Journal of Industrial and Management Optimization

https://www.aimspress.com/journal/jimo

JIMO, 22(8): 4138–4150.
DOI:10.3934/jimo.2026145
Received: 08 May 2026
Revised: 24 June 2026
Accepted: 13 July 2026
Published: 29 July 2026

Theory article

On-line single-machine scheduling to minimize the weighted makespan under (dis)agreeability conditions

Bingbing Fan, Lingfa Lu* and Zhan Shi

School of Mathematics and Statistics, Zhengzhou University, Zhengzhou, Henan 450001, P.R. China

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

Abstract: In this paper, we considered the on-line single-machine scheduling with the objective of minimizing the weighted makespan. For the general case, a lower bound of 2 and two on-line algorithms with the best-possible competitive ratio of 2 were provided in the literature. We considered eight special cases in which all jobs had the agreeability or disagreeability condition on jobs' parameters. For each special case, we provided a lower bound and a corresponding on-line algorithm with the best-possible competitive ratio which matched the given lower bound.

Keywords: on-line scheduling; single-machine scheduling; weighted makespan; (dis)agreeability conditions; competitive ratio
Mathematics Subject Classification: 90B35

1. Problem Formulation

The on-line single-machine scheduling problem to minimize the weighted makespan can be stated as follows: There are n jobs $J_1, J_2, \cdots, J_n$ which are processed on a single machine. Each job J_j is associated with an arrival time r_j, a processing time p_j, and a weight w_j. That is, all jobs arrive over time, and job J_j and its parameters r_j, p_j, and w_j become known only at or after time r_j. Given a feasible schedule, let C_j be the completion time of J_j in this schedule. The weighted makespan (the maximum weighted completion time) is defined by $WC_{\max} = \max\{w_jC_j : j = 1, 2, \cdots, n\}$. Using the notation for scheduling problems, the general on-line single-machine scheduling problem is denoted by $1|r_j, \text{on-line}|WC_{\max}$. For problem $1|r_j, \text{on-line}|WC_{\max}$, Li [1] presented a lower bound of 2 and Chai et al. [2] proposed two 2-competitive on-line algorithms, respectively. Moreover, Li [1] also pointed out that objective function $WC_{\max}$ can be applied in some service systems, where each customer will hope that their service costs (w_jC_j) as small as possible. In this situation, $WC_{\max}$ can avoid the excessively delayed completion time for each job.

Different from Li [1] and Chai et al. [2], we study some similar on-line single-machine scheduling problems under several agreeability or disagreeability assumptions on jobs' parameters. In practice, there are many applications on the agreeability or disagreeability assumptions. For example, in the screen manufacturing of mobile phones or computers, the larger the screen size, the greater the screen cost. In this situation, the size of a screen and its cost satisfies the agreeability assumption. However, in the chip manufacturing, the smaller the chip size, the greater the chip cost. In this situation, the size of a chip and its cost satisfies the disagreeability assumption. In theory, if some agreeability or disagreeability assumptions are satisfied, the difficulty of the studied problems might be decreased so that we can design some approximation (on-line) algorithms with a smaller approximation (competitive) ratio.

More precisely, we say that the jobs have agreeable processing times and weights if $w_i < w_j$ implies $p_i \leq p_j$ for any two jobs J_i and J_j. This assumption for processing times and weights is also called the (p_j, w_j)-agreeability condition, denoted by "(p_j, w_j)-agreeable" throughout this paper. Furthermore, all jobs have disagreeable processing times and weights if $w_i > w_j$ implies $p_i \geq p_j$ for any two jobs J_i and J_j. The corresponding disagreeability condition is denoted by "(p_j, w_j)-disagreeable". Similarly, we can also define other four agreeable and disagreeable conditions: "(r_j, w_j)-agreeable", "(r_j, p_j)-agreeable", "(r_j, w_j)-disagreeable", and "(r_j, p_j)-disagreeable" throughout this paper. Using the notation for scheduling problems, the corresponding on-line scheduling problems under consideration are denoted by

$$1|r_j, \text{on-line}, (p_j, w_j)\text{-agreeable}|WC_{\max}, \tag{1.1}$$

$$1|r_j, \text{on-line}, (p_j, w_j)\text{-disagreeable}|WC_{\max}, \tag{1.2}$$

$$1|r_j, \text{on-line}, (r_j, w_j)\text{-agreeable}|WC_{\max}, \tag{1.3}$$

$$1|r_j, \text{on-line}, (r_j, w_j)\text{-disagreeable}|WC_{\max}, \tag{1.4}$$

$$1|r_j, \text{on-line}, (r_j, p_j)\text{-agreeable}|WC_{\max}, \tag{1.5}$$

and

$$1|r_j, \text{on-line}, (r_j, p_j)\text{-disagreeable}|WC_{\max}, \tag{1.6}$$

respectively.

In addition, we also consider two more special problems (1.7) and (1.8). In problem (1.7), we assume that $w_j = up_j$ holds for each job J_j, where $u > 0$ is a fixed number. That is, the weight w_j of each job J_j is proportional to its processing time p_j. Thus, problem (1.7) can be denoted by

$$1|r_j, \text{on-line}, w_j = up_j|WC_{\max}. \tag{1.7}$$

In problem (1.8), we assume that $w_j p_j = v$ holds for each job J_j, where $v > 0$ is also a fixed number. That is, the weight w_j of each job J_j is inversely proportional to its processing time p_j. Thus, problem (1.8) can be denoted by

$$1|r_j, \text{on-line}, w_j p_j = v|WC_{\max}. \tag{1.8}$$

Clearly, problems (1.7) and (1.8) are the special cases of problems (1.1) and (1.2), respectively.

The rest of this paper is organized as follows. We introduce some related literature in Section 2. In Section 3, for each scheduling problem among problems (1.1–1.8), we analyze its lower bound on any deterministic on-line algorithm and design an on-line algorithm with the best-possible competitive ratio. Finally, Section 4 concludes the paper and discusses future research directions.

2. Literature review

Over the past three decades, on-line scheduling has been widely studied in the literature. In on-line scheduling, no information about future jobs is known in advance. Two different on-line settings are commonly considered in the literature: on-line over-time setting (see Amouzandeh et al. [3]) and on-line over-list setting (see Braun et al. [4]). More recent articles on the on-line scheduling can also been found in [5–10]. In this paper, we adopt the on-line-over-time setting. That is, the characteristics of each job, including its arrival time, processing time, and weight, become known only at or after the job arrives.

We consider the on-line single-machine scheduling problem to minimize the weighted makespan $WC_{\max}$, which is also defined as the maximum weighted completion time in some literature. Feng and Yuan [11] were the first to introduce this objective function. They studied a two-agent single-machine scheduling problem in which the objective for the first agent is to minimize the weighted makespan $WC_{\max}$, while the task for the other agent is to minimize the total weighted completion time $\sum_{j=1}^{n} w_j C_j$. They showed that the problem is strongly NP-hard. Nong et al. [12] further investigated this model and proposed the corresponding approximation algorithms. Since then, the study of $WC_{\max}$ has gradually extended from the off-line complexity and approximation algorithms to the on-line models.

For the on-line single-machine scheduling problem, denoted by $1|r_j, \text{on-line}|WC_{\max}$, Li [1] presented a lower bound of 2 and proposed a 3-competitive on-line algorithm; for the on-line parallel-machine scheduling problem $P|r_j, p_j = p, \text{on-line}|WC_{\max}$, he developed an on-line algorithm with the best-possible competitive ratio of $\frac{1+\sqrt{5}}{2} \approx 1.618$. Furthermore, for the single-machine scheduling problem $1|r_j, \text{on-line}|WC_{\max}$, Chai et al. [2] closed the gap by presenting two 2-competitive on-line algorithms based on the different ideas. Furthermore, the weighted makespan $WC_{\max}$ has also been studied in more complex machine environments. For the bounded parallel-batch machines scheduling problem $P|r_j, p_j = p, \text{on-line, p-batch}, b < n|WC_{\max}$, Li and Chai [13] obtained an on-line algorithm with the best-possible competitive ratio of $\frac{1+\sqrt{5}}{2}$, and also derived a best-possible dense algorithm with the competitive ratio of 2. More recently, Zhang et al. [9] studied the on-line scheduling problem $1|r_j, \text{on-line, p-batch}, b \geq n|WC_{\max}$. They presented a lower bound of 2 and designed a 4-competitive on-line algorithm for the above problem. For the special case in which the jobs' processing times and weights are agreeable, they further obtained an on-line algorithm with the best-possible competitive ratio of $\frac{1+\sqrt{5}}{2}$; when limited restart is allowed, they show that the competitive ratio $\frac{1+\sqrt{5}}{2}$ can be improved to $\frac{11}{7}$. In addition, there have been several articles on $WC_{\max}$ with additional constraints. For example, Li and Yuan [14] studied the on-line single-machine scheduling problem with the non-delayed processing (NDP) constraint; Li and Liu [15] further investigated on-line NDP-constraint scheduling with delivery times. On the other hand, Lu et al. [16] studied the single-machine scheduling problem with rejection, denoted by $1||WC_{\max} + \sum_{J_j \in R} e_j$. They showed that this problem is binary NP-hard and provided a pseudo-polynomial algorithm and a fully

polynomial-time approximation scheme (FPTAS for short). Zheng et al. [17] also considered a single-machine lot scheduling model with the objective of minimizing the weighted makespan $WC_{\max}$.

3. Main results

3.1. Problems (1.2), (1.3), (1.6), and (1.8)

In this subsection, we first consider the special problems (1.2), (1.3), (1.6) and (1.8). Note that, for the more general problem $1|r_j, \text{on-line}|WC_{\max}$, Li [1] proved that there is no deterministic on-line algorithm which has a competitive ratio of less than 2. For completeness, the corresponding lemma and its proof in [1] is provided as follow.

Lemma 3.1. ([1]) For problem $1|r_j, \text{on-line}|WC_{\max}$, there is no deterministic on-line algorithm which has a competitive ratio of less than 2.

Proof. Consider any deterministic on-line algorithm A and the following instance presented by the adversary. Let σ denote the schedule produced by Algorithm A, and let π denote an optimal off-line schedule. Furthermore, we also assume that WC_{on} and WC_{opt} are the objective values corresponding to σ and π, respectively. At time 0, a job J_1 with $p_1 = v$ and $w_1 = 1$ arrives, where v is a sufficiently large positive integer. Assume that algorithm A starts J_1 at time S_1. If $S_1 \geq v - 1$, then no other jobs will arrive, and so $\frac{WC_{on}}{WC_{opt}} \geq \frac{1\times(2v-1)}{1\times v} \to 2$ when $v \to +\infty$.

Suppose that $S_1 < v - 1$. Then, a job J_2 with $p_2 = 1$ and $w_2 = v$ arrives at time $S_1 + \epsilon$, where $\epsilon = \frac{1}{v-1}$ is a sufficiently small positive number. Note that, from algorithm A, J_2 must be processed after J_1 in σ. Thus, we have $WC_{on} \geq w_2C_2(\sigma) \geq v(S_1 + v + 1)$. However, J_2 is processed before J_1 in the optimal schedule π. Thus, we also have $WC_{opt} = w_2C_2(\pi) = v(S_1 + \epsilon + 1)$ since $w_2C_2(\pi) = v(S_1 + \epsilon + 1) \geq 1 \cdot (S_1 + \epsilon + v + 1) = w_1C_1(\pi)$. Note that $S_1 < v - 1$. Hence, we have $\frac{WC_{on}}{WC_{opt}} \geq \frac{v(S_1+v+1)}{v(S_1+1+\epsilon)} \to \frac{v(S_1+v+1)}{v(S_1+1)} = 1 + \frac{v}{S_1+1} \geq 2$ when $\epsilon \to 0^+$. The result follows.

Note that, in the proof of Lemma 3.1, two jobs J_1 and J_2 satisfy $r_1 < r_2$, $p_1 > p_2$, and $w_1 < w_2$. Thus, we have the following corollary.

Corollary 3.2. For problems (1.2), (1.3), and (1.6), there is no deterministic on-line algorithm which has a competitive ratio of less than 2.

Note further that, in the proof of Lemma 3.1, $w_1p_1 = w_2p_2 = v$. Thus, we also have the following corollary.

Corollary 3.3. For problem (1.8), there is no on-line deterministic algorithm which has a competitive ratio of less than 2.

Chai et al. [2] proposed two on-line algorithms for problem $1|r_j, \text{on-line}|WC_{\max}$ with the best-possible competitive ratio 2. Thus, the on-line algorithms in Chai et al. [2] are also best-possible for problems (1.2), (1.3), (1.6), and (1.8).

3.2. Problems (1.1) and (1.5)

Let $\alpha = \frac{\sqrt{5}-1}{2} \approx 0.618$ be the positive root of $\alpha(1+\alpha) = 1$. For the problem $1|r_j, p_j = p, \text{on-line}|WC_{\max}$, Li [1] showed that, for any deterministic on-line algorithm A, its competitive ratio is at least $1 + \alpha = \frac{\sqrt{5}+1}{2}$. Note that, problem $1|r_j, p_j = p, \text{on-line}|WC_{\max}$ is a special case of problems (1.1) and (1.5). Thus, we obtain the following theorem.

Theorem 3.4. For problems (1.1) and (1.5), there is no deterministic on-line algorithm which has a competitive ratio of less than $1 + \alpha$.

Next, we will propose an on-line algorithm A_1 for problems (1.1) and (1.5) and show that its competitive ratio is exactly $1 + \alpha$. Let S be a set of jobs. A job J_k is called a *key job* of S if $w_k = \max\{w_j : J_j \in S\}$. Note that, if there are multiple jobs in S which reach the largest weight, we assume that the key job J_k has the earliest arrival time among them. Based on the concept of key jobs, we design the following algorithm.

Algorithm A_1

Step 1: Set $t = 0$.

Step 2: Let $U(t)$ denote the set of available jobs which have arrived but not yet been processed on the machine. If $U(t) = \emptyset$, let t' be the arrival time of the next job if possible. Set $t = t'$, return to Step 2. If $U(t) \neq \emptyset$, let J_k be a key job in $U(t)$.

Step 2.1: If $t \geq \alpha p_k$ and the machine is idle at time t, then assign J_k to be processed non-preemptively on the machine. Set $t = t + p_k$, return to Step 2.

Step 2.2: If $t < \alpha p_k$, set $t = \min\{t', \alpha p_k\}$, return to Step 2. Here t' is the arrival time of the next job if possible.

Theorem 3.5. For problems (1.1) and (1.5), Algorithm A_1 has the best-possible competitive ratio of $1 + \alpha$.

Proof. Let σ denote the schedule produced by Algorithm A_1, and let π denote an optimal off-line schedule. Let WC_{on} and WC_{opt} be the objective values corresponding to σ and π, respectively. Let $C_j(\sigma)$ and $C_j(\pi)$ be the completion times of J_j in σ and π, respectively. Furthermore, we also assume that J_j denotes the j-th completed job in schedule σ, $j = 1, 2, \ldots, n$. Thus, we have $C_1(\sigma) \leq C_2(\sigma) \leq \cdots \leq C_n(\sigma)$. Let J_k be the critical job corresponding to the objective value WC_{on}. It follows that

$$WC_{on} = WC_{\max}(\sigma) = \max\{w_j C_j(\sigma) : j = 1, 2, \ldots, n\} = w_k C_k(\sigma).$$

In schedule σ, we assume that J_i is the job with the smallest index such that the jobs $J_i, \cdots, J_k$ are processed consecutively on the machine, where $1 \leq i \leq k$. Furthermore, we denote $S_i(\sigma)$ by the starting time of job J_i in schedule σ. Thus, we have

$$C_k(\sigma) = S_i(\sigma) + \sum_{j=i}^{k} p_j. \tag{1}$$

From the definition of J_i, the machine is idle before time $S_i(\sigma)$. Thus, we have $S_i(\sigma) = \max\{r_i, \alpha p_i\}$. Next, we distinguish two cases in the following discussion.

Case 1. $w_k = \min\{w_j : i \le j \le k\}$.

In this case, let J_x be the last completed job among the jobs $J_i, \cdots, J_k$ in the off-line optimal schedule π, where $i \le x \le k$. Clearly, we have $C_x(\pi) \ge \sum_{j=i}^{k} p_j$ and

$$WC_{opt} = WC_{\max}(\pi) \ge w_x C_x(\pi) \ge w_k \sum_{j=i}^{k} p_j. \tag{2}$$

Next, we prove the competitive ratio by considering the value of $S_i(\sigma)$.

Subcase 1.1. $S_i(\sigma) = r_i$.

If $r_i = \min\{r_j : i \le j \le k\}$, then $C_x(\pi) \ge r_i + \sum_{j=i}^{k} p_j$. Furthermore, we have

$$WC_{opt} \ge w_x C_x(\pi) \ge w_x \left(r_i + \sum_{j=i}^{k} p_j \right) \ge w_k \left(r_i + \sum_{j=i}^{k} p_j \right) = WC_{on}.$$

If $r_i \ne \min\{r_j : i \le j \le k\}$, then there exists a job J_h with $i \le h \le k$ such that $w_h = \max\{w_j : r_j < r_i \text{ and } i \le h \le k\}$. Note that $\{J_i, \cdots, J_k\}$ are processed consecutively in σ and the machine is idle before time r_i from the definition of J_i. Thus, we have $\alpha p_h \ge r_i$. It follows that $WC_{opt} \ge w_h p_h \ge w_k p_h$ since $w_k = \min\{w_j : i \le j \le k\} \le w_h$. From equations (1) and (2), we further have

$$\begin{aligned} WC_{on} &= w_k \left(r_i + \textstyle\sum_{j=i}^{k} p_j \right) \\ &\le w_k \left(\alpha p_h + \textstyle\sum_{j=i}^{k} p_j \right) \\ &\le \alpha w_k p_h + w_k \textstyle\sum_{j=i}^{k} p_j \\ &\le (1+\alpha) WC_{opt}. \end{aligned}$$

Subcase 1.2. $S_i(\sigma) = \alpha p_i$.

In this case, we have $WC_{on} = w_k C_k(\sigma) = w_k \left(\alpha p_i + \sum_{j=i}^{k} p_j \right)$. Moreover, we also have $WC_{opt} \ge w_i p_i \ge w_k p_i$ since $w_k = \min\{w_j : i \le j \le k\} \le w_i$. It follows that $WC_{on} = w_k \left(\alpha p_i + \sum_{j=i}^{k} p_j \right) \le (1+\alpha) WC_{opt}$.

Case 2. $w_k \ne \min\{w_j : i \le j \le k\}$.

In this case, let J_y with $i \le y \le k$ be the last job which satisfies $w_y < w_k$. By the definition of J_y, it follows that $w_j \ge w_k > w_y$ for each $j = y+1, \cdots, k$. Note that J_y starts its processing on the machine at time S_y in σ. By the property of Algorithm A_1, J_y must be a key job among all jobs that have already arrived by time S_y, i.e., $w_y = \max\{w_r : J_r \in U(S_y)\}$. Thus, for every job J_j in the set $\{J_{y+1}, \cdots, J_k\}$, we have $r_j > S_y$ since $w_j > w_y$, where $j = y+1, \cdots, k$.

Let J_m and J_z $(y+1 \le z \le k)$ be the first and the last processed jobs among the set $\{J_{y+1}, \cdots, J_k\}$ in the optimal schedule π, respectively. Thus, we have

$$C_z(\pi) \ge r_m + \sum_{j=y+1}^{k} p_j \ge S_y + \sum_{j=y+1}^{k} p_j.$$

It follows that

$$WC_{opt} \ge w_z C_z(\pi) \ge w_z \left(S_y + \sum_{j=y+1}^{k} p_j \right) \ge w_k \left(S_y + \sum_{j=y+1}^{k} p_j \right). \tag{3}$$

Note that $w_y < w_k$ and $r_y \le S_y < r_k$. From the (w_j, p_j)-agreeability condition in problem (1.1) or the (r_j, p_j)-agreeability condition in problem (1.5), we can conclude that $p_y \le p_k$. Note further that $C_k(\pi) \ge r_k + p_k$. Thus,

$$WC_{opt} \ge w_k(r_k + p_k) \ge w_k(S_y + p_k) \ge w_k(\alpha p_y + p_k) \ge w_k(\alpha p_y + p_y) = (1+\alpha) w_k p_y.$$

It follows that $w_k p_y \le \frac{1}{1+\alpha} WC_{opt} = \alpha WC_{opt}$. Finally, from inequality (3), we have

$$\begin{aligned} WC_{on} &= w_k \left(S_y + p_y + \sum_{j=y+1}^{k} p_j \right) \\ &= w_k p_y + w_k \left(S_y + \sum_{j=y+1}^{k} p_j \right) \\ &\le (1+\alpha) WC_{opt}. \end{aligned}$$

Combining the discussions of Case 1 and Case 2, we complete the proof of Theorem 3.5.

Remark. Note that, algorithm A_1 is a delayed largest-weight-first on-line algorithm which is very similar to the known algorithms proposed by Li [1] and Chai et al. [2]. In algorithm A_1, we choose a new threshold value $\alpha = \frac{\sqrt{5}-1}{2}$ and use the agreeability assumptions in the proof of Theorem 3.5.

3.3. Problem (1.4)

In this subsection, we consider problem (1.4), namely, $1|r_j, \text{on-line}, (r_j, w_j)\text{-disagreeable}|WC_{\max}$. Let us first introduce the following algorithm.

Algorithm A_2

All jobs are scheduled on the machine according to the LW (Largest-Weight first) rule. That is, whenever a job is completed or a new job arrives at time t, among all jobs that have been released by time t but are not yet completed, the job with the largest weight is selected for processing. All jobs are processed non-preemptively on the machine, and if there are multiple jobs which reach the largest weight, then we select the job with the earliest arrival time.

Theorem 3.6. For problem (1.4), namely, $1|r_j,$ on-line, (r_j, w_j)-disagreeable$|WC_{\max}$, Algorithm A_2 yields an optimal schedule.

Proof. Let π be an off-line optimal schedule for problem (1.4). Without loss of generality, we assume that J_j is the j-th processed job in π, where $j = 1, \cdots, n$. Furthermore, let $S_j(\pi)$ and $C_j(\pi)$ be the starting time and the completion time of J_j, respectively. If the processing order in π does not coincide with the LW rule, then there must exist two jobs J_k and J_{k+1} in π such that J_k is processed before J_{k+1} with $w_k < w_{k+1}$. From the (r_j, w_j)-disagreeability condition and the fact that $w_k < w_{k+1}$, we can conclude that $r_{k+1} \le r_k \le S_k(\pi)$. That is, both J_k and J_{k+1} are available at time $S_k(\pi)$.

Now, we construct a new schedule π' from π by exchanging the processing order of J_k and J_{k+1}: schedule J_{k+1} for the first at time $S_k(\pi)$, and then process J_k immediately after J_{k+1}. It is easy to see that $C_{k+1}(\pi') \le C_k(\pi') = S_k(\pi) + p_{k+1} + p_k \le C_{k+1}(\pi)$. Furthermore, for every job J_j with $j \ne k, k+1$, we have $C_j(\pi') \le C_j(\pi)$. It follows that

$$w_{k+1}C_{k+1}(\pi') \le w_{k+1}C_{k+1}(\pi) \le WC_{\max}(\pi)$$

and

$$w_kC_k(\pi') \le w_kC_{k+1}(\pi) < w_{k+1}C_{k+1}(\pi) \le WC_{\max}(\pi).$$

Moreover, for all $j \ne k, k+1$, we also have

$$w_jC_j(\pi') \le w_jC_j(\pi) \le WC_{\max}(\pi).$$

Note that π' is a feasible schedule and π' satisfies $WC_{\max}(\pi') \le WC_{\max}(\pi)$. This implies that π' is also an optimal schedule for the problem (1.4). Repeating the above exchange argument a finite number of times, we finally obtain an optimal schedule of the required form, namely, an optimal schedule that coincides the LW rule. This completes the proof of Theorem 3.6.

Remark. Note that, in problem (1.4), we have $r_i < r_j$ implies $w_i \ge w_j$. That is, any two jobs have satisfied the LW rule when they have the distinct arrival times. Thus, in on-line algorithm A_2, we need only to sort those jobs in the LW rule when they arrive at the same time. Assume that there are d distinct arrival times $t_1, t_2, \cdots, t_d$ and there are $n_1, n_2, \cdots, n_d$ jobs which arrive at $t_1, t_2, \cdots, t_d$, respectively. It takes an $O(n_i \log n_i)$ time to sort the jobs arriving at t_i, $i = 1, 2, \cdots, d$. Note further that

$$n_1 \log n_1 + n_2 \log n_2 + \cdots + n_d \log n_d \le n_1 \log n + n_2 \log n + \cdots + n_d \log n = n \log n.$$

Thus, the time complexity of algorithm A_2 is $O(n \log n)$.

3.4. Problem (1.7)

In this subsection, we consider problem (1.7), namely $1|r_j, \text{on-line}, w_j = up_j|WC_{\max}$. For this problem, we present a lower bound of the competitive ratio of $\frac{3}{2}$ and design an on-line algorithm with the best-possible competitive ratio of $\frac{3}{2}$.

Theorem 3.7. For problem $1|r_j, \text{on-line}, w_j = up_j|WC_{\max}$, there is no deterministic on-line algorithm which has a competitive ratio of less than $\frac{3}{2}$.

Proof. Consider any deterministic on-line algorithm A and the following instance presented by the adversary. At time 0, a job J_1 with $p_1 = 1$ and $w_1 = u$ arrives. Assume that algorithm A starts J_1 at time S_1. If $S_1 \geq \frac{1}{2}$, then no other jobs will arrive. In this case, we have $WC_{on} = w_1C_1 = w_1(S_1 + p_1) \geq u(\frac{1}{2} + 1) = \frac{3}{2}u$. However, in the optimal schedule, J_1 starts its processing at time 0. Thus, we have $WC_{opt} = u \cdot (0 + 1) = u$. It follows that $\frac{WC_{on}}{WC_{opt}} \geq \frac{\frac{3}{2}u}{u} = \frac{3}{2}$.

If $S_1 < \frac{1}{2}$, then a job J_2 with $p_2 = \frac{3}{2}$ and $w_2 = \frac{3}{2}u$ arrives at time $S_1 + \epsilon$, where $\epsilon \to 0^+$. In this case, J_2 must be processed after J_1 from algorithm A. Thus, we have $WC_{on} \geq w_2C_2 \geq \frac{3}{2}u \cdot (S_1 + 1 + \frac{3}{2}) = \frac{3}{2}(S_1 + \frac{5}{2})u$. However, in the optimal schedule, J_2 is scheduled before J_1. Thus, we have $WC_{opt} = \max\{u \cdot (S_1 + \epsilon + \frac{5}{2}), \frac{3}{2}u \cdot (S_1 + \epsilon + \frac{3}{2})\}$. Note that

$$u \cdot (S_1 + \epsilon + \frac{5}{2}) - \frac{3}{2}u \cdot (S_1 + \epsilon + \frac{3}{2}) = u \cdot (\frac{1}{4} - \frac{1}{2}S_1 - \frac{1}{2}\epsilon) \geq u \cdot (\frac{1}{4} - \frac{1}{2} \times \frac{1}{2}) = 0.$$

Thus, we have $WC_{opt} = u \cdot (S_1 + \epsilon + \frac{5}{2})$. It follows that $\frac{WC_{on}}{WC_{opt}} \geq \frac{\frac{3}{2}u \cdot (S_1 + \frac{5}{2})}{u \cdot (S_1 + \epsilon + \frac{5}{2})} \to \frac{3}{2}$ when $\epsilon \to 0^+$. This completes the proof of Theorem 3.7.

Next, we will design an on-line algorithm A_3 for problem $1|r_j, \text{on-line}, w_j = up_j|WC_{\max}$ and analyze its competitive ratio.

Algorithm A_3

Step 1: Set $t = 0$.

Step 2: Let $U(t)$ denote the set of available jobs which have arrived but not yet been processed on the machine. If $U(t) = \emptyset$, let t' be the arrival time of the next job if possible. Set $t = t'$, return to Step 2; If $U(t) \neq \emptyset$, let J_k be a key job in $U(t)$.

Step 2.1: If $t \geq \frac{1}{2}p_k$ and the machine is idle at time t, then assign J_k to be processed non-preemptively on the machine. Set $t = t + p_k$, return to Step 2.

Step 2.2: If $t < \frac{1}{2}p_k$, let t' be the arrival time of the next job if possible. Set $t = \min\{t', \frac{1}{2}p_k\}$, return to Step 2.

Theorem 3.8. For problem $1|r_j, \text{on-line}, w_j = up_j|WC_{\max}$, Algorithm A_3 has the best-possible competitive ratio of $\frac{3}{2}$.

Proof. Let σ and π be the schedule produced by Algorithm A_3 and an optimal off-line schedule, respectively. Furthermore, let WC_{on} and WC_{opt} be the objective values corresponding to σ and π, respectively. We assume that J_j denotes the j-th completed job in schedule σ, $j = 1, 2, \ldots, n$. Let $C_j(\sigma)$ be the completion time of J_j in σ. Let J_k be the critical job corresponding to the objective value WC_{on}. That is,

$$WC_{on} = WC_{\max}(\sigma) = \max\{w_jC_j(\sigma) : j = 1, 2, \ldots, n\} = w_kC_k(\sigma).$$

In schedule σ, let i be the smallest index such that the jobs $J_i, \cdots, J_k$ are processed consecutively on the machine, where $1 \leq i \leq k$. Furthermore, we denote $S_i(\sigma)$ by the starting time of job J_i in schedule

σ. Thus, we have

$$WC_{on} = w_k C_k(\sigma) = w_k(S_i(\sigma) + \sum_{j=i}^{k} p_j).$$

From the definition of J_i, we have $S_i(\sigma) = \max\{r_i, \frac{1}{2}p_i\}$. Next, we distinguish two cases in the following discussion.

Case 1. $w_k = \min\{w_j : i \le j \le k\}$.

In this case, whether $S_i = r_i$ or $S_i = \frac{1}{2}p_i$, we can show that $WC_{on} \le \frac{3}{2}WC_{opt}$ by taking $\alpha = \frac{1}{2}$. The proof is very similar to that in Case 1 of Theorem 3.5. Thus, we omit the detailed procedure.

Case 2. $w_k \ne \min\{w_j : i \le j \le k\}$.

In this case, let J_y with $i \le y \le k$ be the last job which satisfies $w_y < w_k$. By the definition of J_y, it follows that $w_j \ge w_k > w_y$ for each $j = y+1, \cdots, k$. Note that J_y starts its processing on the machine at time S_y in σ. By the property of Algorithm A_3, J_y must be a key job among all jobs that have already arrived by time S_y, i.e., $w_y = \max\{w_r : J_r \in U(S_y)\}$. That is, for every job J_j in the set $\{J_{y+1}, \cdots, J_k\}$, we have $r_j > S_y$ since $w_j > w_y$, where $j = y+1, \cdots, k$. Similar to the proof for Theorem 3.5, we can conclude that

$$WC_{opt} \ge w_k\left(S_y + \sum_{j=y+1}^{k} p_j\right). \tag{4}$$

Set $M = \{J_{y+1}, \cdots, J_k\}$ and $|M| = k - y$, where $|M|$ is the number of the jobs in set M. Since $J_k \in M$, we have $|M| \ge 1$. According to the value of $|M|$, we distinguish two subcases in our discussion.

Subcase 2.1. $|M| \ge 2$.

According to the definition of M, for all jobs $J_j \in M$, we have $r_j > S_y$ and $w_j > w_y$. Note that $w_j = up_j$ and $w_y = up_y$, and we have $p_j > p_y$ for each job $J_j \in M$. Thus, we have $WC_{opt} \ge w_k(S_y + |M|p_y) \ge w_k(0 + 2p_y) = 2w_k p_y$. Note that $WC_{on} = w_k(S_y + p_y + \sum_{j=y+1}^{k} p_j)$. Using (4), we further have

$$WC_{on} = w_k p_y + w_k(S_y + \sum_{j=y+1}^{k} p_j) \le \frac{1}{2}WC_{opt} + WC_{opt} = \frac{3}{2}WC_{opt}.$$

Subcase 2.2. $|M| = 1$.

In this case, since $|M| = 1$, we have $M = \{J_{y+1}, \cdots, J_k\} = J_k$. Thus, we have $WC_{on} = w_k(S_y + p_y + p_k)$ from (4).

If $\frac{w_k}{w_y} \ge \frac{3}{2}$, since $w_k = up_k$ and $w_y = up_y$, we have $\frac{p_k}{p_y} \ge \frac{3}{2}$. Note that $WC_{opt} \ge w_k(r_k + p_k)$. Thus, we have

$$\frac{WC_{on}}{WC_{opt}} \leq \frac{w_k(S_y + p_y + p_k)}{w_k(r_k + p_k)} < \frac{S_y + p_y + p_k}{S_y + p_k} = 1 + \frac{p_y}{S_y + p_k} \leq 1 + \frac{p_y}{\frac{1}{2}p_y + p_k} = 1 + \frac{1}{\frac{1}{2} + \frac{p_k}{p_y}} \leq \frac{3}{2}.$$

Next, we assume that $\frac{w_k}{w_y} < \frac{3}{2}$. If J_k starts its processing before J_y in the optimal schedule π, then we have $WC_{opt} \geq w_y(r_k + p_k + p_y) \geq w_y(S_y + p_k + p_y)$. It follows that

$$\frac{WC_{on}}{WC_{opt}} \leq \frac{w_k(S_y + p_y + p_k)}{w_y(S_y + p_k + p_y)} = \frac{w_k}{w_y} < \frac{3}{2}.$$

Note that $w_k > w_y$. From the fact that $w_k = up_k$ and $w_y = up_y$, we also have $p_k > p_y$. If J_y starts its processing before J_k in the optimal schedule π, then we have

$$WC_{opt} \geq w_k(r_y + p_y + p_k) \geq w_k(0 + p_y + p_y) = 2w_k p_y.$$

Furthermore, we have $w_k p_y \leq \frac{1}{2} WC_{opt}$. It follows that

$$\begin{aligned} WC_{on} &= w_k(S_y + p_y + p_k) \\ &\leq w_k p_y + w_k(S_y + p_k) \\ &\leq \frac{1}{2} WC_{opt} + WC_{opt} \\ &= \frac{3}{2} WC_{opt}. \end{aligned}$$

Combining the discussion of Case 1 and Case 2, we complete the proof of Theorem 3.8.

4. Conclusions and future research

This paper studies on-line-over-time scheduling problems on a single machine under agreeability and disagreeability assumptions on jobs' parameters, with the objective of minimizing the weighted makespan. We show that problems (1.2), (1.3), (1.6), and (1.8) have the best-possible competitive ratio of 2; problems (1.1) and (1.5) have the best-possible competitive ratio of $\frac{\sqrt{5}+1}{2}$; problem (1.4) can be solved by an optimal schedule; and problem (1.7) has the best-possible competitive ratio of $\frac{3}{2}$.

For future research, the first interesting direction is to study the problem on $m > 1$ parallel machines under agreeability and disagreeability assumptions. For the special case with $p_j = p$, Li [1] gave an on-line algorithm with the best-possible competitive ratio $\frac{\sqrt{5}+1}{2}$. However, for the more general case with the agreeable processing times and weights, the lower and upper bounds are still open. A second interesting on-line problem is to introduce *restarts* into the model as that in [18]. By allowing restarts, it may be possible to obtain an on-line algorithm with a smaller competitive ratio.

Author contributions

Bingbing Fan contributed to writing original draft and methodology. Lingfa Lu was involved in conceptualization, supervision, and funding acquisition. Zhan Shi contributed to writing and validation. All authors have read and approved the final version of the manuscript.

Use of Generative-AI tools declaration

No Artificial Intelligence (AI) tools were used in the creation of this article.

Acknowledgments

This research was supported in part by the National Natural Science Foundation of China under Grant Numbers 12271491 and 12471305.

Conflict of interest

The authors declare that they have no competing interests.

References

1. 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
2. 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
3. A. Amouzandeh, R. van Stee, Improved online scheduling with restarts on a single machine, *Theory Comput. Syst.*, **70** (2026), 10. https://doi.org/10.1007/s00224-026-10267-w
4. O. Braun, F. Chung, R. L. Graham, Lower bounds for online scheduling on four processors, *J. Sched.*, **28** (2025), 529–544. https://doi.org/10.1007/s10951-025-00843-2
5. Z. L. Chen, Online integrated production and distribution scheduling: review and extensions, *INFORMS J. Comput.*, **37** (2025), 360–380. https://doi.org/10.1287/ijoc.2022.0305
6. C. Escribe, M. Hu, R. Levi, Competitive algorithms for the online minimum peak job scheduling, *Oper. Res.*, **73** (2025), 408–423. https://doi.org/10.1287/opre.2021.0080
7. Y. J. Qu, J. Tian, R. Y. Fu, K. J. Ge, A best possible online algorithm for single-machine scheduling with non-delayed processing constraint and bounded delivery times, *J. Oper. Res. Soc. China*, **14** (2026), 489–501. https://doi.org/10.1007/s40305-024-00541-4
8. J. M. Yang, Z. Y. Tan, Online scheduling with rejection revisited, *Inform. and Comput.,* **310** (2026), 105442. https://doi.org/10.1016/j.ic.2026.105442
9. H. Zhang, L. F. Lu, J. J. Yuan, Online scheduling on an unbounded parallel-batch machine to minimize the weighted makespan, *J. Comb. Optim.,* **49** (2025), 6. https://doi.org/10.1007/s10878-024-01242-7
10. L. Q. Zhang, L. F. Lu, X. K. Sun, L. L. Zuo, On-Line Single Machine Scheduling of Unit Time Jobs with Rejection: Minimizing the Maximum Quadratic Completion Time, *Asia-Pacific J. Oper. Res.,* **42** (2025), 2550009. https://doi.org/10.1142/S0217595925500095

11. Q. Feng, J. J. Yuan, NP-hardness of a multicriteria scheduling on two families of jobs, *OR Transactions*, **114** (2007), 121–126. https://doi.org/10.15960/j.cnki.issn.1007-6093.2007.04.001

12. Q. Q. Nong, T. C. E. Cheng, C. T. Ng, Two-agent scheduling to minimize the total cost, *European J. Oper. Res.*, **215** (2011), 39–44. https://doi.org/10.1016/j.ejor.2011.05.041

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. W. J. Li, J. J. Yuan, Single-machine online scheduling of jobs with non-delayed processing constraint, *J. Comb. Optim.*, **41** (2021), 830–843. https://doi.org/10.1007/s10878-021-00722-4

15. W. J. Li, H. L. Liu, Online NDP-constraint scheduling of jobs with delivery times or weights, *Optim. Lett.*, **17** (2023), 591–612. https://doi.org/10.1007/s11590-022-01889-3

16. L. F. Lu, L. L. Zuo, L. Q. Zhang, J. W. Ou, Order acceptance and scheduling with weighted makespan, *4OR*, **23** (2025), 247–267. https://doi.org/10.1007/s10288-025-00586-y

17. F. F. Zheng, N. Li, M. Liu, Y. Xu, Single machine lot scheduling to minimize maximum weighted completion time, *J. Comb. Optim.,* **50** (2025), 1. https://doi.org/10.1007/s10878-025-01327-x

18. X. X. Liang, L. F. Lu, X. K. Sun, X. Yu, L. L. Zuo, Online scheduling on a single machine with one restart for all jobs to minimize the weighted makespan, *AIMS Math.*, **9** (2024), 2518–2529. https://doi.org/10.3934/math.2024124

AIMS Press