Online Makespan Scheduling with Job Migration on Uniform Machines

Matthias Englert
DIMAP and Department of Computer Science, University of Warwick, Coventry, UK
m.englert@warwick.ac.uk

David Mezlaf
Department of Computer Science, TU Dortmund, Dortmund, Germany
david.mezlaf@tu-dortmund.de

Matthias Westermann
Department of Computer Science, TU Dortmund, Dortmund, Germany
matthias.westermann@cs.tu-dortmund.de

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 reassign up to k jobs to different machines in the final assignment.

For m identical machines, Albers and Hellwig (Algorithmica, 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 a subtle 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.

2012 ACM Subject Classification Theory of computation → Scheduling algorithms

Keywords and phrases online algorithms, competitive analysis, minimum makespan scheduling, job migration

Digital Object Identifier 10.4230/LIPIcs.ESA.2018.26

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 in scheduling has been studied previously, see for example [8, 12, 27, 32, 33, 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.

This model was introduced by Englert, Özmen, and Westermann [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] 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.

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, Özmen, and Westermann [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 $\min\{s_0, \ldots, s_{m-1}\} = 1$. 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 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, Özmen, and Westermann [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

$$w_i = \begin{cases} s_i \cdot \frac{r_{s_0,\ldots,s_{m-1}}}{S}, & \text{if } 0 \leq \sum_{j=0}^{i-1} s_j \leq \frac{r_{s_0,\ldots,s_{m-1}}-1}{r_{s_0,\ldots,s_{m-1}}} \cdot S \\ s_i \cdot \frac{r_{s_0,\ldots,s_{m-1}}-1}{\sum_{j=0}^{i-1} s_j}, & \text{if } \frac{r_{s_0,\ldots,s_{m-1}}-1}{r_{s_0,\ldots,s_{m-1}}} \cdot S < \sum_{j=0}^{i-1} s_j < S \end{cases}.$$

Now, $r_{s_0,\ldots,s_{m-1}}$ 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. Such a solution always exists. (Due to space limitations, the proof of this claim is omitted.) Note that, if $s_0 = \cdots = s_{m-1} = 1$, the weights match those in [2, 14] and that $r_{s_0,\ldots,s_{m-1}} =: r_m$ is equal to the competitive ratio achieved in [2, 14] for m identical machines.

Unfortunately, we do not know a closed-form formula for $r_{s_0,\ldots,s_{m-1}}$, but the value can be calculated for any given $s_0, \ldots, s_{m-1}$ and $1 < r_{s_0,\ldots,s_{m-1}} \leq \mathrm{W}_{-1}(-1/e^2)/(1+\mathrm{W}_{-1}(-1/e^2)) \approx 1.4659$.[3] (Due to space limitations, the proof of this claim is omitted.) Note that, for the optimal competitive ratio r_m for m identical machines, $4/3 \leq r_m \leq \mathrm{W}_{-1}(-1/e^2)/(1+\mathrm{W}_{-1}(-1/e^2))$. Depending on the speeds of the machines, $r_{s_0,\ldots,s_{m-1}}$ can be significantly smaller than r_m.

[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.

[3] 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 $x \cdot e^x = -1/e^2$.

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, $r_m \leq \mathrm{W}_{-1}(-1/e^2)/(1+\mathrm{W}_{-1}(-1/e^2)) \approx 1.4659$. 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, say about $1/\varepsilon$, of very small jobs, say 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 load strictly less than, and bounded away from, w_i. But then another machine must have load 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 $c = \lceil -\ln(2-r)/\ln r \rceil \geq 2$ and $m \geq c^2$, no online algorithm can achieve a competitive ratio less than $r \in (1,2)$ while migrating at most $\lfloor m/c^2 \rfloor - 1$ 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 $\lfloor m/9 \rfloor - 1$ 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 $1 = s_0 \leq \cdots \leq s_{m-1}$, our online algorithm achieves a competitive ratio of $r_{s_0,\ldots,s_{m-1}} + 1/3$ 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 $1 < r_{s_0,\ldots,s_{m-1}} \leq \mathrm{W}_{-1}(-1/e^2)/(1+\mathrm{W}_{-1}(-1/e^2)) \approx 1.4659$, 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, $r_{s_0,\ldots,s_{m-1}}$ 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 $L(M'_0) \leq \cdots \leq L(M'_{m-1})$. As a consequence, 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 $M'_i, \ldots, M'_{m-1}$, which is a crucial property for achieving the optimal competitive ratio for identical machines. In the more general case of uniform machines, this is not always the case. 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, 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 with minimum load, 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, $2.5 \cdot m$ 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 as 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, Özmen, and Westermann [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 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

Due to space limitations, the proofs of the following two theorems are omitted.

▶ **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.*

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

3 Scheduling algorithm

For m uniform machines with speeds $1 = s_0 \leq \cdots \leq s_{m-1}$, 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. For simplicity, define $r = r_{s_0,\ldots,s_{m-1}}$.

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 dynamically 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 $1 = s_0 \leq \cdots \leq s_{m-1}$:

(1a) The jobs are scheduled on the virtual machines optimally, i.e., to minimize the makepsan of the virtual machines.[4] Call the resulting makespan on the virtual machines OPT'. We assume that the resulting loads of the virtual machines

[4] 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.

are sorted, i.e., $L(M'_0) \leq \cdots \leq L(M'_{m-1})$, and that, for each $1 \leq i \leq m-1$, $L(M'_i)/s_i > \mathrm{OPT}'/2$ if $L(M'_{i-1}) > 0$. (See the following Observation 3 items (1) and (2).)

(1b) Each machine M'_i, with

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

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 the machines $M'_{c+1}, \ldots, M'_{m-1}$.
For $i = 0, \ldots, c$ do the following:

- Find the largest $\ell \geq c+1$ such that $(L(M'_i) + L(M'_\ell))/s_\ell \leq 4/3 \cdot \mathrm{OPT}'$. (Due to the following Observation 3 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.
- Resort the loads of the machines such that $L(M'_0) \leq \cdots \leq L(M'_{m-1})$ again. (See the following Observation 3 item (1).)

Finally, for each $0 \leq i \leq m-1$, 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.

Due to space limitations, the proof of the following observation is omitted.

▶ **Observation 3.** *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* $1 \leq i \leq m-1$, $L(M'_i)/s_i > \mathrm{OPT}'/2$ *if* $L(M'_{i-1}) > 0$.
(3) *If* $C \neq \emptyset$, $\{c+1 \leq j \leq m-1 : (L(M'_i) + L(M'_j))/s_j \leq 4/3 \cdot \mathrm{OPT}'\} \neq \emptyset$.
(4) *For each* $0 \leq i \leq m-1$, $L(M'_i)/s_i \leq 4/3 \cdot \mathrm{OPT}'$.

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 4.** *For* m *uniform machines with speeds* $1 = s_0 \leq \cdots \leq s_{m-1}$, *our online algorithm achieves a competitive ratio of* $r_{s_0,\ldots,s_{m-1}} + 1/3$ *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 5.** *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 $t = 1$.

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, due to (a), at most $b \cdot m \cdot s_i/S$ jobs are removed.

To bound the number of jobs removed due to (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, due to (b), at most $b \cdot m \cdot s_i/(S \cdot (r-1)) + 1$ jobs are removed, 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. ◀

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 $s_{m-1} \leq 3/4 \cdot S$, $18/17 \leq r \leq \mathrm{W}_{-1}(-1/e^2)/(1+\mathrm{W}_{-1}(-1/e^2)) \approx 1.4659$. (Due to space limitations, the proof of this claim is omitted.) Hence, due to Lemma 5, the total number of jobs migrated is bounded by

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

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 reminder 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 6.** *At the end of step (1b), for each $0 \leq j \leq m-1$,*

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

Proof. The second inequality is true 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). In the following, we prove the first inequality. 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 $0 \leq j \leq c$, $L(M'_i) = 0$ 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 $L(M'_i)$ and $\hat{L}(M'_i)$ denote the load of machine M'_i 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, resort the loads of the machines again. In detail,

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

i.e., after resorting, $\hat{L}(M'_z) = L(M'_i) + L(M'_\ell)$, 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 $\hat{L}(M'_x)/s_x > \mathrm{OPT}'$. Due to space limitations, the proof of the following observation is omitted.

▶ **Observation 7.** *For each $x+1 \leq j \leq m-1$, $L(M'_j)/s_j \geq 4/5 \cdot \mathrm{OPT}'$.*

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,

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

since $L(M'_c)/s_c - \mathrm{OPT}'/3 \leq L(M'_c)/s_c \leq \mathrm{OPT}'$.

Due to Observation 3 item (2), $\sum_{j=c+1}^{x} L(M'_j) \geq 1/2 \cdot \mathrm{OPT}' \cdot \sum_{j=c+1}^{x} s_j$ and, due to Observation 7, $\sum_{j=x+1}^{m-1} L(M'_j) \geq 4/5 \cdot \mathrm{OPT}' \cdot \sum_{j=x+1}^{m-1} s_j$. Hence,

$$\frac{2}{3} \cdot \mathrm{OPT}' \cdot \left(\sum_{j=c+1}^{x} s_j + \sum_{j=x+1}^{m-1} s_j\right) \geq \sum_{j=c+1}^{m-1} L(M'_j) \geq \frac{1}{2} \cdot \mathrm{OPT}' \cdot \sum_{j=c+1}^{x} s_j + \frac{4}{5} \cdot \mathrm{OPT}' \cdot \sum_{j=x+1}^{m-1} s_j \ ,$$

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

Altogether,

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

since $\hat{L}(M'_x)/s_x \leq 4/3 \cdot \mathrm{OPT}'$ due to Observation 3 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 3 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. 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:

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

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

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

  since $\sum_{j=i}^{m-1} s_j = S - \sum_{j=0}^{i-1} s_j$ and $4/3 \cdot \mathrm{OPT} - L(M'_i)/s_i \geq 4/3 \cdot \mathrm{OPT}' - L(M'_i)/s_i \geq 0$ due to Observation 3 item (4).

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. Due to space limitations, the proof of this observation is omitted.

▶ **Observation 8.** *There exists a machine M_i with $m_b + 1 \leq i \leq m-1$ and completion time of at most $(\sqrt{b}+1)/\sqrt{b} \cdot \mathrm{OPT}$, where*

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

Consider a removed job J that is scheduled in step (2) of the migration phase. Since J is small at time $n+1$, $p(J) \leq T_{n+1}/(b \cdot m) \leq \mathrm{OPT} \cdot S/(b \cdot m)$. According to Observation 8, there exists a machine M_i with $m_b + 1 \leq i \leq m-1$ and completion time of at most $(\sqrt{b}+1)/\sqrt{b} \cdot \mathrm{OPT}$. Since $\sum_{j=0}^{m_b+1} s_j \geq S/(\sqrt{b}+1)$, $s_i \geq \sum_{j=0}^{i} s_j/(i+1) \geq S/((\sqrt{b}+1) \cdot m)$. 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:

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

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 $1 < r \leq \mathrm{W}_{-1}(-1/e^2)/(1 + \mathrm{W}_{-1}(-1/e^2)) \approx 1.4659$. For example, for $b = 8.5827$, $(1 + 1/\sqrt{b})^2 \leq 1.4659 + 1/3$, and, for $b = 41.7847$, $(1 + 1/\sqrt{b})^2 \leq 4/3$.

References

1 Susanne Albers. Better bounds for online scheduling. *SIAM Journal on Computing*, 29(2):459–473, 1999.

2 Susanne Albers and Matthias Hellwig. On the value of job migration in online makespan minimization. *Algorithmica*, 79(2):598–623, 2017.

3 James Aspnes, Yossi Azar, Amos Fiat, Serge A. Plotkin, and Orli Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM*, 44(3):486–504, 1997.

4 Yair Bartal, Amos Fiat, Howard J. Karloff, and Rakesh Vohra. New algorithms for an ancient scheduling problem. *Journal of Computer and System Sciences*, 51(3):359–366, 1995.

5 Yair Bartal, Howard J. Karloff, and Yuval Rabani. A better lower bound for on-line scheduling. *Information Processing Letters*, 50(3):113–116, 1994.

6 Piotr Berman, Moses Charikar, and Marek Karpinski. On-line load balancing for related machines. *Journal of Algorithms*, 35(1):108–121, 2000.

7 Bo Chen, André van Vliet, and Gerhard J. Woeginger. New lower and upper bounds for on-line scheduling. *Operations Research Letters*, 16(4):221–230, 1994.

8 Xin Chen, Yan Lan, Attila Benko, György Dósa, and Xin Han. Optimal algorithms for online scheduling with bounded rearrangement at the end. *Theoretical Computer Science*, 412(45):6269–6278, 2011.

9 Ning Ding, Yan Lan, Xin Chen, György Dósa, He Guo, and Xin Han. Online minimum makespan scheduling with a buffer. *International Journal of Foundations of Computer Science*, 25(5):525–536, 2014.

10 György Dósa and Leah Epstein. Online scheduling with a buffer on related machines. *Journal of Combinatorial Optimization*, 20(2):161–179, 2010.

11 György Dósa and Leah Epstein. Preemptive online scheduling with reordering. *SIAM Journal on Discrete Mathematics*, 25(1):21–49, 2011.

12 György Dósa, Yuxin Wang, Xin Han, and He Guo. Online scheduling with rearrangement on two related machines. *Theoretical Computer Science*, 412(8-10):642–653, 2011.

13 Tomás Ebenlendr and Jirí Sgall. A lower bound on deterministic online algorithms for scheduling on related machines without preemption. *Theory of Computing Systems*, 56(1):73–81, 2015.

14 Matthias Englert, Deniz Özmen, and Matthias Westermann. The power of reordering for online minimum makespan scheduling. *SIAM Journal on Computing*, 43(3):1220–1237, 2014.

15 Leah Epstein and Lene M. Favrholdt. Optimal preemptive semi-online scheduling to minimize makespan on two related machines. *Operations Research Letters*, 30(4):269–275, 2002.

16 Leah Epstein, Asaf Levin, and Rob van Stee. Max-min online allocations with a reordering buffer. *SIAM Journal on Discrete Mathematics*, 25(3):1230–1250, 2011.

17 Ulrich Faigle, Walter Kern, and György Turán. On the performance of on-line algorithms for partition problems. *Acta Cybernetica*, 9(2):107–119, 1989.

18 Rudolph Fleischer and Michaela Wahl. On-line scheduling revisited. *Journal of Scheduling*, 3(6):343–353, 2000.

19 Donald K. Friesen. Tighter bounds for LPT scheduling on uniform processors. *SIAM Journal on Computing*, 16(3):554–560, 1987.

20 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

21 Todd Gormley, Nick Reingold, Eric Torng, and Jeffery Westbrook. Generating adversaries for request-answer games. In *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 564–565, 2000.

22 Ronald L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(1):1563–1581, 1966.

23 Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal of Applied Mathematics*, 17(2):416–429, 1969.

24 Dorit S. Hochbaum and David B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM Journal on Computing*, 17(3):539–551, 1988.

25 David R. Karger, Steven J. Phillips, and Eric Torng. A better algorithm for an ancient scheduling problem. *Journal of Algorithms*, 20(2):400–430, 1996.

26 Hans Kellerer, Vladimir Kotov, Maria Grazia Speranza, and Zsolt Tuza. Semi on-line algorithms for the partition problem. *Operations Research Letters*, 21(5):235–242, 1997.

27 Ming Liu, Yinfeng Xu, Chengbin Chu, and Feifeng Zheng. Online scheduling on two uniform machines to minimize the makespan. *Theoretical Computer Science*, 410(21-23):2099–2109, 2009.

28 Paul Mireault, James B. Orlin, and Rakesh V. Vohra. A parametric worst case analysis of the LPT heuristic for two uniform machines. *Operations Research*, 45(1):116–125, 1997.

29 Kirk Pruhs, Jiri Sgall, and Eric Torng. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, chapter Online Scheduling. CRC Press, 2004.

30 John F. Rudin III. *Improved Bound for the Online Scheduling Problem.* PhD thesis, University of Texas at Dallas, 2001.

31 John F. Rudin III and R. Chandrasekaran. Improved bound for the online scheduling problem. *SIAM Journal on Computing*, 32(3):717–735, 2003.

32 Peter Sanders, Naveen Sivadasan, and Martin Skutella. Online scheduling with bounded migration. *Mathematics of Operations Research*, 34(2):481–498, 2009.

33 Zhiyi Tan and Shaohua Yu. Online scheduling with reassignment. *Operations Research Letters*, 36(2):250–254, 2008.

34 Yuxin Wang, Attila Benko, Xin Chen, György Dósa, He Guo, Xin Han, and Cecilia Sik-Lányi. Online scheduling with one rearrangement at the end: Revisited. *Information Processing Letters*, 112(16):641–645, 2012.