Minimizing the Weighted Makespan with Restarts on a Single Machine

Aflatoun Amouzandeh[1][0009−0009−6912−3344], Klaus Jansen[2][0000−0001−8358−6796], Lis Pirotton[2], Rob van Stee[1][0000−0002−3664−0865], and Corinna Wambsganz[2]

[1] University of Siegen, Department of Mathematics, Germany
{aflatoun.amouzandeh,rob.vanstee}@uni-siegen.de
[2] Kiel University, Department of Computer Science, Germany
{kj,lpi,cwa}@informatik.uni-kiel.de

Abstract. We consider the problem of minimizing the weighted makespan on a single machine with restarts. Restarts are similar to preemptions but weaker: a job can be interrupted, but then it has to be run again from the start instead of resuming at the point of interruption later. The objective is to minimize the weighted makespan, defined as the maximum weighted completion time of jobs.
We establish a lower bound of 1.4656 on the competitive ratio achievable by deterministic online algorithms. For the case where all jobs have identical processing times, we design and analyze a deterministic online algorithm that improves the competitive ratio to better than 1.3098. Finally, we prove a lower bound of 1.2344 for this case.

1 Introduction

Makespan minimization is a fundamental and extensively studied problem in scheduling theory, with a significant body of research devoted to it. A sequence of jobs must be scheduled non-preemptively on m machines. Each job j is characterized by a processing time p_j and a non-negative arrival time r_j, where $1 \leq j \leq n$. In this paper, we assume that jobs arrive over time, i.e., each job becomes available for processing only at its release time. The goal is to minimize the *makespan*, defined as the maximum completion time of any job in the schedule. The problem is obviously NP-hard as it generalizes the makespan problem where all jobs arrive at time 0. Regarding the online setting, the best competitive ratio currently known, for general m, is equal to 1.5 and attained by the LPT (Longest Processing Time First) algorithm [5].

A more general objective is the *weighted makespan*, denoted as $WC_{\max} = \max_j\{w_jC_j\}$, which measures the maximum *weighted completion time* of all jobs. Each job j is assigned a positive weight w_j, reflecting its relative importance. The classical makespan minimization problem is a special case in which all weights are equal. This objective function was first considered only fairly recently and has received some attention [16, 12, 4, 19]. Even on a single machine, it is nontrivial to optimize the weighted makespan non-preemptively if jobs arrive over time. In

the standard three-field notation introduced by Graham et al. [9], this problem is denoted as $1 \mid r_j \mid WC_{\max}$. If all jobs arrive at time 0 or preemptions are allowed, then the algorithm Largest Weight (LW) which always runs the job that has largest weight is 1-competitive.

In the *offline* setting, all job parameters (processing times, release times, and weights) are known in advance, allowing the algorithm to make better decisions. In contrast, the *online setting* assumes that jobs arrive over time and decisions must be made without knowledge of the future input. Once a job arrives, the online algorithm learns its parameters and must decide whether to process it immediately or delay its execution. The effectiveness of an online algorithm is measured by its *competitive ratio* [3, 7, 10]. For a given instance I, the objective values of the online and optimal offline algorithms are denoted as $\textsc{alg}(I)$ and $\textsc{opt}(I)$, respectively. The competitive ratio $\mathcal{R}$ of an online algorithm is defined as:

$$\mathcal{R} = \sup_I \frac{\textsc{alg}(I)}{\textsc{opt}(I)}.$$

This ensures that, in the worst case, the online algorithm's cost will never exceed $\mathcal{R}$ times the optimal cost. An online algorithm may use as much computation as needed to make decisions, but it lacks knowledge of the input.

Feng and Yuan [16] were the first to consider the weighted makespan in the context of single-machine scheduling. Li [12] explored the online version of this problem and established a lower bound of 2 on the competitive ratio for deterministic algorithms. Moreover, for the problem on a single machine, he provided an online algorithm with a competitive ratio of 3. For the case of identical machines where all jobs have the same processing time, Li proposed an online algorithm with the best-possible competitive ratio of $\frac{\sqrt{5}+1}{2}$.

Chai et al. [4] further improved the results on a single machine, by developing two deterministic online algorithms with a competitive ratio of 2. Lu et al. [15] considered the offline single-machine problem with job rejection but without release dates. They showed that this problem is NP-hard and proposed a fully polynomial-time approximation scheme (FPTAS). More recently, Sun [19] proposed a $(2 - \frac{1}{m})$-approximation algorithm for the weighted makespan scheduling problem on m identical machines, again in the setting without release dates $(P||WC_{\max})$. The algorithm runs List Scheduling on the jobs after sorting them by decreasing weight. Sun also introduced a randomized efficient polynomial-time approximation scheme (EPTAS) for this problem, as well as a randomized FPTAS for the case when m is fixed.

One important extension to the classical scheduling framework is the use of *restarts*, which forms a central focus of this paper. In this model, a running job can be interrupted when a more urgent job arrives (e.g., a smaller or heavier job in the case of weighted jobs). However, any work done on the interrupted job is lost, and it must be restarted from the beginning. This model, also known as *preemption with restarts*, differs from the more commonly studied preemption with resume model, where interrupted jobs can be resumed from the point of interruption. The concept of restarts was first introduced by Shmoys et al. [17]

in the context of makespan minimization and was later shown to improve the performance of online algorithms for other objectives on a single machine [1, 2, 11, 18], including for equal-length jobs [6].

Variants of the restart model, including *limited restarts*, which means that each job may be restarted at most once, and k-limited restart where each job can be restarted at most k times and k can be either a positive integer or infinity, have also been studied, particularly in the context of parallel batch machines [14, 8, 20].

Liang et al. [13] recently explored the weighted makespan problem in the restart model for online single-machine scheduling. They proved that when only a single restart is allowed across all jobs, no online algorithm can achieve a competitive ratio lower than 2, matching the lower bound for the problem without restarts. However, in the case where all jobs have unit processing times, they presented the best possible online algorithm with a competitive ratio of 1.4656.

In this paper, we consider the problem of online scheduling with restarts for minimizing the weighted makespan $WC_{\max}$. Specifically, we study the problem $1|r_j, \text{online}, \text{restart}|WC_{\max}$ and establish a new lower bound of 1.4656 on the competitive ratio for deterministic online algorithms in the general setting with restarts. Remarkably, this is the same value that Liang et al. [13] achieved for jobs of unit size and a single allowed restart. However, the problems are different.

Our main result is a deterministic online algorithm that achieves a competitive ratio better than 1.3098 for the case where all jobs have identical processing times. The algorithm balances a greedy approach, favoring heavier jobs, with the consideration that not too much time should be wasted on jobs that are eventually interrupted anyway. If a job arrives that is heavier than the currently running job, the new job is started instead unless the current job has already been running for sufficiently long. What sufficiently long is depends on the current time: we do not interrupt the running job in the case that the new job, if it is started after the current job completes, still completes within a factor of 1.3098 of its smallest possible completion time (which is achieved if the new job starts immediately when it arrives).

We also prove a lower bound of 1.2344 for this case. In determining both lower bounds, we first identified some difficult inputs and then used a computer to optimize the parameters. For the optimized values we then considered in which cases the goal competitive ratio was achieved. This resulted in a system of equalities that we solved to get exact lower bounds.

2 Lower Bound

We now consider the online problem $1|r_j, \text{online}, \text{restart}|WC_{\max}$. The lower bound presented here shows that the problem maintains its difficulty, even when unlimited restarts are allowed.

Theorem 1. *Any algorithm for the problem* $1|r_j, \text{online}, \text{restart}|WC_{\max}$ *is at least* $R_1 \approx 1.4656$ *competitive which is the real root of the equation* $R_1^3 - R_1^2 - 1 = 0$.

Proof. For contradiction, suppose that there exists an online algorithm ALG with a competitive ratio of less than R_1.

We consider the following job instance I. At time $r_1 = 0$, job 1 with $w_1 = 1$ and $p_1 = 1$ arrives. Then job 2 with $w_2 = \frac{1}{R_1 - 1} \approx 2.1478$ and $p_2 = 0$ arrives at time $r_2 = \frac{1}{R_1(R_1-1)} - 1 \approx 0.4655$.

Case 1. If ALG runs job 1 before job 2, no more jobs arrive. Then we have $\text{ALG}(I) \geq \max(1, w_2)$ and $\text{OPT}(I) \leq \max(r_2 w_2, 1 + r_2) = 1 + r_2$ and we get the following contradiction to our assumption

$$\frac{\text{ALG}(I)}{\text{OPT}(I)} \geq \frac{\max(1, w_2)}{\max(r_2 w_2, 1 + r_2)} = \frac{\frac{1}{R_1 - 1}}{\frac{1}{R_1(R_1-1)}} = R_1.$$

Case 2. If ALG runs job 2 first, then job 2 is started and also finishes no earlier than at time r_2 and job 1 is restarted no earlier than at time r_2. Now, two more jobs arrive at time $r_3 = r_4 = 1$: job 3 with $w_3 = \frac{1}{R_1 - 1} \approx 2.1478$ and $p_3 = 0$ and job 4 with $w_4 = 1$ and $p_4 = \frac{1}{R_1 - 1} - 1 \approx 1.1478$. Note that processing job 3 before job 4 does only improve (or maintain) the result, since job 3 and job 4 are released at the same time and $p_3 = 0$ holds. Therefore, we do not consider the other case.

OPT schedules first job 1 and then the jobs 2,3 and 4 in this order, thus $\text{OPT}(I) = \max(1, w_2, w_3, 1 + p_4) = 1 + p_4 = \frac{1}{R_1 - 1}$.

Case 2.1. If ALG runs job 3 and job 4 before job 1, then we have

$$\frac{\text{ALG}(I)}{\text{OPT}(I)} \geq \frac{\max(r_2 w_2, w_3, 1 + p_4, p_4 + 2)}{\max(1, w_2, w_3, 1 + p_4)} = \frac{\frac{1}{R_1 - 1} + 1}{\frac{1}{R_1 - 1}} = R_1.$$

Since job 4 has a higher processing time than job 1, while both jobs have the same weight, the job order 3,4,1 improves the weighted makespan compared to the job order 3,1,4.

Case 2.2. If ALG runs job 1 directly after job 2, then job 3 starts and finishes not before $r_2 + 1$ and job 4 finishes not before time $r_2 + 1 + p_4$. In this case we have $\text{ALG}(I) \geq \max(r_2 w_2, r_2 + 1, (r_2 + 1) w_3, r_2 + 1 + p_4) = (r_2 + 1) w_3$. We have

$$\begin{aligned}
&(r_2 + 1) w_3 \geq r_2 + 1 + p_4 \\
\Leftrightarrow\ & \frac{1}{R_1(R_1 - 1)} \cdot \frac{1}{R_1 - 1} \geq \frac{1}{R_1(R_1 - 1)} + \frac{1}{R_1 - 1} - 1 \\
\Leftrightarrow\ & 1 \geq (R_1 - 1) + (R_1^2 - R_1) - (R_1^3 + -2R_1^2 + R_1) \\
\Leftrightarrow\ & 0 \leq R_1^3 - 3R_1^2 + R_1 + 2 \\
\Leftrightarrow\ & 0 \leq (R_1^3 - R_1^2 - 1) - 2R_1^2 + R_1 + 3 \\
\Leftrightarrow\ & 0 \leq -2R_1^2 + R_1 + 3
\end{aligned}$$

using $R_1^3 - R_1^2 - 1 = 0$ and $-2R_1^2 + R_1 + 3 = (-2R_1 + 3)(R_1 + 1) \geq 0$ for $R_1 \in (-1, 1.5)$. Thus,

$$\frac{\text{ALG}(I)}{\text{OPT}(I)} \geq \frac{\max(r_2 w_2, r_2 + 1, (r_2 + 1)w_3, r_2 + 1 + p_4)}{\max(1, w_2, w_3, 1 + p_4)}$$
$$= \frac{\frac{1}{R_1(R_1-1)} \cdot \frac{1}{R_1-1}}{\frac{1}{R_1-1}} = \frac{1}{R_1(R_1-1)} = R_1$$

using $R_1^3 - R_1^2 - 1 = 0$. □

3 Unit-Sized Jobs

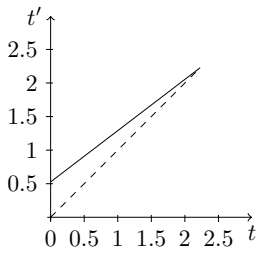
In this section, we consider the problem of online scheduling with restarts for minimizing the weighted makespan, assuming all jobs have identical processing times. We first present our deterministic online algorithm Limited Largest Weight (LLW) which has a competitive ratio of $R \approx 1.3098$, the positive real root of $3R^3 - 2R^2 - R - 2 = 0$. The other roots of this function are complex. We also provide a lower bound for this problem.

We first define two versions of the Largest Weight (LW) strategy.

- *LW with interruptions:* If the machine is idle, the next arriving job is started when it arrives. A running job is interrupted if a new job with higher weight arrives. If a job is interrupted or finishes and there are jobs waiting, the heaviest waiting job starts processing immediately.
- *LW without interruptions:* The same rules apply, except that once a job has started it is never interrupted.

Our algorithm starts the first arriving job when it arrives. Once the algorithm starts a job at time $\frac{2-R}{R-1}$ or later, it runs LW without interruptions until all jobs have finished. Whenever it starts a job at time $t < \frac{2-R}{R-1} \approx 2.2279$, it runs LW with interruptions until time

$$t' = \frac{t+2}{R} - 1. \tag{1}$$



Note that for all $t < \frac{2-R}{R-1}$ it holds that $t < t'$ since

$$t < \frac{2-R}{R-1} \Leftrightarrow tR - t < 2 - R \Leftrightarrow t < \frac{t+2}{R} - 1.$$

During this time, if an interruption occurs at time t_i, then we update $t := t_i$ and set t' according to (1). If time t' is ever reached without an interruption, the running job is allowed to continue until completion without any interruptions. After this, the algorithm resumes running LW either with or without interruptions, depending on the current time as described above.

For any job that starts at some time $t < \frac{2-R}{R-1}$, the interval (t, t') is called an *LW-phase*; however, if an interruption occurs in such an LW-phase, the LW-phase

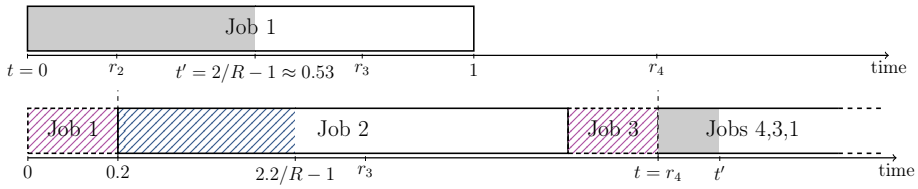


Fig. 1. This example shows the behavior of our algorithm.

ends at that time, and the next LW-phase starts immediately (because this only happens before time $\frac{2-R}{R-1}$). In Fig. 1, we sketched in an example the behavior of the algorithm for the following job sequence: $((0,1),(0.2,1.1),(0.7,1.6),(1.4,2.3))$. Here each pair specifies first a release date and then the weight of the job released at this time. The first figure shows the situation at time $t = 0$ and the first LW-phase with values t and t'. In this LW-phase job 1 gets interrupted because job 2 is heavier. The second figure shows the schedule of the algorithm at time $t = r_4$. Job 2 is not interrupted during its LW-phase, so it continues without interruption. At its completion time the algorithm starts the waiting job 3. Job 3 gets interrupted in its LW-phase and after that the remaining jobs are scheduled with LW in the order $4, 3, 1$.

Analysis

We now prove that LLW is R-competitive and this bound is tight. Recall that R is the positive real root of $3R^3 - 2R^2 - R - 2 = 0$.

Lemma 1. *The competitive ratio of the LLW algorithm is at least R.*

Proof. Suppose a job starts at $\frac{2-R}{R-1}$ and another job j of arbitrarily large weight arrives at $\frac{2-R}{R-1} + \varepsilon$ for some $\varepsilon > 0$. The algorithm does not interrupt, and the optimal schedule processes job j first. As $\varepsilon \to 0$, we find that the competitive ratio of LLW is at least

$$\frac{\frac{2-R}{R-1} + 2}{\frac{2-R}{R-1} + 1} = 1 + \frac{1}{\frac{2-R}{R-1} + 1} = 1 + \frac{R-1}{2 - R + R - 1} = R. \qquad \square$$

In the analysis, we fix an input I. For each job i, let s_i be its (most recent) starting time. Define a *critical job* k to be a job such that $\text{LLW}(I) = w_k(s_k + 1)$, i.e., the weighted completion time of a critical job is the maximum over all jobs and thus defines the performance of LLW. Let k be an arbitrary critical job and let S be the maximal set of jobs that run consecutively without interruptions and ending with the critical job k. W.l.o.g., we can assume that $|S| = k$ and that the jobs in S are numbered from 1 to k according to their order in the LLW-schedule.

We scale the weights of the jobs so that the weight of the critical job is 1. Thus

$$\textsc{llw}(I) = s_1 + k.$$

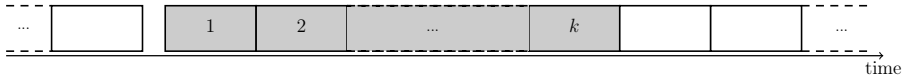


Fig. 2. This figure illustrates the set S with jobs 1 up to k. Job k is the critical job, and the jobs after k are not in S and are lighter.

If OPT runs the jobs in S in the same order as LLW and does not start them too much earlier than LLW, then the competitive ratio is maintained. We formalize this intuition in the next definition.

Definition 1. *Suppose* OPT *starts processing a job in S at time $t^* \geq (s_1 + k)/R - k$ and completes a job which is at least as heavy as the critical job at time $t^* + k$ or later. Then S is* good.

Proposition 1. *If S is good as in definition 1, the competitive ratio of* LLW *is at most R. If the critical job is the lightest job in S, and* OPT *starts running the first job in S at time $(s_1 + k)/R - k$ or later, then S is good.*

Proof. Since $|S| = k$ and the critical job (with weight 1) is by definition the last job in S, we have $\textsc{llw}(I) = s_1 + k$. In the first case, we have $\textsc{opt}(I) \geq (s_1 + k)/R$ because by the given assumptions, OPT completes a job which is at least as heavy as the critical job at time $t^* + k$ or later and $t^* \geq (s_1 + k)/R - k$.

In the second case, OPT completes at least k jobs (namely, those in the set S) that are at least as heavy as the critical job and it does not start any of these k jobs before time $(s_1 + k)/R - k$. □

Note that there may be additional jobs that are run consecutively without interruption immediately after a good set.

Theorem 2. *The LLW algorithm is R-competitive, and this bound is tight.*

Proof. We assume the existence of a *minimal counterexample*. A minimal counterexample has ratio strictly larger than R and consists of the smallest number of jobs. We show by contradiction that such a counterexample does not exist. Without loss of generality, we can assume that at most one job arrives at each time instant, as otherwise, we could postpone the lighter job to the next time instant. Consider the following two cases.

Case 1. The critical job is the lightest job in S. Let I_1 be a (hypothetical) minimal counterexample in which the critical job is the lightest job in S and achieves the highest possible competitive ratio. We establish the following properties for I_1.

Property 1. In I_1, no job completes at time s_1. Job 1 in S arrives at time s_1. If a job in S arrives before s_1, it is not heavier than job 1.

Proof. The first statement follows from the definition of S. It means that job 1 did not wait for another job to complete before starting. Since LLW also does not idle the machine if there is a job available, it starts job 1 immediately when it arrives, at time s_1.

Suppose some job $j \in S$ arrives at time $r_j < s_1$ and is heavier than job 1. When j arrives, LLW may continue the job it is running, but LLW will not start any new job that is lighter than j until it has completed j.

By the definition of S, LLW does not complete any job in S before time s_1. So it completes j at some time $c_j > s_1$. In the interval $[r_j, c_j)$, LLW only starts jobs of weight at least w_j. Since $s_1 \in (r_j, c_j)$, this contradicts the fact that LLW starts job 1 at time s_1. This proves the last statement. □

Property 2. In I_1, a job $h \in S$ was interrupted and then job 1 starts at time

$$R - 1 < s_1 < \frac{2 - R}{R - 1}.$$

OPT starts at least one job in S (not job 1) before time $(s_1 + k)/R - k$. Therefore $s_1 > k(R-1)$ and $2 \leq k \leq 7$.

Proof. If $s_1 \leq R - 1$, OPT starts processing the first job in S at the earliest at time $0 = \frac{R-1+1}{R} - 1 \geq \frac{s_1+1}{R} - 1$. By definition 1 and because the critical job is the lightest job in S, S is therefore good. We conclude that $s_1 > R - 1$ in I_1. More generally, if OPT starts its first job from S at time $\frac{s_1}{R} - k(1 - \frac{1}{R})$ or later, then S is by definition 1 good, as $|S| = k$.

Hence, OPT starts at least one job in S at some time $s' < (s_1 + k)/R - k < s_1$, implying that at least one job in S arrives at or before time s'. From $(s_1 + k)/R - k > s' \geq 0$ we get $s_1 > k(R-1)$.

Since LLW does not complete any job in S before time s_1 but also does not idle if at least one job is available, LLW does not idle in the interval $(s', s_1]$. By Property 1, LLW does not complete a job at time s_1. Since LLW was not idle at time s_1, LLW interrupted some job $h \neq 1$ at time s_1.

If $h \notin S$, then h is lighter than the critical job, because otherwise h would be started not later than s_k by the rules of the algorithm, implying that $h \in S$ after all: a contradiction. Since h is lighter than the critical job which is the lightest job in S, all jobs in S are heavier than h. Then all jobs in S arrived at or after time s_1 because otherwise LLW would not still have been running h at time s_1, contradicting $s' < s_1$.

Thus, $h \in S$ and with this, we have $k \geq 2$. Moreover, since LLW interrupted a job h at time s_1, we get $s_1 < \frac{2-R}{R-1}$ by the rules of the algorithm. From $(R-1)k < s_1 < \frac{2-R}{R-1}$ we get $k \leq 7$. □

Property 3. In I_1, LLW does not complete two jobs before time s_1.

Proof. We prove this by contradiction. Suppose LLW completes two jobs before time s_1, called q_1 and q_2 in order. Then $s_1 \in [2, \frac{2-R}{R-1})$. By Property 2, OPT starts running its first job j in S (say $j \neq 1$) before time

$$\frac{s_1 + k}{R} - k < \frac{\frac{2-R}{R-1} + 2}{R} - 2 < 1.23.$$

Note that for the starting time of job q_2 we have $s_{q_2} \geq 1$ and with this, we get that the LW-phase of job q_2 ends not before

$$\frac{s_{q_2} + 2}{R} - 1 \geq \frac{1+2}{R} - 1 = \frac{3}{R} - 1 > 1.29.$$

Suppose j is heavier than q_2. Then either j is started before q_2 or q_2 is interrupted, as j arrives before or during its LW-phase. In both scenarios LLW completes j before q_2, which means q_2 does not complete before time 3. This contradicts $s_1 < \frac{2-R}{R-1} \approx 2.2279$, since q_2 completes before time s_1. Therefore, j is lighter than q_2.

Now, if j is not heavier than q_1, OPT needs to complete $k+2$ jobs that are at least as heavy as the critical job, so $\text{OPT}(I_1) \geq k+2$ whereas $\text{LLW}(I_1) < \frac{2-R}{R-1} + k \approx 2.2279 + k$. Therefore, the competitive ratio is less than R for $k \geq 2$. Hence, j is heavier than q_1.

Job j must arrive after time $2/R - 1$ as j is heavier than q_1 but does not cause q_1 to be interrupted. This implies $\text{OPT}(I_1) \geq 2/R - 1 + k + 1 \approx 1.5270 + k$ as OPT must also complete q_2 which is heavier than the critical job and heavier than q_1 and can therefore also not arrive before time $2/R - 1$. Since $\text{LLW}(I_1) < \frac{2-R}{R-1} + k \approx 2.2279 + k$, the competitive ratio is less than R for $k \geq 2$.

Consequently, in both cases the competitive ratio is less than R for $k \geq 2$, which contradicts the assumption that I_1 is a counterexample. □

Property 4. In I_1, if LLW completes a job q before time s_1, then q is lighter than the critical job, and OPT starts the first job in S at time $(s_q + 2)/R - 1$ or later.

Proof. We prove this by contradiction. Suppose LLW completes a job q before time s_1 which is at least as heavy as the critical job. Since OPT must complete this job as well, OPT must start its first job in $S \cup \{q\}$ before time $(s_1 + k)/R - k - 1 \approx 0.764 s_1 - 0.237k - 1$. Since $s_1 < \frac{2-R}{R-1} < 2.228$, this implies $k \leq 2$, so $k = 2$ by Property 2. It follows that after time $s_q + 1$, there is exactly one interruption, namely at time s_1. Otherwise it would have to be the two most recent jobs (at time s_1) which are in S, since they are the heaviest, implying that S is good: let t be the last starting time before s_1, then LLW starts running the jobs in S (without further interruptions) at time $s_1 < (t+2)/R - 1$ and OPT starts running them at time t or later because the job that LLW started at time t arrived at that time. Since $t \geq s_q + 1$, this means S is good.

Because LLW does not complete another job at time s_q by Property 3, LLW starts job q when it arrives. Then, as we have seen, it interrupts the next job

it starts at time s_1, and then it completes two jobs. On the other hand, OPT must complete three jobs that are at least as heavy as the critical job starting from time s_q. After completing q, LLW does not idle, otherwise I_1 would not be minimal. Therefore, we have $s_1 < (s_q + 3)/R - 1$ and this means that I_1 is not a counterexample in this case: the competitive ratio is at most

$$\frac{(s_q + 3)/R + 1}{s_q + 3} \leq \frac{3/R + 1}{3} < R.$$

The first statement of the lemma follows, and it implies that all jobs in S are heavier than q. Such a job would cause q to be interrupted if it arrived before time $(s_q + 2)/R - 1$. □

Property 5. In I_1, LLW does not complete a job before time s_1.

Proof. By definition, LLW does not complete a job at time s_1. Let q be the last job that LLW completes before time s_1. By Property 4, q is lighter than the critical job. After completing q, LLW does not idle, otherwise I_1 would not be minimal. Any job started after q but before time s_1 is interrupted. If there is exactly one interruption after time $s_q + 1$ (namely at time s_1), then $\text{LLW}(I_1) \leq (s_q + 3)/R + k - 1$ by (1) and $\text{OPT}(I_1) \geq (s_q + 2)/R + k - 1$ by Property 4; the ratio is

$$\frac{\text{LLW}(I_1)}{\text{OPT}(I_1)} \leq \frac{(s_q + 3)/R + k - 1}{(s_q + 2)/R + k - 1} \leq \frac{(0 + 3)/R + 2 - 1}{(0 + 2)/R + 2 - 1} < 1.303,$$

for $k \geq 2$ and $s_q \geq 0$.

If there are additional interruptions, these are all of jobs that arrived after time $s_q + 1$. Let t be the arrival time of the first job in S. In this case, as in the previous proof, the k most recently arrived jobs at time s_1 are the heaviest k jobs that exist at that time. Hence, S does not contain any job that arrived before the first of those k jobs. Put another way, t is not smaller than the arrival time of the first of those k jobs.

If $k = 2$, we find that I_1 is not a counterexample: we have $\text{OPT}(I_1) \geq t + 2$ and $\text{LLW}(I_1) \leq (t+2)/R+1$. The ratio is less than R. For the remaining bounds we always either use the bound $\text{OPT}(I_1) \geq t+k$ or $\text{OPT}(I_1) \geq (s_q+2)/R+k-1$.

If there are exactly two restarts, then

$$\text{LLW}(I_1) \leq \frac{(s_q + 3)/R + 1}{R} - 1 + k,$$

and the ratio is less than $4.513/3.526 \approx 1.280$ for $k \geq 3$ and $s_q \geq 0$.

Else if $k = 3$ and t is the arrival time of the first job in S we have $\text{OPT}(I_1) \geq t + 3$ and

$$\text{LLW}(I_1) \leq \frac{(t + 2)/R - 1 + 2}{R} - 1 + 3,$$

and the ratio is less than R.

Else if there are exactly three restarts, then

$$\textsc{llw}(I_1) \leq \frac{\frac{(s_q+3)/R+1}{R}+1}{R} - 1 + k,$$

and the ratio is less than $5.682/4.527 \approx 1.256$ for $k \geq 4$ and $s_q \geq 0$.

Finally, if $k = 4$ and t is the arrival time of the first job in S we have $\textsc{opt} \geq t + 4$ and

$$\textsc{llw}(I_1) \leq \frac{\frac{(t+2)/R-1+2}{R}-1+2}{R} - 1 + 4,$$

and the ratio is less than R. In all cases, we now have $\textsc{llw}(I_1) \leq \frac{2-R}{R-1} + k$ and $\textsc{opt}(I_1) \geq \frac{s_q+2}{R+k-1}$. The ratio is less than $7.22815/5.52698 \approx 1.30776$ for $k \geq 5$ and $s_q \geq 0$. □

By Property 5 and Property 2, in I_1 it happens one or more times that a job is interrupted before any job completes. All interrupted jobs started when they arrived. The final interruption before the critical job completes is at time s_1. Let m be the total number of interruptions. As in the proof of Property 5, at time s_1, any job that arrived more than k jobs ago is not in S because it is too light. Recall that $2 \leq k \leq 7$ by Property 2. We also have $k \geq m + 1$ (if k were smaller, we could just remove previous jobs from I_1 without affecting the objective value or increasing the optimal cost). As before $\textsc{llw} = s_1 + k$. We now show for all values m by contradiction that I_1 is not a counterexample.

If $m = 1$, then, since $\textsc{opt}$ starts its first job in S before time $(s_1 + k)/R - k$, we have $s_1 < ((s_1 + k)/R - k + 2)/R - 1$. For $k \geq 2$ this leads to $s_1 < 0.4$:

$$\begin{aligned}
& s_1 < ((s_1 + 2)/R - 2 + 2)/R - 1 \\
\Leftrightarrow\ & s_1 < ((s_1 + 2)/R)/R - 1 \\
\Leftrightarrow\ & s_1 < \frac{s_1}{R^2} + \frac{2}{R^2} - 1 \\
\Leftrightarrow\ & s_1(1 - \frac{1}{R^2}) < \frac{2}{R^2} - 1 \\
\Leftrightarrow\ & s_1 < \frac{\frac{2}{R^2} - 1}{1 - \frac{1}{R^2}}
\end{aligned}$$

contradicting the bound of Property 2 $s_1 \geq 2(R - 1)$.

If $m = 2$, $\textsc{opt}(I_1) \geq \max(0, ((s_1 + 1)R - 1)R - 2) + k$ and $k \geq 3$. The ratio is at most R since $s_1 > 3(R - 1)$. It is exactly R for $s_1 = 3(R - 1)$.

It is easily checked that the ratio is at most R for larger values of m as well.

Case 2. The critical job is not the lightest job in S (but it is the last). Let I_2 be a (hypothetical) counterexample in Case 2 which shows the highest possible competitive ratio. Let the last job in S that is lighter than the critical job have number 0, and number the later jobs (call this set S') from 1 to k. Note that

the critical job is the lightest job in S' and this value k is different from in Case 1, where k was the number of consecutively running jobs. Then

$$\textsc{llw}(I_2) = s_0 + k + 1.$$

The k jobs in S' arrived after time s_0, and they are all at least as heavy as the critical job. We have

$$\textsc{opt}(I_2) \geq s_0 + k.$$

The competitive ratio is at most R as long as $s_0 + k \geq 1/(R-1)$, so the only problematic case is $s_0 < \frac{2-R}{R-1}$. Then no jobs from S' arrive before time $(s_0+2)/R - 1$, so we also have the lower bound

$$\textsc{opt}(I_2) \geq \frac{s_0+2}{R} + k - 1.$$

Therefore

$$\frac{\textsc{llw}(I_2)}{\textsc{opt}(I_2)} \leq \frac{s_0+k+1}{(s_0+2)/R + k - 1}.$$

Setting $s_0 = 0$ and $k = 1$ we get $2/(2/R) = R$. This shows that the competitive ratio is at most R for all values of k and s_0. □

Theorem 3. *Any algorithm for the problem* $1|r_j, online, p_j = 1, restart|WC_{\max}$ *is at least* $R_2 \approx 1.2344$ *competitive which is the real root of the equation* $4R_2^3 - R_2^2 - 6 = 0$.

Proof. For a contradiction, suppose that there exists an online algorithm ALG with a competitive ratio of less than R_2. We consider the following job instance. At time $r_1 = 0$, job 1 with $w_1 = 1$ arrives. Then job 2 arrives with $w_2 = \frac{4}{3}$ at time $r_2 = \frac{3}{R_2} - 2 \approx 0.430$.

Case 1. If ALG completes job 1 before job 2, a final job 3 with $w_3 = \frac{3+r_2}{2+r_2} \approx 1.411$ arrives at time $r_3 = 1 + r_2 = \frac{3}{R_2} - 1 \approx 1.430$. The following schedule is feasible: job 2 is completed at time $1 + r_2$, job 3 is completed at time $2 + r_2$ and job 1 is completed at time $3 + r_2$. Thus, $\textsc{opt}(I) \leq 3 + r_2 = {}^3\!/_{R_2} + 1$.

If ALG completes job 2 before job 3, the job order is $1, 2, 3$. Thus, we have

$$\begin{aligned}
\frac{\textsc{alg}(I)}{\textsc{opt}(I)} &\geq \frac{\max(1, 2w_2, 3w_3)}{{}^3\!/_{R_2} + 1} = \frac{\max({}^8\!/_3, {}^{(9+3r_2)}\!/_{(2+r_2)})}{{}^3\!/_{R_2} + 1} \\
&= \frac{\frac{3(3+({}^3\!/_{R_2}-2))}{{}^3\!/_{R_2}}}{{}^3\!/_{R_2} + 1} \\
&= \frac{\frac{3({}^3\!/_{R_2}+1)}{{}^3\!/_{R_2}}}{{}^3\!/_{R_2} + 1} \\
&= R_2.
\end{aligned}$$

If ALG completes job 3 before job 2, then ALG would schedule job 1, then job 3 and finally job 2. Since $w_3(r_3+1) \leq w_2(r_3+2)$, we obtain

$$\frac{\text{ALG}(I)}{\text{OPT}(I)} \geq \frac{\max(1, w_3(r_3+1), w_2(r_3+2))}{3/R_2+1} = \frac{w_2(r_3+2)}{3/R_2+1}$$
$$= \frac{w_2((3/R_2-1)+2)}{3/R_2+1} = \frac{4}{3} \geq R_2.$$

Case 2. If ALG completes job 2 before job 1, then a (different) third job 3 with $w_3 = 2$ arrives at time $r_3 = \frac{6}{R_2^2} - 3 \approx 0.938$.

If now ALG completes job 2 before job 3, no more jobs arrive and OPT could first schedule job 3, then job 2 and finally job 1. Therefore, $\text{OPT}(I) \leq \max(w_3(r_3+1), w_2(r_3+2), r_3+3) = r_3+3$. However, after completing job 2, in the best case, ALG schedules job 3 after job 2 and before job 1, since job 3 has higher weight. Thus, $\text{ALG}(I) = \max(w_2(r_2+1), w_3(r_2+2), r_2+3) = w_3(r_2+2)$ and we have

$$\frac{\text{ALG}(I)}{\text{OPT}(I)} \geq \frac{w_3(r_2+2)}{r_3+3} = \frac{2((3/R_2-2)+2)}{(6/R_2^2-3)+3} = \frac{6/R_2}{6/R_2^2} = R_2.$$

Now, if ALG completes job 3 before job 2, a final fourth job 4 with $w_4 = 1$ arrives at time $r_4 = 3$. A feasible schedule s would be the job order $1, 3, 2, 4$ with $WC_s = 4$. However, if ALG completes job 3 before job 2, at best, ALG first schedules job 3 at time r_3, then job 2 and finally the two jobs with weight 1 in arbitrary order, since job 2 has higher weight than jobs 1 and 4 which have the same weight. Therefore, we have

$$\frac{\text{ALG}(I)}{\text{OPT}(I)} \geq \frac{\max((r_3+1)w_3, (r_3+2)w_2, r_3+3, r_3+4)}{4}$$
$$= \frac{r_3+4}{4} = \frac{6/R_2^2+1}{4} = R_2,$$

using $4R_2^3 - R_2^2 - 6 = 0$. □

4 Conclusion

We studied the problem of online scheduling with restarts for minimizing the weighted makespan $WC_{\max}$. We presented two new lower bounds on the competitive ratio for deterministic algorithms with restarts, one in the general setting and another in the case where all jobs have identical processing times. We designed a deterministic online algorithm for unit processing times that improves the competitive ratio to better than 1.3098. This leaves a small gap to our lower bound for further improvement. Closing this gap appears to be nontrivial and is an interesting topic for future research. This result may also lead the way to an improved result for general processing times.

Acknowledgments. Aflatoun Amouzandeh was financially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - Project Number STE 1727/4-1. Klaus Jansen, Rob van Stee and Corinna Wambsganz were funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project Number 453769249.

References

1. van den Akker, M., Hoogeveen, H., Vakhania, N.: Restarts can help in the online minimization of the maximum delivery time on a single machine. In: Paterson, M. (ed.) Algorithms - ESA 2000, 8th Annual European Symposium, Saarbrücken, Germany, September 5-8, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1879, pp. 427–436. Springer (2000). https://doi.org/10.1007/3-540-45253-2_39
2. Amouzandeh, A., van Stee, R.: Improved online scheduling with restarts on a single machine. In: Bienkowski, M., Englert, M. (eds.) Approximation and Online Algorithms - 22nd International Workshop, WAOA 2024, Egham, UK, September 5-6, 2024, Proceedings. Lecture Notes in Computer Science, vol. 15269, pp. 16–30. Springer (2024). https://doi.org/10.1007/978-3-031-81396-2_2
3. Borodin, A., El-Yaniv, R.: Online computation and competitive analysis. Cambridge University Press (1998)
4. Chai, X., Lu, L., Li, W., Zhang, L.: Best-possible online algorithms for single machine scheduling to minimize the maximum weighted completion time. Asia-Pacific Journal of Operational Research **35**(06), 1850048 (2018)
5. Chen, B., Vestjens, A.: Scheduling on identical machines: How good is LPT in an on-line setting? Operations Research Letters **21**(4), 165–19 (1997)
6. Chrobak, M., Jawor, W., Sgall, J., Tichý, T.: Online scheduling of equal-length jobs: Randomization and restarts help. SIAM J. Comput. **36**(6), 1709–1728 (2007). https://doi.org/10.1137/S0097539704446608
7. Fiat, A., Woeginger, G.J.: Online algorithms: The state of the art, Lecture Notes in Computer Science, vol. 1442. Springer (1998)
8. Fu, R., Cheng, T.C.E., Ng, C.T., Yuan, J.: Online scheduling on two parallel-batching machines with limited restarts to minimize the makespan. Information Processing Letters **110**(11), 444–450 (2010)
9. Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.H.G.R.: Optimization and approximation in deterministic sequencing and scheduling: a survey. In: Annals of discrete mathematics, vol. 5, pp. 287–326. Elsevier (1979)
10. Komm, D.: An Introduction to Online Computation - Determinism, Randomization, Advice. Texts in Theoretical Computer Science. An EATCS Series, Springer (2016), https://doi.org/10.1007/978-3-319-42749-2
11. Laalaoui, Y., M'Hallah, R.: Enhancing the best-first-search F with incremental search and restarts for large-scale single machine scheduling with release dates and deadlines. Ann. Oper. Res. **351**(1), 303–332 (2025). https://doi.org/10.1007/S10479-024-06386-7
12. Li, W.: A best possible online algorithm for the parallel-machine scheduling to minimize the maximum weighted completion time. Asia-Pacific Journal of Operational Research **32**(04), 1550030 (2015)
13. Liang, X., Lu, L., Sun, X., Yu, X., Zuo, L.: Online scheduling on a single machine with one restart for all jobs to minimize the weighted makespan. AIMS Mathematics **9**(1), 2518–2529 (2024)

14. Liu, H.L., Lu, X.W.: Online scheduling on a parallel batch machine with delivery times and limited restarts. Journal of the Operations Research Society of China pp. 1–19 (2022)
15. Lu, L., Zhang, L., Ou, J.: Single machine scheduling with rejection to minimize the weighted makespan. In: Algorithmic Aspects in Information and Management: 15th International Conference, AAIM 2021, Virtual Event, December 20–22, 2021, Proceedings 15. pp. 96–110. Springer (2021)
16. Qi, F., Jinjiang, Y.: Np-hardness of a multicriteria scheduling on two families of jobs. Operations Research Transactions pp. 121–126 (2007)
17. Shmoys, D.B., Wein, J., Williamson, D.P.: Scheduling parallel machines on-line. In: McGeoch, L.A., Sleator, D.D. (eds.) On-Line Algorithms, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, February 11-13, 1991. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 7, pp. 163–166. DIMACS/AMS (1991). https://doi.org/10.1090/DIMACS/007/13
18. van Stee, R., Poutré, J.A.L.: Minimizing the total completion time on-line on a single machine, using restarts. Journal of Algorithms **57**(2), 95–129 (2005). https://doi.org/10.1016/J.JALGOR.2004.10.001
19. Sun, R.: Randomized approximation schemes for minimizing the weighted makespan on identical parallel machines. Journal of Combinatorial Optimization **47**(3), 34 (2024)
20. Tian, J., Fu, R., Yuan, J.: Online over time scheduling on parallel-batch machines: a survey. Journal of the Operations Research Society of China **2**(4), 445–454 (2014)