mathematics

MDPI

Article

Single Machine Scheduling Proportionally Deteriorating Jobs with Ready Times Subject to the Total Weighted Completion Time Minimization

Zheng-Guo Lv, Li-Han Zhang, Xiao-Yuan Wang and Ji-Bo Wang *

School of Computer, Shenyang Aerospace University, Shenyang 110136, China;
lvzhengguo@stu.sau.edu.cn (Z.-G.L.); zhanglihan@stu.sau.edu.cn (L.-H.Z.); 20012302@sau.edu.cn (X.-Y.W.)
* Correspondence: wangjibo@sau.edu.cn

Abstract: In this paper, we investigate a single machine scheduling problem with a proportional job deterioration. Under release times (dates) of jobs, the objective is to minimize the total weighted completion time. For the general condition, some dominance properties, a lower bound and an upper bound are given, then a branch-and-bound algorithm is proposed. In addition, some meta-heuristic algorithms (including the tabu search (TS), simulated annealing (SA) and heuristic (NEH) algorithms) are proposed. Finally, experimental results are provided to compare the branch-and-bound algorithm and another three algorithms, which indicate that the branch-and-bound algorithm can solve instances of 40 jobs within a reasonable time and that the NEH and SA are more accurate than the TS.

Keywords: scheduling; single machine; proportional job deterioration; release dates; total weighted completion time

MSC: 90B35

check for updates

Citation: Lv, Z.-G.; Zhang, L.-H.; Wang, X.-Y.; Wang, J.-B. Single Machine Scheduling Proportionally Deteriorating Jobs with Ready Times Subject to the Total Weighted Completion Time Minimization. *Mathematics* **2024**, *12*, 610. https://doi.org/10.3390/math12040610

Academic Editor: Pavel Novoa-Hernández

Received: 7 January 2024
Revised: 30 January 2024
Accepted: 2 February 2024
Published: 19 February 2024

1. Introduction

The classical scheduling models assume that the job processing time is a fixed constant. However, in many real-life situations (e.g., scheduling derusting operations (see Gawiejnowicz et al. [1]), and timely medical treatment (Wu et al. [2])), the job processing time is an increasing function of its starting time, i.e., this is the deterioration effect (for more detail on deteriorating jobs, see Mosheiov [3], Gawiejnowicz [4], Oron [5]). Lee et al. [6] and Wang et al. [7] considered single-machine two-agent problems with deteriorating jobs. Wu et al. [8] scrutinized single-machine scheduling with a truncated linear deteriorating effect. For the makespan minimization with ready times, they proposed a branch-and-bound algorithm and some heuristic algorithms. Yin and Kang [9] studied flow shop scheduling with proportional deterioration. They proved that some special cases of makespan minimization are polynomially solvable. Pei et al. [10] considered the single serial-batching machine with deteriorating jobs. Jafari and Lotfi [11] studied single-machine scheduling with deteriorating jobs. For the maximum tardiness minimization, they proposed branch-and-bound and heuristic algorithms. Miao and Zhang [12] discussed the parallel-machine scheduling with step-deteriorating jobs. For the makespan minimization, some NP-hard results were given. Huang [13] investigated the single-machine scheduling problem with deterioration effects. Under the group technology, they proved that the bicriterion minimization problem remains polynomially solvable. Liu et al. [14] studied the single-machine makespan minimization problem with deterioration effects. Under the group technology and ready times, they proposed heuristic and branch-and-bound algorithms. Sun and Geng [15] investigated single-machine scheduling with deteriorating effects. Under machine maintenance, they showed that the makespan and sum of job completion times

minimizations can be solved in polynomial time. Cheng et al. [16] explored single-machine total completion time minimization with step-deteriorating jobs. They showed that this problem is binary NP-hard, and proposed a dynamic programming algorithm to solve the problem. Li and Lu [17] explored single-machine parallel-batch scheduling problem with job rejection. Under deterioration effects, they proved that the makespan (total weighted completion time) minimization is NP-complete under the total rejection penalty is limited. They also proposed dynamic programming algorithms and FPTASs (i.e., fully polynomial time approximation schemes) to solve the problem. Zhang et al. [18] scrutinized parallel-machine scheduling with deterioration effects. Under machine maintenance activities of the non-resumable case and resumable case, the goal is to minimize the expected sum of completion times. They determined the time complexity of various cases, and proposed pseudo-polynomial time algorithms. Huang et al. [19] considered due window assignment scheduling with deteriorating jobs. He et al. [20] considered parallel-machine problems with deteriorating jobs. Under processor maintenance activities, the objective is to minimize the total completion time (machine load). Qian and Han [21] and Qian and Zhan [22] scrutinized single-machine scheduling with proportional job deterioration. Under due-date and due-window assignments, they showed that some earliness-tardiness problems remain polynomially solvable. Miao et al. [23] studied parallel-machine scheduling with step-deterioration effect, where the job deteriorating dates are identical. They proved that the total (weighted) completion time minimization is NP-hard in the strong sense. They also showed that some special cases of the problem are polynomially solvable. Jia et al. [24] and Sun et al. [25] considered single machine scheduling with deteriorating jobs and a maintenance activity. Miao et al. [26] investigated single machine scheduling with deteriorating jobs and delivery times. Wang et al. [27] studied single-machine resource allocation scheduling with deterioration effect. Zhang et al. [28] delved into due-window scheduling with linear proportional deterioration. Shabtay and Mor [29] discussed the proportionate flow shop problems with step-deteriorating. They proved that the makespan and total load minimizations are NP-hard. A book (resp. survey) on time-dependent (deterioration effect) scheduling problems can be found in Gawiejnowicz [30] (resp. Gawiejnowicz [31]).

Recently, Miao [32] explored a single-machine scheduling problem with the proportional job deterioration. Under different ready times (i.e., release dates), she showed that the total weighted completion time minimization is binary NP-hard. For a special condition, Miao [32] proved that this problem can be solved in polynomial time. For the importance of ready times (please refer to Zhong et al. [33]; Bai et al. [34]; Qian et al. [35]) and deteriorating jobs, in this study, we continue the work of Miao [32], but deal with a general case of Miao [32]. The contributions of this article are: (1) we consider the single machine scheduling with proportionally deteriorating jobs; (2) under ready times, we provide the optimality analysis for the total weighted completion time minimization; (3) under the optimal properties of an optimal schedule, we propose some solution algorithms (including the branch-and-bound, tabu search, simulated annealing and heuristic algorithms).

The remainder of this article is organized as follows. Section 2 presents the problem formulation. In Sections 3 and 4, the branch-and-bound and complex algorithms are proposed. Section 5 gives the computational experiments of randomly generated instances. In the last Section, we conclude this article.

2. Problem Formulation

A set of n jobs, $\widehat{N} = \{J_1, J_2, \ldots, J_n\}$, is to be processed (scheduled) on a single machine, and all the jobs are available at time $t_0 > 0$. As in Mosheiov [3], Oron [5] and Miao [32], the following model with the proportional job deterioration will be addressed, i.e., the actual processing time $\hat{p}_j$ of job J_j $(j = 1, 2, \ldots, n)$ is

$$\hat{p}_j = \hat{b}_j \hat{s}_j, \tag{1}$$

where $\hat{b}_j$ is the deterioration rate of job J_j (i.e., the unit growth rate of its starting processing time), and $\hat{s}_j$ is its starting time. Let $\overline{C}_j = \overline{C}_j(\Psi)$ be the completion time of job J_j under some schedule Ψ. The objective is to find a schedule that minimizes the total weighted completion time $\widetilde{TWCT} = \sum_{j=1}^{n} \mu_j \overline{C}_j$, where μ_j is the weight of job J_j. Using the three-field notation (see Gawiejnowicz [30,31]), for problem classification, the problem can be represented as $1|\hat{p}_j = \hat{b}_j\hat{s}_j, r_j|\widetilde{TWCT}$, where $r_j \geq 0$ is the release time (date) of job J_j. The comparison with proportional job deterioration is given in Table 1.

Table 1. Summary results of proportional job deterioration.

Problem	Complexity or Algorithms	Reference
$1\|\hat{p}_j = \hat{b}_j\hat{s}_j, con, deliv-time\|\sum_{j=1}^{n}(\alpha E_j + \beta T_j + \gamma d)\}$	$O(n \log n)$	Qian and Han [21]
$1\|\hat{p}_j = \hat{b}_j\hat{s}_j, slk, deliv-time\|\sum_{j=1}^{n}(\alpha E_j + \beta T_j + \gamma q)\}$	$O(n \log n)$	Qian and Han [21]
$1\|\hat{p}_j = \hat{b}_j\hat{s}_j, dif, deliv-time\|\sum_{j=1}^{n}(\alpha E_j + \beta T_j + \gamma d_j)\}$	$O(n \log n)$	Qian and Han [21]
$1\|\hat{p}_j = \hat{b}_j\hat{s}_j, conw, deliv-time\|\sum_{j=1}^{n}(\alpha E_j + \beta T_j + \gamma d' + \delta(d'' - d'))$	$O(n \log n)$	Qian and Zhan [22]
$1\|\hat{p}_j = \hat{b}_j\hat{s}_j, slkw, deliv-time\|\sum_{j=1}^{n}(\alpha E_j + \beta T_j + \gamma q' + \delta(q'' - q'))$	$O(n \log n)$	Qian and Zhan [22]
$1\|\hat{p}_j = \hat{b}_j\hat{s}_j\|\overline{C}_{\max}$	$O(n)$	Mosheiov [3]
$1\|\hat{p}_j = \hat{b}_j\hat{s}_j\|\sum_{j=1}^{n}\overline{C}_j$	$O(n \log n)$, $\hat{b}_j \uparrow$	Mosheiov [3]
$1\|\hat{p}_j = \hat{b}_j\hat{s}_j\|L_{\max}$	$O(n \log n)$, $d_j \uparrow$	Mosheiov [3]
$1\|\hat{p}_j = \hat{b}_j\hat{s}_j\|\widetilde{TWCT}$	$O(n \log n)$, $\frac{\hat{b}_j}{\mu_j(1+\hat{b}_j)} \uparrow$	Mosheiov [3]
$1\|\hat{p}_j = \hat{b}_j\hat{s}_j\|\sum_{j=1}^{n}\sum_{h=j+1}^{n}(\overline{C}_h - \overline{C}_j)$	open problem	Oron [5]
$1\|\hat{p}_j = \hat{b}_j\hat{s}_j\|f_{\max}$	$O(n)$	Gawiejnowicz [30] (Theorem 7.93)
$1\|\hat{p}_j = \hat{b}_j\hat{s}_j, r_j\|\widetilde{TWCT}$	NP-hard	Miao [32]
$1\|\hat{p}_j = \hat{b}_j\hat{s}_j, \hat{b}_j = \hat{b}, r_h \leq r_j \Rightarrow \mu_h \geq \mu_j\|\widetilde{TWCT}$	$O(n \log n)$, $r_j \uparrow$	Miao [32]
$1\|\hat{p}_j = \hat{b}_j\hat{s}_j, r_j\|\widetilde{TWCT}$	BB and Meta-heuristic algorithms	This paper

$\alpha \geq 0, \beta \geq 0, \gamma \geq 0, \delta \geq 0$ are given integers, E_j (resp. T_j) is the earliness (resp. tardiness) of J_j, $\uparrow$ denotes non-decreasing order, d_j is the due-date of J_j (*con* denotes the common due-date, i.e., $d_j = d$; *slk* denotes the slack due-date, i.e., $d_j = \hat{p}_j + q$; *dif* denotes the different due-dates), $[d'_j, d''_j]$ is the due-window of J_j (*conw* denotes the common due-window, i.e., $d'_j = d'$ and $d''_j = d''$; *slkw* denotes the slack due-window, i.e., $d'_j = \hat{p}_j + q'$ and $d''_j = \hat{p}_j + q''$, $\overline{C}_{\max}$ (resp. $L_{\max}$, $f_{\max}$) denotes makespan (resp. lateness, maximum cost), $d'' - d'$ (resp. $q'' - q'$) is the size of the common (resp. slack) due-window, deliv-time denotes a delivery time

3. Branch-and-Bound Algorithm

Miao [32] proved that the problem $1|\hat{p}_j = \hat{b}_j\hat{s}_j, r_j|\widetilde{TWCT}$ is binary NP-hard. Hence, a branch-and-bound algorithm (where we need some dominance properties, a lower bound and an upper bound) is a good way to solve $1|\hat{p}_j = \hat{b}_j\hat{s}_j, r_j|\widetilde{TWCT}$.

3.1. Dominance Properties

The following dominance properties can efficiently disregard numerous nodes and reduce the computation load. Let $\Psi_1 = (\tau, J_h, J_j, \tau')$ and $\Psi_2 = (\tau, J_j, J_h, \tau')$ be the orders of schedules, i.e., the jobs in the bracket represent an order of schedule, where τ and τ' are partial sequences. Let the completion time of the last job in τ be s, to show that Ψ_1 dominates Ψ_2 (i.e., the objective value of the schedule $\Psi_1 = (\tau, J_h, J_j, \tau')$ is less than or equal to the schedule $\Psi_2 = (\tau, J_j, J_h, \tau')$), it is sufficient to show that $\overline{C}_j(\Psi_1) \leq \overline{C}_h(\Psi_2)$, and $\mu_h\overline{C}_h(\Psi_1) + \mu_j\overline{C}_j(\Psi_1) \leq \mu_j\overline{C}_j(\Psi_2) + \mu_h\overline{C}_h(\Psi_2)$.

Proposition 1. *For any two jobs J_h and J_j, if $r_h = r_j$, $\frac{\hat{b}_h}{\mu_h(1+\hat{b}_h)} \leq \frac{\hat{b}_j}{\mu_j(1+\hat{b}_j)}$, then Ψ_1 dominates Ψ_2.*

Proof. Under $\Psi_1 = (\tau, J_h, J_j, \tau')$, and $\Psi_2 = (\tau, J_j, J_h, \tau')$, we have

$$\overline{C}_h(\Psi_1) = \max\{s, r_h\} + \hat{b}_h \max\{s, r_h\} = \max\{s(1+\hat{b}_h), r_h(1+\hat{b}_h)\}, \tag{2}$$

$$\overline{C}_j(\Psi_1) = \max\{\hat{C}_h(\Psi_1), r_j\}(1+\hat{b}_j) = \max\{s(1+\hat{b}_h)(1+\hat{b}_j), r_h(1+\hat{b}_h)(1+\hat{b}_j), r_j(1+\hat{b}_j)\}, \tag{3}$$

$$\overline{C}_j(\Psi_2) = \max\{s(1+\hat{b}_j), r_j(1+\hat{b}_j)\}, \tag{4}$$

$$\overline{C}_h(\Psi_2) = \max\{s(1+\hat{b}_j)(1+\hat{b}_h), r_j(1+\hat{b}_j)(1+\hat{b}_h), r_h(1+\hat{b}_h)\}. \tag{5}$$

If $r_h = r_j \leq s$, we have $\overline{C}_j(\Psi_1) = \overline{C}_h(\Psi_2) = s(1+\hat{b}_j)(1+\hat{b}_h)$, and

$$\begin{aligned} & \mu_h\overline{C}_h(\Psi_1) + \mu_j\overline{C}_j(\Psi_1) - \mu_j\overline{C}_j(\Psi_2) - \mu_h\overline{C}_h(\Psi_2) \\ = \;& s\mu_h(1+\hat{b}_h) + s\mu_j(1+\hat{b}_h)(1+\hat{b}_j) - s\mu_j(1+\hat{b}_j) - s\mu_h(1+\hat{b}_h)(1+\hat{b}_j) \\ = \;& s\mu_j\mu_h(1+\hat{b}_j)(1+\hat{b}_h)\left[\frac{\hat{b}_h}{\mu_h(1+\hat{b}_h)} - \frac{\hat{b}_j}{\mu_j(1+\hat{b}_j)}\right] \\ \leq \;& 0. \end{aligned}$$

If $r_h = r_j > s$, we have $\overline{C}_j(\Psi_1) = \overline{C}_h(\Psi_2) = r(1+\hat{b}_j)(1+\hat{b}_h)$, and

$$\begin{aligned} & \mu_h\overline{C}_h(\Psi_1) + \mu_j\overline{C}_j(\Psi_1) - \mu_j\overline{C}_j(\Psi_2) - \mu_h\overline{C}_h(\Psi_2) \\ = \;& r\mu_j\mu_h(1+\hat{b}_j)(1+\hat{b}_h)\left[\frac{\hat{b}_h}{\mu_h(1+\hat{b}_h)} - \frac{\hat{b}_j}{\mu_j(1+\hat{b}_j)}\right] \\ \leq \;& 0, \end{aligned}$$

the result follows. □

Similarly, we have the following propositions:

Proposition 2. *For any two jobs J_i and J_j, if $\hat{b}_h = \hat{b}_j$, $r_h \leq r_j$, $\mu_h \geq \mu_j$, then Ψ_1 dominates Ψ_2.*

Proposition 3. *For any two jobs J_i and J_j, if $\hat{b}_h \leq \hat{b}_j$, $r_h \leq r_j$, $\mu_h = \mu_j$, then Ψ_1 dominates Ψ_2.*

3.2. Lower Bound

Lemma 1. *(Rau [36], Kelly [37]) Term $\sum_{j=1}^{n} \mu_j \prod_{h=k+1}^{j}(1+\hat{b}_h)$ can be minimized by the non-decreasing order of $\frac{\hat{b}_j}{\mu_j(1+\hat{b}_j)}$, i.e., $\frac{\hat{b}_{<1>}}{\mu_{<1>}(1+\hat{b}_{<1>})} \leq \frac{\hat{b}_{<2>}}{\mu_{<2>}(1+\hat{b}_{<2>})} \leq \ldots \leq \frac{\hat{b}_{<n>}}{\mu_{<n>}(1+\hat{b}_{<n>})}$.*

Suppose $\Theta = \{\Psi_{ps}, \Psi_{pu}\}$ is a set of jobs in which Ψ_{ps} is the scheduled part, Ψ_{pu} is a unscheduled part, and there are ζ jobs in Ψ_{ps}, then, the completion time of the $(\zeta+1)$th job is

$$\begin{aligned} \overline{C}_{[\zeta+1]}(\Psi_{pu}) &= \max\{\overline{C}_{[\zeta]}(\Psi), r_{[\zeta+1]}\} + \hat{b}_{[\zeta+1]} \max\{\overline{C}_{[\zeta]}(\Psi), r_{[\zeta+1]}\} \\ &= \max\{\overline{C}_{[\zeta]}(\Psi), r_{[\zeta+1]}\}(1+\hat{b}_{[\zeta+1]}), \end{aligned}$$

where $[\zeta]$ denotes some job scheduled in ζth position. Similarly,

$$\begin{aligned}
\overline{C}_{[\zeta+2]}(\Psi_{pu}) &= \max\{\overline{C}_{[\zeta+1]}(\Psi), r_{[\zeta+2]}\}(1+\hat{b}_{[\zeta+2]}) \\
&= \max\{\overline{C}_{[\zeta]}(\Psi)(1+\hat{b}_{[\zeta+1]})(1+\hat{b}_{[\zeta+2]}), r_{[\zeta+1]}(1+\hat{b}_{[\zeta+1]})(1+\hat{b}_{[\zeta+2]}), \\
&\quad r_{[\zeta+2]}(1+\hat{b}_{[\zeta+2]})\}, \\
&\vdots \\
\overline{C}_{[n]}(\Psi_{pu}) &= \max\{\overline{C}_{[\zeta]}(\Psi)\prod_{h=\zeta+1}^{n}(1+\hat{b}_{[h]}), r_{[\zeta+1]}\prod_{h=\zeta+1}^{n}(1+\hat{b}_{[h]}), r_{[\zeta+2]}\prod_{h=\zeta+2}^{n}(1+\hat{b}_{[h]}), \\
&\quad \ldots, r_{[n]}(1+\hat{b}_{[n]})\}.
\end{aligned}$$

The $\widetilde{TWCT}$ is

$$\begin{aligned}
\widetilde{TWCT} &= \sum_{j=1}^{n}\mu_j\overline{C}_j \\
&= \sum_{j=1}^{\zeta}\mu_j\overline{C}_j(\Psi_{ps}) + \sum_{j=\zeta+1}^{n}\mu_{[j]}\overline{C}_{[j]}(\Psi_{pu}) \qquad (6) \\
&\geq \sum_{j=1}^{\zeta}\mu_j\overline{C}_j(\Psi_{ps}) + \overline{C}_{[\zeta]}(\Psi_{ps})\sum_{j=\zeta+1}^{n}\mu_{[j]}\prod_{h=k+1}^{j}(1+\hat{b}_{[h]}).
\end{aligned}$$

Obviously, the terms $\sum_{j=1}^{\zeta}\mu_j\overline{C}_j(\Psi_{ps})$ and $\overline{C}_{[\zeta]}(\Psi_{ps})$ on the right hand side of Equation (6) are fixed. From Lemma 1, we obtain the first lower bound

$$\widehat{LB_1} = \sum_{j=1}^{\zeta}\mu_j\overline{C}_j(\Psi_{ps}) + \overline{C}_{[\zeta]}(\Psi_{ps})\sum_{j=\zeta+1}^{n}\mu_{<j>}\prod_{h=k+1}^{j}(1+\hat{b}_{<h>}), \qquad (7)$$

where $\frac{\hat{b}_{<\zeta+1>}}{\mu_{<\zeta+1>}(1+\hat{b}_{<\zeta+1>})} \leq \frac{\hat{b}_{<\zeta+2>}}{\mu_{<\zeta+2>}(1+\hat{b}_{<\zeta+2>})} \leq \ldots \leq \frac{\hat{b}_{<n>}}{\mu_{<n>}(1+\hat{b}_{<n>})}$ is a nondecreasing order of $\frac{\hat{b}_j}{\mu_j(1+\hat{b}_j)}$ for the remaining unscheduled jobs.

However, if the ready times are large, then this lower bound may not be tight. To overcome this situation, we need to take the ready times into consideration. In general, we have

$$\begin{aligned}
\widetilde{TWCT} &= \sum_{j=1}^{\zeta}\mu_j\overline{C}_j(\Psi_{ps}) + \sum_{j=\zeta+1}^{n}\mu_{[j]}\overline{C}_{[j]}(\Psi_{pu}) \\
&\geq \sum_{j=1}^{\zeta}\mu_j\overline{C}_j(\Psi_{ps}) + \sum_{j=\zeta+1}^{n}\mu_{[j]}r_{[j]}(1+b_{[j]}). \qquad (8)
\end{aligned}$$

It is noticed that the first term (i.e., $\sum_{j=1}^{\zeta}\mu_j\overline{C}_j(\Psi_{pu})$) on the right hand side of Equation (8) is fixed, and $\sum_{j=\zeta+1}^{n}\mu_{[j]}r_{[j]}(1+\hat{b}_{[j]}) = \sum_{j\in\Psi_{pu}}\mu_j r_j(1+\hat{b}_j)$ is a constant. Consequently, we obtain the second lower bound

$$\widehat{LB_2} = \sum_{j=1}^{\zeta}\mu_j\overline{C}_j(\Psi_{ps}) + \sum_{j\in\Psi_{pu}}\mu_j r_j(1+\hat{b}_j). \qquad (9)$$

In addition, we have

$$\widetilde{TWCT} \geq \sum_{j=1}^{\zeta}\mu_j\overline{C}_j(\Psi_{ps}) + r_{[\zeta+1]}\sum_{j=\zeta+1}^{n}\mu_{[j]}\prod_{h=\zeta+1}^{j}(1+\hat{b}_{[h]}). \qquad (10)$$

Let $r_{\min} = \min\{r_j | j \in \Psi_{pu}\}$, from Equation (10), we have the third lower bound:

$$\widehat{LB_3} = \sum_{j=1}^{\zeta} \mu_j \overline{C}_j(\Psi_{ps}) + r_{\min} \sum_{j=\zeta+1}^{n} \mu_{<j>} \prod_{h=\zeta+1}^{j} (1 + \hat{b}_{<h>}). \tag{11}$$

where $\frac{\hat{b}_{<\zeta+1>}}{\mu_{<\zeta+1>}(1+\hat{b}_{<\zeta+1>})} \leq \frac{\hat{b}_{<\zeta+2>}}{\mu_{<\zeta+2>}(1+\hat{b}_{<\zeta+2>})} \leq \ldots \leq \frac{\hat{b}_{<n>}}{\mu_{<n>}(1+\hat{b}_{<n>})}$.

To make the lower bound tighter, the maximum value of Equations (7), (9) and (11) is selected as a lower bound for $1|\hat{p}_j = \hat{b}_j\hat{s}_j, r_j|\widetilde{TWCT}$, i.e., the lower bound for $1|\hat{p}_j = \hat{b}_j\hat{s}_j, r_j|\widetilde{TWCT}$ is

$$\widehat{LB} = \max\{\widehat{LB_1}, \widehat{LB_2}, \widehat{LB_3}\}. \tag{12}$$

3.3. Upper Bound

From Section 3.1, the following Algorithm 1 is given as the upper bound (denoted by UB) of problem $1|\hat{p}_j = \hat{b}_j\hat{s}_j, r_j|\widetilde{TWCT}$.

Algorithm 1: UB

(Step 1). Arrange the jobs by the non-decreasing order of r_j, i.e., $r_1 \leq r_2 \leq \ldots \leq r_n$ (call the obtained schedule Ψ^1).
(Step 2). Arrange the jobs by the non-decreasing order of $\hat{b}_j$, i.e., $\hat{b}_1 \leq \hat{b}_2 \leq \ldots \leq \hat{b}_n$ (call the obtained schedule Ψ^2).
(Step 3). Arrange the jobs by the non-decreasing order of $\frac{\hat{b}_j}{\mu_j(1+\hat{b}_j)}$, i.e., $\frac{\hat{b}_1}{\mu_1(1+\hat{b}_1)} \leq \frac{\hat{b}_2}{\mu_2(1+\hat{b}_2)} \leq \ldots \leq \frac{\hat{b}_n}{\mu_n(1+\hat{b}_n)}$ (call the obtained schedule Ψ^3).
(Step 4). Arrange the jobs by the non-increasing order of μ_j, i.e., $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_n$ (call the obtained schedule Ψ^4).
(Step 5). Find the best schedule from schedules Ψ^1, Ψ^2, Ψ^3 and Ψ^4.

3.4. Branch-and-Bound Algorithm

From Sections 3.1–3.3, the standard branch-and-bound (denoted by BB) Algorithm 2 can be proposed as follows.

Algorithm 2: BB

Step 1. Use Algorithm 1 to obtain an initial solution (i.e., UB).
Step 2. In the κth level node, the first κ positions are occupied by κ specific jobs. Select one of the remaining $n - \kappa$ jobs for the node at level $\kappa + 1$.
Step 3. First apply Propositions 1–3, to eliminate the dominated partial schedules.
Step 4. Calculate the $\widehat{LB}$ for $\widetilde{TWCT}$ of the node. If the lower bound for an unfathomed partial schedule of jobs is larger than or equal to the value of $\widetilde{TWCT}$ of the initial solution, eliminate the node and all the nodes following it in the branch. Calculate the value $\widetilde{TWCT}$ of the completed schedule, if it is less than the initial solution, replace it as the new solution; otherwise, eliminate it.
Step 5. Continue to search all the nodes, and the remaining initial solution is an optimal schedule.

4. Meta-Heuristic Algorithms

4.1. Tabu Search

In this subsection, the tabu search (TS) in Algorithm 3 is used to solve $1|\hat{p}_j = \hat{b}_j\hat{s}_j, r_j|\widetilde{TWCT}$. The initial schedule used in the TS is chosen by using Algorithm 1, and the maximum number of iterations for the TS is set at $1000n$.

Algorithm 3: *TS*

Step 1. Let the tabu list be empty and the iteration number be zero.
Step 2. Set the initial sequence of the *TS* algorithm, calculate its objective function $\widetilde{TWCT}$ and set the current schedule as the best solution Ψ^*.
Step 3. Search the associated neighborhood of the current schedule and resolve if there is a schedule Ψ^{**} with the smallest objective function in associated neighborhoods and it is not in the tabu list.
Step 4. If Ψ^{**} is better than Ψ^*, then let $\Psi^* = \Psi^{**}$. Update the tabu list and the iteration number.
Step 5. If there is not a schedule in associated neighborhoods but it is not in the tabu list or the maximum number of iterations is reached, then output the local optimal schedule Ψ and objective function value $\widetilde{TWCT}$. Otherwise, update the tabu list and go to Step 3.

4.2. NEH Algorithm

Adapted from Nawaz et al. [38], the *NEH* Algorithm 4 can be adopt to solve $1|\hat{p}_j = \hat{b}_j\hat{s}_j, r_j|\widetilde{TWCT}$.

Algorithm 4: *NEH*

(Step 1). Arrange all the jobs by Algorithm 1.
(Step 2). Set $\vartheta = 2$. Select the first two jobs from the sorted list (by **Step 1**) and select the best of the two possible schedules. Do not change the relative positions of these two jobs with respect to each other in the remaining steps of the algorithm. Set $\vartheta = 3$.
(Step 3). Pick the job in the ϑth position of the list generated in **Step 1** and find the best schedule by placing it at all possible ϑ positions in the partial sequence found in the previous step, without changing the relative positions to each other of the already assigned jobs. The number of enumerations at this step is equal to ϑ.
(Step 4). If $\vartheta = n$, STOP; otherwise set $\vartheta = \vartheta + 1$ and go to Step 3.

4.3. Simulated Annealing

In this subsection, the simulated annealing (*SA*) in Algorithm 5 is proposed to solve$1|\hat{p}_j = \hat{b}_j\hat{s}_j, r_j|\widetilde{TWCT}$ (see Lai et al. [39] and Liu et al. [40]).

Algorithm 5: *SA*

Step 1. *Initial schedule can be obtained by Algorithm 1.*
Step 2. *Pairwise interchange (PI) neighborhood generation method will be used.*
Step 3. *When a new schedule is generated (by* **Step 2***), it is accepted if its objective function value (i.e., $\widetilde{TWCT}$) is smaller than that of the original schedule; otherwise, it is accepted with some probability that decreases as the process evolves. The probability of acceptance is generated from an exponential distribution,*

$$P(accept) = exp(-\psi \times \triangle G),$$

where ψ is a control parameter and $\triangle G$ is the change in the objective function value, ψ in the kth iteration is

$$\psi = \frac{k}{\eta}$$

and η is an experimental constant. After preliminary trials, $\eta = 1$ was used in our experiments. If the objective function value increases as a result of a random pairwise interchange, the new schedule is accepted when $P(accept) > \varsigma$, where ς is randomly sampled from the uniform distribution $U(0,1)$.
Step 4. *Stopping condition: 1000n iterations.*

4.4. A Mixed Integer Programming (MIP) Model

Let

$$x_{j,k} = \begin{cases} 1, & \text{if job } J_j \text{ is assigned to position } k, \\ 0, & \text{otherwise.} \end{cases}$$

The optimal schedule of $1|\hat{p}_j = \hat{b}_j\hat{s}_j, r_j|\widetilde{TWCT}$ can be written as the following MIP:

$$\text{Min} \quad \sum_{k=1}^{n} \mu_{[k]}\overline{C}_{[k]} \tag{13}$$

$$s.t. \quad \sum_{j=1}^{n} x_{j,k} = 1, k = 1, 2, \ldots, n, \tag{14}$$

$$\sum_{k=1}^{n} x_{j,k} = 1, j = 1, 2, \ldots, n, \tag{15}$$

$$\hat{b}_{[k]} = \sum_{j=1}^{r} x_{j,k}\hat{b}_j, k = 1, 2, \ldots, n, \tag{16}$$

$$r_{[k]} = \sum_{j=1}^{r} x_{j,k}r_j, k = 1, 2, \ldots, n, \tag{17}$$

$$\mu_{[k]} = \sum_{j=1}^{r} x_{j,k}\mu_j, k = 1, 2, \ldots, n, \tag{18}$$

$$S_{[k]} \geq r_{[k]}, k = 1, 2, \ldots, n, \tag{19}$$

$$S_{[k]} \geq C_{[k-1]}, k = 1, 2, \ldots, n, \tag{20}$$

$$C_{[k]} \geq S_{[k]}(1 + \hat{b}_j)x_{j,k}, j, k = 1, 2, \ldots, n, \tag{21}$$

where (13) is the objective cost of optimization, and (14) and (15) ensure that each job must be processed in a unique position; constraints (16)–(18) denote the deterioration rate the release time and the weight of some job at the kth position, respectively; constraint (19) means that the start time of the job at the kth position is not later than its release time; constraint (20) means that the start time of some job at the kth position is not later than the completion time of some job at the $k-1$st position; (21) is the completion time constraint.

5. Computational Results

In this section, computational experiments are conducted to evaluate the accuracy and efficiency of the UB, TS, NEH, SA and BB algorithms. Detailed programming and testing configurations are as follows.

- C++ version: Visual studio 2019, the max memory allowed was restricted to 64G, for the UB, TS, NEH, SA and BB algorithms.
- Testing computer: One desktop workstation with one AMD R7-4700H CPU (2.9–4.2 GHz, 8 cores, 16 threads) and 128G of physical memory.

The following test parameters are randomly generated:

1. $\hat{b}_j$ is uniformly distributed over [0.05, 0.1] (i.e., $\hat{b}_j \in U[0.05, 0.1]$), [0.1, 0.15] (i.e., $\hat{b}_j \in U[0.1, 0.15]$) and [0.05, 0.15] (i.e., $\hat{b}_j \in U[0.05, 0.15]$);
2. r_j is uniform integers distributed over [1, 50] (i.e., $r_j \in U[1, 50]$), [50, 100] (i.e., $r_j \in U[50, 100]$) and [1, 100] (i.e., $r_j \in U[1, 100]$);
3. μ_j is uniform integers distributed over [1, 10] (i.e., $\mu_j \in U[1, 10]$);
4. n = 15, 20, 25, 30, 35, 40.

For each combination of parameters, 20 randomly generated instances were evaluated, and there were 1080 instances tested in total. For the BB, the average and maximum numbers of nodes and the average and maximum time (in millisecond) were recorded. For the heuristics, the average and maximum error were recorded. The error of the solution produced by a heuristic was calculated as $\frac{\widetilde{TWCT}(HA)}{\widetilde{TWCT}(OPT)}$, where $HA \in \{UB, TS, NEH, SA\}$, and OPT is the optimal schedule by BB. For the CPU time of the BB, from Tables 2–4, it should be noted that the CPU time increases when n becomes bigger, and the maximum CPU time (i.e., $n = 40$ and $r_j \in U[1, 100]$) is 39,870,200 milliseconds. In addition, from

Tables 2–4, we obtain that the CPU time becomes lesser for $r_j \in U[50, 100]$. From Tables 5–7, we obtain that error results of the NEH and SA appear to be more accurate than the UB and TS.

Table 2. CPU Results for $\hat{b}_j \in U[0.05, 0.1]$.

	r_j	*BB*		*UB*		*TS*		*NEH*		*SA*	
n		**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**
15	$r_j \in U[1, 50]$	122.60	564.00	4.85	9.00	5741.75	54,925.00	2.75	8.00	6.50	11.00
	$r_j \in U[50, 100]$	17.10	36.00	6.30	11.00	3667.05	64,462.00	5.90	73.00	8.90	16.00
	$r_j \in U[1, 100]$	75.70	359.00	5.70	9.00	5987.75	61,622.00	2.20	5.00	8.20	26.00
20	$r_j \in U[1, 50]$	29,084.40	151,879.00	4.40	6.00	25,536.40	165,107.00	3.70	24.00	14.35	22.00
	$r_j \in U[50, 100]$	245.60	641.00	6.40	9.00	29,178.70	212,358.00	3.35	5.00	15.65	20.00
	$r_j \in U[1, 100]$	27,701.60	199,894.00	5.00	11.00	19,955.30	192,863.00	3.10	5.00	18.30	78.00
25	$r_j \in U[1, 50]$	512,311.00	3,726,670.00	4.00	9.00	15,949.60	289,186.00	4.60	22.00	16.55	23.00
	$r_j \in U[50, 100]$	1990.10	5876.00	5.25	9.00	40,320.75	433,996.00	5.30	16.00	21.20	31.00
	$r_j \in U[1, 100]$	122,106.00	106,339.00	4.05	6.00	67,496.10	339,880.00	3.75	6.00	19.40	28.00
30	$r_j \in U[1, 50]$	174,832.00	1,329,940.00	4.70	15.00	145,316.60	749,768.00	5.15	9.00	21.70	40.00
	$r_j \in U[50, 100]$	9390.05	17,815.00	4.25	10.00	59,158.80	567,654.00	7.60	50.00	24.30	30.00
	$r_j \in U[1, 100]$	6,343,906.00	47,881,400.00	3.95	10.00	148,253.20	722,566.00	5.50	10.00	26.6	66.00
35	$r_j \in U[1, 50]$	3,148,041.85	9,034,400.00	8.85	71.00	117,052.00	1,359,553.00	6.60	21.00	28.15	46.00
	$r_j \in U[50, 100]$	44,630.25	70,439.00	2.35	4.00	57,802.85	1,054,277.00	6.00	16.00	33.00	41.00
	$r_j \in U[1, 100]$	4,296,541.70	30,627,000.00	3.40	7.00	216,026.10	1,075,473.00	4.80	17.00	22.80	27.00
40	$r_j \in U[1, 50]$	2,150,283.40	10,695,000.00	2.50	8.00	89,874.20	850,713.00	5.50	14.00	29.20	49.00
	$r_j \in U[50, 100]$	102,262.90	159,458.00	3.25	7.00	92,418.60	896,981.00	5.90	11.00	23.55	38.00
	$r_j \in U[1, 100]$	9,961,003.10	39,870,200.00	3.60	13.00	58,957.00	711,500.00	4.90	8.00	34.40	47.00

Table 3. CPU Results for $\hat{b}_j \in U[0.1, 0.15]$.

	r_j	*BB*		*UB*		*TS*		*NEH*		*SA*	
n		**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**
15	$r_j \in U[1, 50]$	102.85	451.00	1.20	2.00	10,484.90	67,690.00	1.30	3.00	7.30	9.00
	$r_j \in U[50, 100]$	23.00	47.00	1.35	3.00	7173.15	68,005.00	1.60	3.00	7.15	9.00
	$r_j \in U[1, 100]$	243.90	3067.00	1.45	3.00	430.75	450.00	1.30	4.00	7.30	11.00
20	$r_j \in U[1, 50]$	14,573.35	168,203.00	1.25	2.00	8838.95	160,111.00	1.75	2.00	10.15	11.00
	$r_j \in U[50, 100]$	237.85	509.00	1.35	3.00	28,075.60	215,786.00	1.75	2.00	10.70	18.00
	$r_j \in U[1, 100]$	3487.35	19,283.00	2.10	5.00	43,418.90	215,419.00	1.95	3.00	13.65	17.00
25	$r_j \in U[1, 50]$	193,834.35	3,177,120.00	1.50	4.00	22,294.45	414,487.00	2.20	3.00	14.65	23.00
	$r_j \in U[50, 100]$	1450.20	2885.00	1.45	3.00	32,890.65	315,447.00	2.10	3.00	14.15	15.00
	$r_j \in U[1, 100]$	17,100.45	192,282.00	1.25	3.00	17,350.00	315,937.00	2.00	3.00	14.25	17.00
30	$r_j \in U[1, 50]$	87,182.50	536,278.00	1.60	3.00	56,730.20	545,618.00	2.95	4.00	18.40	21.00
	$r_j \in U[50, 100]$	12,805.85	53,990.00	1.85	4.00	61,176.00	550,356.00	3.20	5.00	21.35	32.00
	$r_j \in U[1, 100]$	201,922.75	1,906,000.00	1.60	4.00	29,775.95	543,595.00	2.55	4.00	18.35	20.00
35	$r_j \in U[1, 50]$	1,946,500.25	26,744,000.00	2.35	5.00	78,636.70	1,459,851.00	5.15	13.00	37.25	47.00
	$r_j \in U[50, 100]$	63,408.45	108,301.00	2.40	5.00	126,142.50	1,208,994.00	4.20	7.00	35.50	47.00
	$r_j \in U[1, 100]$	248,386.05	2,109,800.00	1.45	3.00	175,928.10	874,246.00	3.60	5.00	26.10	39.00
40	$r_j \in U[1, 50]$	1,773,385.10	11,064,300.00	1.85	3.00	117,793.65	2,193,161.00	5.35	9.00	43.65	65.00
	$r_j \in U[50, 100]$	206,306.40	406,824.00	3.45	16.00	8535.20	10,565.00	5.35	6.00	43.55	69.00
	$r_j \in U[1, 100]$	1,335,705.85	8,545,740.00	3.30	14.00	308,533.45	2,184,809.00	5.00	7.00	41.70	53.00

Table 4. CPU Results for $\hat{b}_j \in U[0.05, 0.15]$.

	r_j	*BB*		*UB*		*TS*		*NEH*		*SA*	
n		**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**
15	$r_j \in U[1,50]$	399.25	4462.00	2.75	5.00	9524.20	90,275.00	1.25	3.00	10.35	14.00
	$r_j \in U[50,100]$	25.00	43.00	3.25	10.00	9380.70	88,716.00	1.20	2.00	10.10	13.00
	$r_j \in U[1,100]$	185.55	833.00	1.25	2.00	13,540.00	87,656.00	1.05	2.00	8.70	14.00
20	$r_j \in U[1,50]$	28,036.50	205,993.00	1.05	2.00	1035.60	1089.00	1.05	2.00	11.75	12.00
	$r_j \in U[50,100]$	391.70	817.00	1.05	2.00	11,391.85	208,018.00	1.05	2.00	12.45	19.00
	$r_j \in U[1,100]$	473,980.00	558,400.00	3.85	5.00	20,517.15	376,660.00	3.15	5.00	16.15	22.00
25	$r_j \in U[1,50]$	143,789.80	1,284,350.00	3.75	6.00	42,196.80	207,211.00	3.65	5.00	10.55	14.00
	$r_j \in U[50,100]$	2369.90	5584.00	3.85	5.00	20,517.15	376,660.00	3.15	5.00	16.15	22.00
	$r_j \in U[1,100]$	1,427,065.00	2,1728,000.00	3.70	5.00	21,393.85	392,099.00	3.20	6.00	16.00	17.00
30	$r_j \in U[1,50]$	5,066,261.60	25,664,200.00	4.10	6.00	99,544.30	956,083.00	3.20	4.00	31.30	38.00
	$r_j \in U[50,100]$	16,604.50	25,070.00	4.00	6.00	4375.70	4461.00	3.60	7.00	31.40	37.00
	$r_j \in U[1,100]$	601,835.20	3,554,380.00	3.50	5.00	86,111.60	481,486.00	4.60	6.00	16.80	21.00
35	$r_j \in U[1,50]$	4,268,488.70	33,343,700.00	3.70	5.00	309,401.00	1,527,302.00	4.70	6.00	39.70	47.00
	$r_j \in U[50,100]$	65,676.20	105,503.00	4.10	6.00	160,268.60	1,546,322.00	4.30	5.00	38.40	47.00
	$r_j \in U[1,100]$	9,298,598.50	51,121,800.00	3.80	7.00	5772.70	7089.00	4.50	7.00	33.90	41.00
40	$r_j \in U[1,50]$	20,820,497.90	52,642,500.00	2.70	4.00	8565.50	8792.00	7.90	30.00	43.60	54.00
	$r_j \in U[50,100]$	190,590.40	368,151.00	3.70	5.00	8691.50	8828.00	5.80	8.00	47.80	63.00
	$r_j \in U[1,100]$	5,528,939.50	27,787,500.00	3.20	7.00	464,756.60	2,319,506.00	6.40	9.00	48.50	59.00

Table 5. Error results for $\hat{b}_j \in U[0.05, 0.1]$.

	r_j	$\frac{\widetilde{TWCT}(UB)}{\widetilde{TWCT}(OPT)}$		$\frac{\widetilde{TWCT}(TS)}{\widetilde{TWCT}(OPT)}$		$\frac{\widetilde{TWCT}(NEH)}{\widetilde{TWCT}(OPT)}$		$\frac{\widetilde{TWCT}(SA)}{\widetilde{TWCT}(OPT)}$	
n		**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**
15	$r_j \in U[1,50]$	1.01130	1.03500	1.00577	1.01900	1.00287	1.01980	1.00233	1.01364
	$r_j \in U[50,100]$	1.09401	1.14836	1.06650	1.10572	1.02484	1.07558	1.01209	1.02651
	$r_j \in U[1,100]$	1.00714	1.02365	1.00273	1.01707	1.00063	1.01201	1.00047	1.00470
20	$r_j \in U[1,50]$	1.03847	1.12905	1.02393	1.10072	1.00966	1.04217	1.01025	1.04472
	$r_j \in U[50,100]$	1.18845	1.34761	1.13616	1.24466	1.06560	1.14771	1.03325	1.05427
	$r_j \in U[1,100]$	1.02219	1.08104	1.01351	1.06151	1.00662	1.04503	1.00626	1.03186
25	$r_j \in U[1,50]$	1.07318	1.19098	1.05217	1.15483	1.02053	1.08778	1.03136	1.08909
	$r_j \in U[50,100]$	1.26706	1.40217	1.20397	1.30864	1.10017	1.18650	1.04449	1.06906
	$r_j \in U[1,100]$	1.09087	1.24151	1.06270	1.15799	1.03807	1.14498	1.02875	1.06715
30	$r_j \in U[1,50]$	1.15371	1.30613	1.10994	1.23900	1.05266	1.12934	1.07038	1.12488
	$r_j \in U[50,100]$	1.35286	1.55295	1.27031	1.45067	1.12680	1.24768	1.05119	1.09875
	$r_j \in U[1,100]$	1.11996	1.19086	1.09196	1.15247	1.03394	1.06876	1.04819	1.08544
35	$r_j \in U[1,50]$	1.21194	1.31187	1.16686	1.24187	1.06455	1.12493	1.10517	1.16566
	$r_j \in U[50,100]$	1.40078	1.51959	1.31443	1.47948	1.11353	1.24715	1.05061	1.06747
	$r_j \in U[1,100]$	1.23176	1.29278	1.17090	1.15716	1.09055	1.15716	1.07416	1.11478
40	$r_j \in U[1,50]$	1.27299	1.35269	1.21620	1.28419	1.07545	1.11756	1.11328	1.16429
	$r_j \in U[50,100]$	1.47695	1.72109	1.38209	1.54249	1.08588	1.23025	1.04938	1.07340
	$r_j \in U[1,100]$	1.26862	1.40423	1.20602	1.32286	1.09902	1.16624	1.10156	1.13899

Table 6. Error results for $\hat{b}_j \in U[0.1, 0.15]$.

n	r_j	$\frac{\widetilde{TWCT}(UB)}{\widetilde{TWCT}(OPT)}$		$\frac{\widetilde{TWCT}(TS)}{\widetilde{TWCT}(OPT)}$		$\frac{\widetilde{TWCT}(NEH)}{\widetilde{TWCT}(OPT)}$		$\frac{\widetilde{TWCT}(SA)}{\widetilde{TWCT}(OPT)}$	
		Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**
15	$r_j \in U[1,50]$	1.05674	1.20623	1.03471	1.16052	1.01539	1.07773	1.00858	1.05591
	$r_j \in U[50,100]$	1.21500	1.48927	1.13875	1.32034	1.06802	1.20538	1.00852	1.02564
	$r_j \in U[1,100]$	1.05543	1.16365	1.03285	1.10070	1.01011	1.06412	1.00527	1.01993
20	$r_j \in U[1,50]$	1.12471	1.33094	1.08136	1.26717	1.03191	1.08737	1.03051	1.03702
	$r_j \in U[50,100]$	1.36101	1.49677	1.22776	1.34897	1.08376	1.24973	1.01623	1.02495
	$r_j \in U[1,100]$	1.12802	1.31321	1.08137	1.23608	1.03908	1.12404	1.02079	1.04593
25	$r_j \in U[1,50]$	1.20745	1.47668	1.12907	1.34243	1.10286	1.75250	1.04990	1.08146
	$r_j \in U[50,100]$	1.33965	1.59951	1.22889	1.36198	1.09476	1.27622	1.01893	1.03304
	$r_j \in U[1,100]$	1.21767	1.35760	1.14856	1.27145	1.07122	1.14937	1.03475	1.08024
30	$r_j \in U[1,50]$	1.33229	1.88696	1.23170	1.62900	1.11628	1.30541	1.08435	1.17439
	$r_j \in U[50,100]$	1.44481	1.79102	1.26830	1.51501	1.07284	1.38083	1.01785	1.02863
	$r_j \in U[1,100]$	1.28357	1.49579	1.19603	1.35222	1.10025	1.25002	1.04630	1.09886
35	$r_j \in U[1,50]$	1.43892	1.77995	1.31996	1.56567	1.16249	1.40452	1.09702	1.24194
	$r_j \in U[50,100]$	1.50847	1.81016	1.33659	1.54965	1.04960	1.26749	1.01678	1.02205
	$r_j \in U[1,100]$	1.51732	1.85319	1.38176	1.65955	1.19011	1.39431	1.04774	1.10317
40	$r_j \in U[1,50]$	1.60847	1.99279	1.44337	1.75120	1.21726	1.45192	1.08691	1.13249
	$r_j \in U[50,100]$	1.54510	1.91770	1.32820	1.56574	1.00914	1.02892	1.01318	1.01880
	$r_j \in U[1,100]$	1.62767	1.93104	1.46706	1.70305	1.22651	1.40117	1.04229	1.06242

Table 7. Error results for $\hat{b}_j \in U[0.05, 0.15]$.

n	r_j	$\frac{\widetilde{TWCT}(UB)}{\widetilde{TWCT}(OPT)}$		$\frac{\widetilde{TWCT}(TS)}{\widetilde{TWCT}(OPT)}$		$\frac{\widetilde{TWCT}(NEH)}{\widetilde{TWCT}(OPT)}$		$\frac{\widetilde{TWCT}(SA)}{\widetilde{TWCT}(OPT)}$	
		Mean	**Max**	**Mean**	**Max**	**Mean**	**Max**	**Mean**	**Max**
15	$r_j \in U[1,50]$	1.03603	1.14926	1.01966	1.09956	1.00647	1.02747	1.00515	1.02818
	$r_j \in U[50,100]$	1.16703	1.31754	1.10459	1.18828	1.07194	1.21232	1.01479	1.04807
	$r_j \in U[1,100]$	1.17680	1.02650	1.09727	1.01306	1.07298	1.00554	1.02797	1.05021
20	$r_j \in U[1,50]$	1.08338	1.22684	1.05476	1.16968	1.02351	1.10015	1.02158	1.05933
	$r_j \in U[50,100]$	1.28262	1.45748	1.18372	1.37350	1.10098	1.20654	1.02341	1.04308
	$r_j \in U[1,100]$	1.40915	1.56852	1.27367	1.44357	1.11325	1.24789	1.03637	1.04900
25	$r_j \in U[1,50]$	1.15285	1.26169	1.10954	1.20068	1.04805	1.11419	1.05521	1.10118
	$r_j \in U[50,100]$	1.40915	1.56852	1.27367	1.44357	1.11325	1.24789	1.03637	1.04900
	$r_j \in U[1,100]$	1.15196	1.28620	1.10511	1.21874	1.04765	1.17361	1.04341	1.09800
30	$r_j \in U[1,50]$	1.26896	1.44796	1.20095	1.33424	1.09360	1.15919	1.10978	1.16231
	$r_j \in U[50,100]$	1.38812	1.59121	1.27372	1.51491	1.07134	1.20100	1.03321	1.04538
	$r_j \in U[1,100]$	1.25045	1.35469	1.17607	1.25551	1.08704	1.20058	1.06258	1.09529
35	$r_j \in U[1,50]$	1.40779	1.77403	1.29813	1.55328	1.13990	1.23679	1.12537	1.16885
	$r_j \in U[50,100]$	1.49615	1.63735	1.28232	1.51277	1.10967	1.33036	1.03519	1.04811
	$r_j \in U[1,100]$	1.33965	1.67159	1.24468	1.51793	1.11782	1.31976	1.07395	1.13762
40	$r_j \in U[1,50]$	1.51804	1.05095	1.39466	1.84225	1.18314	1.36730	1.14527	1.20916
	$r_j \in U[50,100]$	1.51672	1.65812	1.37005	1.52064	1.02675	1.06001	1.02962	1.04718
	$r_j \in U[1,100]$	1.49732	1.85312	1.37096	1.69225	1.18359	1.33773	1.07696	1.11406

To demonstrate the superiority of the *BB* algorithm, experiments were conducted between the *BB* algorithm and the mathematical programming model, where the mathematical programming model (13)–(17) is solved by CPLEX (IBM ILOG Optimization Stdio, version 12.10), n = 20, 25, 30, and 10 testing instances were generated for each combination. In order to make the comparative experiments more objective and simple, we assign special values to each instance, without too much sorting, but only for comparison. This is also the reason why CPU time in the comparative experiment is shorter than that in the previous

experiments. Results reported in Tables 8–10 are the objective values and CPU times. In terms of the CUP time, when $r_j \in U[50, 100]$, the B&B algorithm dominates CPLEX significantly for about 92.22% (83 out of 90) of the examples.

Table 8. Results of *BB* and CPLEX for $\hat{b}_j \in U[0.05, 0.1]$.

		$r_j \in U[1, 50]$		$r_j \in U[50, 100]$		$r_j \in U[1, 100]$	
n	**Instance**	**Optimal Value**	**CPU Time (ms) *BB/CPLEX***	**Optimal Value**	**CPU Time (ms) *BB/CPLEX***	**Optimal Value**	**CPU Time (ms) *BB/CPLEX***
20	1	1877.64 | |1877.641	19 | |74	11,767.39 | |11,767.391	1 | |62	5032.861 | |5032.862	42 | |67
	2	2230.077 | |2230.069	662 | |73	9018.022 | |9018.023	5 | |68	4381.909 | |4381.91	76 | |74
	3	2331.78 | |2331.72	213 | |70	6552.248 | |6552.223	3 | |65	6676.791 | |6676.791	354 | |60
	4	1930.146 | |1930.147	220 | |69	9691.73 | |9691.738	2 | |72	5332.758 | |5332.777	105 | |86
	5	2584.912 | |2584.915	14 | |65	9536.494 | |9536.497	2 | |59	4078.021 | |4078.03	1054 | |75
	6	1916.657 | |1916.632	262 | |63	8971.364 | |8971.365	4 | |69	4770.602 | |4770.602	876 | |88
	7	2349.085 | |2349.09	109 | |72	12,743.678 | |12,743.678	1 | |76	3044.157 | |3044.158	686 | |68
	8	2626.915 | |2626.91	458 | |82	9792.175 | |9792.177	6 | |55	3592.465 | |3592.457	156 | |56
	9	3304.571 | |3304.571	62 | |66	11,234.99 | |11,234.99	2 | |59	4284.786 | |4284.787	253 | |63
	10	3213.249 | |3213.25	20 | |77	8078.607 | |8078.611	3 | |63	6258.689 | |6258.689	11 | |73
25	1	2440.8766 | |2440.8766	29 | |78	13501.21 | |13,501.21	5 | |75	8185.514 | |8185.515	1825 | |75
	2	3651.772 | |3651.773	32 | |82	17,962.842 | |17,962.843	1 | |76	8021.395 | |8021.396	682 | |88
	3	3599.892 | |3599.889	33,846 | |75	12,834.31 | |12,834.32	24 | |80	5102.548 | |5102.549	21,541 | |72
	4	2668.11 | |2668.11	1741 | |75	13,387.565 | |13,387.542	5 | |76	7913.71 | |7913.717	213 | |83
	5	2929.49 | |2929.6	40 | |72	14,708.966 | |14,708.977	15 | |98	5360.984 | |5360.985	5017 | |70
	6	3571.392 | |3571.39	408 | |77	13,661.573 | |13,661.58	78 | |73	4762.564 | |4762.542	35 | |85
	7	3402.194 | |3402.194	614 | |78	14,761.3 | |14,761.3	3 | |73	5567.564 | |5567.56	34 | |70
	8	2098.52 | |2098.525	12,589 | |73	16,402.6 | |16,402.63	3 | |71	5397.842 | |5397.885	96,618 | |89
	9	3081.57 | |3081.562	74,684 | |88	15,497.219 | |15,497.22	30 | |88	5549.425 | |5549.423	28,735 | |87
	10	3054.281 | |3054.288	57 | |69	12,570.492 | |12,570.488	3 | |77	7384.72 | |7384.724	16 | |59
30	1	4761.895 | |4761.895	13,576 | |80	20,825.46 | |20,825.468	2 | |85	7114.616 | |7114.615	4587 | |95
	2	3930.492 | |3930.493	474,584 | |78	23,541.618 | |23,541.618	72 | |85	5594.15 | |5594.158	15,516 | |84
	3	5184.257 | |5184.258	3145 | |82	21,762.827 | |21,762.828	5 | |83	10,639.97 | |10,639.97	78,565 | |89
	4	3445.295 | |3445.295	448 | |98	19,283.161 | |19,283.162	5 | |93	6635.125 | |6635.151	35,242 | |82
	5	5420.63 | |5420.645	3586 | |82	21,259.49 | |21,259.457	5 | |83	8109.199 | |8109.192	5334 | |97
	6	3352.281 | |3352.267	9 | |98	22,234.591 | |22,234.587	1 | |76	4331.347 | |4331.378	87,785 | |87
	7	3452.28 | |3452.281	8 | |35	17,115.157 | |17,115.15	14 | |78	6708.094 | |6708.051	15,5681 | |76
	8	3800.062 | |3800.063	27,475 | |98	19,162.501 | |19,162.503	6 | |89	7251.21 | |7251.22	958,872 | | 79
	9	5389.884 | |5389.885	27,512 | |78	19,553.989 | |195,53.989	1 | |86	6495.424 | |6495.424	419 | |84
	10	3226.58 | |3226.52	8755 | |86	16,351.4 | |16,351.4	10 | |85	7976.82 | |7976.82	987,855 | |92

Table 9. Results of *BB* and CPLEX for $\hat{b}_j \in U[0.1, 0.15]$.

		$r_j \in U[1, 50]$		$r_j \in U[50, 100]$		$r_j \in U[1, 100]$	
n	**Instance**	**Optimal Value**	**CPU Time (ms) *BB/CPLEX***	**Optimal Value**	**CPU Time (ms) *BB/CPLEX***	**Optimal Value**	**CPU Time (ms) *BB/CPLEX***
20	1	3224.206 | |3224.205	6 | |72	13,078.051 | |13,078.052	3 | |78	4581.808 | |4581.808	211 | |75
	2	1920.143 | |1920.144	72 | |76	14,416.229 | |14,416.23	2 | |67	5984.757 | |5984.757	17 | |77
	3	1791.398 | |1791.398	26 | |73	16,763.404 | |16,763.404	14 | |75	4306.999 | |4307.001	1 | |76
	4	3194.432 | |3194.433	199 | |78	13,161.205 | |13,161.189	1 | |69	5299.028 | |5299.029	11 | |69
	5	2599.284 | |2599.284	2 | |70	16,456.45 | |16,456.45	6 | |78	7654.312 | |7654.311	19 | |76
	6	2309.754 | |2309.753	17 | |73	15,081.528 | |15,081.527	355 | |80	6500.897 | |6500.875	86 | |70
	7	2352.418 | |2352.418	10 | |69	16,248.781 | |16,248.783	8 | |75	6551.345 | |6551.352	3 | |78
	8	3382.56 | |3382.558	2 | |72	13,478.75 | |13,478.778	4 | |83	6473.928 | |6473.917	18 | |72
	9	3103.121 | |3103.121	4 | |70	16,133.258 | |16,133.257	996 | |77	5987.109 | |5987.11	10 | |68
	10	2467.09 | |2467.092	28 | |71	15,250.86 | |15,250.87	7 | |70	5011.23 | |5011.230	77 | |77
25	1	3381.737 | |3381.739	224 | |73	24,283.305 | |24,283.274	1 | |71	7095.573 | |7095.567	10 | |70
	2	4686.301 | |4686.296	5 | |74	37,005.444 | |37,005.441	2 | |75	6754.407 | |6754.412	2 | |82
	3	6456.931 | |6456.932	1 | |69	24,295.173 | |24,295.174	1 | |73	6870.603 | |6870.608	9 | |76
	4	4357.681 | |4357.682	130 | |66	25,835.513 | |25,835.527	3 | |76	7434.996 | |7434.992	216 | |75
	5	5452.679 | |5452.684	103 | |77	31,347.823 | |31,347.787	1 | |70	7352.652 | |7352.653	58 | | 76
	6	7397.109 | |7397.108	183 | |72	24,678.259 | |24,678.27	4 | |72	8575.343 | |8575.338	15 | |73
	7	5200.723 | |5200.718	39 | |69	22,434.802 | |22,434.803	1 | |78	8431.151 | |8431.15	90 | |87
	8	3852.579 | |3852.578	464 | |67	28,767.763 | |28,767.782	1 | |74	7703.809 | |7703.809	11 | |72
	9	6625.675 | |6625.675	58 | |68	36,563.797 | |36,563.786	3 | |77	9380.949 | |9380.949	8 | |87
	10	3153.421 | |3153.422	4 | |63	28,022.863 | |28,022.806	1 | |70	5833.467 | |5833.467	27 | |89
30	1	3785.294 | |3785.295	49,995 | |78	55,855.796 | |55,855.714	1 | |88	22,799.14 | |22,799.116	452 | |76
	2	4185.165 | |4185.165	73 | |90	35,568.274 | |35,568.261	9 | |82	7981.994 | |7981.978	673 | |73
	3	4877.96 | |4877.96	1617 | |79	57,761.936 | |57,761.94	1 | |75	15,434.62 | |15,434.639	19 | |80
	4	5692.011 | |5692.012	5 | |69	44,756.236 | |44,756.236	2 | |74	19,177.76 | |19,177.758	3 | |80
	5	7250.617 | |7250.618	5 | |79	37,478.351 | |37,478.332	3 | |72	15816.347 | |15,816.335	37 | |73
	6	5247.79 | |5247.79	34 | |88	55,426.436 | |55,426.501	1 | |77	25,672.837 | |25,672.852	307 | |73
	7	6364.644 | |6364.644	141 | |84	45,668.881 | |45,668.846	5 | |70	12,609.977 | |12,609.978	673 | |76
	8	6525.450 | |6525.450	4 | |76	54,463.341 | |54,463.341	1 | |77	11,243.774 | |11,243.772	16 | |70
	9	5198.277 | |5198.277	26,756 | |82	56,327.158 | |56,327.19	2 | |75	21,031.685 | |21,031.685	525 | |75
	10	5735.88 | |5735.88	113 | |72	37,847.965 | |37,847.972	1 | |77	7605.592 | |7605.592	8665 | |89

Table 10. Results of BB and CPLEX for $\hat{b}_j \in U[0.05, 0.15]$.

		$r_j \in U[1, 50]$		$r_j \in U[50, 100]$		$r_j \in U[1, 100]$	
n	Instance	Optimal Value	CPU Time (ms) *BB/CPLEX*	Optimal Value	CPU Time (ms) *BB/CPLEX*	Optimal Value	CPU Time (ms) *BB/CPLEX*
20	1	3143.66||3143.668	1272||74	10,023.477||10,023.48	1||70	6838.891||6838.892	27||62
	2	3170.929||3170.931	448||74	12,336.148||12,336.15	1||72	9484.171||9484.173	57||64
	3	3210.143||3210.145	254||72	12,840.194||12,840.195	2||68	5133.177||5133.176	4777||65
	4	2251.857||2251.857	18||70	8797.323||8797.323	2||75	6765.709||6765.709	35||67
	5	2091.515||2091.51	2||78	12,620.946||12,620.946	1||64	4780.693||4780.693	11||64
	6	3006.8||3006.803	2401||66	14,114.352||14,114.351	3||68	4873.354||4873.354	1051||64
	7	2740.521||2740.521	381||75	10,801.699||10,801.706	1||61	6112.412||6112.412	210||62
	8	2257.812||2257.813	40||70	13,489.863||13,489.864	2||68	6112.412||6112.412	209||72
	9	2507.171||2507.171	102||66	12,856.903||12,856.906	1||66	4763.482||4763.48	58||69
	10	2620.089||2620.09	42||65	13,178.881||13,178.881	3||76	3380.259||3380.258	2023||66
25	1	2857.342||2857.343	2768||87	16,125.897||16,125.9	1||69	5816.16||5816.158	19,384||86
	2	5265.281||5265.281	5||78	20,263.138||20,263.151	1515||69	6627.122||6627.125	2093||81
	3	2914.465||2914.465	59,283||68	18,216.342||18,216.342	2||73	8545.714||8545.714	7720||77
	4	3698.654||3698.656	6||75	16,756.839||16,756.838	3||77	7557.438||7557.44	12,560||82
	5	3012.074||3012.07	27||80	21,526.798||21,526.798	57||70	10,129.488||10,129.48	855||79
	6	6086.928||6086.931	3||67	16,461.209||16,461.21	43||72	6686.74||6686.739	27||85
	7	4097.154||4097.154	8||68	14,856.578||14,856.587	1||66	6686.74||6686.739	23||84
	8	2242.35||2242.355	13575||70	23,343.777||23,343.791	15||69	5613.54||5613.538	75,757||80
	9	4191.34||4191.34	45123||71	17,754.949||17,754.95	2||79	7486.88||7486.88	14,548||72
	10	3572.956||3572.957	44||70	19,581.42||19,581.412	2||90	8447.198||8447.198	605||76
30	1	5761.119||5761.122	7679||85	25,379.136||25,379.138	1||80	7612.346||7612.344	92||78
	2	2694.968||2694.957	78,897||75	28,587.885||28,587.905	8||77	8778.962||8778.963	9968||84
	3	4941.414||4941.418	1140||80	26,516.25||26,516.251	3777||83	8861.97||8861.976	8861||89
	4	4327.01||4327.012	11,250||94	31,780.398||31,780.403	2||73	12,951.088||12,951.089	235||102
	5	3777.884||3777.885	4769||95	27,277.598||27,277.589	6||72	9645.175||9645.182	239||94
	6	7943.162||7943.168	157||90	32,927.444||32,927.447	75,888||86	9645.175||9645.17	234||86
	7	3849.75||3849.75	27,377||96	25,798.781||25,798.775	1||78	26,305.916||26,305.917	797||86
	8	4951.373||4951.374	2541||97	28,156.49||28,156.489	4||77	24,538.923||24,538.936	2||70
	9	4717.61||4717.614	7888||95	22,417.729||22,417.734	8678||81	11,439.549||11,439.549	261||88
	10	4389.528||4389.531	84||90	24,675.208||24,675.207	3||78	12,783.488||12,783.492	4258||86

6. Conclusions

This paper studies single-machine scheduling with proportional deterioration and ready times simultaneously. The objective function is to minimize $\widetilde{TWCT}$. For the general condition of the problem, we propose the branch-and-bound algorithm and some meta-heuristic algorithms. Experimental study demonstrate that the BB algorithm can solve instances of 40 jobs in less than 39,870,200 milliseconds, and the algorithms of NEH and SA are more accurate than TS. Further research should further explore the $\widetilde{TWCT}$ minimization problem with the general linear deterioration (i.e., $1|\hat{p}_j = \hat{a}_j + \hat{b}_j\hat{s}_j, r_j|\widetilde{TWCT}$, where $\hat{a}_j$ is the normal processing time of job J_j), extend our model to problems with scenario-dependent processing times (Wu et al. [41] and Wu et al. [42]), or consider the extension to parallel machines and hybrid flow shop setting (Yu et al. [43]).

Author Contributions: Methodology, J.-B.W.; writing—original draft, Z.-G.L.; writing—review and editing, Z.-G.L., L.-H.Z., X.-Y.W. and J.-B.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Science Research Foundation of the Educational Department of Liaoning Province (LJKMZ20220532).

Data Availability Statement: The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Gawiejnowicz, S.; Kurc, W.; Pankowska, L. Pareto and scalar bicriterion scheduling of deteriorating jobs. *Comput. Oper. Res.* **2006**, *33*, 746–767. [CrossRef]
2. Wu, Y.; Dong, M.; Zheng, Z. Patient scheduling with periodic deteriorating maintenance on single medical device. *Comput. Oper. Res.* **2014**, *49*, 107–116. [CrossRef]
3. Mosheiov, G. Scheduling jobs under simple linear deterioration. *Comput. Oper. Res.* **1994**, *21*, 653–659. [CrossRef]

4. Gawiejnowicz, S. Scheduling deteriorating jobs subject to job or machine availability constraints. *Eur. J. Oper. Res.* **2007**, *180*, 472–478. [CrossRef]
5. Oron, D. Single machine scheduling with simple linear deterioration to minimize total absolute deviation of completion times. *Comput. Oper. Res.* **2008**, *35*, 2071–2078. [CrossRef]
6. Lee, W.-C.; Wang, W.J.; Shiau, Y.R.; Wu, C.-C. A single-machine scheduling problem with two-agent and deteriorating jobs. *Appl. Math. Model.* **2010**, *34*, 3098–3107. [CrossRef]
7. Wang, Z.Y.; Wei, C.M.; Wu, Y.B. Single machine two-agent scheduling with deteriorating jobs. *Asia-Pac. J. Oper. Res.* **2016**, *33*, 1650034. [CrossRef]
8. Wu, C.-C.; Wu, W.-H.; Wu, W.-H.; Hsu, P.-H.; Yin, Y.; Xu, J. A single-machine scheduling with a truncated linear deterioration and ready times. *Inf. Sci.* **2014**, *256*, 109–125. [CrossRef]
9. Yin, N.; Kang, L. Minimizing makespan in permutation flow shop scheduling with proportional deterioration. *Asia-Pac. J. Oper. Res.* **2015**, *32*, 1550050. [CrossRef]
10. Pei, J.; Liu, X.B.; Pardalos, P.M.; Fan, W.J.; Yang, S.L. Scheduling deteriorating jobs on a single serial-batching machine with multiple job types and sequence-dependent setup times. *Ann. Oper. Res.* **2017**, *249*, 175–195. [CrossRef]
11. Jafari, A.A.; Lotfi, M.M. Single-machine scheduling to minimize the maximum tardiness under piecewise linear deteriorating jobs. *Sci. Iran.* **2018**, *25*, 370–385. [CrossRef]
12. Miao, C.X.; Zhang, Y.Z. Scheduling with step-deteriorating jobs to minimize the makespan. *J. Ind. Manag. Optim.* **2019**, *15*, 1955–1964. [CrossRef]
13. Huang, X. Bicriterion scheduling with group technology and deterioration effect. *J. Appl. Math. Comput.* **2019**, *60*, 455–464. [CrossRef]
14. Liu, F.; Yang, J.; Lu, Y.-Y. Solution algorithms for single-machine group scheduling with ready times and deteriorating jobs. *Eng. Optim.* **2019**, *51*, 862–874. [CrossRef]
15. Sun, X.; Geng, X.-N. Single-machine scheduling with deteriorating effects and machine maintenance. *Int. J. Prod. Res.* **2019**, *57*, 3186–3199. [CrossRef]
16. Cheng, T.C.E.; Kravchenko, S.-A.; Lin, B.M.T. Scheduling step-deteriorating jobs to minimize the total completion time. *Comput. Ind. Eng.* **2020**, *144*, 106329. [CrossRef]
17. Li, D.-W.; Lu, X.-W. Parallel-batch scheduling with deterioration and rejection on a single machine. *Appl. Math. J. Chin. Univ.* **2020**, *35*, 141–156. [CrossRef]
18. Zhang, X.; Liu, S.-C.; Lin, W.-C.; Wu, C.-C. Parallel-machine scheduling with linear deteriorating jobs and preventive maintenance activities under a potential machine disruption. *Comput. Ind. Eng.* **2020**, *145*, 106482. [CrossRef]
19. Huang, X.; Yin, N.; Liu, W.-W.; Wang, J.-B. Common due window assignment scheduling with proportional linear deterioration effects. *Asia-Pac. J. Oper. Res.* **2020**, *37*, 1950031. [CrossRef]
20. He, H.; Hu, Y.; Liu, W.-W. Scheduling with deteriorating effect and maintenance activities under parallel processors. *Eng. Optim.* **2021**, *53*, 2070–2087. [CrossRef]
21. Qian, J.; Han, H. The due date assignment scheduling problem with the deteriorating jobs and delivery time. *J. Appl. Math. Comput.* **2022**, *67*, 2173–2186. [CrossRef]
22. Qian, J.; Zhan, Y. The due window assignment problems with deteriorating job and delivery time. *Mathematics* **2022**, *10*, 1672. [CrossRef]
23. Miao, C.; Kong, F.; Zou, J.; Ma, R.; Huo, Y. Parallel-machine scheduling with step-deteriorating jobs to minimize the total (weighted) completion time. *Asia-Pac. J. Oper. Res.* **2023**, *40*, 2240011. [CrossRef]
24. Jia, X.; Lv, D.-Y.; Hu, Y.; Wang, J.-B.; Wang, Z.; Wang, E. Slack due-window assignment scheduling problem with deterioration effects and a deteriorating maintenance activity. *Asia-Pac. J. Oper. Res.* **2022**, *39*, 2250005. [CrossRef]
25. Sun, X.; Liu, T.; Geng, X.-N.; Hu, Y.; Xu, J.-X. Optimization of scheduling problems with deterioration effects and an optional maintenance activity. *J. Sched.* **2023**, *26*, 251–266. [CrossRef]
26. Miao, C.; Song, J.; Zhang, Y. Single-machine time-dependent scheduling with proportional and delivery times. *Asia-Pac. J. Oper. Res.* **2023**, *40*, 2240011. [CrossRef]
27. Wang, J.-B.; Wang, Y.-C.; Wan, C.; Lv, D.-Y.; Zhang, L. Controllable processing time scheduling with total weighted completion time objective and deteriorating jobs. *Asia-Pac. J. Oper. Res.* **2023**, 2350026. [CrossRef]
28. Zhang, L.-H.; Geng, X.-N.; Xue, J.; Wang, J.-B. Single machine slack due window assignment and deteriorating jobs. *J. Ind. Manag. Optim.* **2024**, *20*, 1593–1614. [CrossRef]
29. Shabtay, D.; Mor, B. Exact algorithms and approximation schemes for proportionate flow shop scheduling with step-deteriorating processing times. *J. Sched.* **2022**. [CrossRef]
30. Gawiejnowicz, S. *Models and Algorithms of Time-Dependent Scheduling*; Springer: Berlin, Germany, 2020.
31. Gawiejnowicz, S. A review of four decades of time-dependent scheduling: Main results, new topics, and open problems. *J. Sched.* **2020**, *23*, 3–47. [CrossRef]
32. Miao, C. Complexity of scheduling with proportional deterioration and release dates. *Iran. J. Sci. Technol. Trans. A Sci.* **2018**, *42*, 1337–1342. [CrossRef]
33. Zhong, X.L.; Pan, Z.M.; Jiang, D.K. Scheduling with release times and rejection on two parallel machines. *J. Comb. Optim.* **2017**, *33*, 934–944. [CrossRef]

34. Bai, D.Y.; Xue, H.; Wang, L.; Wu, C.-C.; Lin, W.-C.; Abdulkadir, D.H. Effective algorithms for single-machine learning-effect scheduling to minimize completion-time-based criteria with release dates. *Expert Syst. Appl.* **2020**, *156*, 113445. [CrossRef]
35. Qian, J.; Lin, H.X.; Kong, Y.F.; Wang, Y.S. Tri-criteria single machine scheduling model with release times and learning factor. *Appl. Math. Comput.* **2020**, *387*, 124543. [CrossRef]
36. Rau, G. Minimizing a function of permutations of n integers. *Oper. Res.* **1971**, *19*, 237–240. [CrossRef]
37. Kelly, F.-P. A remark on search and sequencing problems. *Math. Oper. Res.* **1982**, *7*, 154–157. [CrossRef]
38. Nawaz, M.; Enscore, J.E.E.; Ham, I. A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem. *Omega* **1983**, *11*, 91–95. [CrossRef]
39. Lai, K.; Hsu, P.-H.; Ting, P.-H.; Wu, C.-C. A truncated sum of processing-times-based learning model for a two-machine flowshop scheduling problem. *Hum. Factors Ergon. Manuf. Serv. Ind.* **2014**, *24*, 152–160. [CrossRef]
40. Liu, S.; Wu, W.-H.; Kang, C.-C.; Lin, W.-C.; Cheng, Z. A single-machine two-agent scheduling problem by a branch-and-bound and three simulated annealing algorithms. *Discret. Dyn. Nat. Soc.* **2015**, *2015*, 681854. [CrossRef]
41. Wu, C.-C.; Bai, D.; Zhang, X.; Cheng, S.-R.; Lin, J.-C.; Wu, Z.-L.; Lin, W.-C. A robust customer order scheduling problem along with scenario-dependent component processing times and due dates. *J. Manuf. Syst.* **2021**, *58*, 291–305. [CrossRef]
42. Wu, C.-C.; Bai, D.; Chen, J.-H.; Lin, W.-C.; Xing, L.; Lin, J.-C.; Cheng, S.-R. Everal variants of simulated annealing hyper-heuristic for a single-machine scheduling with two-scenario-based dependent processing times. *Swarm Evol. Comput.* **2021**, *60*, 100765. [CrossRef]
43. Yu, Y.; Pan, Q.; Pang, X.; Tang, X. An attribution feature-based memetic algorithm for hybrid flowshop scheduling problem with operation skipping. *IEEE Trans. Autom. Sci. Eng.* **2024**. [CrossRef]