the total processing time of all the jobs scheduled up to that step. The adversary generates a job whose processing time $r'_t$ satisfies $0.919 + r'_t \geq 1.919(l_{t-1} + r'_t)/m$, i.e. $r'_t = (\frac{1.919}{m} l_{t-1} - 0.919)/(1 - \frac{1.919}{m})$. We first analyze the sum of the processing times when all the $r'$-jobs are scheduled and then show that at any time $B^1 \geq B^2$ and $B^1 \geq B^3$. This proves that the $r'$-jobs must be scheduled on different machines and cannot be assigned to machines containing any $r$-job. We show inductively that $l_t = (L_3 - \frac{c'm}{c})(1 - \frac{c}{m})^{-t} + \frac{c'm}{c}$, where $c = 1.919$ and $c' = c - 1$. The equation is satisfied for $t = 0$. Suppose it holds for $t - 1$. Then

$$\begin{aligned} l_t &= l_{t-1} + r'_t = (l_{t-1} - c')/(1 - \tfrac{c}{m}) \\ &= (L_3 - \tfrac{c'm}{c})(1 - \tfrac{c}{m})^{-t} + \tfrac{c'm}{c}/(1 - \tfrac{c}{m}) - c'/(1 - \tfrac{c}{m}) \\ &= (L_3 - \tfrac{c'm}{c})(1 - \tfrac{c}{m})^{-t} + \tfrac{c'm}{c}. \end{aligned}$$

Thus, when all the $r'$-jobs are scheduled, the total processing time of all the jobs is

$$\begin{aligned} L_4 &= (L_3 - \tfrac{c'm}{c})(1 - \tfrac{c}{m})^{-m'_3} + \tfrac{c'm}{c} \\ &\leq (L_3 - \tfrac{c'm}{c})e^{cm'_3/m} + \tfrac{c'm}{c}. \end{aligned}$$

As $m \to \infty$ we find $L_4 < 0.999m$. The processing times of the $r'$-jobs are increasing, the first job having a size of at least 0.5 because

$$\begin{aligned} r'_1 &= (\tfrac{1.919}{m} L_3 - 0.919)/(1 - \tfrac{1.919}{m}) \\ &= (\tfrac{1.919}{m}(L_2 + 0.5m_3) - 0.919)/(1 - \tfrac{1.919}{m}) \\ &\geq (\tfrac{1.919}{m}(L_2 + 0.5((1 + 2 \cdot 0.919)m/1.919 - 2L_2 - 1)) \\ &\quad -0.919)/(1 - \tfrac{1.919}{m}) \\ &= 0.5. \end{aligned}$$

Thus, for the proof that at any step $B^1 \geq B^2$, it is sufficient to show that $(l_{t-1} + r'_t)/m = l_t/m \geq r'_t$. This is equivalent to showing $(l_{t-1} - c')/m \geq (\frac{c}{m} l_{t-1} - c')$, which in turn is equivalent to $(1 - 1/m) \geq l_{t-1}/m$. Since $l_{t-1} \leq L_4$, this holds as $m \to \infty$. We still have to argue that at any step $B^1 \geq B^3$. While the first $m'_3 - \lceil 0.025m \rceil - 1$ $r'$-jobs are scheduled, $B^3 \leq 2q = 0.738 < L_3/m \leq B^1$ because $L_3 > 0.739m$ as $m \to \infty$. When the last $\lceil 0.025m \rceil + 1$ $r'$-jobs are scheduled, $B^3 \leq q' + r = 0.89$, whereas $B^1 \geq (L_2 + (\lfloor 0.975m \rfloor - 1)r)/m > 0.9$ as $m \to \infty$. At the end of the phase each machine has a load of at least 0.919.

*Phase 4:* The adversary presents a final job with a processing time of 1. The online algorithm has a makespan of 1.919. We have $B^1 \leq 1$ as $m \to \infty$ and $B^2 = B^3 = 1$. Thus the online makespan is $1.919 \max\{B^1_\sigma, B^2_\sigma, B^3_\sigma\}$. □

What happens if, in addition to (2), we consider a larger set of jobs? A generalization of the lower bound (c) on page 2 is (d) $(i + 1)$ times the processing time of the $(im + 1)$-st largest job in $\sigma$, for $i = 2, \ldots, \lfloor (n - 1)/m \rfloor$. We consider again a generalized set of information.

(3) The processing times of the $(im - i + 1)$-st to $(im + 1)$-st largest jobs in $\sigma$, for $i = 2, \ldots, \lfloor (n - 1)/m \rfloor$.

We show that even with this additional information it is impossible to prove considerably better competitive ratios. The proof of the following theorem is omitted is this extended abstract.

THEOREM 2. *Let $A$ be a deterministic online scheduling algorithm. Using only information (1–3) to derive lower bounds on the optimum makespan, it is impossible to prove a competitive ratio smaller than 1.917 on $A$'s performance.*

## 3. THE RANDOMIZED ONLINE ALGORITHM

Our new randomized algorithm, called *Rand*, is a combination of two deterministic algorithms $A_1$ and $A_2$. Initially, when starting the scheduling process, *Rand* chooses $A_1$ with probability $q$ and $A_2$ with probability $1 - q$, where $q = 1/2$, and then serves the entire job sequence using the chosen algorithm. The two algorithms complement each other. On sequences for which $A_1$ has a high makespan, $A_2$'s makespan is low, and vice versa. At any time both algorithms keep a list of their machines sorted in non-decreasing order by current load. The *load* of a machine is the sum of the processing times of the jobs already assigned to it. Consider a job sequence $\sigma = J_1, \ldots, J_n$ and let $p_t$ be the processing time of job $J_t$, $1 \leq t \leq n$. Let $M^t_{i,j}$ be the machine with the $j$-th smallest load in the schedule maintained by $A_i$ after $t$ jobs have been scheduled, $i = 1, 2$ and $j = 1, \ldots, m$. Thus $M^t_{i,1}$ are the machines with the smallest load and $M^t_{i,m}$ are the machines with the largest load. For simplicity, we also refer to the machines with the $j$-th smallest load as the *$j$-th smallest machines*. Let $l_{i,j}^t$ be the load of $M_{i,j}^t$. Let $L^t$ the sum of the processing times of the first $t$ jobs in $\sigma$, i.e. $L^t$ is the sum of the loads on the machines in one of the schedules after $t$ jobs have been assigned.

Algorithm $A_i$, $i \in \{1,2\}$, tries to keep $k_i$ machines lightly loaded and $m - k_i$ machines heavily loaded, where $k_1 = \lceil \frac{9}{25} m \rceil$ and $k_2 = \lceil \frac{3}{8} m \rceil$. Let $\mu_i^t$ be the average load on the $k_i$ smallest machines of $A_i$, i.e. $\mu_i^t = \frac{1}{k_i} \sum_{j=1}^{k_i} l_{i,j}^t$. Algorithm $A_i$ always tries to maintain a schedule in which $\mu_i^t$ is bounded by $\alpha_i$ times the load on the $(2k_i + 1)$-st smallest machine, where $\alpha_1$ and $\alpha_2$ are specific constants needed in the analysis of the algorithms, i.e. $\alpha_1 = 1 - (k_1 - \lfloor 0.074m \rfloor)/(2 \cdot 0.916 k_1)$ and $\alpha_2 = 0.409/0.909$. Formally the algorithms want to maintain $\mu_i^t \leq \alpha_i l_{i,2k_i+1}^t$. In some cases, when several large jobs arrive, it is impossible to maintain the invariant. A schedule is called *critical* if $\mu_i^t > \alpha_i l_{i,2k_i+1}^t$.

Algorithm $A_i$, $i \in \{1,2\}$, always schedules an incoming job either on the machine with the smallest load or on the machine with the $(k_i + 1)$-st smallest load. An algorithm only considers scheduling a job on the $(k_i + 1)$-st smallest machine if its schedule is critical. Algorithm $A_2$, which we describe in detail below, is aggressive. Loosely speaking, it assigns a job $J_t$ to the $(k_2 + 1)$-st smallest machine if the resulting makespan is bounded by $c_2 L^t/m$, where $c_2 = 2$. On the other hand $A_1$ is conservative; it only schedules a job on the $(k_1 + 1)$-st smallest machine if the resulting makspan is at most $c_1 L^t/m$, where $c_1 = 1.832$. Note that $q c_1 L^t/m + (1 - q) c_2 L^t/m = 1.916 L^t/m$, which is at most 1.916 times the optimum makespan.

**Algorithm $A_1$:** Set $c_1 := 1.832$, $k_1 := \lceil \frac{9}{25} m \rceil$ and $\alpha_1 := 1 - (k_1 - \lfloor 0.074m \rfloor)/(2 \cdot 0.916 k_1)$.
Schedule a new job $J_t$ on the machine with the $(k_1+1)$-st smallest load if the schedule is critical and $l_{1,k_1+1}^{t-1} + p_t \leq c_1 L^t/m$. Otherwise schedule $J_t$ on the machine with the smallest load.

In some situations $A_2$ cannot afford a makespan of $c_2 L^t/m$. Suppose that $A_1$'s schedule is critical and that all the machines have approximately the same load. Then a newly arriving job can force a high makespan in $A_1$'s schedule. A makespan of $c_2 L^t/m = 2L^t/m$ in $A_2$'s schedule is then too expensive. Let $\lambda_1^t = \sum_{j=k_1+1}^{m} l_{1,j}^t$ be the total load on machines $M_{1,k_1+1}^t, \ldots, M_{1,m}^t$. We say that $A_1$'s schedule is *balanced* if the total load on the $k_1$ smallest machines is at least $(c_1 - 1)\frac{k_1}{m} L^t$, or equivalently, if $\lambda_1^t \leq \beta L^t$ where $\beta := 1 - (c_1 - 1)\frac{k_1}{m}$.

To find out if $A_1$'s schedule is balanced, algorithm $A_2$ always keeps track of the schedule that $A_1$ would have created. If $A_1$'s schedule is indeed balanced, $A_2$ only places a new job on the $(k_2 + 1)$-st smallest machine if the resulting makespan does not exceed $\max\{c_2' L^t/m, c_2 \beta^{-1} \lambda_1^t/m\}$, where $c_2' = 1.885$. As we shall show in the later analysis (see Section 4.1), this constraint ensures that the expected makespan of *Rand* is always bounded by 1.916 times the optimum makespan.

**Algorithm $A_2$:** Set $c_2 := 2$, $c_2' := 1.885$, $k_2 := \lceil \frac{3}{8} m \rceil$, $\alpha_2 := 0.409/0.909$ and $\beta := 1 - (c_1 - 1)\frac{k_1}{m}$.
At any time the algorithm keeps track of the schedule that $A_1$ would have constructed. When a new job $J_t$ arrives, consider $A_1$'s schedule after $J_t$ was processed by $A_1$. If $A_1$'s schedule is balanced, then set $\gamma := \max\{c_2' L^t/m, c_2 \beta^{-1} \lambda_1^t/m\}$. Otherwise set $\gamma := c_2 L^t/m$. Schedule $J_t$ on the machine with the $(k_2+1)$-st smallest load if the schedule is critical and $l_{1,k_2+1}^{t-1} + p_t \leq \gamma$. Otherwise schedule $J_t$ on the machine with the smallest load.

The main algorithm works as follows.

**Algorithm Rand:** Given a job sequence $\sigma$, with probability $q = 1/2$ execute $A_1$ and with probability $1 - q$ execute $A_2$.

THEOREM 3. *The algorithm Rand is 1.916-competitive as $m \to \infty$.*

The algorithm *Rand* with its components $A_1$ and $A_2$ depends on various parameters, all of which have been optimized. To obtain a small competitive ratio, $c_1$ should be chosen as small as possible. However, if $c_1$ is below 1.832, we are not able to always identify large jobs in the input sequence when the schedules are critical; this is crucial in the analysis. In particular, $k_1$ and $\alpha_1$ are chosen so that we can identify large jobs in $A_1$'s schedule when *Rand*'s expected makespan is above $1.916 L^t/m$ but $A_2$'s makespan is low and its schedule cannot be used to identify large jobs. It turns out that by setting the probability $q$ to a value slightly above 1/2, we are able to improve the competitive ratio. However the improvement is minor and the modified algorithm does not achieve a competitive ratio of at most 1.915. Since we decided to optimize up to a value of 1/1000, we work with $q = 1/2$.

## 4. THE ANALYSIS OF THE ALGORITHM

### 4.1 Analysis of the makespan

We prove Theorem 3 by induction on the number $n$ of jobs to be scheduled. Obviously, the theorem holds for job sequences consisting of only $n = 1$ job. Suppose that it holds for sequences of length $n-1$ and consider any sequence $\sigma = J_1, \ldots, J_n$. We have to prove

$$\begin{aligned} \mathrm{E}[\mathit{Rand}(\sigma)] &= q A_1(\sigma) + (1-q) A_2(\sigma) \\ &= q l_{1,m}^n + (1-q) l_{2,m}^n \\ &\leq 1.916 \cdot \mathit{OPT}(\sigma). \end{aligned} \quad (1)$$

Let $L = L^n$ be the total load of all the jobs in $\sigma$.

We first note that if $A_2$ schedules some job $J_t$ on the machine with the $(k_2+1)$-st smallest load, then the resulting load $l_{2,k_2+1}^{t-1} + p_t$ is bounded by $c_2 L^t/m$. This is obvious if in the scheduling step $\gamma = c_2 L^t/m$ or $\gamma = c_2' L^t/m$. If $\gamma = c_2 \beta^{-1} \lambda_1^t/m$, then $\lambda_1^t \leq \beta L^t$ because $A_1$'s schedule is balanced at time $t$. Hence $\gamma \leq c_2 \beta^{-1} \beta L^t/m = c_2 L^t/m$.

If at time $n$ the makespan of $A_1$ satisfies $l_{1,m}^n \leq c_1 L/m$, then (1) follows easily: If $l_{2,m}^n \leq c_2 L/m$, then $\mathrm{E}[\mathit{Rand}(\sigma)] \leq q c_1 L/m + (1-q) c_2 L/m = 0.5(1.832 + 2) L/m = 1.916 L/m \leq 1.916 \cdot \mathit{OPT}(\sigma)$. On the other hand, if $l_{2,m}^n = (c_2 + \delta) L/m$, for some $\delta > 0$, then by the arguments given in the previous paragraph, the last job assigned to machine $M_{2,m}^n$ was scheduled on the least loaded machine at the time of the assignment and its processing time is at least $(c_2 + \delta) L/m - L/m = (1 + \delta) L/m$ because the least loaded machine always has a load of at most $L/m$. Thus $\mathrm{E}[\mathit{Rand}(\sigma)] \leq q c_1 L/m + (1 - q)(c_2 + \delta) L/m \leq 1.916(1 + \delta) L/m \leq 1.916 \max_{1 \leq t \leq n} p_t \leq 1.916 \cdot \mathit{OPT}(\sigma)$.

In the following we assume that the makespan of $A_1$ satisfies $l_{1,m}^n > c_1 L/m$, which implies that the last job on $M_{1,m}^n$ was scheduled on the least loaded machine at the time of the assignment. If the load on the smallest machine is $l_{1,1}^n \le (c_1 - 1)L/m$, then the analysis is again simple. Let $l_{1,m}^n = (c_1 + \delta_1)L/m$, for some $\delta_1 > 0$. The last job on $M_{1,m}^n$ has a processing time of at least $(1 + \delta_1)L/m$. If $l_{2,m}^n \le c_2 L/m$, then $\mathrm{E}[Rand(\sigma)] \le 1.916(1 + \delta_1)L/m \le 1.916 \max_{1 \le t \le n} p_t \le 1.916 \cdot OPT(\sigma)$. If $l_{2,m}^n = (c_2 + \delta_2)L/m$, for some $\delta_2 > 0$, then the last job on $M_{2,m}^n$ has a processing time of at least $(1 + \delta_2)L/m$. We have $\mathrm{E}[Rand(\sigma)] \le 1.916 \max\{(1 + \delta_1), (1 + \delta_2)\}L/m \le 1.916 \max_{1 \le t \le n} p_t \le 1.916 \cdot OPT(\sigma)$.

Therefore we can restrict ourselves to $l_{1,1}^n > (c_1 - 1)L/m$. There are two cases to consider. (1) The last job of $M_{2,m}^n$ was scheduled on the $(k_2+1)$-st smallest machine at the time of the assignment. This case is analyzed in Section *4.1.1*. (2) The last job of $M_{2,m}^n$ was scheduled on the smallest machine at the time of the assignment. This case is analyzed in Section *4.1.2*.

### *4.1.1 The last job on $M_{2,m}^n$ was scheduled on the $(k_2 + 1)$-st smallest machine*

Suppose that the last job on $M_{2,m}^n$ was scheduled at time $t$ and that in this scheduling step $\gamma \in \{c_2 L^t/m, c_2 \beta^{-1} \lambda_1^t/m\}$, which means that the load $l_{2,m}^n$ is bounded by that value of $\gamma$. If $\gamma = c_2 L^t/m$, then $A_1$'s schedule was not balanced, i.e. $\lambda_1^t \ge \beta L^t$, and thus $\gamma \le c_2 \beta^{-1} \lambda_1^t/m$. Since the $\lambda_1$-values cannot decrease over time, $l_{2,m}^n \le c_2 \beta^{-1} \lambda_1^n/m$. We estimate a combination of the load on the smallest machine in $A_1$'s schedule and the largest machine in $A_2$'s schedule. We have

$$q l_{1,1}^n + (1 - q) l_{2,m}^n \le q(L - \lambda_1^n)/k_1 + (1 - q) c_2 \beta^{-1} \lambda_1^n/m$$

because $l_{1,1}^n$ cannot be larger than the average load on the $k_1$ smallest machines. In the last expression the total factor of $\lambda_1^t$, which is $-q/k_1 + (1 - q)c_2\beta^{-1}/m$, is positive. Since $l_{1,1}^n > (c_1 - 1)L/m$, we have $\lambda_1^n \le \beta L$ and hence

$$\begin{aligned} q l_{1,1}^n + (1 - q) l_{2,m}^n &\le q(L - (1 - (c_1 - 1)(k_1/m))L)/k_1 \\ &\quad + (1 - q) c_2 L/m \\ &= q(c_1 - 1)L/m + (1 - q)c_2 L/m \\ &= 1.916 L/m - qL/m. \end{aligned}$$

If $\mathrm{E}[Rand(\sigma)] = (1.916 + \delta)L/m$, for some strictly positive $\delta$, then the processing time $p_s$ of the last job on $M_{1,m}^n$ satisfies

$$\begin{aligned} p_s &\ge (1/q)((1.916 + \delta)L/m - ((1 - q) l_{2,m}^n + q l_{1,1}^n)) \\ &\ge (1/q)((1.916 + \delta)L/m - (1.916 L/m - qL/m)) \\ &= (1 + \delta/q)L/m \;=\; (1 + 2\delta)L/m \end{aligned}$$

and (1) is established because $OPT(\sigma) \ge (1 + 2\delta)L/m$.

We still have to consider the case that when the last job on $M_{2,m}^n$ was assigned at time $t$, $\gamma = c_2' L^t/m$. Let $c_1' = 1.947$ and note that $q c_1' + (1 - q) c_2' = 1.916$. If $l_{1,m}^n = (c_1' + \delta)L/m$, for some $\delta > 0$, the last job on $M_{1,m}^n$ was scheduled on the least loaded machine at that time. The expected makespan by *Rand* is $\mathrm{E}[Rand(\sigma)] = (1.916 + \delta/2)L/m$. We study the load of that machine immediately before the assignment of that job. If the load was bounded by $(c_1' - 1)L/m$, then we have again identified a job of size at least $(1 + \delta)L/m$ and (1) holds again. For the analysis of the case that the load was larger than $(c_1' - 1)L/m$, we need the following lemma, which holds for both $A_1$ and $A_2$. The proof is omitted.

LEMMA 1. *If at any time the least loaded machine of $A_i$, $i \in \{1, 2\}$, has a load of at least $(0.947 + \epsilon)\frac{L}{m}$, for some $\epsilon$ with $0 \le \epsilon \le 0.053$, then the job sequence scheduled so far contains $m$ jobs with a processing time of at least $(0.5 + \frac{0.5}{0.916}\epsilon)\frac{L}{m}$.*

Obviously, the last job on $M_{1,m}^n$ has processing time of at least $(c_1 - 1)L/m \ge (0.5 + \frac{0.5}{0.916}\epsilon)L/m$ for all $\epsilon \in [0, 0.053]$. Thus, if the least loaded machine of $A_1$ before the assignment of that job has a load of $(c_1' - 1 + \epsilon)L/m$, then using Lemma 1 we have identified $m + 1$ jobs of size at least $(0.5 + \frac{0.5}{0.916}\epsilon)L/m$, two of which must be scheduled on the same machine in OPT's schedule. Hence $OPT(\sigma) \ge \max\{1 + \delta - \epsilon, 1 + \epsilon/0.916\}L/m \ge \mathrm{E}[Rand(\sigma)]/1.916$, for all possible values of $\epsilon \in [0, 0.053]$.

### *4.1.2 The last job on $M_{2,m}^n$ was scheduled on the smallest machine*

We are left with analyzing the scenario that the last jobs on the largest machines $M_{1,m}^n$ and $M_{2,m}^n$ were scheduled on the smallest machines at the time of the assignment. Consider the last job $J_n$. We assume that its assignment changes the makespan in $A_1$'s or in $A_2$'s schedule since otherwise (1) follows from the induction hypothesis. Thus, $J_n$ is scheduled on the smallest machine in one of the schedules. We assume $l_{i,m}^n > c_1 L/m$, $i = 1, 2$, since otherwise there is nothing to show. Thus $J_n$ has a large processing time of at least $p_n \ge (c_1 - 1)L/m$.

Let $l_{1,1}^{n-1}$ and $l_{2,1}^{n-1}$ be the loads of the smallest machines immediately before the assignment of $J_n$. If the expected load on the smallest machine is $q l_{1,1}^{n-1} + (1-q) l_{2,1}^{n-1} \le 0.916\frac{L}{m}$, then (1) is easy to prove. If the makespan of $A_i$'s schedule satisfies $l_{i,m}^n \le l_{i,1}^{n-1} + L/m$, then $\mathrm{E}[Rand(\sigma)] \le 1.916 L/m \le 1.916 \cdot OPT(\sigma)$. If $l_{i,m}^n = l_{i,1}^{n-1} + L/m + \delta_i L/m$, for some $i$ and positive $\delta_i$, then $\sigma$ contains a job of size $(1 + \delta)L/m$, where $\delta = \max_{i=1,2} \delta_i$ and $\mathrm{E}[Rand(\sigma)] \le (1.916 + \delta)L/m \le 1.916 \max_{1 \le t \le n} p_t \le OPT(\sigma)$.

If the expected load on the smallest machine is greater than $0.916 L/m$, then one of the loads must be greater than $0.916 L/m$. We distinguish two cases, depending on which of the two loads is higher.

*Case 1:* $l_{2,1}^{n-1} \ge l_{1,1}^{n-1}$ and $l_{2,1}^{n-1} > 0.916 L/m$.

*Case 2:* $l_{1,1}^{n-1} > l_{2,1}^{n-1}$ and $l_{1,1}^{n-1} > 0.916 L/m$.

**Analysis of Case 1:** Let $l_{2,1}^{n-1} = (0.916 + \epsilon)L/m$, for some $0 < \epsilon \le 0.084$, and $l_{1,1}^{n-1} > (0.916 - \epsilon)L/m$. The next lemma, which we will prove in Section 4.3, is crucial.

MAIN LEMMA 1. *Let $0 \le \epsilon \le 0.084$. If at time $n - 1$ the least loaded machines of $A_1$ and $A_2$ satisfy $l_{2,1}^{n-1} = (0.916 + \epsilon)\frac{L}{m}$ and $l_{1,1}^{n-1} \ge \max\{0.916 - \epsilon, 0.885\}\frac{L}{m}$, then the job sequence $J_1, \ldots, J_{n-1}$ contains $m$ jobs with a processing time of at least $(0.5 + \frac{0.5}{0.916}\epsilon)\frac{L}{m}$.*

If $l_{1,1}^{n-1} \ge 0.885 L/m$, then Main Lemma 1 and the fact that $p_n \ge 0.832 L/m$ imply that $\sigma$ contains $m+1$ jobs with a processing time of $(0.5 + \frac{0.5}{0.916}\epsilon)L/m$. Let $\delta = \max_{i=1,2}\{0, (l_{i,m}^n - l_{i,1}^{n-1} - L/m)/(m/L)\}$. Then $\mathrm{E}[Rand(\sigma)] \le (1.916 + \epsilon + \delta)L/m \le 1.916 \max\{1 + \epsilon/0.916, 1 + \delta\}L/m \le 1.916 \cdot OPT(\sigma)$.

If $l_{1,1}^{n-1} < 0.885 L/m$, then $\epsilon > 0.916 - 0.885 = 0.031$ and $\mathrm{E}[Rand(\sigma)] \le (1.916 + (1 - q)(\epsilon - 0.031) + \delta)L/m < (1.916 + \bar{\epsilon} + \delta)L/m$, with $\bar{\epsilon} := \epsilon - 0.031$ and the same definition of $\delta$ as before. Using Lemma 1, we obtain that $\sigma$ contains $m + 1$ jobs with a processing time of at least

$(0.5+\frac{0.5}{0.916}\bar{\epsilon})L/m$ and hence $OPT(\sigma) \geq \max\{1+\bar{\epsilon}/0.916, 1+\delta\}L/m \geq \mathrm{E}[Rand(\sigma)]/1.916$.

**Analysis of Case 2:** Let $l_{1,1}^{n-1} = (0.916+\epsilon)L/m$, for some $0 < \epsilon \leq 0.084$, and let $l_{2,1}^{n-1} > (0.916-\epsilon)L/m$. We need the following lemma whose proof we sketch in Section 4.4.

Main Lemma 2. *Let $0 \leq \epsilon \leq 0.084$, $0 \leq \epsilon' \leq \min\{0.084+\epsilon, 0.115\}$. If at time $n-1$ the least loaded machines of $A_1$ and $A_2$ satisfy $l_{1,1}^{n-1} = (0.916+\epsilon)\frac{L}{m}$ and $l_{2,1}^{n-1} \geq (\max\{0.916-\epsilon, 0.885\}+\epsilon')\frac{L}{m}$, then the sequence $J_1, \ldots, J_{n-1}$ contains $m$ jobs with a processing time of at least $(0.5+\frac{0.5}{0.916}\min\{\epsilon,\epsilon'\})\frac{L}{m}$.*

We first assume $0 < \epsilon \leq 0.031$. Suppose that $l_{2,1}^{n-1} = (0.916-\epsilon+\epsilon')L/m$, for some $\epsilon' > 0$. Note that $\epsilon' < 2\epsilon$ because $l_{1,1}^{n-1} > l_{2,1}^{n-1}$. Set $\bar{\epsilon} = \min\{\epsilon,\epsilon'\}$ and define $\delta$ as in the analysis of Case 1. Then $\mathrm{E}[Rand(\sigma)] \leq (1.916+(1-q)\epsilon'+\delta)L/m < (1.916+\bar{\epsilon}+\delta)L/m$ and Main Lemma 2 ensures that $OPT(\sigma) \geq \max\{1+\bar{\epsilon}/0.916, 1+\delta\}L/m$. Equation (1) follows.

Now suppose that $\epsilon > 0.031$ and $l_{2,1}^{n-1} = (0.885+\epsilon')L/m$. If $\epsilon \leq \epsilon'$, then $\mathrm{E}[Rand(\sigma)] \leq (1.916+\epsilon+\delta)L/m$ because $l_{1,1}^{n-1} > l_{2,1}^{n-1}$ and Main Lemma 2 ensures that $OPT(\sigma) \geq \max\{1+\epsilon/0.916, 1+\delta\}L/m$. If $\epsilon > \epsilon'$, then $\mathrm{E}[Rand(\sigma)] \leq (1.916+q(\epsilon-0.031)+(1-q)\epsilon'+\delta)L/m \leq (1.916+\bar{\epsilon}+\delta)L/m$, with $\bar{\epsilon} = \max\{\epsilon-0.031,\epsilon'\}$. Main Lemma 2 ensures the existence of $m+1$ jobs of size $(0.5+\frac{0.5}{0.916}\epsilon')L/m$ and Lemma 1 ensures the existence of $m+1$ jobs of size $(0.5+\frac{0.5}{0.916}(\epsilon-0.031))L/m$. Thus $OPT(\sigma) \geq \max\{1+\bar{\epsilon}/0.916, 1+\delta\}L/m$. The final case $\epsilon > 0.031$ and $l_{2,1}^{n-1} < 0.885L/m$ can be handled in the same way as in Case 1.

## 4.2 Basic properties and concepts

It remains to prove Main Lemmas 1 and 2. This section presents important statements and concepts needed in both of the proofs. The proofs of the lemmas are given in the appendix. In Main Lemmas 1 and 2 we have to investigate job sequences $\sigma = J_1, \ldots, J_n$ leading to one of the following scenarios.

(S1) At time $n-1$ the least loaded machines of $A_1$ and $A_2$ satisfy $l_{2,1}^{n-1} = (0.916+\epsilon)L/m$ and $l_{1,1}^{n-1} \geq \max\{0.916-\epsilon, 0.885\}L/m$, for some $\epsilon$ with $0 \leq \epsilon \leq 0.084$.

(S2) At time $n-1$ the least loaded machines of $A_1$ and $A_2$ satisfy $l_{1,1}^{n-1} = (0.916+\epsilon)L/m$ and $l_{2,1}^{n-1} \geq (\max\{0.916-\epsilon, 0.885\}+\epsilon')L/m$, for some $\epsilon$ and $\epsilon'$ with $0 \leq \epsilon \leq 0.084$ and $0 \leq \epsilon' \leq \min\{0.084+\epsilon, 0.115\}$

Given a job sequence leading to (S1), at a any time $t$, $1 \leq t \leq n-1$, a machine of $A_2$ is called full if its load is at least $(0.916+\epsilon)\frac{L}{m}$. A machine of $A_1$ is full its load its load is at least $\max\{0.916-\epsilon, 0.885\}\frac{L}{m}$. For sequences leading to (S2), the definition is similar. A machine of $A_1$ is full if its load is at least $(0.916+\epsilon)\frac{L}{m}$, and a machine of $A_2$ is full if the load is at least $(\max\{0.916-\epsilon, 0.885\}+\epsilon')\frac{L}{m}$. The following lemma implies that the job sequences to be investigated generate critical schedules.

Lemma 2. *If the least loaded machine of $A_i$, $i \in \{1,2\}$, has a load of at least $0.832L/m$, then $A_i$'s schedule is critical. For $i=1$, $A_1$'s schedule is also balanced.*

In a job sequence leading to (S1) or (S2), let $t_i^c$, $i \in \{1,2\}$, be the first point in time such that at least $m-k_i$ machines of $A_i$ are full and the schedule of $A_i$ is critical throughout the time interval $[t_i^c, n-1]$. Let $t^b$ be the *first* point in time with $t_2^c \leq t^b \leq n-1$ such that $A_1$'s schedule was balanced at time $t^b$.

In our analyses we will often have to analyze the total load all the machines at some time $t > t_i^c$. We present a general lemma that will be helpful in estimating that load. Given a schedule of $A_i$, $i \in \{1,2\}$, in which the $(k_i+1)$-st smallest machine has a load of at least $b$, define $L_i(l,b)$ as the total load on all machines except for the load in excess to level $b$ on machines $k_i+1, \ldots, k_i+l$. Figure 1 shows an example; $L_i(l,b)$ is shaded grey. On our analyses will use part a) of the lemma for the algorithm $A_2$ with $B = c_2'\beta^{-1}\lambda_1^t/m$. Part b) of the lemma will be used for $A_1$ with $c = c_1$ and for $A_2$ with $c = c_2$ or $c = c_2'$.

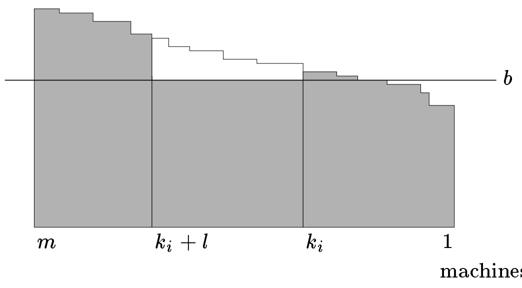

**Figure 1: The load counted in $L_i(l,b)$**

Lemma 3. *Consider $A_i$'s schedule at time $t \geq t_i^c$ and assume that exactly $m-j$ machines are full.*

*a) Suppose that in the next scheduling step, a job cannot be placed in the $(k_i+1)$-st smallest machine if the resulting load exceeds $B$. Then when $m-j+1$ machines are full, $L_i(l-1,b) > L_i(l,b)+B-b$, for any $1 \leq l \leq m-k_i$.*

*b) Suppose that in the next scheduling steps, a job cannot be placed on the $(k_i+1)$-st smallest machine if the resulting load exceeds $c$ times the average load on the machines. If $L_i(l,b) > X$, $1 \leq l \leq m-k_i$, then when exactly $m-j+h$ machines are full $L_i(l-h,b) > (X-bm/c)(1-c/m)^{-h}+bm/c$.*

Lemma 4. *In job sequences leading to (S1) or (S2) at least $m-k_1$ machines of $A_1$ are full at time $t^b$.*

Lemma 5. *Consider a job sequence leading to (S1) or (S2) at some time $t$ with $t > t_2^c$. If $A_2$ cannot schedule $J_t$ on the machine with the $(k_2+1)$-st smallest load, then $p_t \geq (0.5+\frac{0.5}{0.916}\bar{\epsilon})\frac{L}{m}$, where $\bar{\epsilon} = \epsilon$ for sequences leading to (S1) and $\bar{\epsilon} = \epsilon'$ for sequences leading to (S2).*

## 4.3 Proof of Main Lemma 1

We show that each time a machine of $A_2$ becomes full, a job of size at least $(0.5+0.5\epsilon/0.916)L/m$ is scheduled. Since $m$ machines are full at time $n-1$, Main Lemma 1 follows. First consider any time $t > t_2^c$. There are at least $m-k_2$ full machines and hence another full machine can only be created by scheduling a job on the smallest machine. Lemma 5 ensures that the size of the job is at least $(0.5+0.5\epsilon/0.916)L/m$. Next we consider any time $t$, $1 \leq t \leq t_2^c$, and show that whenever another machine becomes full, the job is scheduled on a machine whose load is smaller than $\alpha_2(0.916+\epsilon)L/m$. Thus the size of the job is $p_t > (0.916+\epsilon)L/m - \alpha_2(0.916+\epsilon)L/m > (0.5+0.5\epsilon/0.916)L/m$.

LEMMA 6. *At time $t_2^c$ the average load on the non-full machines of $A_2$ is smaller than $\alpha_2(0.916+\epsilon)L/m$.*

We first finish the proof of Main Lemma 1 and then prove Lemma 6. Let $t'$, $t' \le t_2^c$, be the last point in time when exactly $m-k_2$ machines are full and let $t''$, $t'' \le t'$, be the last point in time when exactly $m-2k_2$ machines are full. At any time $t$, $t' < t \le t_2^c$, a full machine can only be generated by scheduling a job on the smallest machine. Lemma 6 ensures that its load is bounded by $\alpha_2(0.916+\epsilon)L/m$. At time $t$, $t'' \le t \le t'$, the schedule is not critical because $l^t_{2,2k_2+1} \ge (0.916+\epsilon)L/m$ and, by Lemma 6, the average load on the $k_2$ smallest machines is bounded by $\alpha_2$ times this value. Thus jobs are always scheduled on the smallest machine. At time $t''$ the load on the $(k_2+1)$-st smallest machine cannot be larger than the load on the smallest machine at time $t'$, which is at most $\alpha_2(0.916+\epsilon)L/m$. Thus at any time $t$, $1 \le t \le t''$, both machines a job can be assigned to have a low load.

The rest of this section is devoted to proving Lemma 6. The proof is by contradiction. We assume that at time $t_2^c$ the average load on the non-full machine is at least $\alpha_2(0.916+\epsilon)L/m$ and show that this would imply a load of at least $L$ at time $n-1$. This is a contradiction because at time $n$ another job with non-zero processing time is presented and hence $L^{n-1} < L^n = L$.

Let $b_1(\epsilon) = \max\{0.916-\epsilon, 0.885\}L/m$ and $b_2(\epsilon) = (0.916+\epsilon)L/m$. We define several load values $L_1(\epsilon), L_2(\epsilon), L_3(\epsilon)$ and $L_4(\epsilon)$. Essentially, $L_1(\epsilon)$ is the minimum load in the system at time $t_2^c$: At least $m-k_2$ machines are full and by assumption the average load on the non-full machines is at least $\alpha_2(0.916+\epsilon)L/m$. Thus the total load is at least $(m-k_2)b_2(\epsilon) + k_2\alpha_2 b_2(\epsilon) = b_2(\epsilon)m(1+(\alpha_2-1)k_2/m)$. The value $L_2(\epsilon)$ is the minimum load in the system when $A_1$ has a balanced schedule. By Lemma 4 at least $m-k_1$ machines of $A_1$ are full; thus the total load is at least $L_2(\epsilon) = (m-k_1)b_1(\epsilon)\beta^{-1}$. We set $L_3(\epsilon) = (c_2/c_2')L_2(\epsilon)$. Intuitively, this is the load at which the value of $\gamma$ changes in the scheduling process. While the load in the system is smaller than $L_3(\epsilon)$, the value $\gamma = c_2\beta^{-1}\lambda_1^t/m$ dominates $\gamma = c_2' L^t/m$. Afterwards the latter value dominates. Finally, we set $L_4(\epsilon) = L$. This is the final value we want to reach.

$$\begin{aligned}
L_1(\epsilon) &= b_2(\epsilon)m(1+(\alpha_2-1)k_2/m)\\
L_2(\epsilon) &= (m-k_1)b_1(\epsilon)\beta^{-1}\\
L_3(\epsilon) &= (c_2/c_2')L_1(\epsilon)\\
L_4(\epsilon) &= L
\end{aligned}$$

Moreover, we define values $n_1(\epsilon), n_2(\epsilon)$ and $n_3(\epsilon)$. Given a load of $L_i(\epsilon)$, $n_i(\epsilon)$ is the maximum number of large jobs we need to reach a load of $L_{i+1}(\epsilon)$, for $i = 1,2,3$.

$$\begin{aligned}
n_1(\epsilon) &= -\log_B\left(\frac{L_2(\epsilon)-b_2(\epsilon)m/c_2}{L_1(\epsilon)-b_2(\epsilon)m/c_2}\right)\\
&\quad\text{with base } B=(1-c_2/m)\\
n_2(\epsilon) &= (L_3(\epsilon)-L_2(\epsilon))/(c_2\frac{L_2(\epsilon)}{m} - b_2(\epsilon))\\
n_3(\epsilon) &= -\log_{B'}\left(\frac{L_4(\epsilon)-b_2(\epsilon)m/c_2'}{L_3(\epsilon)-b_2(\epsilon)m/c_2'}\right)\\
&\quad\text{with base } B'=(1-c_2'/m)
\end{aligned}$$

Note that $(L_1(\epsilon)-b_2(\epsilon)m/c_2)(1-c_2/m)^{-n_1(\epsilon)} + b_2(\epsilon)m/c_2 = L_2(\epsilon)$. If we have a load of at least $L_2(\epsilon)$ and a job cannot be scheduled on the $(k_2+1)$-st smallest machine in $A_2$'s schedule, then the processing time of the job must be at least $c_2L_2(\epsilon)/m - b_2(\epsilon)$. Thus after at most $n_2(\epsilon)$ large jobs, a load of $L_3(\epsilon)$ is reached. Finally $(L_3(\epsilon) - b_2(\epsilon)m/c_2')(1-c_2'/m)^{-n_3(\epsilon)} + b_2(\epsilon)m/c_2' = L_4(\epsilon) = L$. Let $N_i(\epsilon) = \sum_{j=1}^{i}\lceil n_j(\epsilon)\rceil$, for $j=1,2,3$. Analyzing the first derivatives of the functions $n_i(\epsilon)$, $1 \le i \le 3$, we can show that $N_3(\epsilon)$ is non-increasing in $\epsilon$. The expression $n_1(\epsilon)$ is a concave decreasing function, whereas $n_2(\epsilon)$ and $n_2(\epsilon)$ are convex increasing functions. For $0 \le \epsilon \le 0.031$, the gradient of $n_1(\epsilon)$ is smaller than $-2.3$. The sum $n_2(\epsilon)+n_3(\epsilon)$ can be bounded by a linear function whose gradient is bounded by 1.8. Thus $N_3(\epsilon)$ is non-increasing. For $\epsilon \ge 0.031$ the gradient values changes slightly because $b_1(\epsilon)$ is constant. Here the gradient of $n_1(\epsilon)$ is smaller than $-2.5$ and $n_2(\epsilon) + n_3(\epsilon)$ can be bounded by a linear function whose gradient is smaller than 0.37. Again, $N_3(\epsilon)$ is non-increasing. Evaluating the function for $\epsilon = 0$, we find that $N_3(0) \le \frac{3}{8}m \le k_2$. To prove that the total load in $A_2$'s schedule at time $n-1$ is at least $L$, we define a non-decreasing function $f$ with $f(N_3(\epsilon)) \ge L$. Claim 1 below states that, at any time $t \ge t_2^c$, when exactly $m-k_2+i$ machines of $A_2$ are full, $0 \le i \le k_2$, the load $L_2(k_2-i, b_2(\epsilon)) \ge f(i)$. This shows that when $m$ machines are full $L^{n-1} \ge L_2(0, b_2(\epsilon)) \ge f(k_2) \ge f(N_3(\epsilon)) \ge L$.

$$f(i) = \begin{cases}
\min\{(L_1(\epsilon)-\frac{b_2(\epsilon)m}{c_2})(1-\frac{c_2}{m})^{-i} + \frac{b_2(\epsilon)m}{c_2}, L_2(\epsilon)\} & \\
\qquad\qquad \text{for } 0 \le i \le N_1(\epsilon) & \\
\min\{L_2(\epsilon) + (i-N_1(\epsilon))(c_2\frac{L_2(\epsilon)}{m} - b_2(\epsilon)), L_3(\epsilon)\} & \\
\qquad\qquad \text{for } N_1(\epsilon) < i \le N_2(\epsilon) & \\
\min\{(L_3(\epsilon)-\frac{b_2(\epsilon)m}{c_2'})(1-\frac{c_2'}{m})^{N_2(\epsilon)-i} + \frac{b_2(\epsilon)m}{c_2'}, L\} & \\
\qquad\qquad \text{for } N_2(\epsilon) < i &
\end{cases}$$

CLAIM 1. *Consider a time $t$, $t \ge t_2^c$. If exactly $m-k_2+i$ machines of $A_2$ are full, $0 \le i \le k_2$, then $L_2(k_2-i, b_2(\epsilon)) \ge f(i)$.*

PROOF. Suppose that at time $t_2^c$ exactly $m-k_2+i_0$ machines of $A_2$ are full, $0 \le i_0 \le k_2$. Below we will show that the load of the schedule satisfies $L_2(k_2-i_0, b_2(\epsilon)) \ge f(i_0)$. Given this fact, we first prove inductively that when exactly $m-k_2+i$ machines of $A_2$ are full, $i \ge i_0$, then $L_2(k_2-i, b_2(\epsilon)) \ge f(i)$. This establishes the claim.

So suppose that $L_2(k_2-i, b_2(\epsilon)) \ge f(i)$ and consider the next scheduling step when another machine becomes full. The job is scheduled on the smallest machine while the algorithm would prefer to assign the job to the $(k_2+1)$-st smallest machine because $A_2$'s schedule is critical after scheduling step. We distinguish cases depending on the value of $i$.

*Case 1:* $0 \le i < N_1(\epsilon)$ If the schedule of $A_1$ is not balanced, then $A_2$ sets $\gamma = c_2L^t/m$. Lemma 3 part b) and the induction hypothesis imply

$$\begin{aligned}
&L_2(k_2-(i+1), b_2(\epsilon))\\
&\quad= L_2(k_2-i-1, b_2(\epsilon))\\
&\quad\ge (L_2(k_2-i, b_2(\epsilon)) - b_2(\epsilon)m/c_2)/(1-c_2/m)\\
&\quad\quad + b_2(\epsilon)m/c_2\\
&\quad\ge (L_1(\epsilon) - b_2(\epsilon)m/c_2)/(1-c_2/m)^{-(i+1)} + b_2(\epsilon)m/c_2\\
&\quad\ge f(i+1).
\end{aligned}$$

If the schedule of $A_1$ is balanced, then $\gamma \geq c_2\beta^{-1}\lambda_1^t \geq c_2L_2(\epsilon)$ because by Lemma 4 at least $m-k_1$ machines of $A_1$ are full and have a load of $b_1(\epsilon)$. Thus, by Lemma 3 part a), $L_2(k-(i+1), b_2(\epsilon)) - L_2(k_2-i, b_2(\epsilon)) \geq c_2L_2(\epsilon)/m - b_2(\epsilon)$. We show that $f(i+1) - f(i) \leq c_2L_2(\epsilon)/m - b_2(\epsilon)$. If $i = N_1(\epsilon) - 1$, then

$$\begin{aligned}
&f(i+1) - f(i)\\
&\quad\leq L_2(\epsilon) - ((L_1(\epsilon) - b_2(\epsilon)m/c_2)(1-c_2/m)^{-N_1(\epsilon)+1}\\
&\qquad + b_2(\epsilon)m/c_2)\\
&\quad\leq L_2(\epsilon) - ((L_2(\epsilon) - b_2(\epsilon)m/c_2)(1-c_2/m) + b_2(\epsilon)m/c_2)\\
&\quad= c_2L_2(\epsilon)/m - b_2(\epsilon).
\end{aligned}$$

If $i < N_1(\epsilon) - 1$, then

$$\begin{aligned}
&f(i+1) - f(i)\\
&\quad\leq (L_1(\epsilon) - b_2(\epsilon)m/c_2)(1-c_2/m)^{-i}((1-c_2/m)^{-1} - 1)\\
&\quad= (L_1(\epsilon) - b_2(\epsilon)m/c_2)(1-c_2/m)^{-(i+1)}\tfrac{c_2}{m}\\
&\quad\leq ((L_1(\epsilon) - b_2(\epsilon)m/c_2)(1-c_2/m)^{-n_1(\epsilon)} + b_2(\epsilon)m/c_2\\
&\qquad - b_2(\epsilon)m/c_2)\tfrac{c_2}{m}\\
&\quad= (L_2(\epsilon) - b_2(\epsilon)m/c_2)\tfrac{c_2}{m}\\
&\quad= c_2L_2(\epsilon)/m - b_2(\epsilon).
\end{aligned}$$

*Case 2:* $N_1(\epsilon) \leq i < N_2(\epsilon)$ By induction hypothesis we know $L_2(k-i, b_2(\epsilon)) \geq L_2(\epsilon)$. If the schedule of $A_1$ is not balanced, then $\gamma = c_2L^t/m \geq c_2L_2(\epsilon)/m$. If the schedule of $A_1$ is balanced, then $\gamma = c_2\beta^{-1}\lambda_1^t/m \geq c_2L_2(\epsilon)/m$. Thus by Lemma 3 part b, $L_2(k-(i+1), b_2(\epsilon)) - L_2(k-i, b_2(\epsilon)) \geq \gamma - b_2(\epsilon) \geq c_2L_2(\epsilon)/m - b_2(\epsilon)$ and $f(i+1) - f(i)$ is at most this value.

*Case 3:* $N_2(\epsilon) \leq i$ By induction hypothesis, $f(N_2(\epsilon)) \geq L_3(\epsilon)$. In each of $A_2$'s scheduling steps $\gamma \geq c_2'L^t/m$. The inductive step now follows immediately from Lemma 3 part b).

It remains to show that at time $t_2^c$, $L_2(k_2 - i_0, b_2(\epsilon)) \geq f(i_0)$, where $m - k_2 + i_0$ is the number of full machines. By assumption, the non-full machines have an average load of at least $\alpha_2 b_2(\epsilon)$. Thus the average load on the $k_2$ smallest machines is at least $(i_0b_2(\epsilon) + (k_2 - i_0)\alpha_2b_2(\epsilon))/k_2 = b_2(\epsilon)(1 + (k_2 - i_0)(\alpha_2 - 1)/k_2)$. The schedule was not critical at time $t_2^c - 1$ and hence the load on machine $(2k_2+1)$ is asymptotically at least $1/\alpha_2$ times this value. Thus $L_2(k_2 - i_0, b_2(\epsilon)) \geq L_2(k_2, b_2(\epsilon))$ is at least $g(i_0)$ where

$$\begin{aligned}
&g(i) =\\
&\quad b_2(\epsilon)(1 + (k_2 - i)(\alpha_2 - 1)/k_2)(k_2 + (m - 2k_2)/\alpha_2)\\
&\quad + k_2b_2(\epsilon).
\end{aligned}$$

We show $g(i) \geq f(i)$ for all $0 \leq i \leq k_2$. For $i = 0$ we have

$$\begin{aligned}
g(0) &= b_2(\epsilon)k_2\alpha_2 + (m - k_2)b_2(\epsilon)\\
&= b_2(\epsilon)m(1 + (\alpha_2 - 1)k_2/m))\\
&= L_1(\epsilon)\\
&= f(0).
\end{aligned}$$

In each step the function $g$ increases by $b_2(\epsilon)(1-\alpha_2)(1 + (m - 2k_2)/(\alpha_2k_2)) > 1.2L/m$. We show that the function $f$ increases by at most this value in each step. As shown in Case 1 above, $f(i+1) - f(i) \leq c_2L_2(\epsilon)/m - b_2(\epsilon) \leq 1.1L/m$ if $i \leq N_1(\epsilon) - 1$. The same calculation holds if $N_1(\epsilon) \leq i < N_2(\epsilon)$. For $i \geq N_2(\epsilon)$ we can show as in Case 1, $f(i+1) - f(i) \leq c_2'L_4(\epsilon)/m - b_2(\epsilon)$ and this expression is bounded by $L/m$. □

## 4.4 Proof of Main Lemma 2

Due to space limitations, we only give a sketch of the proof. We have to consider job sequences leading to scenario (S2). To identify large jobs in the sequence we need the following lemmas.

LEMMA 7. *a) If $0 \leq \epsilon \leq 0.007$, then at time $t_2^c$ the average load on the non-full machines of $A_2$ is smaller than $\alpha_2(0.916 - \epsilon)L/m$.*

*b) If $\epsilon \geq 0.007$, the one of the following statement holds: (i) at time $t_1^c$ the average load on the non-full machines of $A_1$ is smaller than $(0.416 + 0.416\epsilon/0.916)L/m$; or (ii) at time $t_2^c$ the average load on the non-full machines of $A_2$ is smaller than $(0.416 - \epsilon)L/m$.*

LEMMA 8. *If at some time $t > t_1^c$, the $(k_1+1)$-st smallest machine of $A_1$ has a load of $(0.916 + d)L/m$, for some $0.007 \leq d \leq 0.084$, and $A_1$ cannot schedule $J_t$ on the machine with the $(k_1+1)$-st smallest load, then $p_t \geq (0.5 + \frac{0.5}{0.916}d)\frac{L}{m}$.*

Using these two lemmas, the identification of large jobs can be done using the same techniques as in the proof of Main Lemma 1. The difficult part of the analysis is to prove part b) of Lemma 7, where we need arguments not yet seen in this paper. Consider an $\epsilon$ with $0.007 \leq \epsilon \leq 0.084$. The proof is again by contradiction. We assume that at time $t_1^c$, the average load on the non-full machines of $A_1$ is at least $(0.416 + 0.416\epsilon/0.916)L/m$ and that at time $t_2^c$, the average load on the non-full machines of $A_2$ is at least $(0.416 - \epsilon)L/m$. We show that these assumptions imply a load of at least $L$ at time $n-1$.

The global structure of the proof is similar to that of Lemma 6, but the technical details differ. Let $b_1(\epsilon) = (0.916 + \epsilon)L/m$ and $b_2(\epsilon) = \max\{0.916 - \epsilon, 0.885\}L/m$. Define $\bar{\epsilon} = \min\{\epsilon, 0.031\}$. Note that at $\epsilon = 0.031$, $b_2(\epsilon)$ becomes constant. Again we define a function $f$ that describes the minimum load in the system when exactly $m - k_2 + i$ machines of $A_2$ are full. Again we need a sequence of loads $L_i(\epsilon)$, $i = 1, \ldots, 4$. In the proof of Lemma 6, $L_1(\epsilon)$ was the load in the system when exactly $m - k_2$ machines of $A_2$ were full, which was also the minimum load in the system when $A_2$'s schedule was critical. For sequences leading to (S2) these two values are different. Here $L_1(\epsilon) = b_2(\epsilon) - k_2 \cdot 0.5L/m$ is the minimum load in the system when exactly $m - k_2$ machines are full. The minimum load in the system when $A_2$'s schedule is critical is $L_2(\epsilon) = b_2(\epsilon)m(1 + (\alpha_2 - 1)k_2/m))$. The value $L_3(\epsilon) = (m - k_1)b_1(\bar{\epsilon})\beta^{-1}$ is the minimum load when $A_1$'s schedule is balanced. For $\epsilon > 0.031$ it is sufficient to work with the smaller value $\bar{\epsilon}$. We do not need the load value $(c_2/c_2')L_3(\epsilon)$ in this analysis.

The main difference in the definitions of the loads is that $L_4(\epsilon)$ is not the final value $L$ but a smaller value, i.e. $L_4(\epsilon) = L - 0.5d(\epsilon)L/m$ where $d(\epsilon) = \lceil(2.933(\bar{\epsilon} - 0.007) + 0.0083)m\rceil$. It turns out that when $m$ machines are full in $A_2$'s schedule, the load is not necessarily $L$ but only $L_4(\epsilon)$ defined above. Nonetheless we are able to derive a contradiction. The intuition is as follows. At time $n-1$, the smallest machine of $A_1$ has a higher load than the smallest machine of $A_2$. Thus during the scheduling process it takes longer to generate full machines in $A_1$'s schedule. In a typical situation after time $t_2^c$, when $i$ machines of $A_2$ are full, only $i - d(\epsilon)$ machines of $A_1$ are full. Thus we get an additional load from jobs needed to fill all the machines of $A_1$. We can show that the critical times satisfy $t_2^c \le t_1^c$. Lemma 8 then implies that at any time after $t_1^c$ when another machines of $A_1$ becomes full, a job of size at least $0.5L/m$ is schedules. We obtain a total load of at least $L$ at times $n-1$ and have the desired contradiction. This idea can be turned into a formal proof; details are presented in the full paper.

## 5. APPENDIX

PROOF OF LEMMA 2. Each machine of $A_i$, $i \in \{1,2\}$, has a load of at least $0.832L/m$. Thus the average load on the $k_i$ smallest machines is $\mu_i \ge 0.832L/m$. If $A_i$'s schedule were not critical, then the load on the $(2k_i+1)$-st smallest machine would be at least $\mu_i/\alpha_i$ and the total load on all machines would be at least $(m-2k_i)\mu_i/\alpha_i + 2k_i\mu_i = 0.832L + (m-2k_i)(1/\alpha_i - 1)0.832L/m > L$ as $m \to \infty$, which is a contradiction. For the proof that $A_1$'s schedule is balanced we simply observe that the load on the $k_1$ smallest machines is at least $0.832k_1L/m \ge (c_1-1)k_1L^t/m$ for any $1 \le t \le n$. □

PROOF OF LEMMA 3. a) Let $e_{k_i+1}$ and $e_{k_i+l}$ be the loads in excess to $b$ on the $(k_i+1)$-st and the $(k_i+l)$-th smallest machines, respectively. Let $\Phi_l$ be the total load in the schedule not counted in $L_i(l,b)$. When the next machine becomes full, the job to be scheduled has a processing time $p$ with $b + e_{k_i+1} + p > B$, i.e. $p + e_{k_i+1} > B - b$. We show that $L_i(l-1,b) - L_i(l,b) \ge p + e_{k_i+1}$. The job is scheduled on the smallest machine in the schedule. The machines are sorted in order of non-decreasing loads after the assignment. If the smallest machine is among the $k_i$ smallest machines after the assignment, then $p$ is fully counted in $L_i(l-1,b)$ and the $\Phi$-value decreases by $e_{k_i+l} \ge e_{k_i+1}$ when going from $L_i(l,b)$ to $L_i(l-1,b)$. Suppose that the smallest machine is among the $m-k_i$ largest machines after the sorting, i.e. its load is $b+e$ for some $e>0$. If the machine is among machines $k_i+l,\dots,m$ (more formally, among machines $M_{i,k_i+l},\dots,M_{i,m}$) after the sorting, then $p$ is fully counted in $L_i(l-1,b)$ and the $\Phi$-value drops by $e_{k_i+1}$ when moving from $L_i(l,b)$ to $L_i(l-1,b)$. If the smallest machine is among machines $k_i+1,\dots,k+l-1$ after the sorting, then the $\Phi$-value drops by $e_{k_i+1}$ during the sorting, an amount of $e$ is not counted in the processing time of $p$ but the $\Phi$-value drops by $e_{k+l} \ge e$ when making the transition from $L_i(l,b)$ to $L_i(l-1,b)$.

b) We prove the statement by induction on $h$. For $h=0$ there is nothing to show. Assume that the statement holds for $h-1$. Consider the point in time when the $(m-j+h)$-th machine becomes full and let $e_{k_i+1}$ be the load above level $b$ on the $(k_i+1)$-st smallest machine. Let $p$ be the processing time of the new job. Algorithm $A_i$ would prefer to assign the job to the $(k_i+1)$-st smallest machine because the schedule is critical. Since this is impossible $b + e_{k_i+1} + p > B$, i.e. $e_{k_i+1} + p > B - b$. The value $B$ is at least $c$ times the average load on the machines after the scheduling step, which means

$$\begin{aligned} B &\ge \tfrac{c}{m}(L_i(l-h+1,b) + e_{k_i+1} + p) \\ &\ge \tfrac{c}{m}(L_i(l-h+1,b) + B - b). \end{aligned}$$

Algebraic manipulations give $B \ge \frac{c}{m}(L_i(l-h+1,b)-b)/(1-\frac{c}{m})$ and $B-b \ge (\frac{c}{m}L_i(l-h+1,b)-b)/(1-\frac{c}{m})$. Using part a) and the induction hypothesis,

$$\begin{aligned} L_i(l-h,b) &\ge L_i(l-h+1,b)/(1-c/m) - b/(1-c/m) \\ &= (X - bm/c)/(1-c/m)^h + bm/c(1-c/m) \\ &\quad -b/(1-c/m) \\ &= (X-bm/c)/(1-c/m)^h + bm/c \end{aligned}$$

and this concludes the proof. □

PROOF OF LEMMA 4. We consider a general setting where we can analyze scenarios (S1) and (S2) simultaneously. Let $\epsilon \in [-0.084, 0.084]$ and set $b_1(\epsilon) = (0.916-\epsilon)L/m$ and $b_2(\epsilon) = (0.916+\epsilon)L/m$. For the analysis of scenario (S1), $\epsilon \ge 0$ and for the analysis of scenario (S2), $\epsilon \le 0$. We assume that less than $m-k_1$ machines are full and derive a contradiction.

We first analyze the average load $\mu_1(\epsilon)$ on the $k_1$ smallest machines of $A_1$ at time $t^b$. In the schedule of $A_2$, at least $m-k_2$ machines are full and the schedule is critical. Thus the total load is at least

$$L^{t^b} \ge (m-k_2)b_2(\epsilon) + k_2\alpha_2 b_2(\epsilon) = (m + (\alpha_2-1)k_2)b_2(\epsilon).$$

Since the schedule of $A_1$ is balanced, the average load on the $k_1$ smallest machines is at least $(c_1-1)L^{t^b}/m$, i.e.

$$\mu_1(\epsilon) = (c_1-1)(1+(\alpha_2-1)k_2/m)b_2(\epsilon).$$

When exactly $m-k_1$ machines of $A_1$ are full, the load $L_1(k_1, b_1(\epsilon))$ of $A_1$'s schedule is at least equal to $L_0(\epsilon) = (m-k_1)b_1(\epsilon) + k_1\mu_1(\epsilon)$. For $i = 0,\dots,k_1$, define

$$f(i,\epsilon) = (C \cdot L_0(\epsilon) - b_1(\epsilon)m/c_1)(1-c_1/m)^{-i} + b_1(\epsilon)m/c_1,$$

where $C = 0.96$. In the above definition of $f$ we scale the load $L_0(\epsilon)$, i.e. we consider a slightly smaller load.

Let $t'$ be the last point in time when exactly $m-k_1$ machines of $A_1$ are full. If $A_1$'s schedule is never non-critical at time $t \ge t'$, then the proof is simple. At time $t'$ we have $L_1(k_1, b_1(\epsilon)) \ge L_0(\epsilon)$ and hence, by Lemma 3 part b), when all the $m$ machines are full

$$L^{n-1} \ge L_1(k_1-k_1, b_1(\epsilon)) = L_1(0, b_1(\epsilon)) \ge f(k_1,\epsilon).$$

We have

$$f(k_1,\epsilon) \ge (1-\tfrac{c_1^2}{m})(C \cdot L_0(\epsilon) - \tfrac{b_1(\epsilon)m}{c_1})e^{k_1c_1/m} + \tfrac{b_1(\epsilon)m}{c_1}.$$

The last term is decreasing in $\epsilon$ and, for $\epsilon = 0.084$, it is strictly greater than $L$ as $m \to \infty$. Thus, if $A_1$'s schedule is never non-critical after time $t'$, then we have a contradiction.

If $A_1$'s schedule is non-critical at some time $t \ge t'$, then we have to analyze more carefully. Suppose that at time $t_1^c$, exactly $m-k_1+i$ machines are full, for some $1 \le i \le k_1$. We show that at this time the load $L_1(k_1, b_1(\epsilon))$ in the schedule satisfies $L_1(k_1, b_1(\epsilon)) \ge f(i,\epsilon)$. Lemma 3 part b) then gives that when all the $m$ machines are full $L_1(k_1-(k_1-i), b_1(\epsilon)) \ge f(k_1,\epsilon)$ and hence $L^{n-1} \ge L_1(k_1-(k_1-i), b_1(\epsilon)) \ge f(k_1,\epsilon) > L$. We have again a contradiction.

Thus we have to estimate the load $L_1(k_1, b_1(\epsilon))$ at time $t_1^c$ when exactly $m-k_1+i$ machines of $A_1$ are full. Among the $k_1$ smallest machines, $i$ machines are full and have a load of $b_1(\epsilon)$. Whenever one of the smallest $k_1$ machines becomes full, a job is assigned to the smallest machine. Thus the average load on the non-full machines at time $t_1^c$ cannot be smaller than the average load on the $k_1$ smallest machines at time $t^b$, which was $\mu_1(\epsilon)$. Hence at time $t_1^c$, the average load on the $k_1$ smallest machines is at least $(ib_1(\epsilon) + (k_1 - i)\mu_1(\epsilon))/k_1$. The load on each of the largest $m - 2k_1$ largest machines is asymptotically at least $1/\alpha_1$ times this value. Thus at time $t_1^c$ we have $L_1(k_1, b_1(\epsilon)) \geq g(i, \epsilon)$, where

$$\begin{aligned} g(i,\epsilon) &= (ib_1(\epsilon) + (k_1 - i)\mu_1(\epsilon))(1 + (m - 2k_1)/(k_1\alpha_1)) \\ &\quad + k_1 b_1(\epsilon). \end{aligned}$$

We show that for any fixed $\epsilon$, $g(i, \epsilon) \geq f(i, \epsilon)$. This establishes $L_1(k_1, b_1(\epsilon)) \geq f(i, \epsilon)$.

Both $f$ and $g$ are increasing in $i$; $f$ grows exponentially while $g$ grows linearly. If we can show that, for any fixed $\epsilon$, the boundary values satisfy $g(0, \epsilon) \geq f(0, \epsilon)$ and $g(k_1, \epsilon) \geq f(k_1, \epsilon)$, then $g(i, \epsilon) \geq f(i, \epsilon)$ holds for all $i$. The first inequality is easy to prove. Obviously $f(0, \epsilon) \leq CL_0(\epsilon)$ and

$$\begin{aligned} g(0,\epsilon) &= k_1\mu_1(\epsilon)(1 + (m - 2k_1)/(k_1\alpha_1)) + k_1 b_1(\epsilon) \\ &= k_1\mu_1(\epsilon) + \mu_1(\epsilon)(m - 2k_1)/\alpha_1 + k_1 b_1(\epsilon) \\ &> C((m - k_1)b_1(\epsilon) + k_1\mu_1(\epsilon)) \\ &= CL_0(\epsilon). \end{aligned}$$

For $i = k_1$,

$$f(k_1, \epsilon) = (C \cdot L_0(\epsilon) - b_1(\epsilon)m/c_1)(1 - c_1/m)^{-k_1} + b_1(\epsilon)m/c_1$$

and

$$g(k_1, \epsilon) = 2k_1 b_1(\epsilon) + (m - 2k_1)b_1(\epsilon)/\alpha_1.$$

Both functions are linear in $\epsilon$. Thus, to establish $g(k_1, \epsilon) \geq f(k_1, \epsilon)$ for all $\epsilon$, it suffices to check the boundary values $\epsilon = -0.084$ and $\epsilon = 0.084$. We have $g(k_1, -0.084) > 1.21L$, $f(k_1, -0.084) < 1.05L$, as well as $g(k_1, 0.084) > 1.01L$, $f(k_1, 0.084) < 1.01L$ as $m \to \infty$. $\square$

PROOF OF LEMMA 5. We first investigate the case that $A_1$'s schedule is not balanced, which implies that $\gamma = c_2 L^t/m$ in the scheduling step. Let $l_{2,k_2+1}^{t-1} = (0.885 + d)L/m$, for some $d > 0$, be the actual load on the $(k_2+1)$-st smallest machine of $A_2$ immediately before $J_t$ is scheduled. Each of the largest $m - k_2$ machines has a load of at least $(0.885 + d)L/m$. $A_2$'s schedule is critical and thus the average load on the $k_2$ smallest machines is at least $\alpha_2(0.885 + d)L/m$. Therefore

$$\begin{aligned} L^t &\geq (m - k_2)(0.885 + d)L/m + k_2\alpha_2(0.885 + d)L/m \\ &= (0.885 + d)(1 + (\alpha_2 - 1)k_2/m)L. \end{aligned}$$

Job $J_t$ cannot be placed on the $(k_2 + 1)$-st smallest machine, which implies $p_t + l_{2,k_2+1}^{t-1} > c_2 L^t/m$ and hence $p_t > c_2 L^t/m - l_{2,k_2+1}^{t-1}$. Thus

$$\begin{aligned} p_t &> c_2 0.885(1 + (\alpha_2 - 1)k_2/m)L/m - 0.885L/m \\ &\quad + c_2 d(1 + (\alpha_2 - 1)k_2/m)L/m - dL/m \\ &> (0.5 + 0.58d)L/m \\ &> (0.5 + \tfrac{0.5}{0.916}d)L/m. \end{aligned}$$

The desired statement follows because in scenario (S1) $d \geq \epsilon$ and in scenario (S2) $d \geq \epsilon'$.

Next we study the case that $A_1$'s schedule is balanced. To analyze sequences leading to (S1) and (S2) simultanously, we consider a more general setting. Let $\epsilon \in [-0.084, 0.084]$ and set $b_1(\epsilon) = \max\{0.916 - \epsilon, 0.885\}L/m$ and $b_2(\epsilon) = \max\{0.916 + \epsilon, 0.885\}L/m$. For the analysis of of (S1) we have $\epsilon \geq 0$ and for the analysis of (S2) we have $\epsilon \leq 0$. We set $\epsilon' = 0$ in scenario (S1). The schedule of $A_1$ is balanced and hence by Lemma 4 at least $m - k_1$ machines of $A_1$ are full, i.e. they have a load of at least $b_1(\epsilon)$. Thus $\lambda_1^t \geq b_1(\epsilon)(m - k_1)$. If $A_2$ cannot schedule $J_t$ on the machine with the $(k_2+1)$-st smallest load, then $l_{2,k+1}^{t-1} + p_t > \gamma$, where $\gamma = c_2\beta^{-1}\lambda_1^t/m$. Thus $p_t > c_2\beta^{-1}\lambda^t/m - l_{2,k_2+1}^{t-1}$. The load on the $(k_2 + 1)$-st smallest machine of $A_2$ is bounded by

$$l_{2,k_2+1}^{max} = (L - k_2 b_2(\epsilon) - k_2\epsilon' L/m)/(m - k_2)$$

since otherwise the total load at time $n-1$ were greater than $(m - k_2)l_{2,k_2+1}^{max} + k_2(b_2(\epsilon) + \epsilon' L/m) = L$. Hence

$$\begin{aligned} p_t &> c_2\beta^{-1}b_1(\epsilon)(m - k_1)/m \\ &\quad - (L - k_2 b_2(\epsilon) - k_2\epsilon' L/m)/(m - k_2). \end{aligned}$$

The last expression is decreasing in $\epsilon$ in the range $-0.084 \leq \epsilon \leq 0.031$ and increasing for $\epsilon \geq 0.031$. Therefore, we evaluate the expression for $\epsilon = 0.031$. For $\epsilon = 0.031$ we obtain $p_t > (0.58 + 0.6\epsilon')L/m$. In any case $p_t > (0.5 + \frac{0.5}{0.916}|\epsilon| + \frac{0.5}{0.916}\epsilon')L/m$ for all $\epsilon \in [-0.084, 0.084]$. $\square$

## 6. REFERENCES

[1] S. Albers. Better bounds for online scheduling. *SIAM Journal on Computing*, 29:459-473, 1999.

[2] 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.

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

[4] B. Chen, A. van Vliet and G.J. Woeginger. A lower bound for randomized on-line scheduling algorithms. *Information Processing Letters*, 51:219–222, 1994.

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

[6] R. Fleischer and M. Wahl. Online scheduling revisited. *Proc. 8th Annual European Symposium on Algorithms*, Springer LNCS, 2000.

[7] 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.

[8] T. Gormley, N. Reingold, E. Torng and J. Westbrook. Generating adversaries for request-answer games. *Proc. 11th ACM-SIAM Symposium on Discrete Algorithms*, 564–565, 2000.

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

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

[11] S.S. Seiden. Online randomized multiprocessor scheduling. *Algorithmica*, 28:173–216, 2000.

[12] S.S.Seiden. Barely random algorithms for multiprocessor scheduling. To appear in *Journal of Scheduling*.

[13] J. Sgall. A lower bound for randomized on-line multiprocessor scheduling. *Information Processing Letters*, 63:51–55, 1997.

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