FOUNDATIONS OF COMPUTING AND DECISION SCIENCES
Vol. 49 (2024) No. 1
ISSN 0867-6356
sciendo DOI: 10.2478/fcds-2024-0006 e-ISSN 2300-3405

# New Results on Single-Machine Scheduling with Rejection to Minimize the Total Weighted Completion Time

Liqi Zhang *, Xue Yu †, Lingfa Lu † ‡

**Abstract.** In this paper, we study eight single-machine scheduling problems with rejection and position-dependent parameters. We consider two position-dependent parameters as follows: (1) position-dependent weights and (2) position-dependent processing times. In addition, we also introduce a weight-modifying activity or a rate-modifying activity into our problems. In the first six problems, the task is to minimize the sum of the total weighted completion time of accepted jobs and the total rejection cost of rejected jobs. We show that all six problems can be solved in polynomial time. In the last two problems, the task is to minimize the total weighted completion time of accepted jobs under the constraint that the total rejection cost of rejected jobs can not exceed a given upper bound. We show that these two problems are binary NP-hard and each problem admits an FPTAS.

**Keywords:** Scheduling with rejection; position-dependent parameters; dynamic programming

## 1 Introduction

### 1.1 Problem formulation

The general single-machine scheduling problem with rejection to minimize the total weighted completion time can be stated as follows. Assume that there are $n$ jobs $J_1, \ldots, J_n$ and a single machine. Each job $J_j$ has a processing time $p_j$, a weight

*College of Information and Management Science, Henan Agricultural University, Zhengzhou, Henan province, P.R. China
†School of Mathematics and Statistics, Zhengzhou University, Zhengzhou, Henan province, P.R. China
‡Corresponding author: lulingfa@zzu.edu.cn

$w_j$ and a rejection cost $e_j$. Here we assume that all $p_j$, $w_j$ and $e_j$ are non-negative integers. Job $J_j$ is either rejected, in which case the rejection cost $e_j$ has to be paid, or accepted and processed on the machine. In a feasible schedule, all jobs are partitioned into two disjoint sets $A$ and $R$, which are the sets of accepted jobs and rejected job, respectively. For each job $J_j$ with $J_j \in A$, let $C_j$ be its completion time in a schedule. Furthermore, we denote $TWC(A) = \sum_{J_j \in A} w_j C_j$ and $e(R) = \sum_{J_j \in R} e_j$ by the total weighted completion time of the accepted jobs and the total rejection cost of the rejected jobs, respectively. If the task is to minimize $TWC(A) + e(R)$, the corresponding scheduling problem can be denoted by $1||TWC(A) + e(R)$. If the task is to minimize $TWC(A)$ under the constraint $e(R) \leq U$, the corresponding scheduling problem can be denoted by $1|e(R) \leq U|TWC(A)$.

We consider two position-dependent parameters as follows: (1) position-dependent weights (PW) and (2) position-dependent processing times (PPT). In the position-dependent weights model, there are $n$ given weights $w_{[1]}, \cdots, w_{[n]}$. If $J_j$ is $i$-th processed job in a schedule, then job $J_j$ is assigned the weight $w_{[i]}$, i.e., $w_j = w_{[i]}$. Under the position-dependent weights model to minimize $TWC(A)+e(R)$, the corresponding problem can be denoted by

$$1|PW|TWC(A) + e(R). \tag{1}$$

In the position-dependent processing times model, there are $n$ processing times $p_{[1]}, \cdots, p_{[n]}$. If $J_j$ is $i$-th scheduled job, then job $J_j$ is assigned the processing time $p_{[i]}$. Under the position-dependent processing time model to minimize $TWC(A) + e(R)$, the corresponding problem can be denoted by

$$1|PPT|TWC(A) + e(R). \tag{2}$$

The scheduler may perform a weight-modifying activity (WMA) or a rate-modifying activity (RMA), which requires $L$ units of time. If job $J_j$ is performed after the weight-modifying activity (WMA), its actual weight will be reduced by multiplying a factor $\alpha$, i.e., $w_j^A = \alpha w_j$, where we assume that $0 < \alpha < 1$ and $w_j^A$ is the actual weight of $J_j$. Similarly, if job $J_j$ is performed after rate-modifying activity (RMA), its actual processing time will be reduced by multiplying a factor $\alpha$, i.e., $p_j^A = \alpha p_j$, where $p_j^A$ is the actual processing time of $J_j$. When the weight-modifying activity or the rate-modifying activity are introduced into our problems, the corresponding problems can be denoted by

$$1|PW, WMA|TWC(A) + e(R), \tag{3}$$

$$1|PW, RMA|TWC(A) + e(R), \tag{4}$$

$$1|PPT, WMA|TWC(A) + e(R), \tag{5}$$

and

$$1|PPT, RMA|TWC(A) + e(R). \tag{6}$$

We show that all six problems can be solved in polynomial time.

Furthermore, we also consider the constrained versions of problems (1) and (2). That is, the objective is to minimize $TWC(A)$ under the constraint $e(R) \leq U$, where

$U$ is a given upper bound. The corresponding problems can be denoted by

$$1|PW, e(R) \leq U|TWC(A), \tag{7}$$

and

$$1|PPT, e(R) \leq U|TWC(A). \tag{8}$$

We show that problems (7) and (8) are binary NP-hard and each problem admits an FPTAS.

### 1.2 Literature review

In this subsection, we will introduce some previous literature on scheduling with rejection, scheduling with position-dependent weights, scheduling with position-dependent processing times, scheduling with a weight-modifying activity and scheduling with a rate-modifying activity, respectively.

Scheduling with rejection was proposed first by Bartal et al. [3] in 2000. They studied the on-line scheduling problem $P|online|C_{\max}(A)+e(R)$ and the off-line scheduling problem $P||C_{\max}(A)+e(R)$, respectively. They presented an online algorithm with the best-possible competitive ratio $\frac{\sqrt{5}+3}{2} \approx 2.618$ for problem $P|online|C_{\max}(A) + e(R)$. They also presented a $(2 - \frac{1}{m})$-approximation algorithm and a polynomial-time approximation scheme (PTAS) for problem $P||C_{\max}(A) + e(R)$. Scheduling with rejection has receive more and more attention in recent two decades. Engels et al. [6] studied the single machine scheduling problem $1||TWC(A)+e(R)$. They showed that this problem is binary NP-hard and provided an FPTAS for this problem. Cao et al. [4] considered the constrained problem $1|e(R) \leq U|TWC(A)$. They also showed that this problem is binary NP-hard and provided an FPTAS for this problem. Zhang et al. [22] further studied other objective functions and provided some similar results. They showed that problem $1|e(R) \leq U|TWC(A)$ is binary NP-hard even when all jobs have the unit weight, i.e., $w_j = 1$ for each job $J_j$. Oron [15] studied four two-agent scheduling problems under rejection budget constraints. They developed some efficient pseudo-polynomial dynamic programming algorithms to obtain optimal solutions.

In additions, Shabtay et al. [16] pointed out, an important application of scheduling with rejection arises in make-to-order production systems with limited production capacity and tight delivery requirements. Another important application of scheduling with rejection occurs in scheduling with an outsourcing option. Moreover, there are many close connections between scheduling with rejection and other scheduling models such as scheduling with controllable processing times, scheduling with costs, and scheduling with due date assignment, etc. For more previous articles on this topic, we refer the reader to a survey on this topic by Shabtay et al. [16].

Scheduling with position-dependent weights was studied first by Kahlbacher [9] in 1992. He considered a single machine scheduling problem with the common due date. Jiang et al. [8] addressed proportionate flow-shop scheduling problems with position-dependent weights. Under the common and slack due date assignment models, two polynomial-time algorithms were also provided to solve these two problems

optimally. Wang et al. [19] studies the single machine scheduling with position-dependent weights and past-sequence-dependent delivery times. For the common and slack due-date assignment, they showed that these problems can be solvable in $O(n \log n)$ time.

Due to the learning or deterioration effects, the actual processing time of a job usually depend on its position in a schedule. In these situations, scheduling problems with position-dependent processing times were studied widely in the literature. Bachman and Janiak [2] studied some single machine scheduling problems with position-dependent processing times. They proved some properties of the special cases of the problems for the following optimization criteria: the makespan, the total completion time and the total weighted completion time. In addition, they also proved the strong NP-hardness of the makespan minimization problem. Zhang et al. [23] considered some single-machine scheduling problems with time and position dependent processing times. They proved that the classical SPT (Shortest Processing Time) rule remains optimal when the objective is to minimize the makespan or the total completion time. For problems of minimizing the total weighted completion time, the maximum lateness, and the discounted total weighted completion time, they presented some heuristic sequencing rules and analyzed the worst-case bounds for performance ratios. Agnetis and Mosheiov [1] further studied the scheduling problem with rejection and position-dependent processing times on proportionate flowshops. They showed that this problem can be solved in polynomial time.

In 1996, Whitaker [20] first studied the scheduling problem with a rate-modifying activity (RMA) on a single machine. She developed a branch-and-bound procedure for the problem with objective of minimizing total completion time problem. Lee and Leon [12] developed a polynomial-time algorithm for the same problem. They also developed some algorithms to solve the problems with other objective functions, including makespan, total weighted completion time and maximum lateness. Scheduling with a rate-modifying-activity has become a popular topic for scheduling researchers in recent years. More recent articles on scheduling with a rate-modifying activity are [10, 11, 13, 24].

Mosheiov and Oron [14] introduced the weight-modifying activity into scheduling problems in 2020. Thus, only a few of papers is published on this topic. Mosheiov and Oron [14] studied the single machine scheduling problem to minimize total weighted completion time. A pseudo-polynomial dynamic programming solution algorithm for this problem is introduced. Several special cases of unit processing time jobs were solved in polynomial time. For the same problem, Chen et al. [5] also proposed two mixed integer linear programming models. Based on some optimality properties, a heuristic algorithm with swap and insert procedures is also developed.

## 2 Scheduling with position-dependent weights

### 2.1 Problem $1|PW|TWC(A)+e(R)$

In this subsection, we will consider the problem $1|PW|TWC(A)+e(R)$. For the general problem $1||TWC(A)+e(R)$, Engels et al. [6] show that this problem is weakly NP-hard and provided a pseudo-polynomial dynamic programming algorithm based on the WSPT (Weighted Shortest Processing Time) rule. However, in our problem $1|PW|TWC(A)+e(R)$, all weights are position-dependent. Thus, the WSPT rule is invalid for our problem. Fortunately, we will show that we can replace the WSPT rule by the SPT (Shortest Processing Time) rule for our problem. We have the following lemma.

**Lemma 1.** For problem $1|PW|TWC(A)+e(R)$, there exists an optimal schedule such that all accepted jobs are processed in the SPT rule.

**Proof.** Lemma 1 can be easily proved by a pairwise exchange argument. Thus, we omit the detailed proof. □

From Lemma 1, we first re-label the jobs in the SPT rule such that $p_1 \leq p_2 \leq \cdots \leq p_n$. Assume that there are $k$ jobs which are accepted in an optimal schedule. Thus, the weights of these $k$ accepted jobs will be $w_{[1]}, \cdots, w_{[k]}$, respectively. Under the assumption that there are $k$ jobs which are accepted in the optimal schedule, the corresponding problem can be denoted by $1|PW, n_A = k|TWC(A)+e(R)$.

For problem $1|PW, n_A = k|TWC(A)+e(R)$, let $f_j(i)$ be the optimal value of the objective function when the jobs in consideration are $J_j, \cdots, J_n$ and the number of the current accepted jobs is exactly $i$. Now, we consider any optimal schedule for the jobs $J_j, \cdots, J_n$ in which the number of the current accepted jobs is $i$. In any such schedule, there are two possible cases: either $J_j$ is rejected or $J_j$ is accepted and processed before other accepted jobs.

**Case 1.** Job $J_j$ is rejected. In this case, when only the jobs $J_{j+1}, \cdots, J_n$ are considered, the number of the current accepted jobs is still $i$. Thus, we have $f_j(i) = f_{j+1}(i) + e_j$.

**Case 2.** Job $J_j$ is accepted. In this case, when only the jobs $J_{j+1}, \cdots, J_n$ are considered, the number of the current accepted jobs is $i-1$. Note that $J_j$ is the $(k-i+1)$-th scheduled job and its weight is exactly $w_{[k-i+1]}$. When $J_j$ is scheduled before other accepted jobs, the total weighted completion time of the accepted jobs is added by a value $p_j(w_{[k-i+1]} + \cdots + w_{[k]})$. Thus, we have $f_j(i) = f_{j+1}(i-1) + p_j(w_{[k-i+1]} + \cdots + w_{[k]})$.

Combining the above two cases, we have the following dynamic programming algorithm DP1.

**Dynamic programming algorithm DP1**

**The boundary conditions:**

$$f_n(i) = \begin{cases} w_{[k]}p_n, & \text{if } i = 1; \\ e_n, & \text{if } i = 0; \\ +\infty, & \text{otherwise.} \end{cases}$$

**The recursive function:**

$$f_j(i) = \min\{f_{j+1}(i) + e_j, f_{j+1}(i-1) + p_j(w_{[k-i+1]} + \cdots + w_{[k]})\}.$$

**The optimal value** is given by $f_1(k)$.

**Theorem 1.** Algorithm DP1 solves problem $1|PW, n_A = k|TWC(A) + e(R)$ in $O(n^2)$ time.

**Proof.** The correctness of algorithm DP1 is guaranteed by the above discussion. The recursive function has at most $O(n^2)$ states and each iteration costs constant time. Thus, the running time of algorithm DP1 is bounded by $O(n^2)$. □

Note that, there are at most $n+1$ choices for the value $k$. Thus, we can enumerate all possibilities for $k = 0, 1, \cdots, n$ and solve the corresponding problem $1|PW, n_A = k|TWC(A) + e(R)$, respectively. Finally, we select the solution with the smallest objective value. Thus, we have the following theorem.

**Theorem 2.** Problem $1|PW|TWC(A) + e(R)$ can be solved in $O(n^3)$ time.

## 2.2 Problem $1|PW, WMA|TWC(A) + e(R)$

In this subsection, we will consider the problem $1|PW, WMA|TWC(A) + e(R)$. It is easy to see that Lemma 1 still holds for problem $1|PW, WMA|TWC(A)+e(R)$. From Lemma 1, we first re-label the jobs in the SPT rule such that $p_1 \leq p_2 \leq \cdots \leq p_n$. Assume that there are $k$ jobs which are accepted in an optimal schedule. Thus, the weights of these $k$ accepted jobs will be $w_{[1]}, \cdots, w_{[k]}$, respectively. Furthermore, we assume that there are $n_w = k'$ jobs which are processed after the weight-modifying activity is performed, where $k' \leq k$. Under the above assumptions, the corresponding problem can be denoted by $1|PW, WMA, n_A = k, n_w = k'|TWC(A) + e(R)$.

For problem $1|PW, WMA, n_A = k, n_w = k'|TWC(A)+e(R)$, let $f_j(i)$ be the optimal value of the objective function when the jobs in consideration are $J_j, \cdots, J_n$ and the number of the current accepted jobs is exactly $i$. Now, we consider any optimal schedule for the jobs $J_j, \cdots, J_n$ in which the number of the current accepted jobs is $i$. In any such schedule, there are two possible cases: either $J_j$ is rejected or $J_j$ is accepted and processed before other accepted jobs.

**Case 1.** Job $J_j$ is rejected. In this case, when only the jobs $J_{j+1}, \cdots, J_n$ are considered, the number of the current accepted jobs is still $i$. Thus, we have $f_j(i) = f_{j+1}(i) + e_j$.

**Case 2.** Job $J_j$ is accepted. In this case, when only the jobs $J_{j+1}, \cdots, J_n$ are considered, the number of the current accepted jobs is $i-1$. Note that $J_j$ is the $(k-i+1)$-th scheduled job and its original weight is exactly $w_{[k-i+1]}$. If $i \leq k'$, then $J_j$ must be processed after the weight-modifying activity is performed. That is, the actual weight is exactly $\alpha w_{[k-i+1]}$. Especially, if $i = k'$, then $J_j$ must be processed immediately after the weight-modifying activity is performed. When the weight-modifying activity and $J_j$ are performed before other accepted jobs, the total weighted completion time of the accepted jobs is added by a value $\alpha(p_j+L)(w_{[k-i+1]}+\cdots+w_{[k]})$. Thus, we have $f_j(i) = V_A^1 = f_{j+1}(i-1) + \alpha(p_j+L)(w_{[k-i+1]} + \cdots + w_{[k]})$. If $i < k'$, there must be some job which is processed between the weight-modifying activity and $J_j$. When $J_j$ is scheduled before other accepted jobs, the total weighted completion time of the accepted jobs is added by a value $\alpha p_j(w_{[k-i+1]}+\cdots+w_{[k]})$. Thus, we have $f_j(i) = V_A^2 = f_{j+1}(i-1)+\alpha p_j(w_{[k-i+1]}+\cdots+w_{[k]})$. If $i > k'$, then $J_j$ must be processed before the weight-modifying activity is performed. When $J_j$ is processed before other accepted jobs the weight-modifying activity, the total weighted completion time of the accepted jobs is added by a value $p_j(w_{[k-i+1]}+\cdots+w_{[k-k']})+\alpha p_j(w_{[k-k'+1]}+\cdots+w_{[k]})$. Thus, we have $f_j(i) = V_A^3 = f_{j+1}(i-1)+p_j(w_{[k-i+1]}+\cdots+w_{[k-k']})+\alpha p_j(w_{[k-k'+1]}+\cdots+w_{[k]})$.

Combining the above two cases, we have the following dynamic programming algorithm DP2.

**Dynamic programming algorithm DP2**

**The boundary conditions:**

$$f_n(i) = \begin{cases} \alpha w_{[k]} p_n, & \text{if } \ i = 1 < k'; \\ \alpha w_{[k]}(p_n + L), & \text{if } \ i = 1 = k'; \\ w_{[k]} p_n, & \text{if } \ i = 1 > k'; \\ e_n, & \text{if } \ i = 0; \\ +\infty, & \text{otherwise.} \end{cases}$$

**The recursive function:**

$$f_j(i) = \begin{cases} \min\{f_{j+1}(i) + e_j, V_A^1\}, & \text{if } \ i = k'; \\ \min\{f_{j+1}(i) + e_j, V_A^2\}, & \text{if } \ i < k'; \\ \min\{f_{j+1}(i) + e_j, V_A^3\}, & \text{if } \ i > k'. \end{cases}$$

**The optimal value** is given by $f_1(k)$.

**Theorem 3.** Algorithm DP2 solves problem $1|PW, WMA, n_A = k, n_w = k'|TWC(A) + e(R)$ in $O(n^2)$ time.

**Proof.** The correctness of algorithm DP2 is guaranteed by the above discussion. The recursive function has at most $O(n^2)$ states and each iteration costs constant time. Thus, the running time of algorithm DP2 is bounded by $O(n^2)$. □

Note that, there are at most $O(n^2)$ choices for the values $k$ and $k'$. Thus, we can enumerate all possibilities and solve the corresponding problem $1|PW, WMA, n_A = k, n_w = k'|TWC(A) + e(R)$, respectively. Finally, we select the solution with the smallest objective value. Thus, we have the following theorem.

**Theorem 4.** Problem $1|PW, WMA|TWC(A)+e(R)$ can be solved in $O(n^4)$ time.

### 2.3 Problem $1|PW, RMA|TWC(A) + e(R)$

In this subsection, we will consider the problem $1|PW, RMA|TWC(A) + e(R)$. It is not hard to verify that Lemma 1 holds still for problem $1|PW, RMA|TWC(A)+e(R)$. From Lemma 1, we first re-label the jobs in the SPT rule such that $p_1 \leq p_2 \leq \cdots \leq p_n$. Assume that there are $k$ jobs which are accepted in an optimal schedule. Thus, the weights of these $k$ accepted jobs will be $w_{[1]}, \cdots, w_{[k]}$. Furthermore, we assume that there are $n_r = k'$ jobs which are processed after the rate-modifying activity is performed, where $k' \leq k$. Under the above assumptions, the corresponding problem can be denoted by $1|PW, RMA, n_A = k, n_R = k'|TWC(A) + e(R)$.

For problem $1|PW, RMA, n_A = k, n_r = k'|TWC(A) + e(R)$, let $f_j(i)$ be the optimal value of the objective function when the jobs in consideration are $J_j, \cdots, J_n$ and the number of the current accepted jobs is exactly $i$. Now, we consider any optimal schedule for the jobs $J_j, \cdots, J_n$ in which the number of the current accepted jobs is $i$. In any such schedule, there are two possible cases: either $J_j$ is rejected or $J_j$ is accepted and processed before other accepted jobs.

**Case 1.** Job $J_j$ is rejected. In this case, when only the jobs $J_{j+1}, \cdots, J_n$ are considered, the number of the current accepted jobs is still $i$. Thus, we have $f_j(i) = f_{j+1}(i) + e_j$.

**Case 2.** Job $J_j$ is accepted. In this case, when only the jobs $J_{j+1}, \cdots, J_n$ are considered, the number of the current accepted jobs is $i-1$. Note that $J_j$ is the $(k-i+1)$-th scheduled job and its weight is exactly $w_{[k-i+1]}$. If $i \leq k'$, then $J_j$ must be processed after the rate-modifying activity is performed. That is, the actual processing time is exactly $\alpha p_j$. Especially, if $i = k'$, then $J_j$ must be processed immediately after the rate-modifying activity is performed. When the rate-modifying activity and $J_j$ are performed before other accepted jobs, the total weighted completion time of the accepted jobs is added by a value $(\alpha p_j + L)(w_{[k-i+1]} + \cdots + w_{[k]})$. Thus, we have $f_j(i) = V_A^1 = f_{j+1}(i-1) + (\alpha p_j + L)(w_{[k-i+1]} + \cdots + w_{[k]})$. If $i < k'$, there must be some job which is processed between the rate-modifying activity and $J_j$. When $J_j$ is scheduled before other accepted jobs, the total weighted completion time of the accepted jobs is added by a value $\alpha p_j(w_{[k-i+1]} + \cdots + w_{[k]})$. Thus, we have $f_j(i) = V_A^2 =$

$f_{j+1}(i-1)+\alpha p_j(w_{[k-i+1]}+\cdots+w_{[k]})$. If $i > k'$, then $J_j$ must be processed before the rate-modifying activity is performed. The actual processing time is still $p_j$. When $J_j$ is processed before other accepted jobs the rate-modifying activity, the total weighted completion time of the accepted jobs is added by a value $p_j(w_{[k-i+1]}+\cdots+w_{[k]})$. Thus, we have $f_j(i) = V_A^3 = f_{j+1}(i-1) + p_j(w_{[k-i+1]}+\cdots+w_{[k]})$.

Combining the above two cases, we have the following dynamic programming algorithm DP3.

**Dynamic programming algorithm DP3**

**The boundary conditions:**

$$f_n(i) = \begin{cases} \alpha w_{[k]} p_n, & \text{if } \ i = 1 < k'; \\ w_{[k]}(\alpha p_n + L), & \text{if } \ i = 1 = k'; \\ w_{[k]} p_n, & \text{if } \ i = 1 > k'; \\ e_n, & \text{if } \ i = 0; \\ +\infty, & \text{otherwise.} \end{cases}$$

**The recursive function:**

$$f_j(i) = \begin{cases} \min\{f_{j+1}(i) + e_j, V_A^1\}, & \text{if } \ i = k'; \\ \min\{f_{j+1}(i) + e_j, V_A^2\}, & \text{if } \ i < k'; \\ \min\{f_{j+1}(i) + e_j, V_A^3\}, & \text{if } \ i > k'. \end{cases}$$

**The optimal value** is given by $f_1(k)$.

**Theorem 5.** Algorithm DP3 solves problem $1|PW, RMA, n_A = k, n_r = k'|TWC(A)+e(R)$ in $O(n^2)$ time.

**Proof.** The correctness of algorithm DP3 is guaranteed by the above discussion. The recursive function has at most $O(n^2)$ states and each iteration costs constant time. Thus, the running time of algorithm DP3 is bounded by $O(n^2)$. □

Note that, there are at most $O(n^2)$ choices for the values $k$ and $k'$. Thus, we can enumerate all possibilities and solve the corresponding problem $1|PW, RMA, n_A = k, n_r = k'|TWC(A) + e(R)$, respectively. Finally, we select the solution with the smallest objective value. Thus, we have the following theorem.

**Theorem 6.** Problem $1|PW, RMA|TWC(A)+e(R)$ can be solved in $O(n^4)$ time.

## 3 Scheduling with position-dependent processing times

### 3.1 Problem $1|PPT|TWC(A)+e(R)$

Assume that there are $k$ accepted jobs in an optimal schedule for problem $1|PPT|TWC(A)+e(R)$. Thus, the processing times of the accept jobs are $p_{[1]}, p_{[2]}, \cdots, p_{[k]}$, respectively. Furthermore, for each $i = 1, \cdots, k$, we also $C_{[i]} = \sum_{r=1}^{i} p_{[r]}$. That is, the completion times of the accepted jobs are always fixed. Thus, to minimize the total weighted completion time of the accepted jobs, all accepted jobs should be processed in the LW (Largest Weight) rule. The LW rule can be stated as follows:

**LW rule:** Whenever the machine is idle, among all unfinished jobs, the job with the largest weight is scheduled.

**Lemma 2.** For problem $1|PPT|TWC(A)+e(R)$, there exists an optimal schedule such that all accepted jobs are processed in the LW rule.

**Proof.** Lemma 2 can be easily proved by a pairwise exchange argument. Here we omit the detailed proof. □

Based on Lemma 2, we first re-label the jobs such that $w_1 \geq w_2 \geq \cdots \geq w_n$. Let $f_j(i)$ be the optimal value of the objective function when the jobs in consideration are $J_1, \cdots, J_j$ and the number of accepted jobs is $i$. If $i > 0$, then the processing times of $i$ accepted jobs are $p_{[1]}, \cdots, p_{[i]}$, respectively.

Now, we consider any optimal schedule for the jobs $J_1, \cdots, J_j$ in which the number of accepted jobs is $i$. In any such schedule, there are two possible cases: either $J_j$ is rejected or $J_j$ is accepted and processed on the machine.

**Case 1.** Job $J_j$ is rejected. In this case, when the jobs $J_1, \cdots, J_{j-1}$ are considered, the number of accepted jobs is still $i$. Thus, we have $f_j(i) = f_{j-1}(i) + e_j$.

**Case 2.** Job $J_j$ is accepted. In this case, when the jobs $J_1, \cdots, J_{j-1}$ are considered, the number of accepted jobs is $i-1$. Note that $J_j$ is the $i$-th processed job. Thus, its processing time is $p_{[i]}$ and then its completion time is $\sum_{r=1}^{i} p_{[r]}$. Thus, we have $f_j(i) = f_{j-1}(i-1) + w_j \sum_{r=1}^{i} p_{[r]}$.

Combining the above two cases, we have the following dynamic programming algorithm DP4.

**Dynamic programming algorithm DP4**

**The boundary conditions:**

$$f_1(i) = \begin{cases} w_1 p_{[1]}, & \text{if } i = 1; \\ e_1, & \text{if } i = 0; \\ +\infty, & \text{otherwise.} \end{cases}$$

**The recursive function**

$$f_j(i) = \min\{f_{j-1}(i) + e_j, f_{j-1}(i-1) + w_j \sum_{r=1}^{i} p_{[r]}\}.$$

**The optimal value is given by**

$$\min\{f_n(i) : 0 \leq i \leq n\}.$$

**Theorem 7.** Algorithm DP4 solves $1|PPT|TWC(A) + e(R)$ in $O(n^2)$ time.

**Proof.** The correctness of the algorithm is guaranteed by the above discussion. Note that the recursive function has at most $O(n^2)$ states and each iteration costs constant time. Thus, the running time is bounded by $O(n^2)$. □

### 3.2 Problem $1|PPT, WMA|TWC(A) + e(R)$

Assume that there are $n_w = k'$ accepted jobs which are processed before the the weight-modification activity in an optimal schedule, where $k' = 0, 1, \cdots, n$. Under the above assumption, the corresponding problem can be denoted by $1|PPT, WMA, n_w = k'|TWC(A) + e(R)$. It is easy to see that Lemma 2 is still valid for problem $1|PPT, WMA, n_w = k'|TWC(A) + e(R)$. Furthermore, similar to the discussion on $k'$ in subsections 2.2 and 2.3, we can obtain a new dynamic programming algorithm for $1|PPT, WMA, n_w = k'|TWC(A) + e(R)$.

Based on Lemma 2, we first re-label the jobs such that $w_1 \geq w_2 \geq \cdots \geq w_n$. For problem $1|PPT, WMA, n_w = k'|TWC(A) + e(R)$, let $f_j(i)$ be the optimal value of the objective function when the jobs in consideration are $J_1, \cdots, J_j$ and the number of accepted jobs is $i$. If $i > 0$, then the processing times of $i$ accepted jobs are $p_{[1]}, \cdots, p_{[i]}$, respectively.

**Dynamic programming algorithm DP5**

**The boundary conditions:**

$$f_1(i) = \begin{cases} w_1 p_{[1]}, & \text{if } i = 1 \leq k'; \\ \alpha w_1 p_{[1]}, & \text{if } i = 1 > k'; \\ e_1, & \text{if } i = 0; \\ +\infty, & \text{otherwise.} \end{cases}$$

**The recursive function**

$$f_j(i) = \begin{cases} \min\{f_{j-1}(i) + e_j, f_{j-1}(i-1) + w_j(p_{[1]} + \cdots + p_{[i]})\} & \text{if } i \le k'; \\ \min\{f_{j-1}(i) + e_j, f_{j-1}(i-1) + \alpha w_j(p_{[1]} + \cdots + p_{[i]} + L)\}, & \text{if } i > k'; \end{cases}$$

**The optimal value is given by**

$$\min\{f_n(i) : k' \le i \le n\}.$$

**Theorem 8.** Algorithm DP5 solves $1|PPT, WMA, n_w = k'|TWC(A) + e(R)$ in $O(n^2)$ time.

**Proof.** The correctness of the algorithm is guaranteed by the above discussion. Note that the recursive function has at most $O(n^2)$ states and each iteration costs a constant time. Thus, the running time is bounded by $O(n^2)$. □

By enumerating all possibilities for $k' = 0, 1, \cdots, n$ and solve the corresponding problem $1|PPT, WMA, n_w = k'|TWC(A) + e(R)$, respectively, we can obtain an optimal algorithm which solves problem $1|PPT, WMA|TWC(A) + e(R)$ in $O(n^3)$ time.

### 3.3 Problem $1|PPT, RMA|TWC(A) + e(R)$

Assume that there are $n_r = k'$ accepted jobs which are processed before the the rate-modification activity in an optimal schedule, where $k' = 0, 1, \cdots, n$. Note that, Lemma 2 is still valid for problem $1|PPT, RMA, n_r = k'|TWC(A) + e(R)$. Furthermore, similar to the discussion on $k'$ in subsection 2.3, we can obtain a new dynamic programming algorithm for $1|PPT, RMA, n_r = k'|TWC(A) + e(R)$.

Based on Lemma 2, we first re-label the jobs such that $w_1 \ge w_2 \ge \cdots \ge w_n$. For problem $1|PPT, RMA, n_r = k'|TWC(A) + e(R)$, let $f_j(i)$ be the optimal value of the objective function when the jobs in consideration are $J_1, \cdots, J_j$ and the number of accepted jobs is $i$.

**Dynamic programming algorithm DP6**

**The boundary conditions:**

$$f_1(i) = \begin{cases} w_1 p_{[1]}, & \text{if } i = 1 \le k'; \\ \alpha w_1 p_{[1]}, & \text{if } i = 1 > k'; \\ e_1, & \text{if } i = 0; \\ +\infty, & \text{otherwise.} \end{cases}$$

**The recursive function**

$$f_j(i) = \begin{cases} \min\{f_{j-1}(i) + e_j, f_{j-1}(i-1) + w_j(p_{[1]} + \cdots + p_{[i]})\} & \text{if } i \le k'; \\ \min\{f_{j-1}(i) + e_j, f_{j-1}(i-1) + w_j(p_{[1]} + \cdots + p_{[k']} \\ +L + \alpha p_{[k'+1]} + \cdots \alpha p_{[i]})\}, & \text{if } i > k'; \end{cases}$$

**The optimal value is given by**

$$\min\{f_n(i) : k' \leq i \leq n\}.$$

**Theorem 9.** Algorithm DP6 solves $1|PPT, RMA, n_r = k'|TWC(A) + e(R)$ in $O(n^2)$ time.

**Proof.** The correctness of the algorithm is guaranteed by the above discussion. Note that the recursive function has at most $O(n^2)$ states and each iteration costs a constant time. Thus, the running time is bounded by $O(n^2)$. □

By enumerating all possibilities for $k' = 0, 1, \cdots, n$ and solve the corresponding problem $1|PPT, RMA, n_r = k'|TWC(A) + e(R)$, respectively, we can obtain an optimal algorithm which solves problem $1|PPT, RMA|TWC(A) + e(R)$ in $O(n^3)$ time.

## 4 The constrained problems

In this section, we consider two constrained problems $1|PW, e(R) \leq U|TWC(A)$ and $1|PPT, e(R) \leq U|TWC(A)$. We show that two constrained problems are binary NP-hard and admit an FPTAS.

### 4.1 Problem $1|PW, e(R) \leq U|TWC(A)$

Set $TC(A) = \sum_{J_j \in A} C_j$. Zhang et al. [22] show that the constrained problem $1|e(R) \leq U|TC(A)$ is binary NP-hard. Thus, by setting $w_{[1]} = \cdots = w_{[n]} = 1$, we have $TWC(A) = TC(A)$. Furthermore, we can conclude that problem $1|PW, e(R) \leq U|TWC(A)$ is also NP-hard.

Clearly, Lemma 1 still holds for problem $1|PW, e(R) \leq U|TWC(A)$. From Lemma 1, we first re-label the jobs in the SPT rule such that $p_1 \leq p_2 \leq \cdots \leq p_n$. Assume that there are $k$ jobs which are accepted in an optimal schedule. Thus, the weights of these $k$ accepted jobs will be $w_{[1]}, \cdots, w_{[k]}$, respectively. Under the assumption that there are $k$ jobs which are accepted in the optimal schedule, the corresponding problem can be denoted by $1|PW, e(R) \leq U, n_A = k|TWC(A)$.

For problem $1|PW, e(R) \leq U, n_A = k|TWC(A)$, let $f_j(i, E)$ be the optimal value of the objective function when the jobs in consideration are $J_j, \cdots, J_n$, the number of the current accepted jobs is exactly $i$ and the total rejection cost of the current rejected jobs is exactly $E$.

**Case 1.** Job $J_j$ is rejected. In this case, when only the jobs $J_{j+1}, \cdots, J_n$ are considered, the number of the current accepted jobs is still $i$ and the total rejection cost of the current rejected jobs is exactly $E - e_j$. Thus, we have $f_j(i, E) = f_{j+1}(i, E - e_j)$.

**Case 2.** Job $J_j$ is accepted. In this case, when only the jobs $J_{j+1}, \cdots, J_n$ are considered, the number of the current accepted jobs is $i - 1$ and the total rejection cost of

the current rejected jobs is still $E$. Note that $J_j$ is the $(k-i+1)$-th scheduled job and its weight is exactly $w_{[k-i+1]}$. When $J_j$ is scheduled before other accepted jobs, the total weighted completion time of the accepted jobs is added by a value $p_j(w_{[k-i+1]} + \cdots + w_{[k]})$. Thus, we have $f_j(i,E) = f_{j+1}(i-1,E) + p_j(w_{[k-i+1]} + \cdots + w_{[k]})$.

Combining the above two cases, we have the following dynamic programming algorithm DP7.

**Dynamic programming algorithm DP7**

**The boundary conditions:**

$$f_n(i,E) = \begin{cases} w_{[k]}p_n, & \text{if } i = 1 \text{ and } E = 0; \\ 0, & \text{if } i = 0 \text{ and } E = e_1 \le U; \\ +\infty, & \text{otherwise.} \end{cases}$$

**The recursive function:**

$$f_j(i,E) = \min\{f_{j+1}(i, E-e_j), f_{j+1}(i-1,E) + p_j(w_{[k-i+1]} + \cdots + w_{[k]})\}.$$

**The optimal value** is given by $\min\{f_1(k,E) : 0 \le E \le U\}$.

**Theorem 10.** Algorithm DP7 solves problem $1|PW, e(R) \le U, n_A = k|TWC(A)$ in $O(n^2U)$ time.

**Proof.** The correctness of algorithm DP7 is guaranteed by the above discussion. The recursive function has at most $O(n^2U)$ states and each iteration costs constant time. Thus, the running time of algorithm DP7 is bounded by $O(n^2U)$. □

Note that, there are at most $n+1$ choices for the value $k$. Thus, we can enumerate all possibilities for $k = 0, 1, \cdots, n$ and solve the corresponding problem $1|PW, e(R) \le U, n_A = k|TWC(A)$, respectively. Finally, we select the solution with the smallest objective value. Thus, we have the following theorem.

**Theorem 11.** Problem $1|PW, e(R) \le U|TWC(A)$ can be solved in $O(n^3U)$ time.

To obtain a fully polynomial-time approximation scheme (FPTAS) for an NP-hard problem, a common way is to transform a optimal but pseudo-polynomial-time dynamic programming algorithm into a near optimal but faster algorithm. The main technique is to remove some yielded data during the execution of the algorithm and clean up the algorithm's memory. As a result, the algorithm becomes faster and the outputted solution is nearly optimal. This approach is called "vector trimming" in the literature (see [17, 21]). By using the vector trimming technique, an FPTAS can be proposed for problem $1|PW, e(R) \le U|TWC(A)$, where we omit the detailed procedure.

### 4.2 Problem $1|PPT, e(R) \leq U|TWC(A)$

Zhang et al. [22] show that the constrained problem $1|e(R) \leq U|TC(A)$ is binary NP-hard. By the modification of their NP-hardness proof, we show that problem $1|p_j = 1, e(R) \leq U|TWC(A)$ is NP-hard. Furthermore, this implies that problem $1|PPT, e(R) \leq U|TWC(A)$ is also NP-hard.

**Theorem 12.** Problem $1|p_j = 1, e(R) \leq U|TWC(A)$ is NP-hard.

**Proof.** The decision version of the problem is clearly in NP. We use the NP-complete Old-Even-Partition problem (Garey and Johnson [7]) for the reduction.

**Old-Even-Partition problem:** Given $2t+1$ positive integers $a_1, a_2, \cdots, a_{2t}, B$ such that $\sum_{i=1}^{2t} a_i = 2B$, is there a subset $S \subset \{1, 2, \cdots, 2t\}$ such that $|S \bigcap \{2i-1, 2i\}| = 1$ for each $i = 1, 2, \cdots, t$ and $\sum_{i \in S} a_i = B$?

For a given instance of the Old-Even-Partition problem, we construct an instance of the decision version of the problem $1|p_j = 1, e(R) \leq U|TWC(A)$ as follows.

- There are $n = 2t$ jobs with $p_j = 1$ for each $j = 1, \cdots, 2t$.
- For with $i$ with $1 \leq i \leq t$, we define

$$w_{2i-1} = \frac{2^{t-i}B + a_{2i-1}}{i}, \quad e_{2i-1} = 2^{t-i}B + a_{2i-1},$$

and

$$w_{2i} = \frac{2^{t-i}B + a_{2i}}{i}, \quad e_{2i} = 2^{t-i}B + a_{2i}.$$

- The upper bound is defined by $U = 2^t B$.
- The threshold value is defined by $Y = 2^t B$.
- The decision version asks whether there is a schedule $\pi$ such that $TWC(A) \leq Y$ under the constraint $e(R) \leq U$.

It can be observed that the above construction can be done in polynomial time. First, we assume that Old-Even-Partition problem has a solution $S \subset \{1, 2, \cdots, 2t\}$ such that $|S \bigcap \{2i-1, 2i\}| = 1$ for each $i = 1, 2, \cdots, t$ and $\sum_{i \in S} a_i = B$. Note that $\min\{w_{2i-1}, w_{2i}\} > \max\{w_{2i+1}, w_{2i+2}\}$ for each $i = 1, 2, \cdots, t-1$. We assign the jobs in $\{J_j : j \in S\}$ to be processed in LW-rule (Largest-Weight first) and reject all other jobs. It is not hard to see that $TWC(A) = 2^t B = Y$ and $e(R) = 2^t B = U$.

Now, we suppose that there is a schedule $\pi$ such that $TWC(A) \leq Y = 2^t B$ and $e(R) \leq U = 2^t B$. We are ready to show that Old-Even-Partition problem has a solution. We have the following claims.

**Claim 1.** $|A\bigcap\{J_{2i-1}, J_{2i}\}| = |R\bigcap\{J_{2i-1}, J_{2i}\}| = 1$ for each $i = 1, 2, \cdots, t$.

Otherwise, suppose that $k$ is the minimum index such that $\{J_{2k-1}, J_{2k}\} \subset A$ or $\{J_{2k-1}, J_{2k}\} \subset R$. By the definition of $k$, we have $|A\bigcap\{J_{2i-1}, J_{2i}\}| = |R\bigcap\{J_{2i-1}, J_{2i}\}| = 1$ for each $i = 1, 2, \cdots, k-1$. Furthermore, we can assume that the accepted jobs are processed in LW-rule in $\pi$. Note that $\min\{w_{2i-1}, w_{2i}\} > \max\{w_{2i+1}, w_{2i+2}\}$ for each $i = 1, 2, \cdots, t-1$. If $\{J_{2k-1}, J_{2k}\} \subset A$, then we have

$$\begin{aligned} TWC(A) &\geq (w_{2k-1} + w_{2k}) \times k + \textstyle\sum_{i<k, J_{2i-1}\in A} w_{2i-1} \times i + \sum_{i<k, J_{2i}\in A} w_{2i} \times i \\ &> 2^{t-k}B + 2^{t-k}B + 2^{t-k+1}B + \cdots + 2^{t-1}B \\ &= 2^tB \\ &= Y, \end{aligned}$$

a contradiction. If $\{J_{2k-1}, J_{2k}\} \subset R$, then we have

$$\begin{aligned} e(R) &\geq e_{2k-1} + e_{2k} + \textstyle\sum_{i<k, J_{2i-1}\in R} e_{2i-1} + \sum_{i<k, J_{2i}\in R} e_{2i} \\ &> 2^{t-k}B + 2^{t-k}B + 2^{t-k+1}B + \cdots + 2^{t-1}B \\ &= 2^tB \\ &= U, \end{aligned}$$

a contradiction again. Thus, we have $|A\bigcap\{J_{2i-1}, J_{2i}\}| = |R\bigcap\{J_{2i-1}, J_{2i}\}| = 1$ for each $i = 1, 2, \cdots, t$.

**Claim 2.** $\sum_{J_j\in A} a_j = \sum_{J_j\in R} a_j = B$.

Since $\sum_{J_j\in R} e_j = \sum_{i=1}^{t} 2^{t-i}B + \sum_{J_j\in R} a_j \leq U = 2^tB$, we have $\sum_{J_j\in R} a_j \leq B$. Thus, we also have $\sum_{J_j\in A} a_j \geq B$. Assume that $\sum_{J_j\in A} a_j > B$, we have

$$\begin{aligned} TWC(A) &= \textstyle\sum_{J_{2i-1}\in A} w_{2i-1} \times i + \sum_{J_{2i}\in A} w_{2i} \times i \\ &= \textstyle\sum_{J_{2i-1}\in A}(2^{t-i}B + a_{2i-1}) + \sum_{J_{2i}\in A}(2^{t-i}B + a_{2i}) \\ &= \textstyle\sum_{i=1}^{t} 2^{t-i}B + \sum_{J_j\in A} a_j \\ &> 2^tB \\ &= Y, \end{aligned}$$

a contradiction. Hence, we have $\sum_{J_j\in A} a_j = \sum_{J_j\in R} a_j = B$.

Set $S = \{j : J_j \in A\}$ and $\overline{S} = \{j : J_j \in R\}$. By Claim 1 and Claim 2, $S$ (or $\overline{S}$) is a solution of Odd-Even-Partition problem. Theorem 12 follows. □

Next, we present a dynamic programming algorithm for problem $1|PPT, e(R) \leq U|TWC(A)$. Clearly, Lemma 2 still holds for problem $1|PPT, e(R) \leq U|TWC(A)$.

From Lemma 2, we first re-label the jobs in the LW rule such that $w_1 \geq w_2 \geq \cdots \geq w_n$. Let $f_j(i, E)$ be the optimal value of the objective function when the jobs in consideration are $J_1, \cdots, J_j$, the number of the current accepted jobs is exactly $i$ and the total rejection cost of the current rejected jobs is exactly $E$.

**Case 1.** Job $J_j$ is rejected. In this case, when only the jobs $J_1, \cdots, J_{j-1}$ are considered, the number of the current accepted jobs is still $i$ and the total rejection cost of the current rejected jobs is exactly $E - e_j$. Thus, we have $f_j(i, E) = f_{j-1}(i, E - e_j)$.

**Case 2.** Job $J_j$ is accepted. In this case, when only the jobs $J_1, \cdots, J_{j-1}$ are considered, the number of the current accepted jobs is $i - 1$ and the total rejection cost of the current rejected jobs is still $E$. Note that $J_j$ is the $i$-th scheduled job and its processing time is exactly $p_{[i]}$. When $J_j$ is scheduled after other accepted jobs, the makespan of the accepted jobs is exactly $(p_{[1]} + \cdots + p_{[i]})$. Thus, we have $f_j(i, E) = f_{j-1}(i - 1, E) + w_j(p_{[1]} + \cdots + p_{[i]})$.

Combining the above two cases, we have the following dynamic programming algorithm DP8.

**Dynamic programming algorithm DP8**

**The boundary conditions:**

$$f_1(i, E) = \begin{cases} w_1 p_{[1]}, & \text{if } \ i = 1 \text{ and } E = 0; \\ 0, & \text{if } \ i = 0 \text{ and } E = e_1 \leq U; \\ +\infty, & \text{otherwise.} \end{cases}$$

**The recursive function:**

$$f_j(i, E) = \min\{f_{j-1}(i, E - e_j), f_{j-1}(i - 1, E) + w_j(p_{[1]} + \cdots + p_{[i]})\}.$$

**The optimal value** is given by $\min\{f_n(i, E) : 0 \leq i \leq n, 0 \leq E \leq U\}$.

**Theorem 13.** Algorithm DP8 solves problem $1|PPT, e(R) \leq U|TWC(A)+$ in $O(n^2U)$ time.

**Proof.** The correctness of algorithm DP8 is guaranteed by the above discussion. The recursive function has at most $O(n^2U)$ states and each iteration costs constant time. Thus, the running time of algorithm DP7 is bounded by $O(n^2U)$. □

Similar to the discussion in Subsection 4.1, by using the vector trimming technique, we can obtain an FPTAS for problem $1|PPT, e(R) \leq U|TWC(A)$.

## 5 Conclusions and future research

In this paper, we first study six single-machine scheduling problems with rejection and position-dependent parameters. The task is to minimize the sum of the total weighted completion time of accepted jobs and the total rejection cost of rejected jobs. We show that all six problems can be solved in polynomial time. Furthermore, we also consider two constrained problems and show that both of them are binary NP-hard. Finally, by using the vector trimming technique, an FPTAS can be obtained for each constrained problem.

In future research, an interesting direction is to consider the online versions of these problems. In addition, we also plan to extend these problems into parallel machine scheduling or batch processing scheduling in the future.

## Acknowledgments

The authors would like to thank two anonymous reviewers for their many suggestions and comments that helped improve the paper presentation. This research was supported by NSFCs (12271491, 11901168 and 11971443).

## References

[1] A. Agnetis, G. Mosheiov, Scheduling with job rejection and position-dependent processing times on proportionate flowshops. Optimization Letters 11, 885-892, 2017.

[2] A. Bachman, A. Janiak, Scheduling jobs with position-dependent processing times. Journal of the Operational Research Society 55, 257-264, 2004.

[3] Y. Bartal, S. Leonardi, A.M. Spaccamela, J. Stougie, Multi-processor scheduling with rejection. SIAM Journal on Discrete Mathematics 13, 64-78, 2000.

[4] Z.G. Cao, Z. Wang, Y.Z. Zhang, S.P. Liu, On several scheduling problems with rejection or discretely compressible processing times. Lecture Notes in Computer Science 3959, 90-98, 2006.

[5] Y.R. Chen, Z.L. Guan, Y.C. Tsai, F.D. Chou, Single machine scheduling problem with a weight-modifying activity to minimize the total weighted completion time. IEEE Access 10, 45444-45456, 2022.

[6] D.W. Engels, D.R. Karger, S.G. Kolliopoulos, S. Sengupta, R.N. Uma, J. Wein, Techniques for scheduling with rejection. Journal of Algorithms 49, 175-191, 2003.

[7] M.R. Garey, D.S. Johnson. Computers and Intractablity: A Guide to the Theory of NP-Completeness. Freeman, San Francisco, CA, 1979.

[8] C. Jiang, D.X. Zou, D.Y. Bai D, J.B. Wang, Proportionate flowshop scheduling with position-dependent weights. Engineering Optimization 52, 37-52, 2020.

[9] Kahlbacher H (1992) Termin-und Ablaufplanung-ein analytischer Zugang. Ph.D. thesis, University of Kaiserslautern

[10] H.J. Kim, B.I. Kim, Optimal sequence for single server scheduling incorporating a rate-modifying activity under job-dependent linear deterioration. European Journal of Operational Research 298, 439-450, 2022.

[11] Y.J. Kim, J.W. Jang, D.S. Kim, B.S. Kim, Batch loading and scheduling problem with processing time deterioration and rate-modifying activities. International Journal of Production Research 60, 1600-1620, 2022.

[12] C.-Y. Lee, V.J. Leon, Machine scheduling with a rate-modifying activity. European Journal of Operational Research 128, 119-128, 2001

[13] G. Mosheiov, D. Oron, A note on scheduling a rate modifying activity to minimize total late work. Computers and Industrial Engineering 154, 107138, 2021.

[14] G. Mosheiov, D. Oron, Scheduling problems with a weight-modifying-activity. Annals of Operations Research 295, 737-745, 2020.

[15] D. Oron, Two-agent scheduling problems under rejection budget constraints. Omega 102, 102313, 2021.

[16] D. Shabtay, N. Gaspar, M. Kaspi, A survey on off-line scheduling with rejection. Journal of Scheduling 16, 3-28, 2013.

[17] P. Schuurman, G.J. Woeginger. Approximation Schemes - A Tutorial. Lectures on Scheduling, LA Wolsey, 2009.

[18] D.J. Wang, Y.Q. Yin, Y. Jin, Parallel-machine rescheduling with job unavailability and rejection. Omega 81, 246-260, 2018.

[19] J.B. Wang, B. Cui, P. Ji, W.W. Liu, Research on single-machine scheduling with position-dependent weights and past-sequence-dependent delivery times. Journal of Combinatorial Optimization 41, 290-303, 2021.

[20] L.O. Whitaker, Integrated production and maintenance activities. M.S. Thesis, Department of Industrial Engineering, Texas A&M University, College Station, TX, 1996.

[21] G.J. Woeginger, When does a dynamic programming formulation guarantee the exitence of an FPTAS? INFORMS Journal of Computting 12, 57-74, 2000.

[22] L.Q. Zhang, L.F. Lu, J.J. Yuan, Single-machine scheduling under the job rejection constraint. Theoretical Computer Science 411, 1877-1882, 2010.

[23] X.G. Zhang, G.L. Yan, W.Z Huang, G.C. Tang, Single-machine scheduling problems with time and position dependent processing times. Annals of Operations Research 186, 345-356, 2011.

[24] Z.G. Zhu, F.F Zheng, C.B. Chu, Multitasking scheduling problems with a rate-modifying activity. International Journal of Production Research 55, 296-312, 2017.

*Received 19.02.2023, Accepted 12.09.2023*