AIMS Mathematics, 9(1): 2518–2529.
DOI: 10.3934/math.2024124
Received: 23 October 2023
Revised: 04 December 2023
Accepted: 13 December 2023
Published: 25 December 2023

http://www.aimspress.com/journal/Math

Research article

Online scheduling on a single machine with one restart for all jobs to minimize the weighted makespan

Xiaoxiao Liang, Lingfa Lu*, Xueke Sun, Xue Yu and Lili Zuo

School of Mathematics and Statistics, Zhengzhou University, Zhengzhou, Henan 450001, China

* **Correspondence:** Email: lulingfa@zzu.edu.cn.

Abstract: In this paper, we consider the online scheduling problem on a single machine to minimize the weighted makespan. In this problem, all jobs arrive over time and they are allowed to be restarted only once. For the general case when the processing times of all jobs are arbitrary, we show that there is no online algorithm with a competitive ratio of less than 2, which matches the lower bound of the problem without restart. That is, only one restart for all jobs is invalid for improving the competitive ratio in the general case. For the special case when all jobs have the same processing time, we present the best possible online algorithm with a competitive ratio of 1.4656, which improves the competitive ratio of $\frac{1+\sqrt{5}}{2} \approx 1.618$ for the problem without restart.

Keywords: scheduling; restart; the weighted makespan; online algorithm; competitive ratio
Mathematics Subject Classification: 90B35, 68M20, 68Q17

1. Introduction

In this paper, we consider the online scheduling problem with restarts on a single machine, with the aim of minimizing the weighted makespan, which refers to the maximum weighted completion time of the jobs. For the majority of online scheduling problems, no online algorithm can generate a solution that is equally effective as an offline optimal solution because in the online scheduling, the exact data of each job is unknown until it arrives. In the offline scheduling, the decision maker could make better decisions since all information about the jobs are given in advance. The arrival of jobs can be categorized into two groups in online scheduling: Over time and over list. Jobs arriving over time means that each job has an arrival time, after which it can be processed, and the online algorithm does not have to schedule this job immediately. Jobs arriving over list means that the next job arrives after the current job has been scheduled on the machine. In this paper, all jobs arrive over time.

As far as we know, the concept of restarts was first proposed for online scheduling by Akker et al. [1]. Restart (Hoogeveen et al. [2]) means that a job being processed is interrupted and

loses all the work already done on it. In other words, the processing time previously spent on the job is wasted. The interrupted job then becomes available again as an unprocessed job, which can be subsequently processed and restarted, and we call it a restarted job. As further jobs arrive over time, we obtain more information about the job instance. Allowing restarts actually prevents us from making wrong decisions, since we have the opportunity to reconsider the schedule. However, frequent restarts can result in resource waste and harm jobs in practice. "Limited restart" was first proposed by Fu et al. [3], in which "limited restart" means that each job can be restarted only once, and "k-limited restart" supposes that each job cannot be interrupted more than k times, where k can be either a positive integer or infinity. Different from the definition of "k-limited restart," the problem we investigate in this paper is "1-restart", which means that the total number of restarts allowed for all jobs is 1.

A common approach for assessing the performance of an online algorithm is usually to calculate and analyze its competitive ratio. Assuming that the instance of an online scheduling problem under consideration is I, and $\mathcal{A}(I)$ is the objective function value obtained by running Algorithm $\mathcal{A}$, $OPT(I)$ signifies the offline optimal objective value. For the minimization problem, we say that Algorithm $\mathcal{A}$ is ρ-competitive if the inequality $\mathcal{A}(I) \leq \rho OPT(I)$ holds. That is, once an online algorithm can generate a schedule whose objective value is not lower than ρ times the value of the offline optimal schedule, it will be considered as a ρ-competitive algorithm. The infimum of ρ is regarded as the competitive ratio of Algorithm $\mathcal{A}$. For the online scheduling problem considered in our paper, the competitive ratio of the algorithm is the minimum value of ρ such that it is ρ-competitive. Moreover, an online algorithm is the best possible if no algorithm can be discovered that possesses a smaller competitive ratio than it does.

Roughly speaking, over the past few decades, there has been a wealth of researches on restarts in the field of online scheduling. In the model of parallel batch machine, Liu et al. [4] studied the online scheduling problem with "k-limited restart" on a bounded parallel batch machine with the goal of minimizing the makespan, and the jobs are equal in length. Depending on the capacity of the parallel batch, they presented the best possible online algorithm for the corresponding problem when $k \geq 1$. Furthermore, Liu et al. [5] investigated the online scheduling problem on an unbounded parallel batch machine with "limited restart", and the delivery time of each job is no longer than its processing time. The goal is to minimize the transportation completion time. They provided the best possible online algorithm with a competitive ratio of $\frac{3}{2}$. For the problem that minimizing the makespan using restarts on a parallel batch processing system, Fu et al. [6] showed that the lower bound is $\frac{5-\sqrt{5}}{2}$, and designed an online algorithm with a competitive ratio of $\frac{3}{2}$. The problem was further investigated and Yuan et al. [7] solved the gap and gave the best possible online algorithm. Additionally, for online scheduling problem on two parallel batch machines with "limited restart" to minimize the makespan, Fu et al. [8] presented the best possible online algorithm with a competitive ratio of $\frac{\sqrt{3}+1}{2}$ under the assumption of "second-restart". There are many other outcomes about restarts, details of which can be found in Tian et al. [9].

In certain service systems, it is imperative to keep service costs low. Hence, this paper draws inspiration from customers' desires for lower costs. Customers' orders often arrive at the service provider at any time in practice, and the system aims to minimize the highest service costs as low as possible for each client, so the online scheduling problem discussed in this paper that minimizes the weighted makespan may also be used to assess the expenses associated with work-in-progress inventory and customer satisfaction. Therefore, considering the issue that minimizes the weighted

makespan is of clear interest. Actually, there are fruitful achievements focused on the objective that minimize the weighted makespan. Li [10] demonstrated that no deterministic online algorithm has a competitive ratio of less than 2. Moreover, for the problem on a single machine, he offered a 3-competitive algorithm. For the problem on parallel machines and jobs with identical processing times, Li [10] presented the best possible online algorithm. The gap in Li [10] was later solved by Chai et al. [11]. Lu et al. [12] considered the problem that minimizes the weighted makespan plus the rejection cost and demonstrated that the problem is NP-hard, as well as provided a 2-approximation algorithm. For the online scheduling problem on identical bounded parallel batch machines to minimize the maximum weighted makespan, and the jobs with the same processing time arrive over time, Li et al. [13] presented the best possible online algorithm with the competitive ratio of $\frac{\sqrt{5}+1}{2}$. Furthermore, they also gave the best possible online dense algorithm with a competitive ratio of 2. Sun [14] considered the single machine scheduling problem with rejection to minimize the weighted makespan, and showed that the problem is fixed-parameter tractable with respect to some parameters.

For convenience, we present some notations that will be used subsequently. The problem addressed in our paper may be written as $1|r_j, online, 1\text{-}restart|WC_{\max}$ and $1|r_j, online, p_j = p, 1\text{-}restart|WC_{\max}$, where $WC_{\max} = \max\{w_jC_j : J_j \in I\}$ if the job instance I is given. Unless ambiguity would result, we simplify the objective values of an online algorithm and an optimal algorithm by $WC_{\text{on}}(I)$ and $WC_{\text{opt}}(I)$ for instance I, respectively. In addition, we use C_j to stand for the completion time of job J_j. In order to distinguish the online scheduling with restarts from the offline scheduling, we denote the i-th starting time of job J_j by S_{ji}. Furthermore, since only one restart is considered in this paper, we abbreviate S_{j1} as S_j. Let ε be an infinitely small positive number, and $\beta \approx 0.4656$ is the positive root of equation $x(1 + x)^2 = 1$.

Throughout this paper, in Section 2, we consider problem $1|r_j, online, 1\text{-}restart|WC_{\max}$ and prove that there is no online algorithm with a competitive ratio of less than 2, which matches the lower bound of the problem without restart. That is, "1-restart" is invalid for improving the competitive ratio of problem $1|r_j, online|WC_{\max}$. In Section 3, we provide the best possible online algorithm with a competitive ratio of $1 + \beta \approx 1.4656$ for problem $1|r_j, online, p_j = p, 1\text{-}restart|WC_{\max}$. Li [8] presented the best possible online algorithm with a competitive ratio of $\frac{1+\sqrt{5}}{2} \approx 1.618$ for problem $1|r_j, online, p_j = p|WC_{\max}$, so in our paper, we improve the result in Li [8]. In Section 4, we summarize the results of this paper and present some problems that can be investigated in the future.

2. The general case with the arbitrary processing times

In this section, we consider the problem $1|r_j, online, 1\text{-}restart|WC_{\max}$. For the online scheduling problem to minimize the weighted makespan, i.e., $1|r_j, online|WC_{\max}$, Chai et al. [11] gave two best possible online algorithms with a competitive ratio of 2 based on the preemptive optimal schedule and the concept of delay, respectively.

It is obvious that the competitive ratio of problem $1|r_j, online|WC_{\max}$ is likely to be improved with the use of restarts. Hence, for problem $1|r_j, online, 1\text{-}restart|WC_{\max}$, its competitive ratio may be less than or equal to 2. In the following, we demonstrate that the lower bound for problem $1|r_j, online, 1\text{-}restart|WC_{\max}$ is still 2, which means that "1-restart" does not lead to a better competitive ratio for problem $1|r_j, online|WC_{\max}$.

Theorem 1. For problem $1|r_j, online, 1\text{-}restart|WC_{\max}$, there is no online algorithm with a competitive ratio of less than 2.

Proof. For an arbitrary online algorithm H, we construct an online instance I, in which the jobs are presented by the adversary.

There are four jobs J_1, J_2, J_3, J_4 contained in I and the jobs arrive in the non-decreasing order of weights, i.e., $w_i \leq w_j$ if job J_j arrives after job J_i. We write $I_m = \{J_j : 1 \leq j \leq m\}$. At time 0, job J_1 with processing time 1 and weight 1 comes, i.e., $(r_1, p_1, w_1) = (0, 1, 1)$.

Case 1. $S_1 \geq 1$.

If $S_1 \geq 1$, then no jobs arrive later. In this case, $WC_{\text{opt}}(I_1) = w_1 p_1 = 1$ and the objective value of algorithm H is $WC_{\text{on}}(I_1) = w_1(S_1 + 1) \geq 2 = 2WC_{\text{opt}}$. Thus, in order to guarantee the competitive ratio, we assume that $S_1 < 1$ in the following discussion.

Case 2. $S_1 < 1$.

When algorithm H processes job J_1 at time S_1, then job J_2 arrives at time $S_1 + \varepsilon$, and $(r_2, p_2, w_2) = (S_1 + \varepsilon, \varepsilon, M)$, where M is a sufficiently large positive integer and ε is a positive number that converges infinitely to 0. In the following we discuss the scenarios according to the value of S_2. Note that $S_1 < 2S_1 + 3\varepsilon \leq S_1 + 1$.

Case 2.1. $S_2 \geq 2S_1 + 3\varepsilon$.

If $S_2 \geq 2S_1 + 3\varepsilon$, then we have $WC_{\text{on}}(I_2) \geq w_2(S_2 + p_2) = M(S_2 + \varepsilon)$. In an optimal schedule, job J_2 has to be processed first at time r_2, then job J_1 is processed at the completion time of job J_2. So, we have $WC_{\text{opt}}(I_2) = M(S_1 + 2\varepsilon)$, then we conclude that

$$\frac{WC_{\text{on}}(I_2)}{WC_{\text{opt}}(I_2)} \geq \frac{M(S_2 + \varepsilon)}{M(S_1 + 2\varepsilon)} \geq \frac{2S_1 + 4\varepsilon}{S_1 + 2\varepsilon} = 2.$$

Case 2.2. $S_2 < 2S_1 + 3\varepsilon$.

We assume that $S_2 < 2S_1 + 3\varepsilon \leq S_1 + 1$, then job J_3 arrives after jobs J_1 and J_2 are completed, i.e., $(r_3, p_3, w_3) = (S_1 + 2\varepsilon + 1, M, M)$. In the following we discuss the scenarios according to the value of S_3.

Case 2.2.1. $S_3 \geq M - 1$.

If $S_3 \geq M - 1$, then no jobs arrive. Since all jobs are finished before job J_3 arrives, then we have $WC_{\text{on}}(I_3) = w_3(S_3 + p_3) \geq w_3(M - 1 + M) = w_3(2M - 1)$. However, for the instance I_3, the optimal schedule will process job J_3 at time r_3 and jobs J_1 and J_2 are finished before r_3. Thus, we have $WC_{\text{opt}}(I_3) = w_3(r_3 + p_3) = w_3(S_1 + 2\varepsilon + 1 + M)$. Furthermore,

$$\frac{WC_{\text{on}}(I_3)}{WC_{\text{opt}}(I_3)} \geq \frac{w_3(2M - 1)}{w_3(S_1 + 2\varepsilon + 1 + M)} = \frac{2 - \frac{1}{M}}{\frac{S_1}{M} + \frac{2\varepsilon}{M} + \frac{1}{M} + 1} \to 2,$$

when $M \to \infty$, $\varepsilon \to 0^+$. As a result, we assume that $1 < r_3 < S_3 < M - 1$ in the following discussion.

Case 2.2.2. $r_3 < S_3 < M - 1$.

Once algorithm H processes job J_3 at time S_3, job J_4 with processing time ε and weight $2M^2$ comes at time $S_3 + \varepsilon$, i.e., $(r_4, p_4, w_4) = (S_3 + \varepsilon, \varepsilon, 2M^2)$. Due to the fact that all jobs are allowed to be restarted only once, then the processing of job J_3 cannot be interrupted. Therefore, job J_4 will be processed after the completion of job J_3 in algorithm H. We have $WC_{\text{on}}(I_4) = \max\{w_3(S_3 + p_3), w_4(S_3 + p_3 + \varepsilon)\} = w_4(S_3 + p_3 + \varepsilon) = 2M^2(S_3 + M + \varepsilon)$. The optimal schedule for instance I_4 is to process all jobs in the

order of J_2, J_1, J_4, J_3, and $WC_{\text{opt}}(I_4) = \max\{w_4(S_3+2\varepsilon), w_3(S_3+p_3+2\varepsilon)\} = \max\{2M^2(S_3+2\varepsilon), M(S_3+M+2\varepsilon)\}$. Note that

$$\begin{aligned} 2M^2(S_3+2\varepsilon) - M(S_3+M+2\varepsilon) &= M^2S_3 + M^2S_3 + 4M^2\varepsilon - MS_3 - M^2 - 2M\varepsilon \\ &= (M^2-M)S_3 + M^2(S_3-1) + (4M^2-2M)\varepsilon > 0, \end{aligned}$$

thus, we have $WC_{\text{opt}}(I_4) = 2M^2(S_3+2\varepsilon)$ and

$$\frac{WC_{\text{on}}(I_4)}{WC_{\text{opt}}(I_4)} = \frac{2M^2(S_3+M+\varepsilon)}{2M^2(S_3+2\varepsilon)} \to 1 + \frac{M}{S_3} > 1 + \frac{M}{M-1} > 2,$$

when $M \to \infty,\ \varepsilon \to 0^+$. This completes the proof of Theorem 1.

3. A special case with the same processing time

In this section, we consider a special case of problem $1|r_j, online, 1\text{-}restart|WC_{\max}$, in which all jobs are of equal length. We denote the problem by $1|r_j, online, p_j = p, 1\text{-}restart|WC_{\max}$. Without loss of generality, it's feasible to consider the problem in which all jobs have the unit processing time. Thus, we study problem $1|r_j, online, p_j = 1, 1\text{-}restart|WC_{\max}$. For this problem, we present a lower bound of $1+\beta$, and the best possible online algorithm that matches the lower bound. For problem $1|r_j, online, p_j = p|WC_{\max}$, Li [10] gave the best possible online algorithm with a competitive ratio of $\frac{1+\sqrt{5}}{2} \approx 1.618$. Hence we improve the result of Li [10] and this indicates that the competitive ratio of problem $1|r_j, online|WC_{\max}$ can be improved when all jobs have the same processing time and the number of restarts allowed for all jobs is 1.

3.1. The lower bound

Recall that $\beta \approx 0.4656$ is a positive real solution of equation $x(1+x)^2 = 1$.

Theorem 2. There exists no online algorithm with a competitive ratio of less than $1+\beta$ for the scheduling problem $1|r_j, online, p_j = 1, 1\text{-}restart|WC_{\max}$.

Proof. By the contradiction, suppose that there exists an online algorithm H with a competitive ratio of less than $1+\beta$. We consider the following job instance I provided by the adversary.

At time $r_1 = 0$, job J_1 with $w_1 = 1$ arrives. Suppose that algorithm H starts to process it at time S_1. In order to guarantee the competitive ratio of algorithm H, we discuss the value of S_1.

Case 1. $S_1 \geq \beta$.

If $S_1 \geq \beta$, then the adversary will inform us that no jobs arrive later, so $WC_{\text{opt}} = w_1p_1 = 1$ and $WC_{\text{on}} = w_1C_1 = S_1 + p_1 \geq 1+\beta = (1+\beta)WC_{\text{opt}}$.

Case 2. $S_1 < \beta$.

In this case, job J_2 with $w_2 = 2$ arrives at time $S_1 + \varepsilon$. Suppose that algorithm H starts to process it at time S_2.

Case 2.1. $S_2 \geq S_1 + 1$.

If $S_2 \geq S_1 + 1$, i.e., algorithm H does not restart the job J_1, then no other jobs arrive. In this case, we have $WC_{\text{on}} = w_2C_2 = w_2(S_2+p_2) \geq 2(S_1+2)$. In the offline schedule, we can process job J_2 before job J_1 and $WC_{\text{opt}} \leq \max\{w_2(r_2+p_2), w_1(r_2+p_2+p_1)\} = \max\{2(S_1+\varepsilon+1), S_1+\varepsilon+2\} = 2(S_1+\varepsilon+1)$.

Thus, we have

$$\frac{WC_{\text{on}}}{WC_{\text{opt}}} \geq \frac{2(S_1+2)}{2(S_1+\varepsilon+1)} \to \frac{S_1+2}{S_1+1} = 1+\frac{1}{S_1+1} > 1+\frac{1}{\beta+1} > 1+\beta.$$

Case 2.2. $r_2 \leq S_2 < S_1 + 1 < 1+\beta$.

Consequently, in this case, algorithm H restarts job J_1 at time S_2. In the following we discuss the scenarios according to the value of S_2.

Case 2.2.1. $S_2 \geq (1+\beta)^2 - 1$.

If $S_2 \geq (1+\beta)^2 - 1$, then no jobs appear subsequently because algorithm H restarts job J_1 too late. Hence, we have $WC_{\text{on}} = w_2C_2 = w_2(S_2+p_2) \geq 2(1+\beta)^2$, and as mentioned above, $WC_{\text{opt}} \leq 2(S_1+\varepsilon+1)$. This implies that

$$\frac{WC_{\text{on}}}{WC_{\text{opt}}} \geq \frac{2(1+\beta)^2}{2(S_1+\varepsilon+1)} > \frac{(1+\beta)^2}{1+\beta} = 1+\beta, \varepsilon \to 0^+.$$

Case 2.2.2. $r_2 \leq S_2 < (1+\beta)^2 - 1$.

If $r_2 \leq S_2 < (1+\beta)^2 - 1$, then job J_3 with $w_3 = 4$ arrives at time $S_2 + \varepsilon$. Job J_3 cannot start to be processed until job J_2 is completed, since all jobs are allowed to be restarted only once. Assume that the starting time of job J_3 under algorithm H is S_3, then $S_3 = S_2 + 1$ and $WC_{\text{on}} = w_3C_3 = 4(S_2+2)$. Note that the optimal schedule will process these jobs on the machine in the order of J_3, J_2, J_1, then by comparison, we can obtain $WC_{\text{opt}} = 4(S_2+\varepsilon+1)$. Hence,

$$\frac{WC_{\text{on}}}{WC_{\text{opt}}} = \frac{4(S_2+2)}{4(S_2+\varepsilon+1)} \to 1+\frac{1}{S_2+1} > 1+\frac{1}{(1+\beta)^2} = 1+\beta, \text{ when } \varepsilon \to 0^+.$$

This completes the proof of Theorem 2.

3.2. An online algorithm

In this subsection, we will present an online algorithm for problem $1|r_j, online, p_j = 1, 1\text{-}restart|WC_{\max}$ and analyze its competitive ratio.

In the present moment t, we use $U(t)$ to denote the set of jobs that have arrived but not yet been processed. Furthermore, if $U(t) \neq \emptyset$, we find the job with the largest weight in $U(t)$ and denote it as job J_k, i.e., $w_k = \max\{w_j : J_j \in U(t)\}$. The following algorithm $\mathcal{A}$ is provided to solve problem $1|r_j, online, p_j = 1, 1\text{-}restart|WC_{\max}$.

Algorithm $\mathcal{A}$

Step 1. Set $t := \beta$ and the machine is idle in $[0, \beta)$.

Step 2. At time t, if $U(t) \neq \emptyset$ and the machine is idle, then schedule job J_k non-preemptively on the machine. Otherwise, go to Step 4.

Step 3. In the time interval $(t, t+1)$, if no jobs arrive, then set $t := t+1$ and go to Step 2. Otherwise, let $\mathcal{F}$ be the set of all jobs arrive in the time interval $(t, t+1)$, i.e., $\mathcal{F} = \{J_j \mid t < r_j < t+1\}$, then we write $r_{\min} = \min\{r_j \mid J_j \in \mathcal{F}\}$ and do the following:

Step 3.1. If $r_{\min} > (1+\beta)^2 - 1$, reset $t := t+1$, go to Step 2.

Step 3.2. If $r_{\min} \leq (1+\beta)^2 - 1$, let $\tilde{\mathcal{F}} = \{J_j \mid t < r_j \leq (1+\beta)^2 - 1\}$. If there exists some jobs J_j in $\tilde{\mathcal{F}}$ satisfying the condition $w_j > (1+\beta)w_k$, then find the job with the largest weight from these jobs and schedule it at time $(1+\beta)^2 - 1$, then go to Step 3. Otherwise, continue to process the job J_k until it is completed, reset $t := t+1$ and go to Step 2.

Step 4. If there are still some jobs arriving, set t be the earliest arrival time of these jobs, go to Step 2. Otherwise, do nothing but wait.

From Algorithm $\mathcal{A}$, we have the following observations.

Observation 1. If job i is restarted and job $i+1$ is processed at time $(1+\beta)^2-1$, then we write the job that is processed immediately after job $i+1$ as job $i+2$. If the arrival time of job $i+2$ is less than β, then job $i+2$ is job i. Otherwise, the arrival time of job $i+2$ is greater than β.

Observation 2. If the starting time of job i satisfies that $S_i < (1+\beta)^2-1$, then we have $S_i \le r_i+\beta$.

Theorem 3. For problem $1|r_j, online, p_j = 1, 1\text{-}restart|WC_{\max}$, algorithm $\mathcal{A}$ is the best possible online algorithm with the competitive ratio of $1+\beta$.

Proof. Let σ be the schedule obtained by algorithm $\mathcal{A}$, $S_j(\sigma)$ and $C_j(\sigma)$ be the starting time and the completion time of job J_j, respectively. Denote the offline optimal schedule as π, and we can use WC_{on} and WC_{opt} to represent the objective function values of schedule σ and π, respectively.

For each $j = 1, \cdots, n$, we assume that $J_{\sigma[j]}$ is the j-th completed job in schedule σ. Moreover, in the following proof, we can suppose that $J_{\sigma[k]}$ is the critical job that assumes the objective value of schedule σ. That is,

$$WC_{\max}(\sigma) = \max\{w_jC_j(\sigma) : j = 1, \cdots, n\} = w_{\sigma[k]}C_{\sigma[k]}(\sigma).$$

In schedule σ, let job $J_{\sigma[i]}$ with $1 \le i \le k$ be the job with the smallest index such that jobs $J_{\sigma[i]}, \cdots, J_{\sigma[k]}$ are processed consecutively. Similarly, let $S_{\sigma[i]}(\sigma)$ be the starting time of job $J_{\sigma[i]}$, then we have $C_{\sigma[k]}(\sigma) = S_{\sigma[i]}(\sigma) + \sum_{j=i}^{k} p_{\sigma[j]}$. In the following, we distinguish two cases in our discussion.

Case 1. $w_{\sigma[k]} = \min\{w_{\sigma[j]} : i \le j \le k\}$.

In this case, we assume that job $J_{\sigma[x]}$ with $i \le x \le k$ is the final completed job among $J_{\sigma[i]}, \cdots, J_{\sigma[k]}$ in an optimal schedule π. Hence, we have $C_{\sigma[x]}(\pi) \ge \sum_{j=i}^{k} p_{\sigma[j]}$, then $WC_{\text{opt}} \ge w_{\sigma[x]}C_{\sigma[x]}(\pi) \ge w_{\sigma[k]}\sum_{j=i}^{k} p_{\sigma[j]}$. Since $WC_{\text{on}} = w_{\sigma[k]}C_{\sigma[k]}(\sigma) = w_{\sigma[k]}(S_{\sigma[i]}(\sigma)+\sum_{j=i}^{k} p_{\sigma[j]})$, we can prove the competitive ratio of algorithm $\mathcal{A}$ by discussing the value of $S_{\sigma[i]}$ in the subcases. From algorithm $\mathcal{A}$, we have $S_{\sigma[i]}(\sigma) \ge \beta$.

Case 1.1. $S_{\sigma[i]}(\sigma) = \beta$.

Note that $S_{\sigma[i]}(\sigma) = \beta$ holds if and only if job $J_{\sigma[i]}$ arrives at and before time β, i.e., $0 \le r_{\sigma[i]} \le \beta$, then we have

$$WC_{\text{opt}} \ge w_{\sigma[i]}C_{\sigma[i]}(\pi) \ge w_{\sigma[i]}(r_{\sigma[i]}+1) \ge w_{\sigma[k]},$$

thus,

$$WC_{\text{on}} = w_{\sigma[k]}C_{\sigma[k]}(\sigma) = w_{\sigma[k]}(S_{\sigma[i]}(\sigma) + \sum_{j=i}^{k} p_{\sigma[j]}) = w_{\sigma[k]}(\beta + \sum_{j=i}^{k} p_{\sigma[j]}) \le (1+\beta)WC_{\text{opt}}.$$

Case 1.2. $S_{\sigma[i]}(\sigma) = (1+\beta)^2-1$.

There are two possible reasons for $S_{\sigma[i]}(\sigma) = (1+\beta)^2-1$: One in which a restart occurs at time $(1+\beta)^2-1$ and the other in which there is no restart occurance and the arrival time of job $J_{\sigma[i]}$ is $(1+\beta)^2-1$.

Case 1.2.1. A restart occurs at time $S_{\sigma[i]}(\sigma)$.

We can assume that job $J_{\sigma[h]}$ is the one that being processed when job $J_{\sigma[i]}$ arrives, and it is easy to find that $r_{\sigma[i]} > \beta$. By algorithm $\mathcal{A}$, we conclude that the restart occurs when condition $w_{\sigma[i]} >$

$w_{\sigma[h]}(1+\beta)$ holds. In this case, we have

$$WC_{\text{on}} = w_{\sigma[k]}C_{\sigma[k]}(\sigma) = w_{\sigma[k]}(S_{\sigma[i]}(\sigma) + \sum_{j=i}^{k} p_{\sigma[j]}) = w_{\sigma[k]}((1+\beta)^2 - 1 + \sum_{j=i}^{k} p_{\sigma[j]}),$$

then we estimate the optimal objective function value WC_{opt}.

If $|\{J_{\sigma[j]} : i \le j \le k\}| = 2$ and $h \in \{i+1, \cdots, k\}$, then we have $h = i+1 = k$ and $w_{\sigma[h]} = w_{\sigma[k]}$. In an optimal schedule π, if job $J_{\sigma[k]}$ is scheduled before job $J_{\sigma[i]}$, we can conclude that

$$WC_{\text{opt}} \ge w_{\sigma[i]}(r_{\sigma[k]} + 2) > 2w_{\sigma[h]}(1+\beta) = 2w_{\sigma[k]}(1+\beta) > w_{\sigma[k]}(2+\beta).$$

If job $J_{\sigma[i]}$ is scheduled before job $J_{\sigma[k]}$, then we have

$$WC_{\text{opt}} \ge w_{\sigma[k]}(r_{\sigma[i]} + 2) > w_{\sigma[k]}(2+\beta).$$

All in all, we can find that $WC_{\text{opt}} > w_{\sigma[k]}(2+\beta)$, then

$$WC_{\text{on}} = w_{\sigma[k]}((1+\beta)^2 - 1 + \sum_{j=i}^{k} p_{\sigma[j]}) = w_{\sigma[k]}(\beta(\beta+2) + \sum_{j=i}^{k} p_{\sigma[j]}) \le (1+\beta)WC_{\text{opt}}.$$

If $|\{J_{\sigma[j]} : i \le j \le k\}| = 2$ and $h \notin \{i+1, \cdots, k\}$, then by Observation 1, job $J_{\sigma[i]}$ and job $J_{\sigma[k]}$ arrive after time β. Thus, we can obtain $WC_{\text{opt}} \ge w_{\sigma[x]}C_{\sigma[x]}(\pi) \ge w_{\sigma[k]}(\beta + \sum_{j=i}^{k} p_{\sigma[j]}) = w_{\sigma[k]}(\beta+2)$, then

$$WC_{\text{on}} = w_{\sigma[k]}((1+\beta)^2 - 1 + \sum_{j=i}^{k} p_{\sigma[j]}) = w_{\sigma[k]}(\beta(\beta+2) + \sum_{j=i}^{k} p_{\sigma[j]}) \le (1+\beta)WC_{\text{opt}}.$$

If $|\{J_{\sigma[j]} : i \le j \le k\}| \ge 3$, i.e., the number of jobs in $\{J_{\sigma[i]}, \cdots, J_{\sigma[k]}\}$ is at least 3 and $w_{\sigma[j]} \ge w_{\sigma[k]}$ holds for each $i \le j \le k$, then we can obtain that $WC_{\text{opt}} \ge 3w_{\sigma[k]} > w_{\sigma[k]}(2+\beta)$; thus,

$$WC_{\text{on}} = w_{\sigma[k]}((1+\beta)^2 - 1 + \sum_{j=i}^{k} p_{\sigma[j]}) = w_{\sigma[k]}(\beta(\beta+2) + \sum_{j=i}^{k} p_{\sigma[j]}) \le (1+\beta)WC_{\text{opt}}.$$

Case 1.2.2. No restart occurs at time $S_{\sigma[i]}(\sigma)$.

If no restart occurs at time $S_{\sigma[i]}(\sigma)$, which indicates that $S_{\sigma[i]}(\sigma) = r_{\sigma[i]}$, then it can be seen that job $J_{\sigma[i]}$ is the first job that arrives among $J_{\sigma[i]}, \cdots, J_{\sigma[k]}$. Therefore,

$$WC_{\text{opt}} \ge w_{\sigma[x]}C_{\sigma[x]}(\pi) \ge w_{\sigma[k]}(r_{\sigma[i]} + \sum_{j=i}^{k} p_{\sigma[j]}) = w_{\sigma[k]}(S_{\sigma[i]}(\sigma) + \sum_{j=i}^{k} p_{\sigma[j]}) = WC_{\text{on}}.$$

In this case, schedule σ is obviously the optimal schedule.

Case 1.3. $S_{\sigma[i]}(\sigma) > (1+\beta)^2 - 1$ or $\beta < S_{\sigma[i]}(\sigma) < (1+\beta)^2 - 1$.

In either situation, we have $S_{\sigma[i]}(\sigma) = r_{\sigma[i]}$. Similar to Case 1.2.2, we can deduce that σ is indeed the optimal schedule.

Case 2. $w_{\sigma[k]} > \min\{w_{\sigma[j]} : i \le j \le k\}$.

In this case, we suppose that job $J_{\sigma[y]}$ with $i \le y \le k$ is the last one such that $w_{\sigma[y]} < w_{\sigma[k]}$. From the definition of $J_{\sigma[y]}$, we have $w_{\sigma[j]} \ge w_{\sigma[k]} > w_{\sigma[y]}$ for each $j = y+1, \cdots, k$. By algorithm $\mathcal{A}$, job $J_{\sigma[y]}$ is picked out and processed at time $S_{\sigma[y]}(\sigma)$, then job $J_{\sigma[y]}$ must have the largest weight in $U(t)$, where $t = S_{\sigma[y]}(\sigma)$. Furthermore, since $w_{\sigma[j]} \ge w_{\sigma[k]} > w_{\sigma[y]}$, then we have $r_{\sigma[j]} > S_{\sigma[y]}(\sigma) \ge \beta$. Otherwise, job $J_{\sigma[y]}$ will be processed after job $J_{\sigma[j]}$ for each $j = y+1, \cdots, k$. We assume that job $J_{\sigma[z]}$ with $y+1 \le z \le k$ is the last finished job among $J_{\sigma[y+1]}, \cdots, J_{\sigma[k]}$ in an optimal schedule π, then we have $C_{\sigma[z]}(\pi) \ge S_{\sigma[y]}(\sigma) + \sum_{j=y+1}^{k} p_{\sigma[j]}$ and

$$WC_{\text{opt}} \ge w_{\sigma[z]}C_{\sigma[z]}(\pi) \ge w_{\sigma[k]}(S_{\sigma[y]}(\sigma) + \sum_{j=y+1}^{k} p_{\sigma[j]}).$$

If $S_{\sigma[y]}(\sigma) \ge (1+\beta)^2 - 1$ or at least one job from set $\{J_{\sigma[j]} : j = y+1, \cdots, k\}$ has an arrival time of greater than or equal to $(1+\beta)^2 - 1$, we can estimate that

$$WC_{\text{opt}} \ge w_{\sigma[k]}C_{\sigma[k]}(\pi) \ge w_{\sigma[k]}(r_{\sigma[k]} + 1) > w_{\sigma[k]}(S_{\sigma[y]}(\sigma) + 1) \ge w_{\sigma[k]}(1+\beta)^2.$$

As a result, the objective value of schedule σ satisfies

$$\begin{aligned} WC_{\text{on}} &= w_{\sigma[k]}C_{\sigma[k]}(\sigma) = w_{\sigma[k]}(S_{\sigma[y]}(\sigma) + p_{\sigma[y]} + \textstyle\sum_{j=y+1}^{k} p_{\sigma[j]}) \\ &= w_{\sigma[k]}(S_{\sigma[y]}(\sigma) + \textstyle\sum_{j=y+1}^{k} p_{\sigma[j]} + \beta(1+\beta)^2) \le (1+\beta)WC_{\text{opt}}. \end{aligned}$$

So, we only need to consider the case that $S_{\sigma[y]}(\sigma) < (1+\beta)^2 - 1$ and the arrival time of all jobs in set $\{J_{\sigma[j]} : j = y+1, \cdots, k\}$ is less than $(1+\beta)^2 - 1$. In this case, we can deduce that no restart occurs at time $(1+\beta)^2 - 1$ and the condition $w_{\sigma[j]} > (1+\beta)w_{\sigma[y]}$ in algorithm $\mathcal{A}$ is not satisfied, i.e., $w_{\sigma[j]} \le (1+\beta)w_{\sigma[y]}$ for each $j = y+1, \cdots, k$. Let $\Gamma = \{J_j : w_{\sigma[j]} > w_{\sigma[y]}, j = y+1, \cdots, k\}$, and we have the following two cases.

Case 2.1. $\Gamma = \{J_{\sigma[k]}\}$.

If $\Gamma = \{J_{\sigma[k]}\}$, then we have $WC_{\text{on}} = w_{\sigma[k]}(S_{\sigma[y]} + 2)$.

When job $J_{\sigma[y]}$ is scheduled before job $J_{\sigma[k]}$ in an optimal schedule π, we have $WC_{\text{opt}} \ge w_{\sigma[k]}(r_{\sigma[y]} + 2)$. If $r_{\sigma[y]} = S_{\sigma[y]}(\sigma)$, it is clear that σ is the optimal schedule. If $r_{\sigma[y]} \ne S_{\sigma[y]}(\sigma)$, because $S_{\sigma[y]}(\sigma) < (1+\beta)^2 - 1$, then from Observation 2 we have $S_{\sigma[y]}(\sigma) \le r_{\sigma[y]} + \beta$. Thus,

$$\frac{WC_{\text{on}}}{WC_{\text{opt}}} \le \frac{S_{\sigma[y]}(\sigma) + 2}{r_{\sigma[y]} + 2} \le \frac{r_{\sigma[y]} + \beta + 2}{r_{\sigma[y]} + 2} \le 1 + \frac{\beta}{r_{\sigma[y]} + 2} < 1 + \beta.$$

When job $J_{\sigma[k]}$ is scheduled before job $J_{\sigma[y]}$ in an optimal schedule π, we have $WC_{\text{opt}} \ge w_{\sigma[y]}(r_{\sigma[k]} + 2)$. Note that $r_{\sigma[k]} > S_{\sigma[y]}(\sigma)$ and $w_{\sigma[k]} \le (1+\beta)w_{\sigma[y]}$, then we have

$$\frac{WC_{\text{on}}}{WC_{\text{opt}}} \le \frac{w_{\sigma[k]}(S_{\sigma[y]}(\sigma) + 2)}{w_{\sigma[y]}(r_{\sigma[k]} + 2)} < 1 + \beta.$$

Case 2.2. $\Gamma \ne \{J_{\sigma[k]}\}$.

If $\Gamma \ne \{J_{\sigma[k]}\}$, in this situation, there exists at least one job that satisfies $w_{\sigma[j]} \ge w_{\sigma[k]}$ for each $j = y+1, \cdots, k-1$. Moreover, at least two jobs have arrival times that are larger than β, so we have

$$WC_{\text{opt}} \ge w_{\sigma[k]}(\beta + 2) \Rightarrow \beta WC_{\text{opt}} \ge w_{\sigma[k]},$$

and

$$WC_{\text{on}} = w_{\sigma[k]}C_{\sigma[k]}(\sigma) = w_{\sigma[k]}(S_{\sigma[y]}(\sigma) + p_{\sigma[y]} + \sum_{j=y+1}^{k} p_{\sigma[j]}) \leq (1+\beta)WC_{\text{opt}}.$$

From the above discussion, we complete the proof of Theorem 3.

Next, we present a numerical example to demonstrate the working of Algorithm $\mathcal{A}$. The details of this example can be found in Table 1.

Table 1. Example for problem $1|r_j, online, p_j = 1, 1\text{-}restart|WC_{\max}$.

Jobs (J_j)	J_1	J_2	J_3
Release time (r_j)	0	1	3
Weights (w_j)	6	10	20

Note that algorithm $\mathcal{A}$ will process job J_1 at time β. When job J_1 is being processed, job J_2 arrives at time 1 and $w_2 > (1+\beta)w_1$, then job J_2 will interrupt the processing of job J_1 and starts at time $(1+\beta)^2-1$. Thus, job J_1 will become a restarted job. After the completion of job J_2, the machine is idle and only job J_1 is currently available, so algorithm $\mathcal{A}$ will process job J_1 at time $(1+\beta)^2$, and process job J_3 at time $(1+\beta)^2+1$. We then deduce that the objective value of algorithm $\mathcal{A}$ is $w_3C_3 = 20\times[(1+\beta)^2+2] \approx 83$. The optimal schedule is to process job J_1 at time 0, then process job J_2 and job J_3 in turn. Thus, the optimal objective value is $w_3(r_3+1) = 20\times 4 = 80$, and the competitive ratio is $\frac{83}{80} \approx 1.0375 < 1+\beta$.

4. Conclusions

In this paper, we have shown that allowing all jobs to be restarted only once will not improve the competitive ratio of problem $1|r_j, online|WC_{\max}$. However, when all jobs are of equal length, the competitive ratio can be revised from $\frac{1+\sqrt{5}}{2} \approx 1.618$ to $1+\beta \approx 1.4656$ by using "1-restart". In subsequent investigations, the problem with "k-restart"($k \geq 2$) and the jobs with the same processing time is still open. As proposed by Fu et al. [8] and Nouinou et al. [15], the same problem with "limited restart" or "semi-online" is worthy of further research. It is also interesting to consider the online scheduling problem with other kinds of objectives.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

We are grateful to the editor and three anonymous reviewers for their constructive comments and helpful suggestions.

This work was supported by the National Natural Science Foundation of China under Grant Numbers 12271491 and 11971443.

Conflict of interest

The authors declare that they have no competing interests.

References

1. M. V. D. Akker, H. Hoogeveen, N. Vakhania, Restarts can help in the on-line minimization of the maximum delivery time on a single machine, *J. Scheduling*, **3** (2000), 333–341. https://doi.org/10.1002/1099-1425(200011/12)3:6<333::AID-JOS53>3.0.CO;2-8
2. H. Hoogeveen, C. N. Potts, G. J. Woeginger, On-line scheduling on a single machine: Maximizing the number of early jobs, *Oper. Res. Lett.*, **27** (2000), 193–197. https://doi.org/10.1016/S0167-6377(00)00061-4
3. R. Y. Fu, J. Tian, J. J. Yuan, C. He, On-line scheduling on a batch machine to minimize makespan with limited restarts, *Oper. Res. Lett.*, **36** (2008), 255–258. https://doi.org/10.1016/j.orl.2007.07.001
4. H. L. Liu, J. J. Yuan, Online scheduling of equal length jobs on a bounded parallel batch machine with restart or limited restart, *Theor. Comput. Sci.*, **543** (2014), 24–36. https://doi.org/10.1016/j.tcs.2014.05.021
5. H. L. Liu, X. W. Lu, Online scheduling on a parallel batch machine with delivery times and limited restarts, *J. Oper. Res. Soc. China*, **10** (2022), 113–131. https://doi.org/10.1007/s40305-021-00356-7
6. R. Y. Fu, T. Ji, J. J. Yuan, Y. X. Lin, Online scheduling in a parallel batch processing system to minimize makespan using restarts, *Theor. Comput. Sci.*, **374** (2007), 196–202. https://doi.org/10.1016/j.tcs.2006.12.040
7. J. J. Yuan, R. Y. Fu, C. T. Ng, T. C. E. Cheng, A best online algorithm for unbounded parallel-batch scheduling with restarts to minimize makespan, *J. Scheduling*, **14** (2011), 361–369. https://doi.org/10.1007/s10951-010-0172-2
8. R. Y. Fu, T. C. E. Cheng, C. T. Ng, J. J. Yuan, Online scheduling on two parallel-batching machines with limited restarts to minimize the makespan, *Inform. Process. Lett.*, **110** (2010), 444–450. https://doi.org/10.1016/j.ipl.2010.04.008
9. J. Tian, R. Y. Fu, J. J. Yuan, Online over time scheduling on parallel-batch machines: A survey, *J. Oper. Res. Soc. China*, **2** (2014), 445–454. https://doi.org/10.1007/s40305-014-0060-0
10. W. J. Li, A best possible online algorithm for the parallel-machine scheduling to minimize the maximum weighted completion time, *Asia-Pac. J. Oper. Res.*, **32** (2015), 1550030. https://doi.org/10.1142/S021759591550030X
11. X. Chai, L. F. Lu, W. H. Li, L. Q. Zhang, Best-possible online algorithms for single machine scheduling to minimize the maximum weighted completion time, *Asia-Pac. J. Oper. Res.*, **35** (2018), 1850048. https://doi.org/10.1142/S0217595918500483
12. L. F. Lu, L. Q. Zhang, J. W. Ou, *Single machine scheduling with rejection to minimize the weighted makespan*, In: Algorithmic Aspects in Information and Management, AAIM 2021, Lecture Notes in Computer Science, Springer, Cham, **13153** (2021), 96–110. https://doi.org/10.1007/978-3-030-93176-6_9

13. W. H. Li, X. Chai, Online scheduling on bounded batch machines to minimize the maximum weighted completion time, *J. Oper. Res. Soc. China*, **6** (2018), 455–465. https://doi.org/10.1007/s40305-017-0179-X

14. R. Q. Sun, *On the parameterized tractability of single machine scheduling with rejection to minimize the weighted makespan*, In: Theoretical Computer Science, NCTCS 2022, Communications in Computer and Information Science, Springer, Singapore, **1693** (2022), 236–247. https://doi.org/10.1007/978-981-19-8152-4_17

15. H. Nouinou, T. Arbaoui, A. Yalaoui, Minimising total weighted completion time for semi-online single machine scheduling with known arrivals and bounded processing times, *Int. J. Prod. Res.*, 2023, 1–14. https://doi.org/10.1080/00207543.2023.2217294

AIMS Press