Algorithmica
https://doi.org/10.1007/s00453-021-00852-5

Online Makespan Scheduling with Job Migration on Uniform Machines

Matthias Englert[1] · David Mezlaf[2] · Matthias Westermann[2]

Received: 26 May 2019 / Accepted: 26 June 2021
© The Author(s) 2021

Abstract
In the classic minimum makespan scheduling problem, we are given an input sequence of n jobs with sizes. A scheduling algorithm has to assign the jobs to m parallel machines. The objective is to minimize the makespan, which is the time it takes until all jobs are processed. In this paper, we consider online scheduling algorithms without preemption. However, we allow the online algorithm to change the assignment of up to k jobs at the end for some limited number k. For m identical machines, Albers and Hellwig (Algorithmica 79(2):598–623, 2017) give tight bounds on the competitive ratio in this model. The precise ratio depends on, and increases with, m. It lies between 4/3 and ≈ 1.4659. They show that $k = O(m)$ is sufficient to achieve this bound and no $k = o(n)$ can result in a better bound. We study m uniform machines, i.e., machines with different speeds, and show that this setting is strictly harder. For sufficiently large m, there is a $\delta = \Theta(1)$ such that, for m machines with only two different machine speeds, no online algorithm can achieve a competitive ratio of less than $1.4659 + \delta$ with $k = o(n)$. We present a new algorithm for the uniform machine setting. Depending on the speeds of the machines, our scheduling algorithm achieves a competitive ratio that lies between 4/3 and ≈ 1.7992 with $k = O(m)$. We also show that $k = \Omega(m)$ is necessary to achieve a competitive ratio below 2. Our algorithm is based on maintaining a specific imbalance with respect to the completion times of the machines, complemented by a bicriteria approximation algorithm that minimizes the makespan and maximizes the average completion time for certain sets of machines.

Keywords Online algorithms · Competitive analysis · Minimum makespan scheduling · Job migration

A preliminary version appeared in Proceedings of the 26th European Symposium on Algorithms (ESA), pages 26:1–26:14, 2018.

✉ Matthias Englert
m.englert@warwick.ac.uk

Extended author information available on the last page of the article

Published online: 19 August 2021

1 Introduction

In the classic minimum makespan scheduling problem, we are given an input sequence of n jobs with sizes. A scheduling algorithm has to assign the jobs to m parallel machines. The objective is to minimize the makespan, which is the time it takes until all jobs are processed. This problem is NP-hard in the strong sense [20]. In this paper, we consider online scheduling without preemption.

An online algorithm does not have knowledge about the input sequence in advance. Instead, it gets to know the input sequence job by job without knowledge about the future. An online algorithm is called c-competitive if the makespan of the algorithm is at most c times the makespan of an optimal offline solution.

Extensive work has been done to narrow the gap between lower and upper bounds on the competitive ratio for online minimum makespan scheduling. Increasingly sophisticated algorithms and complex analyses were developed. Nevertheless, even for the most basic case of identical machines, in which each job has the same processing time, i.e., its size, on every machine, there is still a gap between the best known lower and upper bounds on the competitive ratio of 1.880 [30] and 1.9201 [18], respectively. In the setting with uniform machines, in which different machines may run at different speeds, the best known lower and upper bounds on the competitive ratio are 2.564 [13] and 5.828 [6], respectively.

In this work, we study to what extent the ability to migrate a limited number of jobs can help an online algorithm in terms of the competitive ratio in the uniform machine setting. In this model, the online algorithm has to assign jobs to machines as they arrive. However, after all jobs have arrived, the algorithm may remove up to k jobs from the machines and reassign them to different machines.

Job migration is a useful tool to balance loads and it is natural to study how many jobs need to be migrated to achieve certain load guarantees. Indeed, job migration in scheduling has been studied previously, see for example [8, 12, 27, 32–34], but in particular, Albers and Hellwig [2] studied this problem for m identical machines[1] and gave tight bounds on the competitive ratio for this case. Roughly speaking, $k = \Theta(m)$ job migrations are sufficient and necessary to achieve this tight bound. Allowing more job migrations does not result in further improvements as long as $k = o(n)$, where n denotes the total number of arriving jobs.

We provide related results for the more general setting of uniform machines, which introduces new technical challenges. Our contribution also implies new results on a different but related problem: online reordering for scheduling. In this model, a so-called *reordering buffer* can be used to reorder the input sequence of jobs in a restricted fashion. Arriving jobs are first stored in the reordering buffer which has capacity to store up to k jobs. When the buffer is full, the online scheduling algorithm has to decide which of the jobs to remove from the buffer and to assign (irrevocably) to a machine. When no more jobs arrive, all jobs remaining in the buffer have to be assigned to machines as well.

[1] Technically, they allow job migration to be performed before all jobs have arrived as long as the total number of migration is still bounded by k. However, performing all migrations at the end cannot increase the competitive ratio.

This model was introduced by Englert et al. [14] and the work by Albers and Hellwig [2] generalizes their results for identical machines to the setting were no buffer is used, but a limited number of job migrations are permitted. It is not known what the relationship between the two models is in general. However, Albers and Hellwig note that any online algorithm for the job migration model that satisfies a certain monotonicity property can be transformed into an online algorithm for the corresponding reordering buffer problem which has the same competitive ratio. If the algorithm migrates k jobs, the transformed algorithm requires a buffer of size k. The aforementioned monotonicity property is as follows: if the algorithm would not migrate a job at time t if we pretend that the input sequence ends at that time, then the algorithm does not migrate the job at any later time either.

Both the algorithm by Albers and Hellwig and the algorithm we present in this work satisfy the monotonicity property. Therefore, our results also directly imply an improved upper bound for the online minimum makespan scheduling problem with a reordering buffer on uniform machines.

1.1 The Model and Our Contribution

We present a lower bound on the competitive ratio showing that the problem is strictly harder for uniform machines than for identical machines. We give the first online algorithm for uniform machines with job migration. Depending on the speeds of the m machines, our scheduling algorithm achieves a competitive ratio that lies between 4/3 and ≈ 1.7992 and performs $O(m)$ job migrations. In addition, we show that $\Omega(m)$ job migrations are necessary to achieve a competitive ratio of less than 2.

For the corresponding problem of online minimum makespan scheduling with a reordering buffer, Englert et al. [14] present a greedy algorithm that achieves a competitive ratio of 2 (or $2+\varepsilon$ if the algorithm is supposed to be efficient) with a reordering buffer of size m. Subsequently, Ding et al. [9] improved the competitive ratio to $2-1/m$ with a buffer of size $m+1$.[2] Therefore, we also obtain a significant improvement over these previously known results for the reordering buffer version of the problem, since our upper bound translates to this model as well.

Before we explain our contribution in more detail, we define the model more formally and introduce some useful notation and definitions. The $m \geq 2$ *machines* are denoted by $M_0, \ldots, M_{m-1}$. For each $0 \leq i \leq m-1$, the *speed* of machine M_i is denoted by s_i, with $\min\{s_0, \ldots, s_{m-1}\} = 1$. Later, for our upper bounds, we will assume that the machines are sorted in ascending order of their speeds, i.e., $1 = s_0 \leq \ldots \leq s_{m-1}$, but in our lower bound construction this is not necessarily the case. The *sum of speeds* is denoted by $S = \sum_{i=0}^{m-1} s_i$. The *size* of a job J is denoted by $p(J)$. The *load* $L(M_i)$ of a machine M_i is defined as the sum of the sizes of the jobs assigned to machine M_i. The *completion time* of a machine M_i is defined as the

[2] Note that in this and several of the following papers, the model differs from the model in [14] in that arriving jobs can bypass the buffer and may directly be assigned to a machine. This is equivalent to increasing the buffer size in the model from [14] by 1. We express buffer sizes in terms of the model from [14] here.

load $L(M_i)$ of machine M_i divided by the speed s_i of machine M_i. The objective is to minimize the makespan, i.e., the maximum completion time.

As in previous works of Englert et al. [14] and Albers and Hellwig [2], our algorithm attempts to maintain a specific (and not balanced) load distribution on the machines. The desired load on a machine M_i is defined by the so-called *weight* w_i of the machine. The weight is defined as

$$w_i := \min\left\{ s_i \cdot \frac{r}{S}, s_i \cdot \frac{r-1}{\sum_{j=0}^{i-1} s_j} \right\} = \begin{cases} s_i \cdot \frac{r}{S}, & \text{if } 0 \le \sum_{j=0}^{i-1} s_j \le \frac{r-1}{r} \cdot S \\ s_i \cdot \frac{r-1}{\sum_{j=0}^{i-1} s_j}, & \text{if } \frac{r-1}{r} \cdot S < \sum_{j=0}^{i-1} s_j < S \cdot \end{cases}$$

Now, r is the smallest positive solution to $\sum_{i=0}^{m-1} w_i = 1$, i.e., we ensure that the weights of all machines sum up to 1. Due to Corollary 16 in the "Appendix", such a solution always exists. It is important to note that r depends on the number of machines m as well as the machine speeds $s_0, \ldots, s_{m-1}$. If $s_0 = \cdots = s_{m-1} = 1$, the weights match those in [2, 14] and $r =: r_m$ is equal to the competitive ratio achieved in [2, 14] for m identical machines.

Unfortunately, we do not know a closed-form formula for r, but the value can be calculated for any given m and speeds $s_0, \ldots, s_{m-1}$. Due to Corollary 16 in the "Appendix",

$$1 < r \le \mathrm{W}_{-1}(-1/e^2)/(1 + \mathrm{W}_{-1}(-1/e^2)) \approx 1.4659,$$

where W_{-1} is the lower branch of the Lambert W function, i.e., $\mathrm{W}_{-1}(-1/e^2)$ is the smallest real solution to $x \cdot e^x = -1/e^2$. Note that, for the optimal competitive ratio r_m for m identical machines,

$$4/3 \le r_m \le \mathrm{W}_{-1}(-1/e^2)/(1 + \mathrm{W}_{-1}(-1/e^2)).$$

Depending on the speeds of the machines, r can be significantly smaller than r_m.

Our results are as follows.

- We prove that a $\delta = \Theta(1)$ exists such that, for m uniform machines with only two different machine speeds, m sufficiently large, no online algorithm can achieve a competitive ratio less than $\mathrm{W}_{-1}(-1/e^2)/(1 + \mathrm{W}_{-1}(-1/e^2)) + \delta \approx 1.4659 + \delta$ while migrating $o(n)$ jobs. Recall that, for the optimal competitive ratio r_m for m identical machines, $r_m \le \mathrm{W}_{-1}(-1/e^2)/(1 + \mathrm{W}_{-1}(-1/e^2)) \approx 1.4659$. Hence, the more general problem of uniform machines is strictly harder than the special case of identical machines. The lower bound construction differs from the previous ones for identical machines in [2, 14]. The previous constructions used a very large number ($1/\varepsilon$ many) of very small jobs (of size ε), which the online algorithm has to schedule on the machines. The adversary then identifies a machine with load of at least w_i, i.e., a machine with a load that is not below the "target load" and, roughly

speaking, produces just enough large jobs so that one of them has to be assigned to a machine with load w_i. Migrating small jobs is ineffective and the large jobs cannot all avoid a machine with load w_i. This technique alone however is no longer sufficient to obtain a lower bound that is strictly larger than the known one. Using a larger number of possible continuations of the initial input, we can show that to handle these additional continuations, the online algorithm would have to have a significant number of machines with completion time strictly less than, and bounded away from, w_i. But then another machine must have completion time strictly above w_i (rather than just equal to w_i). We remark that the same lower bound can be constructed for the reordering buffer model with uniform machines.

- We show that, for m uniform machines, $\Omega(m)$ migrations are necessary to achieve a competitive ratio of less than 2. Specifically, for $c = \lceil -\ln(2-r)/\ln r \rceil \geq 2$, no online algorithm can achieve a competitive ratio less than $r \in (1, 2)$ while migrating at most $(m-c)/(c^2+c)$ jobs. For example, $r \approx 1.8393 > \mathrm{W}_{-1}(-1/e^2)/(1 + \mathrm{W}_{-1}(-1/e^2)) + 1/3$ if at most $(m-3)/12$ job migrations are allowed. Again, we remark that the same lower bound can be constructed for the reordering buffer model with uniform machines.
- For m uniform machines with speeds $1 = s_0 \leq \ldots \leq s_{m-1}$, our online algorithm achieves a competitive ratio of $r + 1/3$ with $O(m)$ job migrations. If an efficient algorithm is desired, there is an additional additive loss of ε in the competitive ratio due to the use of a PTAS by Hochbaum and Shmoys [24] in a subroutine. Note that $1 < r \leq \mathrm{W}_{-1}(-1/e^2)/(1 + \mathrm{W}_{-1}(-1/e^2)) \approx 1.4659$, i.e., the competitive ratio is at most an additive 1/3 larger than in the identical machines case. However, depending on the speeds of the machines, r can also be significantly smaller than r_m in which case the difference between the competitive ratios can also be smaller than 1/3. The basic structure of our algorithm is similar to the algorithm for the special case of identical machines [2]: Jobs are classified into small and large jobs according to their relative size compared to the total load on all machines. Ignoring the contribution of large jobs, the small jobs are scheduled in such a way that an imbalance with respect to the completion times of the machines is maintained. Roughly speaking, faster machines are kept at lower completion times than slower ones. After all jobs have arrived, some jobs are migrated. The rough intuition is that the largest jobs should be reassigned to improve the solution. For this, we first remove some jobs from machines. Then, we schedule the largest ones optimally on m empty virtual machines $M'_0, \ldots, M'_{m-1}$ with $L(M'_0) \leq \cdots \leq L(M'_{m-1})$. For m identical machines, this means that, for each $0 \leq i \leq m-1$, the completion time of machine M'_i is less than or equal to the average completion time of the machines $M'_i, \ldots, M'_{m-1}$, and this is a crucial property for achieving the optimal competitive ratio for the identical machine case. In the more general case of uniform machines, this property does not always hold. For example, if M'_0 has speed 1 and $M'_1, \ldots, M'_{m-1}$ have speed 3/2, then m jobs of size 1 are optimally scheduled with makespan 1, but the completion time of M'_0 is 1, which is strictly greater than the average completion time of the machines $M'_0, \ldots, M'_{m-1}$. To address this new complication, our algorithm contains a crucial additional balancing step in which the average completion time for certain sets of virtual machines is increased at the cost of a small increase in the maximum completion time (which is responsible

for the additive loss of 1/3). Finally, the smaller jobs that were removed from their machines are reassigned greedily one by one. The analysis of this step is also more involved than the corresponding one for identical machines because a more straightforward naive argument would introduce a factor of s_{m-1}/s_0 into the number of job migrations. Obviously, once we determine which jobs to migrate, we could just assign those jobs optimally to the existing machines. However, it is not clear how to analyze such a procedure directly. We state a specific algorithm for the reassignment step because it provides us with important properties that enable us to analyze the competitive ratio.

1.2 Related Work

Minimum makespan scheduling has been extensively studied. See the survey by Pruhs, Sgall, and Torng [29] for an overview. For m identical machines, the currently best upper and lower bounds are 1.9201 [18] and 1.880 [30], respectively. These bounds were the last ones in a long series of successive improvements for general or specific values of m [1, 4, 5, 7, 17, 21, 22, 25, 31].

For uniform machines, Aspnes et al. [3] present the first algorithm that achieves a constant competitive ratio. Due to Berman, Charikar and Karpinski [6], the best known upper bound on the competitive ratio is 5.828, and, due to Ebenlendr and Sgall [13], the best known lower bound on the competitive ratio is 2.564.

In a semi-online variant of the problem the jobs arrive in decreasing order of their size. The greedy LPT algorithm, which assigns each job to a machine on which it will be completely processed as early as possible, was considered in this setting. For m identical machines, Graham [23] shows that the LPT algorithm achieves a competitive ratio of $4/3-1/(3m)$. For uniform machines, the LPT algorithm achieves a competitive ratio of 1.66 and a lower bound of 1.52 on its competitive ratio is known [19]. A detailed and tight analysis for two uniform machines is given by Mireault, Orlin, and Vohra [28] and Epstein and Favrholdt [15].

For m identical machines, Albers and Hellwig [2] present an algorithm that is r_m-competitive, which is optimal as long as at most $o(n)$ jobs can be migrated. For $m \geq 11$, the algorithm migrates at most $7m$ jobs. For smaller m, $8m$ to $10m$ jobs may be migrated. They further give some results on the trade-off between the number of job migrations and the competitive ratio. For example, $2.5m$ job migrations are sufficient to achieve a competitive ratio of 1.75.

Tan and Yu [33] study two identical machines. They give a tight bound of 4/3 on the competitive ratio and this bound is achievable by migrating a single job. They also explore two other models. One in which, at the end, for each machine, the last job that was assigned to the machine may be migrated. And another in which, at the end, the k jobs that arrived last in the input may be migrated.

Chen et al. [8] give an optimal algorithm for two uniform machines. Using independent techniques and algorithms, Wang et al. [34] show bounds which are similar, but not quite optimal for all machine speeds. Both improve upon work by Liu et al. [27].

Dósa et al. [12] consider a variant in which up to k jobs can be migrated after every job arrival, which is a relaxation of online scheduling with a reordering buffer of size k. Sanders, Sivadasan, and Skutella [32] introduce another model in which, after every job arrival, a number of jobs can be reassigned as long as the total size of the reassigned jobs is bounded by some linear function of the size of the arriving job.

Numerous variants related to online minimum makespan scheduling with reordering buffers have been studied. Kellerer et al. [26] present, for two identical machines, an algorithm that achieves an optimal competitive ratio of 4/3 with a reordering buffer of size 2, i.e., the smallest buffer size allowing reordering.

For m identical machines, Englert et al. [14] present a tight and, in comparison to the problem without reordering, improved bound on the competitive ratio for minimum makespan scheduling with reordering buffers. Depending on m, their scheduling algorithm achieves the optimal competitive ratio r_m with a buffer of size $\Theta(m)$. Further, they show that larger buffer sizes do not result in an additional advantage and that a buffer of size $\Omega(m)$ is necessary to achieve this competitive ratio.

Ding et al. [9] give, for m identical machines, a 1.5-competitive algorithm with a buffer of size $1.5m + 1$ and, for three identical machines, a (15/11)-competitive algorithm with a buffer of size 7.

Dósa and Epstein [10] study minimum makespan scheduling on two uniform machines with speed ratio $s \geq 1$. They show that, for any $s > 1$, a buffer of size 3 is sufficient to achieve an optimal competitive ratio (i.e. even a larger buffer cannot result in a smaller competitive ratio) and, in the case $s \geq 2$, a buffer of size 2 already allows to achieve an optimal ratio.

Dósa and Epstein [11] further study preemptive scheduling, as opposed to non-preemptive scheduling, on m identical machines with a reordering buffer. They present a tight bound on the competitive ratio for any m. This bound is 4/3 for even values of m and slightly lower for odd values of m. They show that a buffer of size $\Theta(m)$ is sufficient to achieve this bound, but a buffer of size $o(m)$ does not reduce the best overall competitive ratio of $e/(e-1)$ that is known for the case without reordering.

Epstein, Levin, and van Stee [16] study the objective to maximize the minimum completion time. For m identical machines, they present an upper bound on the competitive ratio of $H_{m-1}+1$ for a buffer of size m and a lower bound of H_m for any fixed buffer size. For m uniform machines, they show that a buffer of size $m+2$ is sufficient to achieve the optimal competitive ratio m.

2 Lower Bounds

Theorem 1 *A $\delta = \Theta(1)$ exists such that, for m uniform machines with only two machine speeds, m sufficiently large, no online algorithm can achieve a competitive ratio of less than $\mathrm{W}_{-1}(-1/e^2)/(1+\mathrm{W}_{-1}(-1/e^2)) + \delta \approx 1.4659 + \delta$ while migrating $o(n)$ jobs, where n denotes the total number of arriving jobs.*

Proof Only two machine speeds 1 and 3/2 are used. Let m_s denote the number of *slow machines* with speed 1 and $m_f = m - m_s$ denote the number of *fast machines* with

speed 3/2. Note that the sum of speeds $S = m_s + 3/2 \cdot m_f$. Define m_s in such a way that $m_s = \lceil (r_\infty - 1)/r_\infty \cdot S \rceil$, with $r_\infty = \mathrm{W}_{-1}(-1/e^2)/(1 + \mathrm{W}_{-1}(-1/e^2)) \approx 1.4659$.

Consider an online algorithm A that uses at most $k(n) = o(n)$ job migrations. We start with the following initial input sequence: S/ε *small jobs* of size $\varepsilon > 0$ arrive. Depending on the actions of the online algorithm up to at most m additional larger jobs arrive later on. Therefore, in total our input sequence will contain no more than $S/\varepsilon + m$ jobs (i.e. $S/\varepsilon \leq n \leq S/\varepsilon + m$). In the remainder of the proof, we will frequently use that $\lim_{\varepsilon \to 0^+} \varepsilon \cdot k(n) = 0$ which is a simple consequence of this. Let $M_0, \ldots, M_{m-1}$ denote the m uniform machines on which A has scheduled these jobs. Let $s_0, \ldots, s_{m-1}$ denote the respective speeds of these machines, with $L(M_0)/s_0 \geq \cdots \geq L(M_{m-1})/s_{m-1}$. According to this order of the machine speeds, define the weight w_i of a machine M_i.

In the following, we show that the competitive ratio of A is at least $r + 1/19 \cdot (r-1)^4/r^3$. Due to Observation 17 in the "Appendix", $\lim_{m \to \infty} r = \mathrm{W}_{-1}(-1/e^2)/(1 + \mathrm{W}_{-1}(-1/e^2)) = r_\infty$, as long as all machine speeds are upper bounded by some constant which is independent of m. Hence, for m sufficiently large, $1/19 \cdot (r-1)^4/r^3 = \Theta(1)$. As a consequence, for m sufficiently large, there exists a $\delta = \Theta(1)$ such that no online algorithm can achieve a competitive ratio of less than $r_\infty + \delta$.

Due to Corollary 16 in the "Appendix", $1 < r \leq r_\infty \approx 1.4659$. This gives the following observation.

Observation 2

$$\frac{r-1}{r} \leq \frac{r_\infty - 1}{r_\infty} < \frac{1}{3} \quad \text{and} \quad r + \frac{1}{19} \cdot \frac{(r-1)^4}{r^3} < 1 + \frac{r}{3} < \frac{3}{2}.$$

Assume for contradiction that A achieves a competitive ratio of $r_A < r + 1/19 \cdot (r-1)^4/r^3$.

The lower bound construction for the identical machines case, is based on the following idea: there must exist a machine M_ℓ with load of at least $w_\ell \cdot S$ after the initial sequence, since otherwise, the total scheduled load would be strictly less than $\sum_{i=0}^{m-1} w_i \cdot S = S$. Since at most $k(n)$ of these jobs can be migrated at the end, at least $w_\ell \cdot S/\varepsilon - k(n)$ of them are guaranteed to stay on M_ℓ which is still almost all the load for sufficiently small ε since $\lim_{\varepsilon \to 0^+} \varepsilon \cdot k(n) = 0$. The high load on M_ℓ can now be exploited by continuing the input sequence in the right way.

However, for uniform machines we now want to get a larger lower bound than for the identical machine case. Hence, a significant contribution is the following lemma that gives an improved lower bound on the completion time. Specifically, it shows a lower bound on the completion time that is larger than the original $w_\ell/s_\ell \cdot S$ by an additive $1/19 \cdot (r-1)^3/r^2$, resulting in an improved lower bound on the competitive ratio of A.

Lemma 3 *After the initial input sequence, a machine M_ℓ with completion time of at least $w_\ell/s_\ell \cdot S + 1/19 \cdot (r-1)^3/r^2$ exists.*

Proof We show that the lemma holds as otherwise a continuation of the input sequence that leads to a contradiction to A being r_A-competitive exists. We distinguish two cases.

- The number of fast machines with weight $w_i = 3/2 \cdot r/S$ is at least $\lceil 2/11 \cdot (r-1)/r \cdot S \rceil$: In addition, $m_f - \lceil 1/11 \cdot (r-1)/r \cdot S \rceil$ *large jobs* of size $3/2 \cdot S/S_\varepsilon \geq 3/2$ arrive, with

$$\begin{aligned} S_\varepsilon &= m_s + \frac{3}{2} \cdot \left\lceil \frac{1}{11} \cdot \frac{r-1}{r} \cdot S \right\rceil \\ &= \left\lceil \frac{r_\infty - 1}{r_\infty} \cdot S \right\rceil + \frac{3}{2} \cdot \left\lceil \frac{1}{11} \cdot \frac{r-1}{r} \cdot S \right\rceil \\ &\geq \frac{r_\infty - 1}{r_\infty} \cdot S + \frac{3}{2} \cdot \frac{1}{11} \cdot \frac{r-1}{r} \cdot S \\ &\geq \frac{25}{22} \cdot \frac{r-1}{r} \cdot S. \end{aligned}$$

An optimal offline algorithm can schedule each large job on a separate fast machine and evenly distribute the small jobs among the remaining machines. These remaining machines, among which the small jobs are distributed, consist of all m_s slow machines and the $\lceil 1/11 \cdot (r-1)/r \cdot S \rceil$ fast machines that do not get assigned any of the large jobs. Therefore, the sum of their speeds is exactly S_ε. Hence, the optimal makespan is at most $S/S_\varepsilon + \varepsilon$. If, in the final schedule after migrations, A schedules one large job on a slow machine or two large jobs on the same fast machine, the completion time of such a machine is at least $3/2 \cdot S/S_\varepsilon$ and, therefore, the competitive ratio of A is at least

$$\frac{3/2 \cdot S/S_\varepsilon}{S/S_\varepsilon + \varepsilon},$$

which is strictly larger than r_A if ε is sufficiently small. As a consequence, A schedules each of the large jobs on a separate fast machine. Let $\mathcal{U}$ be the set of such machines which have weight $w_i = 3/2 \cdot r/S$. Then for any machine $M_i \in \mathcal{U}$, for the load $L_\varepsilon(M_i)$ on M_i caused by small jobs,

$$L_\varepsilon(M_i) \leq w_i \cdot S - \frac{137}{12 \cdot 19} \cdot \frac{(r-1)^2}{r},$$

since otherwise the completion time of M_i is at least

$$\begin{aligned} \frac{S}{S_\varepsilon} + \frac{L_\varepsilon(M_i)}{s_i} &\geq \frac{S}{S_\varepsilon} + \frac{w_i}{s_i} \cdot S - \frac{1}{s_i} \cdot \frac{137}{12 \cdot 19} \cdot \frac{(r-1)^2}{r} \\ &= \frac{S}{S_\varepsilon} + \frac{3/2 \cdot r/S}{s_i} \cdot S - \frac{1}{s_i} \cdot \frac{137}{12 \cdot 19} \cdot \frac{(r-1)^2}{r} \\ &= \frac{S}{S_\varepsilon} + r - \frac{2}{3} \cdot \frac{137}{12 \cdot 19} \cdot \frac{(r-1)^2}{r} \\ &\geq \frac{S}{S_\varepsilon} + \frac{S}{S_\varepsilon} \cdot \frac{25}{22} \cdot \frac{r-1}{r} \cdot r - \frac{S}{S_\varepsilon} \cdot \frac{2}{3} \cdot \frac{137}{12 \cdot 19} \cdot \frac{(r-1)^2}{r} \end{aligned}$$

$$
\begin{aligned}
&= \frac{S}{S_\epsilon} \cdot \left(1 + \frac{25}{22} \cdot (r-1) - \frac{137}{18 \cdot 19} \cdot \frac{(r-1)^2}{r}\right) \\
&= \frac{S}{S_\epsilon} \cdot \left(r + \frac{3}{22} \cdot (r-1) - \frac{137}{18 \cdot 19} \cdot \frac{(r-1)^2}{r}\right) \\
&\geq \frac{S}{S_\epsilon} \cdot \left(r + \frac{3}{22} \cdot (r-1) \cdot 3 \cdot \frac{r-1}{r} - \frac{137}{18 \cdot 19} \cdot \frac{(r-1)^2}{r}\right) \\
&= \frac{S}{S_\epsilon} \cdot \left(r + \left(\frac{3}{22} \cdot 3 - \frac{137}{18 \cdot 19}\right) \cdot \frac{(r-1)^2}{r}\right) \\
&\geq \frac{S}{S_\varepsilon} \cdot \left(r + \left(\frac{3}{22} \cdot 3 - \frac{137}{18 \cdot 19}\right) \cdot \frac{(r-1)^2}{r} \cdot 3^2 \cdot \frac{(r-1)^2}{r^2}\right) \\
&\geq \frac{S}{S_\varepsilon} \cdot \left(r + \frac{1}{19} \cdot \frac{(r-1)^4}{r^3}\right),
\end{aligned}
$$

which is a contradiction to A being r_A-competitive. For the fourth step, recall that

$$
\frac{22}{25} \cdot \frac{r}{r-1} \geq \frac{S}{S_\varepsilon} \geq 1,
$$

and for steps seven and nine, note that Observation 2 gives $3 \cdot (r-1)/r < 1$. The number of large jobs that are scheduled on a machine with weight $3/2 \cdot r/S$ is

$$
\begin{aligned}
|\mathcal{U}| \geq \left\lceil \frac{2}{11} \cdot \frac{r-1}{r} \cdot S \right\rceil - \left\lceil \frac{1}{11} \cdot \frac{r-1}{r} \cdot S \right\rceil &\geq \left(\frac{2}{11} - \frac{142}{11 \cdot 137}\right) \cdot \frac{r-1}{r} \cdot S \\
&= \frac{12}{137} \cdot \frac{r-1}{r} \cdot S,
\end{aligned}
$$

for $S = m_s + 3/2 \cdot m_f$ sufficiently large. We conclude that there must be a machine $M_\ell \notin \mathcal{U}$ such that

$$
\frac{L_\varepsilon(M_\ell)}{s_\ell} \geq \frac{w_\ell}{s_\ell} \cdot S + \frac{1}{19} \cdot \frac{(r-1)^3}{r^2},
$$

as otherwise

$$
\begin{aligned}
&\sum_{M_\ell \notin \mathcal{U}} L_\varepsilon(M_\ell) + \sum_{M_i \in \mathcal{U}} L_\varepsilon(M_i) \\
&\qquad < \sum_{M_\ell \notin \mathcal{U}} \left(w_\ell \cdot S + \frac{1}{19} \frac{(r-1)^3}{r^2} \cdot s_\ell\right) \\
&\qquad\quad + \sum_{M_i \in \mathcal{U}} \left(w_i \cdot S - \frac{137}{12 \cdot 19} \frac{(r-1)^2}{r}\right) \\
&\qquad = S \cdot \sum_{i=0}^{m-1} w_i + \sum_{M_\ell \notin \mathcal{U}} \frac{1}{19} \frac{(r-1)^3}{r^2} \cdot s_\ell - \sum_{M_i \in \mathcal{U}} \frac{137}{12 \cdot 19} \frac{(r-1)^2}{r}
\end{aligned}
$$

$$\leq S \cdot \sum_{i=0}^{m-1} w_i + \frac{1}{19}\frac{(r-1)^3}{r^2} \cdot S - |\mathcal{U}| \cdot \frac{137}{12 \cdot 19}\frac{(r-1)^2}{r}$$
$$\leq S,$$

which is a contradiction to the fact that the total size of all small jobs combined is S.

- The number of fast machines with weight $w_i = 3/2 \cdot r/S$ is at most $\lceil 2/11 \cdot (r-1)/r \cdot S \rceil - 1$: In addition, $m_f + \lfloor 1/2 \cdot m_s \rfloor$ *large jobs* of size $S/S_\varepsilon \geq 1$ arrive, with

$$S_\varepsilon = \left\lceil \frac{1}{2} \cdot m_s \right\rceil + \frac{1}{3} \cdot \frac{3}{2} \cdot m_f \geq \frac{1}{3} \cdot S + \frac{1}{6} \cdot m_s \geq S \cdot \left(\frac{1}{3} + \frac{1}{6} \cdot \frac{r_\infty - 1}{r_\infty} \right)$$
$$\geq S \cdot \left(\frac{1}{3} + \frac{1}{6} \cdot \frac{r-1}{r} \right).$$

An optimal offline algorithm can schedule each large job on a separate machine of the set of all fast and $\lfloor 1/2 \cdot m_s \rfloor$ slow machines. Now, the fast machines can process additional jobs while the slow machines are working on large jobs. Therefore, the small jobs can be distributed among the remaining slow machines and the fast machines, which give a weighted sum of speeds of S_ε. Hence, the optimal makespan is at most $S/S_\varepsilon + \varepsilon$. If A schedules two large jobs on the same slow machine or three large jobs on the same fast machine, the competitive ratio of A is at least

$$\frac{2 \cdot S/S_\varepsilon}{S/S_\varepsilon + \varepsilon},$$

which is strictly larger than r_A if ε is sufficiently small. As a consequence, A schedules at most one large job on each slow machine and at most two large jobs on each fast machine. Let $\mathcal{U}$ be the set of all slow machines that have weight $w_i = r/S$ and receive one large job and all fast machines that receive two large jobs. Then for any slow machine $M_i \in \mathcal{U}$, for the load $L_\varepsilon(M_i)$ on M_i caused by small jobs,

$$L_\varepsilon(M_i) \leq w_i \cdot S - \frac{1}{4} \cdot \frac{(r-1)^2}{r},$$

since otherwise the completion time of M_i is at least

$$\begin{aligned}
\frac{S}{S_\varepsilon} + \frac{L_\varepsilon(M_i)}{s_i} &\geq \frac{S}{S_\varepsilon} + \frac{w_i}{s_i} \cdot S - \frac{1}{s_i} \cdot \frac{1}{4} \cdot \frac{(r-1)^2}{r} \\
&\geq \frac{S}{S_\varepsilon} \cdot \left(1 + r \cdot \left(\frac{1}{3} + \frac{1}{6} \cdot \frac{r-1}{r} \right) - \frac{1}{4} \cdot \frac{(r-1)^2}{r} \right) \\
&\geq \frac{S}{S_\varepsilon} \cdot \left(1 + r \cdot \left(\frac{1}{3} + \frac{1}{6} \cdot \frac{(r-1)^2}{r^2} \cdot 3 \right) - \frac{1}{4} \cdot \frac{(r-1)^2}{r} \right) \\
&\geq \frac{S}{S_\varepsilon} \cdot \left(1 + \frac{r}{3} + \left(\frac{1}{6} \cdot 3 - \frac{1}{4} \right) \cdot \frac{(r-1)^2}{r} \right)
\end{aligned}$$

$$\geq \frac{S}{S_\varepsilon} \cdot \left(1 + \frac{r}{3}\right)$$
$$\geq \frac{S}{S_\varepsilon} \cdot \left(r + \frac{1}{19} \cdot \frac{(r-1)^4}{r^3}\right),$$

which is a contradiction to A being r_A-competitive. For the second step, recall that

$$\left(\frac{1}{3} + \frac{1}{6} \cdot \frac{r-1}{r}\right)^{-1} \geq \frac{S}{S_\varepsilon} \geq 1,$$

and note that the last step follows form Observation 2. In addition, for any fast machine $M_i \in \mathcal{U}$, for the load $L_\varepsilon(M_i)$ on M_i caused by small jobs,

$$L_\varepsilon(M_i) \leq w_i \cdot S - \frac{1}{4} \cdot \frac{(r-1)^2}{r},$$

since otherwise the completion time of M_i is at least

$$\begin{aligned}
\frac{4}{3} \cdot \frac{S}{S_\varepsilon} + \frac{L_\varepsilon(M_i)}{s_i} &\geq \frac{4}{3} \cdot \frac{S}{S_\varepsilon} + \frac{w_i}{s_i} \cdot S - \frac{1}{s_i} \cdot \frac{1}{4} \cdot \frac{(r-1)^2}{r} \\
&\geq \frac{4}{3} \cdot \frac{S}{S_\varepsilon} + (r-1) - \frac{1}{s_i} \cdot \frac{1}{4} \cdot \frac{(r-1)^2}{r} \\
&\geq \frac{S}{S_\varepsilon}\left(\frac{4}{3} + (r-1)\left(\frac{1}{3} + \frac{1}{6} \cdot \frac{r-1}{r}\right) - \frac{2}{3} \cdot \frac{1}{4} \cdot \frac{(r-1)^2}{r}\right) \\
&= \frac{S}{S_\varepsilon}\left(1 + \frac{r}{3}\right) \\
&\geq \frac{S}{S_\varepsilon}\left(r + \frac{1}{19} \cdot \frac{(r-1)^4}{r^3}\right),
\end{aligned}$$

which is a contradiction to A being r_A-competitive. For the second step, note that by the definition of the weights, $w_i \geq s_i \cdot (r-1)/S$ for all i. If the number of fast machines with weight $w_i = 3/2 \cdot r/S$ is at most $\lceil 2/11 \cdot (r-1)/r \cdot S \rceil - 1$ (as is the case here), then their combined speed is at most $3/2 \cdot \lceil 2/11 \cdot (r-1)/r \cdot S \rceil - 3/2$. The sum of the speeds of all machines with weight $w_i = 3/2 \cdot r/S$ is at least $S \cdot (r-1)/r - 3/2$ by the definition of the weights. Then the number of slow machines with weight $3/2 \cdot r/S$ is at least $S \cdot (r-1)/r - 3/2 - (3/2 \cdot \lceil 2/11 \cdot (r-1)/r \cdot S \rceil - 3/2) = S \cdot (r-1)/r - 3/2 \cdot \lceil 2/11 \cdot (r-1)/r \cdot S \rceil$.

We conclude that the number of slow machines with a weight different from $3/2 \cdot r/S$ is at most $m_s - S \cdot (r-1)/r + 3/2 \cdot \lceil 2/11 \cdot (r-1)/r \cdot S \rceil$. Thus, for the $\lfloor m_s/2 \rfloor + m_f$ large jobs, the number of large jobs that are scheduled on a slow machine with weight r/S or together with another large job on a fast machine is

$$
\begin{aligned}
|\mathcal{U}| &\geq \left\lfloor \frac{1}{2} \cdot m_s \right\rfloor - \left(m_s - \left(\frac{r-1}{r} \cdot S \right) + \frac{3}{2} \cdot \left\lceil \frac{2}{11} \cdot \frac{r-1}{r} \cdot S \right\rceil \right) \\
&\geq -\left\lceil \frac{1}{2} \cdot \left\lceil \frac{r_\infty - 1}{r_\infty} \cdot S \right\rceil \right\rceil + \left(\frac{r-1}{r} \cdot S \right) - \frac{3}{2} \cdot \left\lceil \frac{2}{11} \cdot \frac{r-1}{r} \cdot S \right\rceil \\
&\geq -\left(\frac{105}{209} \cdot \frac{r_\infty - 1}{r_\infty} \cdot S \right) + \left(\frac{r-1}{r} \cdot S \right) - \frac{3}{2} \cdot \left(\frac{116}{627} \cdot \frac{r-1}{r} \cdot S \right) \\
&= -\frac{105}{209} \cdot S \cdot \left(\frac{r_\infty - 1}{r_\infty} - \frac{r-1}{r} \right) + \left(\frac{46}{209} \cdot \frac{r-1}{r} \cdot S \right) \\
&\geq \left(-\frac{2}{209} + \frac{46}{209} \right) \cdot \frac{r-1}{r} \cdot S = \frac{4}{19} \cdot \frac{r-1}{r} \cdot S.
\end{aligned}
$$

The second step holds for $S = m_s + 3/2 \cdot m_f$ sufficiently large. For the fourth step, note that, for m sufficiently large, $(r_\infty - 1)/r_\infty - (r-1)/r \leq 2/105 \cdot (r-1)/r$, since $(r-1)/r \leq (r_\infty - 1)/r_\infty$ and $\lim_{m \to \infty} r = r_\infty$. We conclude that there must be a machine $M_\ell \notin \mathcal{U}$ such that

$$
\frac{L_\varepsilon(M_\ell)}{s_\ell} \geq \frac{w_\ell}{s_\ell} \cdot S + \frac{1}{19} \cdot \frac{(r-1)^3}{r^2},
$$

as otherwise

$$
\begin{aligned}
&\sum_{M_\ell \notin \mathcal{U}} L_\varepsilon(M_\ell) + \sum_{M_i \in \mathcal{U}} L_\varepsilon(M_i) \\
&\qquad < \sum_{M_\ell \notin \mathcal{U}} \left(w_\ell \cdot S + \frac{1}{19} \frac{(r-1)^3}{r^2} \cdot s_\ell \right) \\
&\qquad\quad + \sum_{M_i \in \mathcal{U}} \left(w_i \cdot S - \frac{1}{4} \frac{(r-1)^2}{r} \right) \\
&\qquad = S \cdot \sum_{i=0}^{m-1} w_i + \sum_{M_\ell \notin \mathcal{U}} \frac{1}{19} \frac{(r-1)^3}{r^2} \cdot s_\ell - \sum_{M_i \in \mathcal{U}} \frac{1}{4} \frac{(r-1)^2}{r} \\
&\qquad \leq S \cdot \sum_{i=0}^{m-1} w_i + \frac{1}{19} \frac{(r-1)^3}{r^2} \cdot S - |\mathcal{U}| \cdot \frac{1}{4} \frac{(r-1)^2}{r} \\
&\qquad \leq S,
\end{aligned}
$$

which is a contradiction to the fact that the total size of all small jobs combined is S. □

Let M_ℓ denote a machine which, after the initial input sequence of jobs of size ε, has a completion time of at least $w_\ell/s_\ell \cdot S + 1/19 \cdot (r-1)^3/r^2$. We distinguish two cases.

- $w_\ell = s_\ell \cdot r/S$:

  No more jobs arrive. An optimal offline algorithm can evenly distribute all jobs among the machines. Hence, the optimal makespan is at most $S/S + \varepsilon = 1 + \varepsilon$. Finally, the competitive ratio of A is at least

$$\frac{r + 1/19 \cdot (r-1)^3/r^2 - \varepsilon \cdot k(n)}{1+\varepsilon} \geq \frac{r + 1/19 \cdot (r-1)^4/r^3}{1+\varepsilon} - \frac{\varepsilon \cdot k(n)}{1+\varepsilon},$$

  which is strictly larger than r_A if ε is sufficiently small.

- $w_\ell = s_\ell \cdot (r-1)/\sum_{j=0}^{\ell-1} s_j$:

  In addition, $\min\{m_f, \lfloor 2/3 \cdot (S - \sum_{j=0}^{\ell-1} s_j)\rfloor\}$ *large 3/2-jobs* of size $3/2 \cdot S/\sum_{j=0}^{\ell-1} s_j$ and $\max\{0, \lfloor m_s - \sum_{j=0}^{\ell-1} s_j \rfloor\} = \max\{0, \lfloor S - \sum_{j=0}^{\ell-1} s_j - 3/2 \cdot m_f \rfloor\}$ *large 1-jobs* of size $S/\sum_{j=0}^{\ell-1} s_j$ arrive. An optimal offline algorithm can schedule each large x-job, with $x \in \{1, 3/2\}$, on a separate machine with speed x and evenly distribute the small jobs among the remaining machines. Hence, the optimal makespan is at most $S/\sum_{j=0}^{\ell-1} s_j + \varepsilon$.

  If A schedules one large 3/2-job on a slow machine or two large jobs on the same machine, the competitive ratio of A is at least

$$\frac{3/2 \cdot S/\sum_{j=0}^{\ell-1} s_j}{S/\sum_{j=0}^{\ell-1} s_j + \varepsilon},$$

  which is strictly larger than r_A if ε is sufficiently small.

  If $m_f > \lfloor 2/3 \cdot (S - \sum_{j=0}^{\ell-1} s_j)\rfloor$, the number of 3/2-jobs that arrive is greater than the number of fast machines $M_{\ell'}$ which have an index of $\ell' \geq \ell$. Since these jobs can only be scheduled on fast machines, at least one of them has to be scheduled on a machine with index $\ell' < \ell$. If $m_f \leq \lfloor 2/3 \cdot (S - \sum_{j=0}^{\ell-1} s_j)\rfloor$ the number of 3/2-jobs that arrive is equal to the total number of fast machines m_f. If there exists a fast machine which has an index $\ell' \leq \ell$, at least one of these jobs has to be scheduled on such a machine. On the other hand, if all fast machines have an index of $\ell' > \ell$, then we observe that the total number of 1-jobs and 3/2-jobs is $m_f + m_s - \sum_{j=0}^{\ell-1} s_j = m - \ell$. Hence, at least one of the jobs has to be scheduled on a machine with index $\ell' \leq \ell$ (and this machine is slow). We conclude that A schedules at least one large x-job, with $x \in \{1, 3/2\}$, on a machine with speed x that, after the initial assignment of the jobs of size ε, that, after the initial assignment of the jobs of size ε, already has a completion time of at least $w_\ell/s_\ell \cdot S + 1/19 \cdot (r-1)^3/r^2$.

  By definition of w_ℓ, $\sum_{j=0}^{\ell-1} s_j/S \geq (r-1)/r$. Finally, the competitive ratio of A is at least

$$\begin{aligned}&\frac{(1 + r - 1) \cdot S/\sum_{j=0}^{\ell-1} s_j + 1/19 \cdot (r-1)^3/r^2 - \varepsilon \cdot k(n)}{S/\sum_{j=0}^{\ell-1} s_j + \varepsilon} \\ &\geq \frac{\left(r + 1/19 \cdot (r-1)^4/r^3\right) \cdot S/\sum_{j=0}^{\ell-1} s_j}{S/\sum_{j=0}^{\ell-1} s_j + \varepsilon} - \frac{\varepsilon \cdot k(n)}{S/\sum_{j=0}^{\ell-1} s_j + \varepsilon},\end{aligned}$$

  which is strictly larger than r_A if ε is sufficiently small. □

Theorem 4 *For $c = \lceil -\ln(2-r)/\ln r \rceil \geq 2$, no online algorithm can achieve a competitive ratio of less than $r \in (1,2)$ while migrating at most $(m-c)/(c^2+c)$ jobs.*

Proof Let $1 < r < 2$ and $c = \lceil -\ln(2-r)/\ln r \rceil \geq 2$. For each $0 \leq i \leq c-1$, there are $\lfloor m/c \rfloor$ machines with speed r^i. Add machines of speed 1 such that there are m machines in total. Consider an online algorithm A that migrates at most $k = \lfloor (m-c)/(c^2+c) \rfloor$ jobs.

The input sequence consists of at most c consecutive phases. In phase $0 \leq i \leq c-1$, $\lfloor m/c \rfloor$ jobs of size r^i arrive. Let k_i denote the number of jobs of size r^i that are assigned by A to machines with speed strictly less than r^{c-1} or to machines where at least one job of size r^i is already scheduled. If $k_i > k$, stop at the end of this phase. Otherwise, if $i < c-1$, continue with phase $i+1$.

If the input sequence stops at the end of phase $0 \leq i \leq c-1$ due to the fact that $k_i > k$, the competitive ratio of A is at least

$$\frac{\min\{r^i/r^{c-2}, 2 \cdot r^i/r^{c-1}\}}{r^i/r^{c-1}} = r.$$

Otherwise, we focus on the $\lfloor m/c \rfloor$ machines with speed r^{c-1}. In each phase i, at least $\lfloor m/c \rfloor - k$ of these machines are assigned a job of size r^i. This means that after the last phase, there must be $\lfloor m/c \rfloor - c \cdot k$ such machines which each were assigned one job from each phase. We can remove jobs from at most k such machines in the migration phase. Therefore, after the migration phase, at least

$$\left\lfloor \frac{m}{c} \right\rfloor - (c+1) \cdot k = \left\lfloor \frac{m}{c} \right\rfloor - (c+1) \cdot \left\lfloor \frac{m-c}{c^2+c} \right\rfloor \geq \left\lfloor \frac{m}{c} \right\rfloor - \left\lfloor \frac{m-c}{c} \right\rfloor \geq 1$$

machines with speed r^{c-1} exist, to which, for each $0 \leq i \leq c-1$, A has assigned at least one job of size r^i. Hence, the competitive ratio of A is at least

$$\frac{\sum_{i=0}^{c-1} r^i}{r^{c-1}} = \frac{r^c - 1}{(r-1) \cdot r^{c-1}} \geq r,$$

since $c \geq -\ln(2-r)/\ln r$. □

3 Scheduling Algorithm

For m uniform machines with speeds $1 = s_0 \leq \ldots \leq s_{m-1}$, our algorithm consists of two phases: In the scheduling phase, arriving jobs are assigned to (or scheduled on) machines online. In the migration phase, which starts after all jobs have arrived, some jobs are removed from their machines and reassigned to other machines.

More specifically, the scheduling phase consists of steps $1, \ldots, n$, where n denotes the total number of arriving jobs. In step t, the t-th job arrives and is assigned to a machine. For $t > 1$, let T_t denote the total size of the $t-1$ jobs that have arrived up

to and including step $t-1$. In addition, define $T_1 = 0$. A job J is called *small in step* t, if $p(J) \leq T_t/(b \cdot m)$, where b is a constant that will be defined later. Otherwise, J is called *large in step* t. Note that during the scheduling phase, a job that is large in step t can become small in step $t+1$.

Further, let T_t^s denote the total size of the jobs that have arrived up to and including step $t-1$ and that are small in step t. Finally, let $L_t(M_i)$ denote the total size of the jobs that are scheduled on machine M_i at the end of step $t-1$, i.e., after the $(t-1)$-th job is assigned to a machine, and let $L_t^s(M_i)$ denote the total size of the jobs that are scheduled on machine M_i at the end of step $t-1$ and that are small in step t.

We use two different algorithms. The first algorithm, which is used when $s_{m-1} > 3/4 \cdot S$, schedules every job on machine M_{m-1} and does not migrate any jobs. The second algorithm, which is used when $s_{m-1} \leq 3/4 \cdot S$, is more interesting and works as follows.

- *Scheduling phase*: The t-th arriving job J is scheduled in step t as follows.
  - If J is small in step t, J is assigned to a machine M_i with $L_t^s(M_i) \leq w_i \cdot T_t^s$. (Since $\sum_{j=0}^{m-1} w_i = 1$ and $\sum_{i=0}^{m-1} L_t^s(M_i) = T_t^s$, such a machine always exists.)
  - If J is large in step t, J is assigned to a machine M_i that has minimum completion time $L_t(M_i)/s_i$ among all machines.
- *Migration phase*: Throughout the migration phase, we remove jobs from machines and reassign them. At any point during this process, let $L(M_i)$ denote the load of machine M_i at that point, i.e., the $L(M_i)$ values are changing throughout the migration phase.

At the start of the migration phase, after all n jobs have arrived, we have, for each $0 \leq i \leq m-1$, $L(M_i) = L_{n+1}(M_i)$. Then do the following. For each machine M_i, as long as $L(M_i) > w_i \cdot T_{n+1}^s$ and $L(M_i) > (r-1) \cdot T_{n+1} \cdot s_i/S$, remove the job of largest size from M_i.

The removed jobs can now be reassigned optimally to the machines, i.e., in such a way that the resulting makespan is minimized. However, as stated before, it is difficult to analyze the resulting makespan directly. In the following, we therefore present a more specific procedure for this reassignment step which provides us with certain properties that enable us to analyze the competitive ratio. The resulting bound is of course also an upper bound on the competitive ratio achieved through an optimal reassignment.

(1) Those removed jobs that are large at time $n+1$ are scheduled on m empty virtual machines $M'_0, \ldots, M'_{m-1}$ with speeds $1 = s_0 \leq \ldots \leq s_{m-1}$:

  (1a) The jobs are scheduled on the virtual machines optimally, i.e., to minimize the makespan of the virtual machines.[3] Call the resulting makespan on the virtual machines OPT′. We assume that the resulting loads of the virtual machines are sorted, i.e., $L(M'_0) \leq \ldots \leq L(M'_{m-1})$, and that, for each $1 \leq i \leq m-1$, $L(M'_i)/s_i > \text{OPT}'/2$ if $L(M'_{i-1}) > 0$. (See the following Observation 5 items (1) and (2).)

[3] If computational efficiency is a concern, the PTAS by Hochbaum and Shmoys [24] may be used instead, resulting in an additive loss of ε in the competitive ratio.

(1b) Each machine M'_i, with $i \in C$ where

$$C = \left\{0 \le i \le m-1 : \sum_{j=0}^{m-1} L(M'_j) \le \left(\frac{L(M'_i)}{s_i} - \frac{\mathrm{OPT}'}{3}\right) \cdot \sum_{j=i}^{m-1} s_j\right\},$$

is called *critical*. If $C \neq \emptyset$, all jobs from the machines $M'_0, \ldots, M'_c$, with $c = \max(C) < m-1$, are reassigned to $M'_{c+1}, \ldots, M'_{m-1}$.
For $i = 0, \ldots, c$ do the following:

- Find the largest $\ell \ge c+1$ such that $(L(M'_i) + L(M'_\ell))/s_\ell \le 4/3 \cdot \mathrm{OPT}'$. (Due to the following Observation 5 item (3), such a machine always exists.)
- Reassign all jobs from M'_i to M'_ℓ, i.e., $L(M'_\ell)$ is increased by $L(M'_i)$ and $L(M'_i)$ is set to 0.
- Re-sort the loads of the machines such that $L(M'_0) \le \ldots \le L(M'_{m-1})$ again. (See the following Observation 5 item (1).)

Finally, for each $0 \le i \le m-1$, assign the jobs from M'_i to the real machine M_i.

(2) Those removed jobs that are small at time $n+1$ are scheduled according to the greedy algorithm that assigns a job to a machine finishing it first.

Observation 5 For the migration phase, the following observations can be made.

(1) Sorting according to the load does not increase the makespan.
(2) We can assume that, for each $1 \le i \le m-1$, $L(M'_i)/s_i > \mathrm{OPT}'/2$ if $L(M'_{i-1}) > 0$.
(3) If $C \neq \emptyset$, then for each $0 \le i \le c$, $\{c+1 \le j \le m-1 : (L(M'_i) + L(M'_j))/s_j \le 4/3 \cdot \mathrm{OPT}'\} \neq \emptyset$.
(4) For each $0 \le i \le m-1$, $L(M'_i)/s_i \le 4/3 \cdot \mathrm{OPT}'$.

Proof

(1) Assume that $L(M'_i) > L(M'_j)$, with $0 \le i < j \le m-1$. Since $s_i \le s_j$, swapping the loads of M'_i and M'_j does not increase the makespan.
(2) While a $1 \le i \le m-1$ exists with $L(M'_i)/s_i \le \mathrm{OPT}'/2$ and $L(M'_{i-1}) > 0$, reassign the jobs from M'_{i-1} to M'_i, i.e., $L(M'_i)$ is increased by $L(M'_{i-1})$ and $L(M'_{i-1})$ is set to 0, and sort according to the load. This does not increase the makespan, since $L(M'_{i-1}) \le L(M'_i)$ and due to item (1). Further, this process terminates, since after each iteration there is one more machine with no load.
(3) Assume for contradiction that, for each $c+1 \le j \le m-1$, $L(M'_j)/s_j > 2/3 \cdot \mathrm{OPT}'$. This yields the following contradiction to the fact that M'_c is critical:

$$\sum_{j=c}^{m-1} L(M'_j) > \left(\frac{L(M'_c)}{s_c} - \frac{\mathrm{OPT}'}{3}\right) \cdot s_c + \sum_{j=c+1}^{m-1} \frac{2}{3} \cdot \mathrm{OPT}' \cdot s_j$$
$$\geq \left(\frac{L(M'_c)}{s_c} - \frac{\mathrm{OPT}'}{3}\right) \cdot \sum_{j=c}^{m-1} s_j,$$

since $L(M'_c)/s_c \leq \mathrm{OPT}'$. Then, there exists a $c+1 \leq j \leq m-1$, with $L(M'_j)/s_j \leq 2/3 \cdot \mathrm{OPT}'$. Since $L(M'_i) \leq L(M'_j)$, $(L(M'_i) + L(M'_j))/s_j \leq 4/3 \cdot \mathrm{OPT}'$.

(4) Clearly, at the beginning of step (1b), for each $0 \leq i \leq m-1$, $L(M'_i)/s_i \leq \mathrm{OPT}'$. Then, after each reassignment in step (1b), for each $0 \leq i \leq m-1$, $L(M'_i)/s_i \leq 4/3 \cdot \mathrm{OPT}'$ due to items (3) and (1). □

3.1 Analysis of the Algorithm

The analysis of the algorithm consists of two parts. The first part provides a bound on the number of migrated jobs. The second part provides a bound on the competitive ratio of the algorithm. These two parts together give the following theorem.

Theorem 6 *For m uniform machines with speeds $1 = s_0 \leq \ldots \leq s_{m-1}$, our online algorithm achieves a competitive ratio of $r + 1/3$ with $O(m)$ job migrations.*

3.1.1 Bounding the Number of Migrated Jobs

The following lemma gives an upper bound on the number of jobs removed from a single machine.

Lemma 7 *For each $0 \leq i \leq m-1$, in the migration phase, at most $r/(r-1) \cdot b \cdot m \cdot s_i/S + 1$ jobs are removed from machine M_i.*

Proof If the final load of M_i at the end of the scheduling phase satisfies $L_{n+1}(M_i) \leq w_i \cdot T^s_{n+1}$ or $L_{n+1}(M_i) \leq (r-1) \cdot T_{n+1} \cdot s_i/S$, no job is removed from M_i. Otherwise, let t be the last time at which $L^s_t(M_i) \leq w_i \cdot T^s_{n+1}$ or $L_t(M_i) \leq (r-1) \cdot T_{n+1} \cdot s_i/S$. Such a time t exists because the condition is met for $t = 1$. Note that the condition is slightly different than the negation of the condition for job removals in the migration phase because we are using $L^s_t(M_i)$ rather than $L_t(M_i)$ in the first part. We do this so that the first part of the condition aligns with the condition for the placement of small jobs in the scheduling phase.

It is sufficient to remove the following jobs from M_i to guarantee $L(M_i) \leq w_i \cdot T^s_{n+1}$ or $L(M_i) \leq (r-1) \cdot T_{n+1} \cdot s_i/S$.

(a) All jobs that are large at time t and are scheduled on M_i before the arrival of the t-th job and
(b) all jobs assigned to M_i in step t or after.

At any time t' (before the arrival of the t'-th job), there are at most $b \cdot m \cdot s_i/S$ jobs that are large at time t' scheduled on M_i. Suppose this is not true and let t' be the first time at which this is not true. Then there were $b \cdot m \cdot s_i/S$ jobs of size greater than $T_{t'}/(b \cdot m)$ scheduled on M_i at time $t' - 1$ and in step $t' - 1$ one more such job J is assigned to M_i. However, before the assignment of J, the load of M_i is $L_{t'-1}(M_i) > T_{t'} \cdot s_i/S \geq T_{t'-1} \cdot s_i/S$. Then M_i cannot be a machine with minimum completion time among all machines in step t' and therefore a large job J would not be assigned to it. We conclude that at most $b \cdot m \cdot s_i/S$ jobs are removed in (a).

To bound the number of jobs removed in (b), we observe that in steps $t+1, \ldots, n$ our algorithm only allocates jobs to M_i that are large at the time of allocation. This is due to the fact that by definition of t, for each $t' \geq t+1$, $L^s_{t'}(M_i) > w_i \cdot T^s_{t'}$. Therefore, whenever a job J is assigned to M_i in a step $t' \geq t+1$, it is a large job, which is assigned to a machine of minimum completion time. But then, for each $0 \leq j \leq m-1$, $L_{t'}(M_j) > (r-1) \cdot T_{n+1} \cdot s_j/S$, because we also have $L_{t'}(M_i) > (r-1) \cdot T_{n+1} \cdot s_i/S$. Hence $T_{t'} = \sum_{j=0}^{m-1} L_{t'}(M_j) > (r-1) \cdot T_{n+1}$. Since job J is large at the time of assignment, its size has to be greater than $(r-1) \cdot T_{n+1}/(b \cdot m)$. After assigning $b \cdot m \cdot s_i/(S \cdot (r-1))$ such jobs to M_i in steps after t, the load of M_i exceeds $T_{n+1} \cdot s_i/S$. After that, no further such jobs are assigned to M_i, because a machine with load greater than $T_{n+1} \cdot s_i/S$ can never be a machine that has the smallest completion time among all machines. We conclude that, at most $b \cdot m \cdot s_i/(S \cdot (r-1)) + 1$ jobs are removed in (b), where the additive 1 is due to the job that is assigned to machine M_i in step t.

In total, it is sufficient to remove these $b \cdot m \cdot s_i/S + b \cdot m \cdot s_i/(S \cdot (r-1)) + 1 = r/(r-1) \cdot b \cdot m \cdot s_i/S + 1$ many jobs, and, because the algorithm removes jobs from M_i in decreasing order of size, the number of jobs removed is bounded by the same number. □

Due to Lemma 7, the total number of jobs migrated is bounded by

$$\sum_{i=0}^{m-1} \left(\frac{r}{r-1} \cdot b \cdot m \cdot \frac{s_i}{S} + 1 \right) = \left(\frac{r}{r-1} \cdot b + 1 \right) \cdot m.$$

Recall, that we only migrate jobs when $s_{m-1} \leq 3/4 \cdot S$, as otherwise, we simply schedule all jobs on machine M_{m-1}. If $s_{m-1} \leq 3/4 \cdot S$, according to Corollary 16 and Observation 18 in the "Appendix", $16/13 \leq r \leq \mathrm{W}_{-1}(-1/e^2)/(1 + \mathrm{W}_{-1}(-1/e^2)) \approx 1.4659$. Hence, we number of migrated jobs is at most

$$\left(\frac{r}{r-1} \cdot b + 1 \right) \cdot m = \Theta(m).$$

3.1.2 Bounding the Competitive Ratio

If $s_{m-1} > 3/4 \cdot S$, we assign all jobs to machine M_{m-1}. The resulting makespan is $L_{n+1}(M_{m-1})/s_{m-1} = T_{n+1}/s_{m-1} < 4/3 \cdot T_{n+1}/S \leq 4/3 \cdot \mathrm{OPT}$, where OPT denotes the optimal makespan. Hence the competitive ratio is bounded by $1 + 1/3$.

For the remainder of the paper, we consider the case $s_{m-1} \leq 3/4 \cdot S$. The following lemma shows that, at the end of step (1b), there are no critical machines. In fact, it gives a lower bound on $\sum_{j=0}^{m-1} L(M'_j)$.

Lemma 8 *At the end of step (1b), for each* $0 \leq j \leq m-1$,

$$\sum_{k=0}^{m-1} L(M'_k) \geq \left(\frac{L(M'_j)}{s_j} - \frac{\mathrm{OPT}}{3} \right) \cdot \sum_{k=j}^{m-1} s_k.$$

Proof Clearly we have

$$\left(\frac{L(M'_j)}{s_j} - \frac{\mathrm{OPT}'}{3} \right) \cdot \sum_{k=j}^{m-1} s_k \geq \left(\frac{L(M'_j)}{s_j} - \frac{\mathrm{OPT}}{3} \right) \cdot \sum_{k=j}^{m-1} s_k,$$

because $\mathrm{OPT}' \leq \mathrm{OPT}$ (optimally scheduling a subset of all jobs can only result in a smaller makespan than optimally scheduling all jobs). Therefore, it only remains to show

$$\sum_{k=0}^{m-1} L(M'_k) \geq \left(\frac{L(M'_j)}{s_j} - \frac{\mathrm{OPT}'}{3} \right) \cdot \sum_{k=j}^{m-1} s_k.$$

If $C = \emptyset$, the lemma is true by definition of C. In the following, we consider the case $C \neq \emptyset$. At the end of step (1b), for each $0 \leq j \leq c$, $L(M'_j) = 0$ and, as a consequence, the lemma is true for these machines. In the following, we show that the lemma is true for $M'_{c+1}, \ldots, M'_{m-1}$ after each reassignment in step (1b), if it is true for these machines before this reassignment.

Initially, at the beginning of step (1b), for each $c+1 \leq j \leq m-1$, M'_j is not critical by definition of c, i.e., the lemma is true for M'_j.

Now, consider a reassignment in step (1b). For each $0 \leq j \leq m-1$, let $L(M'_j)$ and $\hat{L}(M'_j)$ denote the load of machine M'_j before and after this reassignment, respectively. Assume that the lemma is true for $M'_{c+1}, \ldots, M'_{m-1}$ before this reassignment.

In this reassignment, all jobs from M'_i, with $0 \leq i \leq c$, are reassigned to M'_ℓ, with

$$\ell = \max \left\{ c+1 \leq j \leq m-1 : \frac{L(M'_i) + L(M'_j)}{s_j} \leq \frac{4}{3} \cdot \mathrm{OPT}' \right\}.$$

Then, re-sort the loads of the machines again. Specifically,

$$z = \max \left\{ \ell \leq j \leq m-1 : L(M'_j) < L(M'_i) + L(M'_\ell) \right\},$$

i.e., after re-sorting, $\hat{L}(M'_z) = L(M'_i) + L(M'_\ell)$, and, for each $\ell \leq j \leq z-1$, $\hat{L}(M'_j) = L(M'_{j+1})$. In addition, for each $j \in \{c+1, \ldots, m-1\} \setminus \{\ell, \ldots, z\}$, $\hat{L}(M'_j) = L(M'_j)$.

Note that, for each $c+1 \leq j \leq m-1$, $L(M'_j) \leq \hat{L}(M'_j)$ and, if $j+1 \leq m-1$, $\hat{L}(M'_j) \leq L(M'_{j+1})$.

It remains to show that the lemma is true for $M'_\ell, \ldots, M'_z$. Consider machine M'_x with $\ell \leq x \leq z$. If $\hat{L}(M'_x)/s_x \leq \mathrm{OPT}'$, then

$$\sum_{j=0}^{m-1} \hat{L}(M'_j) \geq \sum_{j=x}^{m-1} \hat{L}(M'_j) \geq \frac{\hat{L}(M'_x)}{s_x} \cdot s_x + \sum_{j=x+1}^{m-1} \frac{2}{3} \cdot \mathrm{OPT}' \cdot s_j \geq \left(\frac{\hat{L}(M'_x)}{s_x} - \frac{\mathrm{OPT}'}{3} \right) \cdot \sum_{j=x}^{m-1} s_j,$$

since, by definition of ℓ, for each $\ell+1 \leq j \leq m-1$, $\hat{L}(M'_j)/s_j \geq L(M'_j)/s_j \geq (L(M'_i)+L(M'_j))/(2s_j) > 2/3 \cdot \mathrm{OPT}'$.

In the following, we consider the case $\hat{L}(M'_x)/s_x > \mathrm{OPT}'$.

Observation 9 For each $x+1 \leq j \leq m-1$, $L(M'_j)/s_j \geq 4/5 \cdot \mathrm{OPT}'$.

Proof Assume for contradiction that there exists a $x+1 \leq j \leq m-1$ with $L(M'_j)/s_j < 4/5 \cdot \mathrm{OPT}'$. Then,

$$\frac{4}{5} \cdot \mathrm{OPT}' > \frac{L(M'_j)}{s_j} \geq \frac{L(M'_{x+1})}{s_j} \geq \frac{\hat{L}(M'_x)}{s_j} > \frac{\mathrm{OPT}' \cdot s_x}{s_j},$$

i.e., $4/5 \cdot s_j > s_x \geq s_\ell$. This yields the following contradiction to the fact that all jobs from M'_i are reassigned to M'_ℓ: $\ell < x+1 \leq j$ and

$$L(M'_i) + L(M'_j) \leq \frac{2}{3} \cdot \mathrm{OPT}' \cdot s_\ell + \frac{4}{5} \cdot \mathrm{OPT}' \cdot s_j \leq \left(\frac{2}{3} \cdot \frac{4}{5} + \frac{4}{5} \right) \cdot \mathrm{OPT}' \cdot s_j \leq \frac{4}{3} \cdot \mathrm{OPT}' \cdot s_j,$$

since, by definition of ℓ, $L(M'_i)/s_\ell \leq (L(M'_i)+L(M'_\ell))/(2s_\ell) \leq 2/3 \cdot \mathrm{OPT}'$. □

Due to the fact that M'_c is critical,

$$\sum_{j=c}^{m-1} L(M'_j) \leq \sum_{j=0}^{m-1} L(M'_j) \leq \left(\frac{L(M'_c)}{s_c} - \frac{\mathrm{OPT}'}{3} \right) \cdot \sum_{j=c}^{m-1} s_j.$$

As a consequence, by subtracting $L(M'_c)/s_c \geq L(M'_c)/s_c - \mathrm{OPT}'/3$,

$$\sum_{j=c+1}^{m-1} L(M'_j) \leq \left(\frac{L(M'_c)}{s_c} - \frac{\mathrm{OPT}'}{3} \right) \cdot \sum_{j=c+1}^{m-1} s_j \leq \frac{2}{3} \cdot \mathrm{OPT}' \cdot \sum_{j=c+1}^{m-1} s_j,$$

where the second step follows from $L(M'_c)/s_c \leq \mathrm{OPT}'$.

Due to Observation 5 item (2),

$$\sum_{j=c+1}^{x} L(M'_j) \geq \frac{1}{2} \cdot \mathrm{OPT}' \cdot \sum_{j=c+1}^{x} s_j$$

and, due to Observation 9,

$$\sum_{j=x+1}^{m-1} L(M'_j) \geq \frac{4}{5} \cdot \mathrm{OPT}' \cdot \sum_{j=x+1}^{m-1} s_j.$$

Hence,

$$\begin{aligned}\frac{2}{3} \cdot \mathrm{OPT}' \cdot \sum_{j=c+1}^{m-1} s_j &\geq \sum_{j=c+1}^{m-1} L(M'_j) \\ &\geq \frac{1}{2} \cdot \mathrm{OPT}' \cdot \sum_{j=c+1}^{x} s_j + \frac{4}{5} \cdot \mathrm{OPT}' \cdot \sum_{j=x+1}^{m-1} s_j.\end{aligned}$$

As a consequence,

$$\frac{2}{3} \cdot \sum_{j=c+1}^{x} s_j + \frac{2}{3} \cdot \sum_{j=x+1}^{m-1} s_j \geq \frac{1}{2} \cdot \sum_{j=c+1}^{x} s_j + \frac{4}{5} \cdot \sum_{j=x+1}^{m-1} s_j,$$

i.e.,

$$\sum_{j=c+1}^{x} s_j \geq \frac{4}{5} \cdot \sum_{j=x+1}^{m-1} s_j.$$

Altogether,

$$\begin{aligned}\sum_{j=0}^{m-1} \hat{L}(M'_j) &\geq \sum_{j=c+1}^{x-1} L(M'_j) + \hat{L}(M'_x) + \sum_{j=x+1}^{m-1} L(M'_j) \\ &\geq \frac{1}{2} \cdot \mathrm{OPT}' \cdot \sum_{j=c+1}^{x-1} s_j + \frac{\mathrm{OPT}'}{3} \cdot s_x + \left(\frac{\hat{L}(M'_x)}{s_x} - \frac{\mathrm{OPT}'}{3} \right) \cdot s_x \\ &\quad + \frac{4}{5} \cdot \mathrm{OPT}' \cdot \sum_{j=x+1}^{m-1} s_j \\ &\geq \frac{1}{3} \cdot \mathrm{OPT}' \cdot \sum_{j=c+1}^{x} s_j + \left(\frac{\hat{L}(M'_x)}{s_x} - \frac{\mathrm{OPT}'}{3} \right) \cdot s_x\end{aligned}$$

$$
\begin{aligned}
&+ \frac{4}{5} \cdot \mathrm{OPT}' \cdot \sum_{j=x+1}^{m-1} s_j \\
&\geq \left(\frac{\hat{L}(M'_x)}{s_x} - \frac{\mathrm{OPT}'}{3} \right) \cdot s_x + \left(\frac{1}{3} \cdot \frac{4}{5} + \frac{4}{5} \right) \cdot \mathrm{OPT}' \cdot \sum_{j=x+1}^{m-1} s_j \\
&\geq \left(\frac{\hat{L}(M'_x)}{s_x} - \frac{\mathrm{OPT}'}{3} \right) \cdot \sum_{j=x}^{m-1} s_j,
\end{aligned}
$$

where the second Step uses Observation 5 item (2) and Observation 9, the fourth step uses $\sum_{j=c+1}^{x} s_j \geq 4/5 \cdot \sum_{j=x+1}^{m-1} s_j$, and the fifth step uses $\hat{L}(M'_x)/s_x \leq 4/3 \cdot \mathrm{OPT}'$ which holds due to Observation 5 item (4). □

Next, we give a bound on the makespan at the end of step (1) of the migration phase. We distinguish two cases.

- $L(M_i) \leq (r-1) \cdot T_{n+1} \cdot s_i / S$ after the removal of jobs:
  Then, $L(M_i) \leq (r-1) \cdot T_{n+1} \cdot s_i / S \leq (r-1) \cdot \mathrm{OPT} \cdot s_i$. The completion time of machine M_i at the end of step (1) of the migration phase is $(L(M_i) + L(M'_i))/s_i \leq (r-1) \cdot \mathrm{OPT} + 4/3 \cdot \mathrm{OPT} \leq (r+1/3) \cdot \mathrm{OPT}$, since $L(M'_i)/s_i \leq 4/3 \cdot \mathrm{OPT}' \leq 4/3 \cdot \mathrm{OPT}$ due to Observation 5 item (4).
- $L(M_i) > (r-1) \cdot T_{n+1} \cdot s_i / S$ after the removal of jobs:
  Then, $L(M_i) \leq w_i \cdot T^s_{n+1}$ after the removal of jobs. $S \cdot \mathrm{OPT}$ is an upper bound on the total size of all jobs in the input and the virtual machines only contain jobs which are large at time $n+1$. Therefore $T^s_{n+1} \leq S \cdot \mathrm{OPT} - \sum_{j=0}^{m-1} L(M'_j)$. We distinguish two sub-cases.

  - $w_i = s_i \cdot r / S$:
    By definition of w_i, $\sum_{j=0}^{i-1} s_j \leq (r-1)/r \cdot S$ and, as a consequence,

$$
\sum_{j=i}^{m-1} s_j = S - \sum_{j=0}^{i-1} s_j \geq S - \frac{r-1}{r} \cdot S = \frac{S}{r}.
$$

    Then we can bound the completion time of machine M_i at the end of step (1) of the migration phase as follows:

$$
\begin{aligned}
\frac{L(M_i)}{s_i} &\leq \frac{w_i}{s_i} \left(S \cdot \mathrm{OPT} - \sum_{j=0}^{m-1} L(M'_j) \right) + \frac{L(M'_i)}{s_i} \\
&\leq \frac{r}{S} \left(S \cdot \mathrm{OPT} - \max \left\{ 0, \frac{L(M'_i)}{s_i} - \frac{\mathrm{OPT}}{3} \right\} \sum_{j=i}^{m-1} s_j \right) + \frac{L(M'_i)}{s_i} \\
&\leq \frac{r}{S} \left(S \cdot \mathrm{OPT} - \max \left\{ 0, \frac{L(M'_i)}{s_i} - \frac{\mathrm{OPT}}{3} \right\} \frac{S}{r} \right) + \frac{L(M'_i)}{s_i} \\
&\leq r \cdot \mathrm{OPT} + \frac{1}{3} \cdot \mathrm{OPT},
\end{aligned}
$$

where the second step uses Lemma 8.

- $w_i = s_i \cdot (r-1)/\sum_{j=0}^{i-1} s_j$:
  By definition of w_i,

$$\frac{r-1}{r} \cdot S \leq \sum_{j=0}^{i-1} s_j.$$

Then we can bound the completion time of machine M_i at the end of step (1) of the migration phase as follows:

$$\begin{aligned}
\frac{L(M_i)}{s_i} &\leq \frac{w_i}{s_i} \cdot \left(S \cdot \mathrm{OPT} - \sum_{j=0}^{m-1} L(M'_j) \right) + \frac{L(M'_i)}{s_i} \\
&\leq \frac{r-1}{\sum_{j=0}^{i-1} s_j} \cdot \left(S \cdot \mathrm{OPT} - \max\left\{0, \frac{L(M'_i)}{s_i} - \frac{\mathrm{OPT}}{3}\right\} \cdot \sum_{j=i}^{m-1} s_j \right) \\
&\quad + \frac{L(M'_i)}{s_i} \\
&\leq \frac{r-1}{\sum_{j=0}^{i-1} s_j} \cdot S \cdot \left(\mathrm{OPT} - \max\left\{0, \frac{L(M'_i)}{s_i} - \frac{\mathrm{OPT}}{3}\right\} \right) \\
&\quad + (r-1) \cdot \max\left\{0, \frac{L(M'_i)}{s_i} - \frac{\mathrm{OPT}}{3}\right\} + \frac{L(M'_i)}{s_i} \\
&\leq r \cdot \mathrm{OPT} - \max\left\{0, \frac{L(M'_i)}{s_i} - \frac{\mathrm{OPT}}{3}\right\} + \frac{L(M'_i)}{s_i} \\
&\leq r \cdot \mathrm{OPT} + \frac{1}{3} \cdot \mathrm{OPT},
\end{aligned}$$

where the second step uses Lemma 8, the third step uses $\sum_{j=i}^{m-1} s_j = S - \sum_{j=0}^{i-1} s_j$ and $4/3 \cdot \mathrm{OPT} - L(M'_i)/s_i \geq 4/3 \cdot \mathrm{OPT}' - L(M'_i)/s_i \geq 0$ which holds due to Observation 5 item (4), and the fourth step uses $\sum_{j=0}^{i-1} s_j \geq (r-1)/r \cdot S$.

In all cases, the makespan is at most $(r + 1/3) \cdot \mathrm{OPT}$ at the end of step (1) of the migration phase.

Finally, we analyze the makespan at the end of step (2) of the migration phase. We start with the following observation.

Observation 10 There exists a machine M_i with $m_b \leq i \leq m-1$ and completion time of at most $(\sqrt{b}+1)/\sqrt{b} \cdot \mathrm{OPT}$, where

$$m_b = \min\left\{ 0 \leq i \leq m-1 : \sum_{j=0}^{i} s_j \geq \frac{S}{\sqrt{b}+1} \right\}.$$

Proof Assume for contradiction that, for each $m_b \le j \le m-1$, $L(M_j)/s_j > (\sqrt{b}+1)/\sqrt{b} \cdot \mathrm{OPT}$. This yields the following contradiction:

$$\begin{aligned}\mathrm{OPT} \cdot S \ge \sum_{j=m_b}^{m-1} L(M_j) &> \sum_{j=m_b}^{m-1} \frac{\sqrt{b}+1}{\sqrt{b}} \cdot \mathrm{OPT} \cdot s_j \\ &= \frac{\sqrt{b}+1}{\sqrt{b}} \cdot \mathrm{OPT} \cdot \left(S - \sum_{j=0}^{m_b-1} s_j \right) \\ &\ge \frac{\sqrt{b}+1}{\sqrt{b}} \cdot \mathrm{OPT} \cdot S \cdot \frac{\sqrt{b}}{\sqrt{b}+1},\end{aligned}$$

since $\sum_{j=0}^{m_b-1} s_j < S/(\sqrt{b}+1)$ by the definition of m_b. □

Consider a removed job J that is scheduled in step (2) of the migration phase. Since J is small at time $n+1$, $p(J) \le T_{n+1}/(b \cdot m) \le \mathrm{OPT} \cdot S/(b \cdot m)$. According to Observation 10, there exists a machine M_i with $m_b \le i \le m-1$ and completion time of at most $(\sqrt{b}+1)/\sqrt{b} \cdot \mathrm{OPT}$. Since $\sum_{j=0}^{m_b} s_j \ge S/(\sqrt{b}+1)$, $s_i \ge \sum_{j=0}^{i} s_j/(i+1) \ge S/((\sqrt{b}+1) \cdot m)$. In step (2) of the migration phase, J is assigned to a machine finishing it first. Then, we can bound the completion time of this machine after J is assigned to it as follows:

$$\begin{aligned}\frac{L(M_i)}{s_i} + \frac{p(J)}{s_i} &\le \frac{\sqrt{b}+1}{\sqrt{b}} \cdot \mathrm{OPT} + \mathrm{OPT} \cdot \frac{S}{b \cdot m} \cdot \frac{(\sqrt{b}+1) \cdot m}{S} \\ &= \left(\frac{\sqrt{b}+1}{\sqrt{b}} \right)^2 \cdot \mathrm{OPT}.\end{aligned}$$

At the end of the migration phase, the makespan is at most $\max\{r+1/3, (1+1/\sqrt{b})^2\} \cdot \mathrm{OPT}$. Recall that $1 < r \le \mathrm{W}_{-1}(-1/e^2)/(1+\mathrm{W}_{-1}(-1/e^2)) \approx 1.4659$. For example, for $b = 8.5827$, $(1+1/\sqrt{b})^2 \le 1.4659+1/3$, and, for $b = 41.7847$, $(1+1/\sqrt{b})^2 \le 4/3$.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

Appendix: Properties of *r*

For $y \geq 1$, define

$$f(x, y) = \begin{cases} \frac{y}{S}, & \text{if } x \leq \frac{y-1}{y} \cdot S \\ \frac{y-1}{x}, & \text{if } x > \frac{y-1}{y} \cdot S \end{cases}.$$

Further, for each $0 \leq i \leq m-1$, define $c_i = \sum_{j=0}^{i-1} s_j$. Then, for each $0 \leq i \leq m-1$, $w_i = s_i \cdot f(c_i, r)$.

Observation 11 For any $x < S$, $f(x, y)$ is continuous and monotonically increasing in y.

Proof For any fixed $x \leq 0$, $f(x, y) = y/S$ is continuous and monotonically increasing in y. For any fixed $0 < x < S$, each of the branches is continuous and monotonically increasing in y and for the limits at $y = S/(S-x)$:

$$\lim_{y \to S/(S-x)^-} f(x, y) = \frac{S}{S \cdot (S-x)} = \frac{x}{x \cdot (S-x)} = \lim_{y \to S/(S-x)^+} f(x, y).$$

□

Observation 12 For any $y > 1$, $f(x, y)$ is continuous and monotonically decreasing in x.

Proof For any fixed $y > 1$, each of the branches is continuous and monotonically decreasing in x and for the limits at $x = (y-1)/y \cdot S$:

$$\lim_{x \to (y-1)/y \cdot S^-} f(x, y) = \frac{(y-1) \cdot y}{(y-1) \cdot S} = \lim_{x \to (y-1)/y \cdot S^+} f(x, y).$$

□

Observation 13 For any constant $t \geq 0$ and any $y \geq 1$,

$$\int_{x=0}^{S} f(x-t, y)\, dx = \frac{y}{S} \cdot t + (y-1) \cdot \left(1 + \ln\left(\frac{y}{y-1}\right) + \ln\left(\frac{S-t}{S}\right)\right).$$

Proof

$$\begin{aligned} \int_{x=0}^{S} f(x-t, y)\, dx &= \int_{x=0}^{(y-1)/y \cdot S+t} \frac{y}{S}\, dx + \int_{x=(y-1)/y \cdot S+t}^{S} \frac{y-1}{x-t}\, dx \\ &= \frac{y}{S} \cdot t + (y-1) \cdot \left(1 + \ln\left((S-t) \cdot \frac{y}{(y-1) \cdot S}\right)\right) \\ &= \frac{y}{S} \cdot t + (y-1) \cdot \left(1 + \ln\left(\frac{y}{y-1}\right) + \ln\left(\frac{S-t}{S}\right)\right). \end{aligned}$$

□

Corollary 14 *For any constant $t \geq 0$,*

$$\lim_{S\to\infty} \int_{x=0}^{S} f(x-t, y)\, dx = (y-1) \cdot \left(1 + \ln\left(\frac{y}{y-1}\right)\right) = \int_{x=0}^{S} f(x, y)\, dx.$$

Observation 15 (i) For $r \geq 1$, $\sum_{i=0}^{m-1} w_i$ is continuous in r.
(ii) For $r = 1$, $\sum_{i=0}^{m-1} w_i < 1$.
(iii) For $r = \mathrm{W}_{-1}(-1/e^2)/(1 + \mathrm{W}_{-1}(-1/e^2))$, $\sum_{i=0}^{m-1} w_i \geq 1$.

Proof (i) Due to Observation 11 and $0 = c_0 < \cdots < c_{m-1} < S$, $\sum_{i=0}^{m-1} w_i = \sum_{i=0}^{m-1} s_i \cdot f(c_i, r)$ is continuous in r.
(ii) For $r = 1$, $\sum_{i=0}^{m-1} w_i = w_0 = s_0 \cdot r/S < 1$, since $w_1 = \cdots = w_{m-1} = 0$ and $s_0 < S$.
(iii) Due to Observation 12,

$$\begin{aligned}\sum_{i=0}^{m-1} w_i = \sum_{i=0}^{m-1} s_i \cdot f(c_i, r) &\geq \sum_{i=0}^{m-1} \int_{x=c_i}^{c_{i+1}} f(x, r)\, dx \\ &= \int_{x=0}^{S} f(x, r)\, dx,\end{aligned}$$

since, for each $0 \leq i \leq m-1$, $c_{i+1} = c_i + s_i$.
Due to Obs. 13, $\int_{x=0}^{S} f(x, r)\, dx = 1$ for $r = \mathrm{W}_{-1}(-1/e^2)/(1 + \mathrm{W}_{-1}(-1/e^2))$. □

The above observation and the intermediate value theorem gives the following corollary.

Corollary 16 *A solution r to $\sum_{i=0}^{m-1} w_i = 1$ exists, with*

$$1 < r \leq \frac{\mathrm{W}_{-1}(-1/e^2)}{1 + \mathrm{W}_{-1}(-1/e^2)}.$$

The latter term is approximately 1.4659 and in particular less than 3/2.

Observation 17 For any constant $t \geq 1$, with $s_i \leq t$ for all $0 \leq i \leq m-1$,

$$\lim_{m\to\infty} r = \frac{\mathrm{W}_{-1}(-1/e^2)}{1 + \mathrm{W}_{-1}(-1/e^2)}.$$

Proof Due to Observation 12, for $0 \leq i \leq m-1$ and $x \leq c_i + s_i = c_{i+1}$, $f(c_i, r) \leq f(x - s_i, r) \leq f(x - t, r)$. As a consequence,

$$\int_{x=0}^{S} f(x,r)\,dx \le \sum_{i=0}^{m-1} w_i = \sum_{i=0}^{m-1} s_i \cdot f(c_i,r)$$
$$\le \sum_{i=0}^{m-1} \int_{x=c_i}^{c_{i+1}} f(x-t,r)\,dx$$
$$= \int_{x=0}^{S} f(x-t,r)\,dx\,.$$

Now, due to Observation 14 and $\lim_{m\to\infty} S = \infty$, $\lim_{m\to\infty} \sum_{i=0}^{m-1} w_i = (r-1)\cdot(1+\ln(r/(r-1)))$. Finally, $\lim_{m\to\infty} r = \mathrm{W}_{-1}(-1/e^2)/(1+\mathrm{W}_{-1}(-1/e^2))$ for $\lim_{m\to\infty} \sum_{i=0}^{m-1} w_i = 1$. □

Observation 18 If $s_0 \le \cdots \le s_{m-1}$ and $s_{m-1} \le 3/4 \cdot S$, $r \ge 16/13$.

Proof Pick the smallest ℓ such that $\sum_{i=0}^{\ell} s_i \ge 1/4 \cdot S$. Then $\sum_{i=0}^{\ell-1} s_i < 1/4 \cdot S$. Note that, because $s_{m-1} \le 3/4 \cdot S$, we know that $\ell \le m-2$ and therefore $s_\ell + s_{m-1} \le S$. Hence, because $s_\ell \le s_{m-1}$, $s_\ell \le S/2$. Thus, $\sum_{i=0}^{\ell} s_i = s_\ell + \sum_{i=0}^{\ell-1} s_i \le 1/2 \cdot S + 1/4 \cdot S = 3/4 \cdot S$. To summarize, we have

$$\frac{1}{4}\cdot S \le \sum_{i=0}^{\ell} s_i \le \frac{3}{4}\cdot S.$$

Due to the definition of the weights w_i, we also have

$$1 = \sum_{i=0}^{m-1} w_i \le \sum_{i=0}^{\ell} s_i \cdot \frac{r}{S} + \sum_{i=\ell+1}^{m-1} s_i \cdot \frac{r-1}{\sum_{j=0}^{i-1} s_j}$$
$$\le \sum_{i=0}^{\ell} s_i \cdot \frac{r}{S} + \sum_{i=\ell+1}^{m-1} s_i \cdot \frac{r-1}{\sum_{j=0}^{\ell} s_j}$$
$$= \sum_{i=0}^{\ell} s_i \cdot \frac{r}{S} + \left(S - \sum_{i=0}^{\ell} s_i\right) \cdot \frac{r-1}{\sum_{j=0}^{\ell} s_j}$$

which implies

$$r \ge \frac{S^2}{S\cdot(S-\sum_{i=0}^{\ell} s_i) + (\sum_{i=0}^{\ell} s_i)^2}.$$

Together with $1/4 \cdot S \le \sum_{i=0}^{\ell} s_i \le 3/4 \cdot S$, this implies $r \ge 16/13$. □

References

1. Albers, S.: Better bounds for online scheduling. SIAM J. Comput. **29**(2), 459–473 (1999)
2. Albers, S., Hellwig, M.: On the value of job migration in online makespan minimization. Algorithmica **79**(2), 598–623 (2017)
3. Aspnes, J., Azar, Y., Fiat, A., Plotkin, S.A., Waarts, O.: On-line routing of virtual circuits with applications to load balancing and machine scheduling. J. ACM **44**(3), 486–504 (1997)
4. Bartal, Y., Fiat, A., Karloff, H.J., Vohra, R.: New algorithms for an ancient scheduling problem. J. Comput. Syst. Sci. **51**(3), 359–366 (1995)
5. Bartal, Y., Karloff, H.J., Rabani, Y.: A better lower bound for on-line scheduling. Inf. Process. Lett. **50**(3), 113–116 (1994)
6. Berman, P., Charikar, M., Karpinski, M.: On-line load balancing for related machines. J. Algorithms **35**(1), 108–121 (2000)
7. Chen, B., van Vliet, A., Woeginger, G.J.: New lower and upper bounds for on-line scheduling. Oper. Res. Lett. **16**(4), 221–230 (1994)
8. Chen, X., Lan, Y., Benko, A., Dósa, G., Han, X.: Optimal algorithms for online scheduling with bounded rearrangement at the end. Theor. Comput. Sci. **412**(45), 6269–6278 (2011)
9. Ding, N., Lan, Y., Chen, X., Dósa, G., Guo, H., Han, X.: Online minimum makespan scheduling with a buffer. Int. J. Found. Comput. Sci. **25**(5), 525–536 (2014)
10. Dósa, G., Epstein, L.: Online scheduling with a buffer on related machines. J. Combin. Optim. **20**(2), 161–179 (2010)
11. Dósa, G., Epstein, L.: Preemptive online scheduling with reordering. SIAM J. Discrete Math. **25**(1), 21–49 (2011)
12. Dósa, G., Wang, Y., Han, X., Guo, H.: Online scheduling with rearrangement on two related machines. Theor. Comput. Sci. **412**(8–10), 642–653 (2011)
13. Ebenlendr, T., Sgall, J.: A lower bound on deterministic online algorithms for scheduling on related machines without preemption. Theory Comput. Syst. **56**(1), 73–81 (2015)
14. Englert, M., Özmen, D., Westermann, M.: The power of reordering for online minimum makespan scheduling. SIAM J. Comput. **43**(3), 1220–1237 (2014)
15. Epstein, L., Favrholdt, L.M.: Optimal preemptive semi-online scheduling to minimize makespan on two related machines. Oper. Res. Lett. **30**(4), 269–275 (2002)
16. Epstein, L., Levin, A., van Stee, R.: Max-min online allocations with a reordering buffer. SIAM J. Discrete Math. **25**(3), 1230–1250 (2011)
17. Faigle, U., Kern, W., Turán, G.: On the performance of on-line algorithms for partition problems. Acta Cybern. **9**(2), 107–119 (1989)
18. Fleischer, R., Wahl, M.: On-line scheduling revisited. J. Schedul. **3**(6), 343–353 (2000)
19. Friesen, D.K.: Tighter bounds for LPT scheduling on uniform processors. SIAM J. Comput. **16**(3), 554–560 (1987)
20. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman (1979)
21. Gormley, T., Reingold, N., Torng, E., Westbrook, J.: Generating adversaries for request-answer games. In: Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 564–565 (2000)
22. Graham, R.L.: Bounds for certain multiprocessing anomalies. Bell Syst. Tech. J. **45**(1), 1563–1581 (1966)
23. Graham, R.L.: Bounds on multiprocessing timing anomalies. SIAM J. Appl. Math. **17**(2), 416–429 (1969)
24. Hochbaum, D.S., Shmoys, D.B.: A polynomial approximation scheme for scheduling on uniform processors: using the dual approximation approach. SIAM J. Comput. **17**(3), 539–551 (1988)
25. Karger, D.R., Phillips, S.J., Torng, E.: A better algorithm for an ancient scheduling problem. J. Algorithms **20**(2), 400–430 (1996)
26. Kellerer, H., Kotov, V., Speranza, M.G., Tuza, Z.: Semi on-line algorithms for the partition problem. Oper. Res. Lett. **21**(5), 235–242 (1997)
27. Liu, M., Xu, Y., Chu, C., Zheng, F.: Online scheduling on two uniform machines to minimize the makespan. Theor. Comput. Sci. **410**(21–23), 2099–2109 (2009)
28. Mireault, P., Orlin, J.B., Vohra, R.V.: A parametric worst case analysis of the LPT heuristic for two uniform machines. Oper. Res. **45**(1), 116–125 (1997)

29. Pruhs, K., Sgall, J., Torng, E.: Handbook of Scheduling: Algorithms, Models, and Performance Analysis, chap, Online Scheduling. CRC Press (2004)
30. Rudin, J.F., III: Improved Bound for the Online Scheduling Problem. Ph.D. thesis, University of Texas at Dallas (2001)
31. Rudin, J.F., III., Chandrasekaran, R.: Improved bound for the online scheduling problem. SIAM J. Comput. **32**(3), 717–735 (2003)
32. Sanders, P., Sivadasan, N., Skutella, M.: Online scheduling with bounded migration. Math. Oper. Res. **34**(2), 481–498 (2009)
33. Tan, Z., Yu, S.: Online scheduling with reassignment. Oper. Res. Lett. **36**(2), 250–254 (2008)
34. Wang, Y., Benko, A., Chen, X., Dósa, G., Guo, H., Han, X., Sik-Lányi, C.: Online scheduling with one rearrangement at the end: revisited. Inf. Process. Lett. **112**(16), 641–645 (2012)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Matthias Englert[1] · David Mezlaf[2] · Matthias Westermann[2]

David Mezlaf
david.mezlaf@tu-dortmund.de

Matthias Westermann
matthias.westermann@cs.tu-dortmund.de

[1] DIMAP and Department of Computer Science, University of Warwick, Coventry, UK

[2] Department of Computer Science, TU Dortmund, Dortmund, Germany