The Power of Reordering for Online Minimum Makespan Scheduling*

Matthias Englert
Department of Computer Science
RWTH Aachen University, Germany
englert@cs.rwth-aachen.de

Deniz Özmen
Department of Computer Science
RWTH Aachen University, Germany
deniz.oezmen@rwth-aachen.de

Matthias Westermann
Department of Computer Science
RWTH Aachen University, Germany
marsu@cs.rwth-aachen.de

Abstract

In the classic minimum makespan scheduling problem, we are given an input sequence of jobs with processing times. 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 do not require that each arriving job has to be assigned immediately to one of the machines. A reordering buffer with limited storage capacity can be used to reorder the input sequence in a restricted fashion so as to schedule the jobs with a smaller makespan. This is a natural extension of lookahead.

We present an extensive study of the power and limits of online reordering for minimum makespan scheduling. As main result, we give, for m identical machines, tight and, in comparison to the problem without reordering, much improved bounds on the competitive ratio for minimum makespan scheduling with reordering buffers. Depending on m, the achieved competitive ratio lies between 4/3 and 1.4659. This optimal ratio is achieved with a buffer of size $\Theta(m)$*. We 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. Further, we present several algorithms for different buffer sizes. Among others, we introduce, for every buffer size* $k \in [1,(m+1)/2]$*, a* $(2-1/(m-k+1))$*-competitive algorithm, which nicely generalizes the well-known result of Graham.*

For m uniformly related machines, we give a scheduling algorithm that achieves a competitive ratio of 2 *with a reordering buffer of size m. Considering that the best known competitive ratio for uniformly related machines without reordering is 5.828, this result emphasizes the power of online reordering further more.*

1. Introduction

In the classic minimum makespan scheduling problem, we are given an input sequence of jobs with processing times. 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 [18]. In this paper, we consider online scheduling algorithms 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.

Extensive work has been done to narrow the gap between upper and lower 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 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 [16], respectively.

Adding lookahead is common practice to improve the quality of solutions for online problems. The impact of lookahead has been studied for various problems, e.g., paging [26, 32], the list update problem [1], the k-server problem [8], and bin packing [22]. However, lookahead alone is not sufficient to improve the quality of solutions for the minimum makespan scheduling problem. The lookahead

*Supported by DFG grant WE 2842/1.

window can always be rendered useless by flooding it with unimportant jobs having arbitrary small processing times.

However, for many problems, including minimum makespan scheduling, it is reasonable to not only provide a lookahead to a certain number of future jobs, but additionally to allow the algorithm to choose one of these jobs for processing next and, therefore, to reorder the input sequence. The paradigm of online reordering is more powerful than lookahead alone and has received a lot of attention [3, 4, 11, 12, 15]. It has been studied, e.g., by Albers [3] and Feder et al. [15] for the problem of web caching.

We present an extensive study of the power and limits of online reordering for minimum makespan scheduling. Over a decade ago, Kellerer et al. [25] studied the impact of online reordering for minimum makespan scheduling on two identical machines. To the best of our knowledge, these are the only previously known results related to online reordering in the context of minimum makespan scheduling.

In our model, a reordering buffer can be used to reorder the input sequence of jobs in a restricted fashion. At each point in time, the reordering buffer contains the first k jobs of the input sequence that have not been assigned so far. An online scheduling algorithm has to decide which job to assign to which machine next. Upon its decision, the corresponding job is removed from the buffer and assigned to the corresponding machine, and thereafter the next job in the input sequence takes its place.

As main result, we give, for m identical machines, tight and, in comparison to the problem without reordering, much improved bounds on the competitive ratio for minimum makespan scheduling with reordering buffers. Depending on m, the achieved competitive ratio lies between 4/3 and 1.4659. This optimal ratio is achieved with a buffer of size $\Theta(m)$. We 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.

More precisely, for m identical machines, we present the following results.

- We prove a lower bound of r_m on the competitive ratio of this problem with m identical machines and a reordering buffer whose size does not depend on the input sequence. The precise value of r_m is given in Section 1.1. For example, $r_2 = 4/3$ and $\lim_{m\to\infty} r_m = \text{LambertW}_{-1}(-1/e^2)/(1 + \text{LambertW}_{-1}(-1/e^2)) \approx 1.4659$.[1]

- We introduce a fairly simple scheduling algorithm for m identical machines matching this lower bound with a reordering buffer of size $\lceil (1+2/r_m)\cdot m\rceil + 2 \leq \lceil 2.5\cdot m\rceil + 2$.

- We show a lower bound of $3/2 > r_m$ on the competitive ratio of this problem with m identical machines and a reordering buffer of size at most $\lfloor m/2\rfloor$. This lower bound improves to $1+1/\sqrt{2} \approx 1.7071$ for a reordering buffer of size at most $\lfloor m/8\rfloor$ if $m \geq 8$.

For m uniformly related machines, i.e., for m machines with different speeds, we give a scheduling algorithm that achieves a competitive ratio of 2 with a reordering buffer of size m. Our algorithm and analysis are extremely simple. Considering that the best known lower and upper bounds on the competitive ratio for uniformly related machines without reordering are 2.438 and 5.828 [9], respectively, this result emphasizes the power of online reordering further more.

In addition, we present, for m identical machines, several algorithms for different buffer sizes. In particular, we show that buffers of size $\lceil (2/3+2/(1+\ln 3))\cdot m\rceil + 1 \approx 1.6197\cdot m + 1$ and $m+1$ are sufficient to achieve the competitive ratios $3/2$ and $1+r_m/2 \leq 1.733$, respectively. For every buffer size $k \in [1,(m+1)/2]$, we introduce a $(2-1/(m-k+1))$-competitive algorithm, which nicely generalizes the well-known result of Graham [20].

In the following table, we compare, for m identical machines, the competitive ratios of our algorithm and the best known lower and upper bounds on the competitive ratio for the case that reordering is not allowed.

m	our results reordering buffer	lower bounds no reordering	upper bounds no reordering
2	1.3333	1.5	1.5
3	1.3636	1.6667	1.6667
4	1.375	1.7321	1.7333
∞	1.4659	1.8800	1.9201

Note that our results are tight, i.e., we show matching lower and upper bounds, in contrast to the problem without reordering for which there are still gaps between the lower and upper bounds.

1.1. Notations and the value of r_m

The *processing time* or *size* of a job J is denoted by $p(J)$. The *load* $L(M)$ of a machine M is defined as the sum of the sizes of the jobs assigned to machine M. The *total scheduled load* T is defined as the sum of the load of all machines. The m machines are denoted by $M_0,\ldots,M_{m-1}$.

We frequently make use of the *weight* w_i of a machine M_i which is defined as $w_i := \min\{r_m/m, (r_m-1)/i\}$ or equiva-

[1] $\text{LambertW}_{-1}(-1/e^2)$ is the smallest real solution to $x\cdot e^x = -1/e^2$.

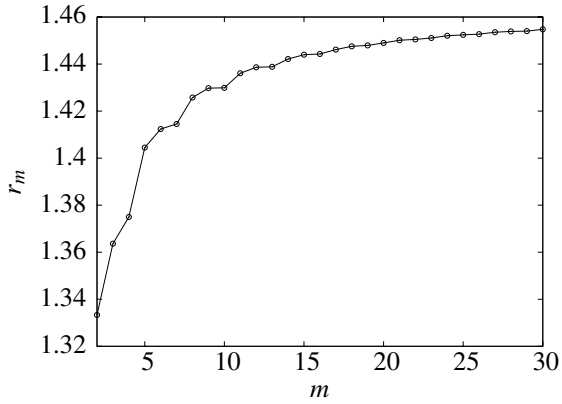

Figure 1. The values of r_m for $2 \leq m \leq 30$.

lently

$$w_i := \begin{cases} \frac{r_m}{m}, & \text{if } 0 \leq i < \frac{r_m-1}{r_m} \cdot m \\ \frac{r_m-1}{i}, & \text{if } \frac{r_m-1}{r_m} \cdot m \leq i \leq m-1 \end{cases}.$$

Now, r_m is the smallest positive solution to $\sum_{i=0}^{m-1} w_i = \lceil m - m/r_m \rceil \cdot r_m/m + (r_m - 1) \cdot \sum_{i=\lceil m-m/r_m \rceil}^{m-1} 1/i = 1$, i.e., we ensure that the weights of all machines sum up to 1.

Unfortunately, we do not know a closed-form formula for r_m, but the value can be easily calculated for any given m. The values of r_m for $2 \leq m \leq 30$ are depicted in Figure 1. We can derive $\lim_{m\to\infty} r_m = \text{LambertW}_{-1}(-1/e^2)/(1 + \text{LambertW}_{-1}(-1/e^2))$, by using $\lim_{m\to\infty} \lceil m - m/x \rceil \cdot x/m = (x-1)$ and $\lim_{m\to\infty} \sum_{i=\lceil m-m/x \rceil}^{m-1} 1/i = -\ln(1-1/x)$. Further, r_m is monotonically increasing with m. This follows from the fact that $\lceil m - m/x \rceil \cdot x/m + (x-1) \cdot \sum_{i=\lceil m-m/x \rceil}^{m-1} 1/i$ monotonically increases with x and monotonically decreases with m. A more detailed argument will be given in a full version.

1.2. Previous work

Kellerer et al. [25] present, for two identical machines, an algorithm that achieves a competitive ratio of 4/3 with a reordering buffer of size 2, i.e., the smallest buffer size allowing reordering. They also show that this bound is tight. Although, we are not aware of any other results related to online minimum makespan scheduling with reordering buffers, the paradigm of online reordering has been studied for several different scheduling problems.

In [12], a reordering buffer of size k is used to minimize the sum of the distances between consecutive elements in a sequence of points from a metric space. A randomized online algorithm is presented that achieves a competitive ratio of $O(\log^2 k \cdot \log n)$, where n denotes the number of distinct points in the metric space. A possible application is the acceleration of rendering in computer graphics [27].

Alborzi et al. [4] consider the similar k-client problem. In this problem, we are given k clients, each of which generates an input sequence of requests for service in a metric space. At each point in time, a scheduling algorithm has to decide which client's request to serve next. They present a deterministic online algorithm that achieves a competitive ratio of $2k-1$.

Web caching with request reordering extends the classic paging model by allowing reordering of requests under the constraint that a request is delayed by no longer than a predetermined number of time steps (see, e.g., [3, 15]). Albers [3] presents a deterministic algorithm that achieves an optimal competitive ratio of $k+1$, where k denotes the cache size. Feder et al. [15] introduce a randomized algorithm that achieves an asymptotically optimal competitive ratio of $\Theta(\log k)$.

Divakaran and Saks [11] consider an online scheduling problem with job set-ups. Each job has a release time, a processing time, and a type. Processing a job takes its processing time and in addition a job-type specific set-up time. However, this set-up time is not needed if the previously processed job was of the same type. The objective is to minimize the maximum flow time. They present an $O(1)$-competitive algorithm for this problem.

Minimum makespan scheduling has been extensively studied. See the survey by Pruhs, Sgall, and Torng [29] for an overview.

For m identical machines, Graham [20] shows that the greedy algorithm, which schedules each arriving job on a machine with minimum load, is $(2-1/m)$-competitive. This is optimal for $m \leq 3$ [14]. However, better bounds are known for larger m. For $m = 4$, the best known lower and upper bounds on the competitive ratio are 1.7321 [31] and 1.7333 [10], respectively. For large m, the best known lower bound on the competitive ratio was improved from 1.837 [7] over 1.852 [2] and 1.854 [19] to 1.880 [30]. The first upper bound on the competitive ratio below 2 was 1.986 [6]. This upper bound was improved to 1.945 [24], then to 1.923 [2], and finally to 1.9201 [16].

For uniformly related machines, Aspnes et al. [5] present the first algorithm that achieves a constant competitive ratio. Due to Berman, Charikar and Karpinski [9], the best known lower and upper bounds on the competitive ratio are 2.438 and 5.828, respectively.

In the semi-online variant of the problem, the jobs arrive in decreasing order of their processing time. To the best of our knowledge, only the greedy LPT algorithm, which assigns each job to a machine with minimum load, was considered in this setting. For m identical machines, Graham [21] shows that the LPT algorithm achieves a competitive ratio of $4/3 - 1/(3m)$. For related machines, the LPT algorithm achieves a competitive ratio of 1.66 and a lower bound of 1.52 on its competitive ratio is known [17]. A detailed and

tight analysis for two related machines is given by Mireault, Orlin, and Vohra [28] and Epstein and Favrholdt [13].

2. The algorithm for uniformly related machines

We start with the algorithm for uniformly related machines, since this simple algorithm overviews well the basic structure of all our algorithms. They consist of two different phases. Initially, the first $k-1$ jobs are stored in the reordering buffer where k denotes the buffer size. Then, the algorithms start with the iteration phase. As long as new jobs arrive, this phase is iterated. After all jobs have arrived, the algorithms schedule the remaining jobs in the final phase.

A generic version of the final phase is to schedule the $k-1$ jobs remaining in the buffer optimally on the machines. However, since the minimum makespan scheduling problem is NP-hard, it is not known how to perform this generic final phase efficiently. Although efficiency is usually not considered for online algorithms, we provide, for all our algorithms for identical machines, very simple and efficient alternatives to the generic approach without deteriorating the competitive ratio. For uniformly related machines, we replace the generic final phase by the PTAS due to Hochbaum and Shmoys [23]. This deteriorates the competitive ratio from 2 to $2+\varepsilon$ for any $\varepsilon > 0$.

The algorithm for scheduling a sequence of jobs on m uniformly related machines $M_0, \ldots, M_{m-1}$ uses a reordering buffer of size m. For each $0 \leq i \leq m-1$, let α_i denote the speed of machine M_i, i.e., if load T is assigned to machine M_i then the completion time of machine M_i is T/α_i. Suppose that $\alpha_0 \leq \ldots \leq \alpha_{m-1}$. The objective is to minimize the makespan, i.e., the maximum completion time. The iteration and final phase are defined as follows.

- *Iteration phase*: When a new job arrives, store this new job in the reordering buffer, and remove a job J of smallest size from the buffer. Let M_i be a machine with load at most

$$\frac{\alpha_i}{\sum_{j=0}^{m-1} \alpha_j} \cdot (T + m \cdot p(J)) - p(J) \ ,$$

  where T denotes the total scheduled load. (Obviously, there always exists such a machine.) Then, schedule job J on machine M_i, i.e., the total scheduled load T is increased by $p(J)$.

- *Final phase*: The $m-1$ remaining jobs in the reordering buffer are virtually scheduled with the PTAS from Hochbaum and Shmoys [23] on m empty machines $M'_0, \ldots, M'_{m-1}$, where, for each $0 \leq i \leq m-1$, machine M'_i has speed α_i. With this scheme an $(1+\varepsilon)$-approximation is achieved. Then, for each $0 \leq i \leq m-1$, schedule the jobs from M'_i on the real machine M_i.

Theorem 1. *For m uniformly related machines, our algorithm achieves the competitive ratio $2+\varepsilon$ with a reordering buffer of size m.*

Proof. Fix an input sequence of jobs. Let OPT denote the minimum makespan achieved by an optimal offline algorithm.

At the end of the iteration phase, for each $0 \leq i \leq m-1$, the completion time of machine M_i is at most

$$\frac{1}{\sum_{j=0}^{m-1} \alpha_j} \cdot (T + (m-1) \cdot p(J_i)) \ ,$$

where T denotes the total scheduled load at the end of the iteration phase and J_i denotes the last job scheduled on machine M_i in the iteration phase. Obviously, for each $0 \leq i \leq m-1$,

$$\frac{1}{\sum_{j=0}^{m-1} \alpha_j} \cdot (T + (m-1) \cdot p(J_i)) \leq \mathrm{OPT} \ ,$$

since $m-1$ jobs are stored in the reordering buffer at the end of the iteration phase and the size of each of these jobs is at least $p(J_i)$.

In the final phase, for each $0 \leq i \leq m-1$, the completion time of the machine M'_i in the virtual schedule is at most $(1+\varepsilon) \cdot \mathrm{OPT}$, due to the polynomial time approximation scheme. As a consequence, the makespan of our algorithm is at most $(2+\varepsilon) \cdot \mathrm{OPT}$. □

3. Lower bounds

In this section, we present lower bounds for m identical machines. As main result, we prove that no online algorithm can achieve a competitive ratio less than r_m with a reordering buffer whose size does not depend on the input sequence. Further, we show that this general lower bound can be improved to $3/2 > r_m$ for a reordering buffer of size at most $\lfloor m/2 \rfloor$, and to $1+1/\sqrt{2} \approx 1.7071$ for a reordering buffer of size at most $\lfloor m/8 \rfloor$ if $m \geq 8$.

Theorem 2. *For m identical machines, no online algorithm can achieve a competitive ratio less than r_m with a reordering buffer whose size does not depend on the input sequence.*

Proof. Assume for contradiction that there exists an online algorithm A that achieves a competitive ratio $r < r_m$ with a reordering buffer of size k. Consider the following input sequence. At first, $(1/\varepsilon + k)$ jobs of size ε arrive. Since only k of these jobs can be stored in the reordering buffer, $1/\varepsilon$ of them have to be scheduled on machines. Let $M_0, \ldots, M_{m-1}$ denote the m identical machines with $L(M_0) \geq \cdots \geq L(M_{m-1})$. Then, there exists a machine

M_j with load at least w_j, since otherwise, the total scheduled load would be strictly less than $\sum_{i=0}^{m-1} w_i = 1$.

We distinguish two different cases.

- If $w_j = r_m/m$, no more jobs arrive. In the optimal schedule, all jobs are evenly distributed between the machines. Hence, the optimal makespan is at most $(1+k\cdot\varepsilon)/m+\varepsilon$. As a consequence, the competitive ratio of A is at least

$$\frac{r_m/m}{(1+(k+m)\cdot\varepsilon)/m} = \frac{r_m}{1+(k+m)\cdot\varepsilon} \ ,$$

  which is strictly larger than r if ε is chosen appropriately small.

- If $w_j = (r_m-1)/j$, $(m-j)$ additional jobs of size $1/j$ arrive. It is possible, to assign each of the $(m-j)$ additional jobs to a different machine and to evenly distribute the remaining $(1/\varepsilon+k)$ jobs between the remaining j machines. Hence, the optimal makespan is at most $(1+k\cdot\varepsilon)/j+\varepsilon$.

  If A schedules two jobs of size $1/j$ on the same machine, the competitive ratio of A is at least

$$\frac{2/j}{(1+(k+j)\cdot\varepsilon)/j} = \frac{2}{1+(k+j)\cdot\varepsilon} \ ,$$

  which is strictly larger than r if ε is chosen appropriately small.

  Otherwise, i.e., A schedules at least one of the jobs of size $1/j$ on a machine that already has load at least $(r_m-1)/j$, the competitive ratio of A is at least

$$\frac{r_m/j}{(1+(k+j)\cdot\varepsilon)/j} = \frac{r_m}{1+(k+j)\cdot\varepsilon} \ ,$$

  which is strictly larger than r if ε is chosen appropriately small.

This concludes the proof of the theorem. □

The above general lower bound can be improved for small reordering buffers.

Theorem 3. *For m identical machines:*

- *No online algorithm can achieve a competitive ratio less than* $3/2$ *with a reordering buffer of size at most* $\lfloor m/2\rfloor$.
- *No online algorithm can achieve a competitive ratio less than* $1+1/\sqrt{2}$ *with a reordering buffer of size at most* $\lfloor m/8\rfloor$ *if* $m \geq 8$.

Proof. The following input sequences are similar to the ones used by Faigle, Kern, and Turán [14] for lower bounds on the problem without reordering.

- Consider an online algorithm A with a reordering buffer of size at most $\lfloor m/2\rfloor$. The input sequence consists of at most two consecutive phases.
  - In the first phase, m jobs of size 1 arrive.
  - In the second phase, $\lfloor m/2\rfloor$ jobs of size 2 arrive if A schedules at most one job on each machine in the first phase.

  If the input sequence consists only of the first phase, the competitive ratio of A is at least 2. Otherwise, at the end of the first phase, the load on at least $m-(\lfloor m/2\rfloor-1)$ machines is 1, and hence, the competitive ratio of A is at least $3/2$.

- Consider an online algorithm A with a reordering buffer of size at most $\lfloor m/8\rfloor$. Assume that $m \geq 8$. The input sequence consists of at most three consecutive phases.
  - In the first phase, m jobs of size 1 arrive.
  - In the second phase, m jobs of size $1+\sqrt{2}$ arrive if A schedules at most one job on each machine in the first phase.
  - In the third phase, $\lfloor m/4\rfloor$ jobs of size $2+2\sqrt{2}$ arrive if the second phase exists and if A schedules at most two jobs of size $1+\sqrt{2}$ and no job of size 1 or at most one job of size $1+\sqrt{2}$ and further jobs of size 1 on each machine in the first two phases.

  If the input sequence consists only of the first phase, the competitive ratio of A is at least 2. If the input sequence consists only of the first two phases, the competitive ratio of A is at least $\frac{1+2(1+\sqrt{2})}{1+(1+\sqrt{2})} = 1+1/\sqrt{2}$.

  Otherwise, at the end of the second phase, the load on at least $m-2(\lfloor m/8\rfloor-1) \geq m-\lfloor m/4\rfloor+2$ machines is at least $1+(1+\sqrt{2})$, since the load on at least $m-(\lfloor m/8\rfloor-1)$ machines is 1 at the end of the first phase and at least $m-(\lfloor m/8\rfloor-1)$ jobs of size $1+\sqrt{2}$ are scheduled in the second phase. Hence, the competitive ratio of A is at least $\frac{1+(1+\sqrt{2})+(2+2\sqrt{2})}{2+2\sqrt{2}} = 1+1/\sqrt{2}$ in this case.

This concludes the proof of the theorem. □

4. Algorithms for identical machines

In this section, we present scheduling algorithms for m identical machines $M_0,\ldots,M_{m-1}$. As main result, we introduce a fairly simple algorithm that achieves the competitive ratio r_m. First, we prove this matching upper bound for a reordering buffer of size $3m$. Then, with a refined analysis, we improve the buffer size to $\lceil(1+2/r_m)\cdot m\rceil+2$.

Further, for buffer size $k \in [1,(m+1)/2]$, we present a $(2-1/(m-k+1))$-competitive algorithm, which nicely generalizes the well-known result of Graham [20]. Finally, we give a 3/2-competitive algorithm using a buffer of size $\lceil(2/3+2/(1+\ln 3))\cdot m\rceil+1$ and a $(1+r_m/2)$-competitive algorithm using a buffer of size $m+1$.

4.1. The optimal algorithm

The following algorithm uses a reordering buffer of size $k \geq m$ which is a parameter of the algorithm. The iteration and final phase are defined as follows.

- *Iteration phase*: When a new job arrives, store this new job in the reordering buffer, and remove a job J of smallest size from the buffer. Let M_i be a machine with load at most

  $$w_i \cdot (T + m \cdot p(J)) - p(J) \ ,$$

  where T denotes the total scheduled load. (Due to Observation 4, there always exists such a machine.) Then, schedule job J on machine M_i, i.e., the total scheduled load T is increased by $p(J)$.

- *Final phase*: This phase consists of two steps.

  In the first step, some of the $k-1$ remaining jobs in the reordering buffer are virtually scheduled on m empty machines $M'_0,\ldots,M'_{m-1}$: The jobs are considered in descending order of their size and assigned to a machine with minimum load. Note that this is just the LPT algorithm. However, we abort the process if the makespan is at least three times the size of the smallest job assigned so far. When the process is aborted, the last assigned job, which is also the smallest assigned job, is removed from the virtual schedule. Note that, in particular, at most two jobs are assigned to each machine in the virtual schedule. Assume that $L(M'_0) \leq \cdots \leq L(M'_{m-1})$. Then, for each $0 \leq i \leq m-1$, schedule the jobs from M'_i on the real machine M_i.

  In the second step, schedule the remaining jobs according to the greedy algorithm, which allocates each job on a machine with minimum load.

Observation 4. *There always exists a machine M_i with load at most $w_i \cdot (T + m \cdot p(J)) - p(J)$.*

Proof. Assume for contradiction that, for each $0 \leq i \leq m-1$, machine M_i has load strictly greater than $w_i \cdot (T + m \cdot p(J)) - p(J)$. This yields the following contradiction

$$\begin{aligned} T &> \sum_{i=0}^{m-1} (w_i \cdot (T + m \cdot p(J)) - p(J)) \\ &= (T + m \cdot p(J)) - m \cdot p(J) \\ &= T \ , \end{aligned}$$

since by definition $\sum_{i=0}^{m-1} w_i = 1$. □

The following theorem shows that the above online algorithm is optimal.

Theorem 5. *For m identical machines, our online algorithm achieves the optimal competitive ratio r_m with a reordering buffer of size $k = 3m$.*

Proof. Fix an input sequence of jobs. Let OPT denote the minimum makespan achieved by an optimal offline algorithm. We show that the makespan of our algorithm is at most $r_m \cdot \mathrm{OPT}$.

At the end of the iteration phase, for each $0 \leq i \leq m-1$, the load of machine M_i is at most

$$w_i \cdot (T + (m-1) \cdot p(J_i)) \ ,$$

where T denotes the total scheduled load at the end of the iteration phase and J_i denotes the last job scheduled on machine M_i. Let $p_{\min}$ denote the smallest size of all remaining jobs in the reordering buffer at the end of the iteration phase. Obviously, $p(J_i) \leq p_{\min}$ and hence, for each $0 \leq i \leq m-1$,

$$w_i \cdot (T + (m-1) \cdot p(J_i)) \leq w_i \cdot (T + (m-1) \cdot p_{\min}) \ .$$

In the final phase, the algorithm schedules the remaining $3m-1$ jobs in the reordering buffer. This phase consists of two steps. First, we analyze, for a fixed $0 \leq i \leq m-1$, the load on machine M_i at the end of the first step. In this step, some of the remaining jobs in the buffer are virtually scheduled on m empty machines. Let $M'_0,\ldots,M'_{m-1}$ denote the machines in the final virtual schedule with $L(M'_0) \leq \cdots \leq L(M'_{m-1})$.

The virtual schedule is optimal. This is due to the fact that at most two jobs are assigned to each machine in the virtual schedule. Scheduling three jobs on the same machine cannot improve the makespan, since, by definition of our algorithm, the combined size of the three smallest jobs in the virtual schedule is larger than the makespan. It is a well-known fact that the LPT algorithm produces an optimal schedule if at most two jobs are assigned to each machine in an optimal schedule. Hence, for each $0 \leq j \leq m-1$, $L(M'_j) \leq \mathrm{OPT}$.

At the end of the first step, for each $0 \leq j \leq m-1$, the jobs from M'_j are scheduled on the real machine M_j. Thus, the load of machine M_i is at most

$$w_i \cdot (T + (m-1) \cdot p_{\min}) + L(M'_i) \ .$$

It remains to show that

$$w_i \cdot (T + (m-1) \cdot p_{\min}) + L(M'_i) \leq r_m \cdot \mathrm{OPT} \ .$$

Clearly,

$$\frac{T + (m-1) \cdot p_{\min} + \sum_{j=0}^{m-1} L(M'_j)}{m} \leq \mathrm{OPT} \ ,$$

since at least $m-1$ jobs remain in the buffer at the end of the first step and the size of each of these jobs is at least $p_{\min}$.

Thus, for each $0 \leq \ell \leq m-1$,

$$T + (m-1)\cdot p_{\min} \leq m\cdot \mathrm{OPT} - \sum_{j=0}^{m-1} L(M'_j) \leq m\cdot \mathrm{OPT} - (m-\ell)\cdot L(M'_\ell) \ . \quad (1)$$

We distinguish two cases.

- If $w_i = (r_m-1)/i$, then $(r_m-1)\cdot m/r_m \leq i \leq m-1$ and it follows

$$\begin{aligned} & w_i\cdot(T+(m-1)\cdot p_{\min}) + L(M'_i) \\ & \leq \frac{r_m-1}{i}\cdot(m\cdot\mathrm{OPT} - (m-i)\cdot L(M'_i)) + L(M'_i) \\ & = \frac{(r_m-1)\cdot m}{i}\cdot(\mathrm{OPT} - L(M'_i)) + r_m\cdot L(M'_i) \\ & \leq r_m\cdot(\mathrm{OPT} - L(M'_i)) + r_m\cdot L(M'_i) \\ & = r_m\cdot\mathrm{OPT} \ , \end{aligned}$$

since $L(M'_i) \leq \mathrm{OPT}$.

- If $w_i = r_m/m$, then $0 \leq i \leq (r_m-1)\cdot m/r_m$ and it follows

$$\begin{aligned} & w_i\cdot(T+(m-1)\cdot p_{\min}) + L(M'_i) \\ & \leq \frac{r_m}{m}\cdot(T+(m-1)\cdot p_{\min}) + L(M'_{\lfloor (r_m-1)\cdot m/r_m\rfloor}) \\ & \leq \frac{r_m}{m}\cdot(m\cdot\mathrm{OPT} \\ & \quad - (m-((r_m-1)\cdot m/r_m))\cdot L(M'_{\lfloor (r_m-1)\cdot m/r_m\rfloor})) \\ & \quad + L(M'_{\lfloor (r_m-1)\cdot m/r_m\rfloor}) \\ & = r_m\cdot\mathrm{OPT} \ . \end{aligned}$$

In both cases, the makespan is at most $r_m\cdot\mathrm{OPT}$ at the end of the first step.

Finally, we analyze the makespan at the end of the second step. Let $p_{\max}$ denote the largest size of all jobs remaining in the reordering buffer at the end of the first step. Then, the virtual scheduling process in the first step aborts when a job of size $p_{\max}$ is assigned to a machine. Recall that this job of size $p_{\max}$ is removed from the virtual schedule. Consider an optimal schedule of all jobs allocated in step one and one additional job of size $p_{\max}$ on m empty machines. Since the makespan of this schedule is at least three times the size of the smallest assigned job and since all jobs in this schedule have a size of at least $p_{\max}$, we conclude that $p_{\max} \leq \mathrm{OPT}/3$.

In the second step, the remaining jobs in the reordering buffer are scheduled according to the greedy algorithm. Since the average load is always bounded by OPT, there always exists a machine with load at most OPT. After scheduling a job J according to the greedy algorithm, the makespan is at most $r_m\cdot\mathrm{OPT}$, since $p(J) \leq p_{\max} \leq \mathrm{OPT}/3 \leq (r_m-1)\cdot\mathrm{OPT}$. This concludes the proof of the theorem. □

In the following theorem we give a refined analysis of the optimal algorithm showing that the size of the buffer can be further reduced.

Theorem 6. *For m identical machines, our online algorithm achieves the optimal competitive ratio r_m with a reordering buffer of size $k = \lceil (1+2/r_m)\cdot m\rceil + 2$.*

Proof. To improve upon Theorem 5, we observe that the proof even goes through if Equation (1) only holds for $\lfloor (r_m-1)\cdot m/r_m\rfloor \leq \ell \leq m-1$. In the following, we argue that Equation (1) indeed holds for these ℓ if we only have a reordering buffer of size $k = \lceil (1+2/r_m)\cdot m\rceil + 2$.

In the beginning of the first step, $\lceil (1+2/r_m)\cdot m\rceil + 1$ jobs are stored in the reordering buffer. Let n' denote the number of jobs scheduled in the final virtual schedule. The number of jobs that are stored in the reordering buffer and that are not scheduled on the virtual machines $M'_{\lfloor (r_m-1)\cdot m/r_m\rfloor}, \ldots, M'_{m-1}$ is at least

$$\frac{r_m+2}{r_m}\cdot m + 1 - n' + \max\left\{0, n' - 2\left(\frac{m}{r_m}+1\right)\right\} \geq m-1 \ ,$$

since $m - \lfloor (r_m-1)\cdot m/r_m\rfloor \leq m/r_m + 1$. As a consequence, for each $\lfloor (r_m-1)\cdot m/r_m\rfloor \leq \ell \leq m-1$,

$$T + (m-1)\cdot p_{\min} \leq m\cdot\mathrm{OPT} - \sum_{j=\lfloor (r_m-1)\cdot m/r_m\rfloor}^{m-1} L(M'_j) \leq m\cdot\mathrm{OPT} - (m-\ell)\cdot L(M'_\ell) \ .$$

Hence, the proof of Theorem 5 goes through if we only have a reordering buffer of size $k = \lceil (1+2/r_m)\cdot m\rceil + 2$. □

4.2. The extended Graham algorithm

The following algorithm uses a reordering buffer of size $k \in [1, (m+1)/2]$. The iteration and final phase are defined as follows.

- *Iteration phase*: When a new job arrives, store this new job in the reordering buffer, and remove a job J of smallest size from the buffer. Schedule J on a machine with minimum load among the machines $M_0, \ldots, M_{m-k}$.
- *Final phase*: If $k > 1$, schedule each of the $k-1$ remaining jobs exclusively on one of the machines $M_{m-k+1}, \ldots, M_{m-1}$.

Theorem 7. *For m identical machines, our online algorithm achieves the competitive ratio $2 - 1/(m-k+1)$ with a reordering buffer of size $k \in [1, (m+1)/2]$,*

Proof. Fix an input sequence of jobs. Let OPT denote the minimum makespan achieved by an optimal offline algorithm. We show that the makespan $L_{\max}$ of our algorithm is at most $(2 - 1/(m-k+1))\cdot\mathrm{OPT}$.

Suppose that the load of a machine M_i with $m-k+1 \leq i \leq m-1$ is $L_{\max}$. Then, $L_{\max} \leq \mathrm{OPT}$, since no job is scheduled on M_i in the iteration phase and exactly one job is scheduled on M_i in the final phase.

Suppose that the load of a machine M_i with $0 \leq i \leq m-k$ is $L_{\max}$. On such a machine M_i, jobs are only scheduled in the iteration phase. Let J denote the last job scheduled on M_i. Then, the load of each machine M_j with $0 \leq j \leq m-k$ is at least $L_{\max} - p(J)$ just before job J should be scheduled. Hence, $(m-k+1) \cdot (L_{\max} - p(J)) + p(J) \leq m \cdot \mathrm{OPT} - (k-1) \cdot p(J)$, because the size of each of the $k-1$ jobs in the reordering buffer is at least $p(J)$ when job J is scheduled. Finally,

$$\begin{aligned} L_{\max} &\leq \frac{m \cdot \mathrm{OPT} - k \cdot p(J)}{m-k+1} + p(J) \\ &= \frac{m \cdot \mathrm{OPT} + (m-2k+1) \cdot p(J)}{m-k+1} \\ &\leq \frac{(2m-2k+1) \cdot \mathrm{OPT}}{m-k+1} \\ &= \left(2 - \frac{1}{m-k+1}\right) \cdot \mathrm{OPT} \ , \end{aligned}$$

since by definition $0 \leq m-2k+1$. □

4.3. The 3/2-competitive algorithm

The following algorithm uses a reordering buffer of size $k := \lceil (2/3 + 2/(1+\ln 3)) \cdot m \rceil + 1$. The iteration and final phase are defined as follows.

- *Iteration phase*: When a new job arrives, store this new job in the reordering buffer, and remove a job J of smallest size from the buffer. Let M_i be a machine with load at most

$$w_i' \cdot \left(T + \frac{2m}{1+\ln 3} \cdot p(J)\right) - p(J) \ ,$$

  where $w_i' := \min\{3/(2m), 1/(2i)\}$ and T denotes the total scheduled load. (Similar to Observation 4, it can be shown that such a machine always exists). Then, schedule job J on machine M_i, i.e., the total scheduled load T is increased by $p(J)$.

- *Final phase*: Consider the $k-1$ remaining jobs in the reordering buffer in descending order of their size and schedule them according to the greedy algorithm, which allocates each job on a machine with minimum load.

Theorem 8. *For m identical machines, our online algorithm achieves the competitive ratio* $3/2$ *with a reordering buffer of size* $\lceil (2/3 + 2/(1+\ln 3)) \cdot m \rceil + 1$.

Proof. Fix an input sequence of jobs. Let OPT denote the minimum makespan achieved by an optimal offline algorithm. We show that the makespan of our algorithm is at most $3/2 \cdot \mathrm{OPT}$.

Let T denote the total scheduled load at the end of the iteration phase, and let $p_{\min}$ denote the smallest size of all remaining jobs in the reordering buffer at the end of the iteration phase. As in Theorem 5, we can conclude that at the end of the iteration phase, for each $0 \leq i \leq m-1$, the load of machine M_i is at most

$$w_i' \cdot \left(T + \left(\frac{2m}{1+\ln 3} - 1\right) \cdot p_{\min}\right) \leq 3/2 \cdot \mathrm{OPT} \ ,$$

since $w_i' \leq 3/(2m)$, $k-1 = \lceil (2/3 + 2/(1+\ln 3)) \cdot m \rceil$ jobs remain in the reordering buffer, and the size of each of these jobs is at least $p_{\min}$.

In the final phase, the remaining jobs in the reordering buffer are scheduled according to the greedy algorithm. Let $J_1, \ldots, J_{k-1}$ denote the remaining jobs in the buffer with $p(J_1) \geq \cdots \geq p(J_{k-1})$.

First, we consider the jobs $J_1, \ldots, J_m$. A fixed job J_i with $1 \leq i \leq m$ is scheduled on a machine with load at most $w_{m-i}' \cdot (T + (2m/(1+\ln 3) - 1) \cdot p_{\min})$, since $w_0' \leq \cdots \leq w_{m-1}'$. Hence, it remains to show that

$$w_{m-i}' \cdot \left(T + \left(\frac{2m}{1+\ln 3} - 1\right) \cdot p_{\min}\right) + p(J_i) \leq 3/2 \cdot \mathrm{OPT} \ .$$

Clearly,

$$\frac{T + (\frac{2m}{1+\ln 3} - 1) \cdot p_{\min} + \sum_{j=1}^{\lceil 2m/3 \rceil} p(J_j)}{m} \leq \mathrm{OPT} \ ,$$

since at least $\lceil 2m/(1+\ln 3) \rceil - 1$ jobs remain in the buffer after the scheduling of the jobs $J_1, \ldots, J_{\lceil 2m/3 \rceil}$ and the size of each of these jobs is at least $p_{\min}$. Thus, for each $1 \leq \ell \leq \lceil 2m/3 \rceil$,

$$\begin{aligned} T + \left(\frac{2m}{1+\ln 3} - 1\right) \cdot p_{\min} &\leq m \cdot \mathrm{OPT} - \sum_{j=1}^{\lceil 2m/3 \rceil} p(J_j) \\ &\leq m \cdot \mathrm{OPT} - \ell \cdot p(J_\ell) \ . \end{aligned}$$

We distinguish two cases.

- If $w_{m-i}' = 1/(2(m-i))$, then $1 \leq i \leq 2m/3$ and it follows

$$\begin{aligned} & w_{m-i}' \cdot \left(T + \left(\frac{2m}{1+\ln 3} - 1\right) \cdot p_{\min}\right) + p(J_i) \\ &\leq \frac{1}{2(m-i)} \cdot (m \cdot \mathrm{OPT} - i \cdot p(J_i)) + p(J_i) \\ &= \frac{m \cdot \mathrm{OPT} + (2m-3i) \cdot p(J_i)}{2(m-i)} \\ &\leq \frac{m \cdot \mathrm{OPT} + (2m-3i) \cdot \mathrm{OPT}}{2(m-i)} \\ &\leq 3/2 \cdot \mathrm{OPT} \ . \end{aligned}$$

- If $w'_{m-i} = 3/(2m)$, then $2m/3 \leq i \leq m$ and it follows

$$\begin{aligned} & w'_{m-i} \cdot \left(T + \left(\frac{2m}{1+\ln 3} - 1 \right) \cdot p_{\min} \right) + p(J_i) \\ & \leq \frac{3}{2m} \cdot \left(m \cdot \mathrm{OPT} - \left\lceil \frac{2m}{3} \right\rceil \cdot p(J_{\lceil 2m/3 \rceil}) \right) + p(J_{\lceil 2m/3 \rceil}) \\ & \leq \frac{3}{2} \cdot \mathrm{OPT} \ . \end{aligned}$$

In both cases, the makespan is at most $3/2 \cdot \mathrm{OPT}$ after the scheduling of the jobs $J_1, \ldots, J_m$.

Finally, we consider the jobs $J_{m+1}, \ldots, J_{k-1}$. For a fixed job J_i with $m+1 \leq i \leq k-1$, $p(J_i) \leq \mathrm{OPT}/2$, because $p(J_1) \geq \cdots \geq p(J_{k-1})$. Since the average load is always bounded by OPT, there always exists a machine with load at most OPT. After scheduling job J_i according to the greedy algorithm, the makespan is at most $3/2 \cdot \mathrm{OPT}$, since $p(J_i) \leq \mathrm{OPT}/2$. □

4.4. The $(1 + r_m/2)$-competitive algorithm

The following algorithm uses a reordering buffer of size $m+1$. The iteration and final phase are defined as follows.

- *Iteration phase*: When a new job arrives, store this new job in the reordering buffer, and remove a job J of smallest size from the buffer. Let M_i be a machine with load at most $w_i \cdot T$, where T denotes the total scheduled load. (Since $\sum_{i=0}^{m-1} w_i = 1$, there always exists such a machine.) Then, schedule job J on machine M_i, i.e., the total scheduled load T is increased by $p(J)$.
- *Final phase*: Consider the m remaining jobs in the reordering buffer in descending order of their size and schedule them according to the greedy algorithm, which allocates each job on a machine with minimum load.

Theorem 9. *For m identical machines, our online algorithm achieves the competitive ratio $1 + r_m/2$ with a reordering buffer of size $m+1$.*

Proof. Fix an input sequence of jobs. Let OPT denote the minimum makespan achieved by an optimal offline algorithm. We show that the makespan of our algorithm is at most $(1 + r_m/2) \cdot \mathrm{OPT}$.

Let T denote the total scheduled load at the end of the iteration phase, and let $p_{\max}$ denote the largest size of all jobs scheduled in the iteration phase. We can conclude that at the end of the iteration phase, for each $0 \leq i \leq m-1$, the load of machine M_i is at most

$$w_i \cdot T + p_{\max} \leq \frac{r_m}{m} \cdot (m \cdot \mathrm{OPT} - m \cdot p_{\max}) + p_{\max} \leq r_m \cdot \mathrm{OPT} \ ,$$

since m jobs remain in the reordering buffer and the size of each of these jobs is at least $p_{\max}$.

In the final phase, the remaining jobs in the reordering buffer are scheduled according to the greedy algorithm. Let $J_1, \ldots, J_m$ denote the remaining jobs in the buffer with $p(J_1) \geq \cdots \geq p(J_m)$. A fixed job J_i with $1 \leq i \leq m$ is scheduled on a machine with load at most $w_{m-i} \cdot T + p_{\max}$, since $w_0 \leq \cdots \leq w_{m-1}$.

Hence, it remains to show that

$$w_{m-i} \cdot T + p_{\max} + p(J_i) \leq (1 + r_m/2) \cdot \mathrm{OPT} \ .$$

Obviously,

$$\begin{aligned} T &= m \cdot \mathrm{OPT} - \sum_{j=1}^{m} p(J_j) \\ &\leq m \cdot \mathrm{OPT} - (m-i) \cdot p_{\max} - i \cdot p(J_i) \\ &= m \cdot (\mathrm{OPT} - p(J_i)) + (m-i) \cdot (p(J_i) - p_{\max}) \ . \end{aligned}$$

We distinguish two cases.

- If $w_{m-i} = (r_m - 1)/(m-i)$, then $(r_m - 1) \cdot m / r_m \leq m - i \leq m - 1$ and it follows

$$\begin{aligned} & w_{m-i} \cdot T + p_{\max} + p(J_i) \\ & = \frac{(r_m - 1) \cdot m}{m - i} \cdot (\mathrm{OPT} - p(J_i)) \\ & \qquad + (r_m - 1) \cdot (p(J_i) - p_{\max}) + p_{\max} + p(J_i) \\ & \leq r_m \cdot (\mathrm{OPT} - p(J_i)) + r_m \cdot p(J_i) + (2 - r_m) \cdot p_{\max} \\ & \leq (1 + r_m/2) \cdot \mathrm{OPT} \ , \end{aligned}$$

  since $p(J_i) \leq \mathrm{OPT}$ and $p_{\max} \leq \mathrm{OPT}/2$.

- If $w_{m-i} = r_m/m$, then $0 \leq m - i \leq (r_m - 1) \cdot m / r_m$ and it follows

$$\begin{aligned} & w_{m-i} \cdot T + p_{\max} + p(J_i) \\ & = r_m \cdot (\mathrm{OPT} - p(J_i)) \\ & \qquad + \frac{r_m \cdot (m-i)}{m} \cdot (p(J_i) - p_{\max}) + p_{\max} + p(J_i) \\ & \leq r_m \cdot \mathrm{OPT} - (r_m - 1) \cdot p(J_i) \\ & \qquad + (r_m - 1) \cdot (p(J_i) - p_{\max}) + p_{\max} \\ & \leq (1 + r_m/2) \cdot \mathrm{OPT} \ , \end{aligned}$$

  since $p_{\max} \leq p(J_i)$ and $p_{\max} \leq \mathrm{OPT}/2$.

In both cases, the makespan is at most $(1 + r_m/2) \cdot \mathrm{OPT}$ at the end of the final phase. □

5. Acknowledgments

We thank an anonymous reviewer for many helpful comments and for pointing out an error in a previous version of this work. We also thank Heiner Ackermann for discussions on this problem.

References

[1] S. Albers. A competitive analysis of the list update problem with lookahead. *Theoretical Compuer Science*, 197(1–2):95–109, 1998.

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

[3] S. Albers. New results on web caching with request reordering. In *Proceedings of the 16th ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 84–92, 2004.

[4] H. Alborzi, E. Torng, P. Uthaisombut, and S. Wagner. The k-client problem. *Journal of Algorithms*, 41(2):115–173, 2001.

[5] J. Aspnes, Y. Azar, A. Fiat, S. A. Plotkin, and O. Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM*, 44(3):486–504, 1997.

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

[7] Y. Bartal, H. J. Karloff, and Y. Rabani. A better lower bound for on-line scheduling. *Inf. Process. Lett.*, 50(3):113–116, 1994.

[8] S. Ben-David and A. Borodin. A new measure for the study of on-line algorithms. *Algorithmica*, 11(1):73–91, 1994.

[9] P. Berman, M. Charikar, and M. Karpinski. On-line load balancing for related machines. *Journal of Algorithms*, 35(1):108–121, 2000.

[10] B. Chen, A. van Vliet, and G. J. Woeginger. New lower and upper bounds for on-line scheduling. *Operations Research Letters*, 16(4):221–230, 1994.

[11] S. Divakaran and M. Saks. An online scheduling problem with job set-ups. Technical report, DIMACS, 2000.

[12] M. Englert, H. Räcke, and M. Westermann. Reordering buffers for general metric spaces. In *Proceedings of the 39th ACM Symposium on Theory of Computing (STOC)*, pages 556–564, 2007.

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

[14] U. Faigle, W. Kern, and G. Turán. On the performance of on-line algorithms for partition problems. *Acta Cybernetica*, 9(2):107–119, 1989.

[15] T. Feder, R. Motwani, R. Panigrahy, S. S. Seiden, R. van Stee, and A. Zhu. Combining request scheduling with web caching. *Theoretical Compuer Science*, 324(2–3):201–218, 2004.

[16] R. Fleischer and M. Wahl. On-line scheduling revisited. *Journal of Scheduling*, 3(6):343–353, 2000.

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

[18] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[19] T. Gormley, N. Reingold, E. Torng, and J. Westbrook. Generating adversaries for request-answer games. In *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 564–565, 2000.

[20] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(1):1563–1581, 1966.

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

[22] E. F. Grove. Online bin packing with lookahead. In *Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 430–436, 1995.

[23] D. S. Hochbaum and D. 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.

[24] D. R. Karger, S. J. Phillips, and E. Torng. A better algorithm for an ancient scheduling problem. *Journal of Algorithms*, 20(2):400–430, 1996.

[25] H. Kellerer, V. Kotov, M. G. Speranza, and Z. Tuza. Semi on-line algorithms for the partition problem. *Operations Research Letters*, 21(5):235–242, 1997.

[26] E. Koutsoupias and C. H. Papadimitriou. Beyond competitive analysis. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 394–400, 1994.

[27] J. Krokowski, H. Räcke, C. Sohler, and M. Westermann. Reducing state changes with a pipeline buffer. In *Proceedings of the 9th International Fall Workshop Vision, Modeling, and Visualization (VMV)*, pages 217–224, 2004.

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

[29] K. Pruhs, J. Sgall, and E. Torng. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, chapter Online Scheduling. CRC Press, 2004.

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

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

[32] E. Torng. A unified analysis of paging and caching. *Algorithmica*, 20(2):175–200, 1998.