Journal of Industrial and Management Optimization

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

JIMO, 22(8): 3968–3998.
DOI:10.3934/jimo.2026141
Received: 23 April 2026
Revised: 05 July 2026
Accepted: 15 July 2026
Published: 24 July 2026

Research article

Solution algorithms for minimizing total weighted completion time scheduling with release times and time-dependent deterioration effects

Li-Han Zhang[1,2], Xiao-Yuan Wang[3,*], Na Yin[3], Zheng-Guo Lv[1] and Ji-Bo Wang[1,2]

[1] School of Mechatronics Engineering, Shenyang Aerospace University, Shenyang 110136, P.R. China
[2] Key Laboratory of Rapid Development & Manufacturing Technology for Aircraft (Shenyang Aerospace University), Ministry of Education, Shenyang 110136, P.R. China
[3] School of Science, Shenyang Aerospace University, Shenyang 110136, P. R. China

* **Correspondence:** Email: 20012302@sau.edu.cn, wxy5099@126.com.

Abstract: This article addresses a single-machine scheduling problem with release times and deterioration effects, where the processing time of a job is a non-decreasing function of its start time, and all jobs share the same deterioration rate. The objective was to determine an optimal schedule that minimizes the total weighted completion time. Since the problem is NP-hard, several heuristic algorithms were proposed. Furthermore, several dominance properties were established, along with an upper bound and a lower bound, after which a branch-and-bound (bb) algorithm was developed. Experimental results were presented to compare the performance of the heuristic algorithms with that of the branch-and-bound algorithm.

Keywords: scheduling; deterioration effect; release time; branch-and-bound; heuristic
Mathematics Subject Classification: 90B35

1. Introduction

Scheduling problems with deteriorating jobs, where processing times increase as start times are delayed, are prevalent in many real-world production systems. These problems model scenarios where task execution becomes more challenging over time, such as in steel hot-rolling where interruptions cause billet temperature drops, necessitating longer re-heating and processing times (deterioration effects) (Mosheiov [1], Kononov [2]), or in equipment maintenance where wear prolongs service times (Browne and Yechiali [3]). A detailed discussion of applications for scheduling with deterioration across various industries can be found in Chapter 6 of Gawiejnowicz [4].

Research on linear deterioration scheduling without release times ($r_i = 0$) has uncovered many special structures solvable in polynomial time. Examples include Mosheiov's work [1] on the $p_i = b_i t$ model and Cheng et al.'s study [5] on models involving a common due-date. However, these results do not hold when distinct release times are introduced. The computational complexity of scheduling with deterioration effects is well-established. For basic problems like $1 \mid p_i = b_i t \mid \sum C_i$, polynomial-time algorithms often exist (Strusevich and Rustogi [6]). However, the introduction of release times r_i significantly increases complexity. Labetoulle et al. [7] proved that the classical problem $1 \mid r_i \mid \sum w_i C_i$ is strongly NP-hard (for the original NP-hardness proof, see Lenstra et al. [8]), establishing the foundation for the complexity of deterioration scheduling variants with release times.

Miao et al. [9] considered proportional deterioration ($p_i = b_i t$) with release times and proved the NP-hardness of $1 \mid r_i, p_i = b_i t \mid \sum w_i C_i$. Lv et al. [10] studied the exact same problem and proposed heuristic and branch-and-bound algorithms. Their work is the most direct precursor to the model addressed here. Notably, our model $p_i = a_i + bt$, which includes job-specific normal processing times a_i, is more general than the proportional model $p_i = b_i t$, leading to different problem characteristics and solution strategies. In broader studies on linear deterioration scheduling, Atsmony and Mosheiov [11] considered a single-machine problem with processing times $p_i = 1 + b_i t$ and introduced a new lower bound for minimizing total completion time. Although their model differs from ours (having job-dependent deterioration rates b_i and a constant term), it provides insights for designing lower-bounding techniques. Kong et al. [12] studied scheduling with deteriorating jobs and delivery times. Furthermore, a limited number of studies have examined other related variants combining release times and deterioration, such as scheduling integrated with maintenance activities (He et al. [13], Liu et al. [14], Sun and Geng [15]), due-date assignment (Qian and Zhang [16], Lv et al. [17], Zhang et al. [18]), group scheduling (Liu et al. [19], Yin and Gao [20], Yin et al. [21]), flow shop scheduling (Lv and Wang [22,23]), parallel-machine scheduling (Miao and Zou [24], Li and Lu [25]), and problems with learning effects (Bai et al. [26], Qian et al. [27], Sun et al. [28], Wang [29], Song et al. [30], Lu and Li [31]) or controllable processing times (Wang et al. [32], Zhang et al. [33], Wu et al. [34]). These works further underscore the complexity of this problem domain and the need for dedicated algorithms.

This paper addresses a computationally challenging and practically significant variant: single-machine scheduling with distinct release times and linear deterioration effects. Specifically, each job J_i has a normal processing time a_i and a release time r_i. Its actual processing time p_i is a linear function of its start time t, given by $p_i = a_i + bt$, where $b > 0$ is a common deterioration rate for all jobs. The objective is to find a schedule that minimizes the total weighted completion time $\sum_{i=1}^{n} w_i C_i$. This model aims to capture practical settings where jobs arrive at different times and delays lead to reduced efficiency, as also motivated by studies considering release times in various scheduling contexts (Wu et al. [35], Bai et al. [36]).

Following the standard three-field notation (Graham et al. [37]), our problem is denoted as $1 \mid r_i, p_i = a_i + bt \mid \sum w_i C_i$. This problem is strongly NP-hard. First, for problems with simple linear deterioration $p_i = b_i t$, the introduction of distinct release times r_i renders even the total completion time minimization NP-hard, as indicated by the complexity of related models (Bachman et al. [38]). Second, the fundamental classical problem $1 \mid r_i \mid \sum w_i C_i$ without deterioration is itself strongly NP-hard (Labetoulle et al. [7]). Consequently, our problem, which combines release times with a more general linear deterioration model, is also strongly NP-hard. Given its computational complexity, developing a solution framework combining exact and efficient heuristic algorithms is of significant value, similar

to approaches used for other complex scheduling problems (Pei et al. [39], Cheng et al. [40], Kim et al. [41], Wu et al. [42], Miao et al. [43], Mao et al. [44], Lu et al. [45], Li et al. [46], Wang and Liu [47], Sun et al. [48], Sun et al. [49]).

The main contributions and innovations of this paper are as follows:

(1) Novel dominance properties. We establish several new dominance properties (Propositions 3.1–3.6) for the $1 \mid r_i, p_i = a_i + bt \mid \sum w_i C_i$ problem, which significantly reduce the search space of the branch-and-bound (bb) algorithm.

(2) Composite lower bound. We design a branch-and-bound (bb) algorithm incorporating a composite lower bound derived from three distinct bounds (inspired by principles from Hardy et al. [50]) to enhance pruning efficiency.

(3) Adapted heuristics. We adapt several heuristic and metaheuristic algorithms — including a constructive heuristic (ub), tabu search (ts), and simulated annealing (sa) — to the single-machine deterioration setting, providing efficient near-optimal solutions for large-scale instances.

(4) Extensive computational validation. We conduct extensive computational experiments to systematically evaluate the performance of the proposed algorithms and compare them with results from the commercial solver Gurobi, following the experimental rigor seen in recent scheduling studies (Wang and Liu [51], Wang et al. [52]).

The remainder of this paper is organized as follows. Section 2 provides the problem description and formulation. Section 3 presents the dominance properties, lower bounds, and the branch-and-bound algorithm. Section 4 describes the heuristic and metaheuristic algorithms. Section 5 reports the computational experiments and results analysis. Finally, Section 6 concludes the paper and suggests future research directions.

2. Problem formulation

There is a set $\widetilde{JOB} = \{J_1, J_2, ..., J_n\}$ of n jobs to be processed on a single-machine. As in Cheng et al. [5], the actual processing time of the job J_i is

$$p_i = a_i + bt, \tag{1}$$

where a_i is the normal processing time of J_i $(i = 1, 2, ..., n)$, and b and t are the common deterioration factor and starting time, respectively. For generality, the release times will be considered in this paper. Let w_i (resp, C_i) be the weight (resp, completion time) of J_i, and the aim is to identify the optimal sequence S^* so that $\sum_{i=1}^{n} w_i C_i(S^*)$ (i.e., and total weighted completion time) is minimized. By the 3-field notation, this problem is represented as

$$1 \,|r_i, p_i = a_i + bt| \sum_{i=1}^{n} w_i C_i, \tag{2}$$

where r_i represents the release time of J_i.

3. A branch-and-bound algorithm

Labetoulle et al. [7] proved that the problem $1 \,|r_i| \sum_{i=1}^{n} w_i C_i$ is strongly NP-hard, hence, the problem $1 \,|r_i, p_i = a_i + bt| \sum_{i=1}^{n} w_i C_i$ is also strongly NP-hard. We intend to develop a branch-and-

bound algorithm and further propose some heuristic algorithms aimed at obtaining both optimal and near-optimal solutions.

3.1. Dominance properties

The ensuing dominance rules can effectively eliminate a large number of nodes and thus lighten the computational burden. Let $S = (\pi, J_i, J_j, \pi')$, $S' = (\pi, J_j, J_i, \pi')$, where π and π' are partial sequences, and we obtain the following findings:

Proposition 1. *For any two jobs J_i and J_j, if $a_i \le a_j$, $r_i = r_j$, and $w_i \ge w_j$, then S dominates S'.*

Proof Let t denote the completion time of the final job in the schedule π. To demonstrate that S outperforms S' in terms of dominance, it suffices to prove that $C_j(S) \le C_i(S')$ and $w_iC_i(S) + w_jC_j(S) \le w_jC_j(S') + w_iC_i(S')$. Under S and S', we have

$$C_i(S) = \max\{t, r_i\} + a_i + b\max\{t, r_i\} = a_i + (1+b)\max\{t, r_i\},$$

$$\begin{aligned} C_j(S) &= a_j + (1+b)\max\left\{C_i(S), r_j\right\} \\ &= a_j + (1+b)\max\left\{a_i + (1+b)t, a_i + (1+b)r_i, r_j\right\}, \end{aligned}$$

$$C_j(S') = \max\left\{t, r_j\right\} + a_j + b\max\left\{t, r_j\right\} = a_j + (1+b)\max\left\{t, r_j\right\},$$

$$\begin{aligned} C_i(S') &= a_i + (1+b)\max\left\{C_j(S'), r_i\right\} \\ &= a_i + (1+b)\max\left\{a_j + (1+b)t, a_j + (1+b)r_j, r_i\right\}. \end{aligned}$$

Let $r_i = r_j = r$, and if $r \le t$, we have

$$\begin{aligned} C_j(S) - C_i(S') &= a_j + (1+b)a_i - a_i - (1+b)a_j \\ &= a_j - a_i + (1+b)(a_i - a_j) \\ &= b(a_i - a_j) \\ &\le 0 \end{aligned}$$

and

$$\begin{aligned} & w_iC_i(S) + w_jC_j(S) - w_jC_j(S') - w_iC_i(S') \\ = \;& w_i(1+b)t - w_j(1+b)t + w_j(1+b)a_i + w_j(1+b)^2t \\ & - w_i(1+b)a_j - w_i(1+b)^2t \\ = \;& w_iw_j(1+b)\left[\left(\frac{a_i}{w_i} - \frac{a_j}{w_j}\right) + tb\left(\frac{1}{w_i} - \frac{1}{w_j}\right)\right] \\ \le \;& 0. \end{aligned}$$

If $r > t$, we have

$$\begin{aligned} C_j(S) - C_i(S') &= a_j + (1+b)a_i - a_i - (1+b)a_j + (1+b)^2 r_i - (1+b)^2 r_j \\ &= a_j - a_i + (1+b)(a_i - a_j) + (1+b)^2(r_i - r_j) \\ &= b(a_i - a_j) \\ &\leq 0 \end{aligned}$$

and

$$\begin{aligned} & w_i C_i(S) + w_j C_j(S) - w_j C_j(S') - w_i C_i(S') \\ = \; & w_i(1+b)r_i - w_j(1+b)r_j + w_j(1+b)a_i + w_j(1+b)^2 r_i \\ & - w_i(1+b)a_j - w_i(1+b)^2 r_j \\ = \; & w_i w_j (1+b)\left[\left(\frac{a_i}{w_i} - \frac{a_j}{w_j}\right) + br\left(\frac{1}{w_i} - \frac{1}{w_j}\right)\right] \\ \leq \; & 0. \end{aligned}$$

This completes the proof. □

Remark 3.1. *Proposition 1 directly implies the following special cases, which were previously stated as separate propositions:*

- $a_i = a_j$, $r_i = r_j$, *and* $w_i \geq w_j$.
- $a_i \leq a_j$, $r_i = r_j$, *and* $w_i = w_j$.

Proposition 2. *For any two jobs J_i and J_j, if $a_i \leq a_j$, $r_i \leq r_j$, and $w_i = w_j$, then S dominates S'.*

Proposition 3. *For any two jobs J_i and J_j, if $a_i = a_j$, $r_i \leq r_j$, and $w_i \geq w_j$, and additionally $t(1+b) \leq r_j$ and $w_i r_i \leq w_j r_j$, then S dominates S'.*

3.2. Heuristic algorithm

From Subsection 3.1, i.e., the dominance properties, the subsequent algorithm produces an upper bound (denoted by ub) of the problem $1\,|p_i = a_i + bt, r_i|\sum_{i=1}^{n} w_i C_i$.

Algorithm 1: ub

Input: a_i, r_i, w_i, b

Output: The best job sequence S_{best} and $\sum w_i C_i(S_{best})$

Phase 1: Initialization
Sort jobs in non-decreasing order of a_i;
Sort jobs in non-decreasing order of r_i;
Sort jobs in non-increasing order of w_i;
Sort jobs in non-decreasing order of r_i/w_i;
Sort jobs in non-decreasing order of a_i/w_i;
Compute $t_i = r_i + a_i$, sort by t_i/w_i;
Select the sequence from the above with minimum $\sum w_i C_i$;

Phase 2: Pairwise interchange improvement
Let S_{best} be the initial sequence from Phase 1;
$k = 1$;
while $k < n$ **do**
    $j = k + 1$;
    while $j \leq n$ **do**
        Swap the k-th and j-th jobs (where j ranges from $k + 1$ to n, allowing non-adjacent swaps at arbitrary positions) to obtain a new sequence S_{new};
        if $\sum w_i C_i(S_{new}) < \sum w_i C_i(S_{best})$ **then**
            $S_{best} = S_{new}$;
        end
        $j = j + 1$;
    end
    $k = k + 1$;
end
return *S_{best} and its objective value.*

3.3. Lower bounds

Lemma 1. *(Hardy et al. [50]) For the sum $\sum_{i=1}^{n} x_i y_i$, if the sequence $\{x_1, x_2, \ldots, x_n\}$ is arranged in non-decreasing order and $\{y_1, y_2, \ldots, y_n\}$ is arranged in non-increasing order, then this pairing minimizes the sum.*

Assume $S = (s_p, s_u)$ represents a job sequence, where π_{ps} denotes the scheduled segment, s_p contains k jobs, and s_u is the unscheduled segment, then the completion time of the $(k + 1)$th job (i.e., $J_{[k+1]}$) is

$$C_{[k+1]}(S) = a_{[k+1]} + \max\left\{C_{[k]}(S)(1+b), r_{[k+1]}(1+b)\right\},$$

$$\begin{aligned} C_{[k+2]}(S) = a_{[k+2]} + \max\big\{&a_{[k+1]}(1+b) + C_{[k]}(S)(1+b)^2, \\ &a_{[k+1]}(1+b) + r_{[k+1]}(1+b)^2, r_{[k+2]}(1+b)\big\}, \end{aligned}$$

...

$$C_{[n]}(S) = a_{[n]} + \max \left\{ \begin{array}{l} \sum_{j=k+1}^{n-1} a_{[j]}(1+b)^{n-j} + C_{[k]}(S)(1+b)^{n-k}, \\ \sum_{j=k+1}^{n-1} a_{[j]}(1+b)^{n-j} + r_{[k+1]}(1+b)^{n-k}, \\ \sum_{j=k+2}^{n-1} a_{[j]}(1+b)^{n-j} + r_{[k+2]}(1+b)^{n-k-1}, \\ \vdots \\ r_{[n]}(1+b) \end{array} \right\}.$$

Hence

$$\begin{aligned} \sum_{j=1}^{n} w_j C_j(S) &= \sum_{j=1}^{k} w_{[j]} C_{[j]}(S) + \sum_{j=k+1}^{n} w_{[j]} C_{[j]}(S) \\ &\geq \sum_{j=1}^{k} w_{[j]} C_{[j]}(S) + C_{[k]}(S) \sum_{j=k+1}^{n} w_{[j]}(1+b)^{j-k} \\ &\quad + \sum_{j=k+1}^{n} w_{[j]} \left[\sum_{i=k+1}^{j} a_{[i]}(1+b)^{j-i} \right], \end{aligned} \tag{3}$$

where $\sum_{i=k+1}^{k} a_{[i]}(1+b)^{j-i} = 0$.

From Eq. (3), $\sum_{j=1}^{k} w_{[j]} C_{[j]}(S)$ and $C_{[k]}(S)$ are fixed constants. From Lemma 1, $\sum_{j=k+1}^{n} w_{[j]}(1+b)^{j-k}$ is minimized by the non-increasing order of w_i. If $a_i \leq a_j$ implies $w_i \geq w_j$ for all the jobs J_i and J_j, $\sum_{j=k+1}^{n} w_{[j]} \left[\sum_{i=k+1}^{j} a_{[i]}(1+b)^{j-i} \right]$ is minimized by the non-increasing (resp, non-decreasing) order of w_i (resp, a_i). So, the first lower bound is:

$$\begin{aligned} LB_1 &= \sum_{j=1}^{k} w_{[j]} C_{[j]}(S) + C_{[k]}(S) \sum_{j=k+1}^{n} w_{<j>}(1+b)^{j-k} \\ &\quad + \sum_{j=k+1}^{n} w_{<j>} \left[\sum_{i=k+1}^{j} a_{<<i>>}(1+b)^{j-i} \right], \end{aligned} \tag{4}$$

where $a_{<<k+1>>} \leq a_{<<k+2>>} \leq \ldots \leq a_{<<n>>}$, $w_{<k+1>} \geq w_{<k+2>} \geq \ldots \geq w_{<n>}$ (note that $a_{<<i>>}$ and $w_{<i>}$ may not refer to the same job).

However, if r_i is larger, it follows that

$$\sum_{j=1}^{n} w_j C_j(S) = \sum_{j=1}^{k} w_{[j]} C_{[j]}(S) + \sum_{j=k+1}^{n} w_{[j]} C_{[j]}(S)$$

$$\geq \sum_{j=1}^{k} w_{[j]}C_{[j]}(S) + (1+b)\sum_{j=k+1}^{n} w_{[j]}r_{[j]}. \tag{5}$$

From Eq. (5), $\sum_{j=1}^{k} w_{[j]}C_{[j]}(S)$ and $(1+b)\sum_{j=k+1}^{n} w_{[j]}r_{[j]} = (1+b)\sum_{j\in s_u} w_j r_j$ are fixed constants. Consequently, the second lower bound is:

$$LB_2 = \sum_{j=1}^{k} w_{[j]}C_{[j]}(S) + (1+b)\sum_{j\in s_u} w_j r_j. \tag{6}$$

In addition, we have

$$\begin{aligned}\sum_{j=1}^{n} w_j C_j(S) &= \sum_{j=1}^{k} w_{[j]}C_{[j]}(S) + \sum_{j=k+1}^{n} w_{[j]}C_{[j]}(S)\\ &\geq \sum_{j=1}^{k} w_{[j]}C_{[j]}(S) + r_{\min}\sum_{j=k+1}^{n} w_{[j]}(1+b)^{j-k}\\ &\quad + \sum_{j=k+1}^{n} w_{[j]}\left[\sum_{i=k+1}^{j} a_{[i]}(1+b)^{j-i}\right],\end{aligned} \tag{7}$$

where $r_{\min} = \min\{r_j | j \in s_u\}$. Similarly, from Eq. (7), the third lower bound is:

$$\begin{aligned}LB_3 &= \sum_{j=1}^{k} w_{[j]}C_{[j]}(S) + r_{\min}\sum_{j=k+1}^{n} w_{<j>}(1+b)^{j-k}\\ &\quad + \sum_{j=k+1}^{n} w_{<j>}\left[\sum_{i=k+1}^{j} a_{<<i>>}(1+b)^{j-i}\right],\end{aligned} \tag{8}$$

where $a_{<<k+1>>} \leq a_{<<k+2>>} \leq \ldots \leq a_{<<n>>}$, $w_{<k+1>} \geq w_{<k+2>} \geq \ldots \geq w_{<n>}$ (note that $a_{<<i>>}$ and $w_{<i>}$ do not invariably refer to an identical job).

To render the lower bound more precise, the maximum value among equations (4), (6), and (8) is chosen as a lower bound for $1\,|p_i = a_i + bt, r_i|\sum_{i=1}^{n} w_i C_i$, i.e., the lower bound for $1\,|p_i = a_i + bt, r_i|$ $\sum_{i=1}^{n} w_i C_i$ is

$$LB = \max\{LB_1, LB_2, LB_3\}. \tag{9}$$

3.4. Details of the branch-and-bound algorithm

The branch-and-bound (bb) algorithm uses the depth-first search strategy.

Algorithm 2: bb

Input: a_i, r_i, w_i, b

Output: The optimal job sequence S^* and $\sum w_i C_i(S^*)$

Phase 1: Initialization
Obtain an initial sequence using Algorithm 1;
Phase 2: Branch-and-bound search
Initialize the incumbent solution as the sequence from Phase 1;
Explore the search tree in a depth-first manner;
At each node, select an unscheduled job for the next position;
Apply dominance properties to discard dominated partial sequences;
Compute the lower bound LB using Eq. (9);
if $LB \geq$ *incumbent value* **then**
    Prune the node and its subsequent branches;
end
if *a complete schedule is obtained* **then**
    Compute its $\sum w_i C_i$;
    if $\sum w_i C_i <$ *incumbent value* **then**
        Update the incumbent solution;
    end
end
Continue traversing all remaining nodes;
return S^* *and* $\sum w_i C_i(S^*)$.

3.5. An example

In order to substantiate the validity of the aforementioned bb algorithm as well as the lower bounds when addressing the problem $1\,|p_i = a_i + bt, r_i|\sum_{i=1}^{n} w_i C_i$, an illustrative case is provided in this subsection. Take the following instance into consideration, where $n = 5$, $b = 0.25$, and other parameters of the jobs are given in the following table:

Table 1. Parameters of jobs.

	J_1	J_2	J_3	J_4	J_5
a_i	16	39	66	69	25
r_i	1	41	18	38	36
w_i	6	6	9	3	6

According to the data in Table 1, the sequence obtained by the ub algorithm is $J_1 \rightarrow J_3 \rightarrow J_5 \rightarrow J_2 \rightarrow J_4$, and its corresponding objective function value is 3953.93. Next, the specific steps of algorithm 2 (bb) are described in detail. When the processing sequence J_1 is fixed, the lower bound $LB = max\{LB_1, LB_2, LB_3\} = 3170.70$ can be obtained by comparing Eq. (4), Eq. (6), and Eq. (8). To

compute LB_1, the sorted sequences are:

$$a_{<<2>>} \le a_{<<3>>} \le a_{<<4>>} \le a_{<<5>>} = 25 \le 39 \le 66 \le 69,$$

$$w_{<2>} \ge w_{<3>} \ge w_{<4>} \ge w_{<5>} = 9 \ge 6 \ge 6 \ge 3.$$

$$\begin{aligned} LB_1 &= w_1 C_1(S) + C_1(S) \sum_{j=2}^{5} w_{<j>} (1+b)^{j-1} + \sum_{j=2}^{5} w_{<j>} \left[\sum_{i=2}^{j} a_{<<i>>} (1+b)^{j-i} \right] \\ &= 6 \times 17.25 + 17.25 \times \left(9 \times 1.25^1 + 6 \times 1.25^2 + 6 \times 1.25^3 + 3 \times 1.25^4 \right) + 9 \times 25 \\ &\quad + 6 \times \left(25 \times 1.25^1 + 39 \right) + 6 \times \left(25 \times 1.25^2 + 39 \times 1.25^1 + 66 \right) \\ &\quad + 3 \times \left(25 \times 1.25^3 + 39 \times 1.25^2 + 66 \times 1.25^1 + 69 \right) \\ &= 3141.09. \end{aligned}$$

$$\begin{aligned} LB_2 &= w_1 C_1(S) + (1+b) \sum_{j=2}^{5} w_j r_j \\ &= 6 \times 17.25 + 1.25 \times (6 \times 41 + 9 \times 18 + 3 \times 38 + 6 \times 36) \\ &= 1026. \end{aligned}$$

$$\begin{aligned} LB_3 &= w_1 C_1(S) + r_{\min} \sum_{j=2}^{5} w_{<j>} (1+b)^{j} + \sum_{j=2}^{5} w_{<j>} \left[\sum_{i=2}^{j} a_{<<i>>} (1+b)^{j-i} \right] \\ &= 6 \times 17.25 + 18 \times \left(9 \times 1.25^1 + 6 \times 1.25^2 + 6 \times 1.25^3 + 3 \times 1.25^4 \right) + 9 \times 25 \\ &\quad + 6 \times \left(25 \times 1.25^1 + 39 \right) + 6 \times \left(25 \times 1.25^2 + 39 \times 1.25^1 + 66 \right) \\ &\quad + 3 \times \left(25 \times 1.25^3 + 39 \times 1.25^2 + 66 \times 1.25^1 + 69 \right) \\ &= 3170.70. \end{aligned}$$

Therefore, according to Eq. (9), we can get: $LB = \max\{LB_1, LB_2, LB_3\} = 3170.70$. The other lower bounds are calculated similarly.

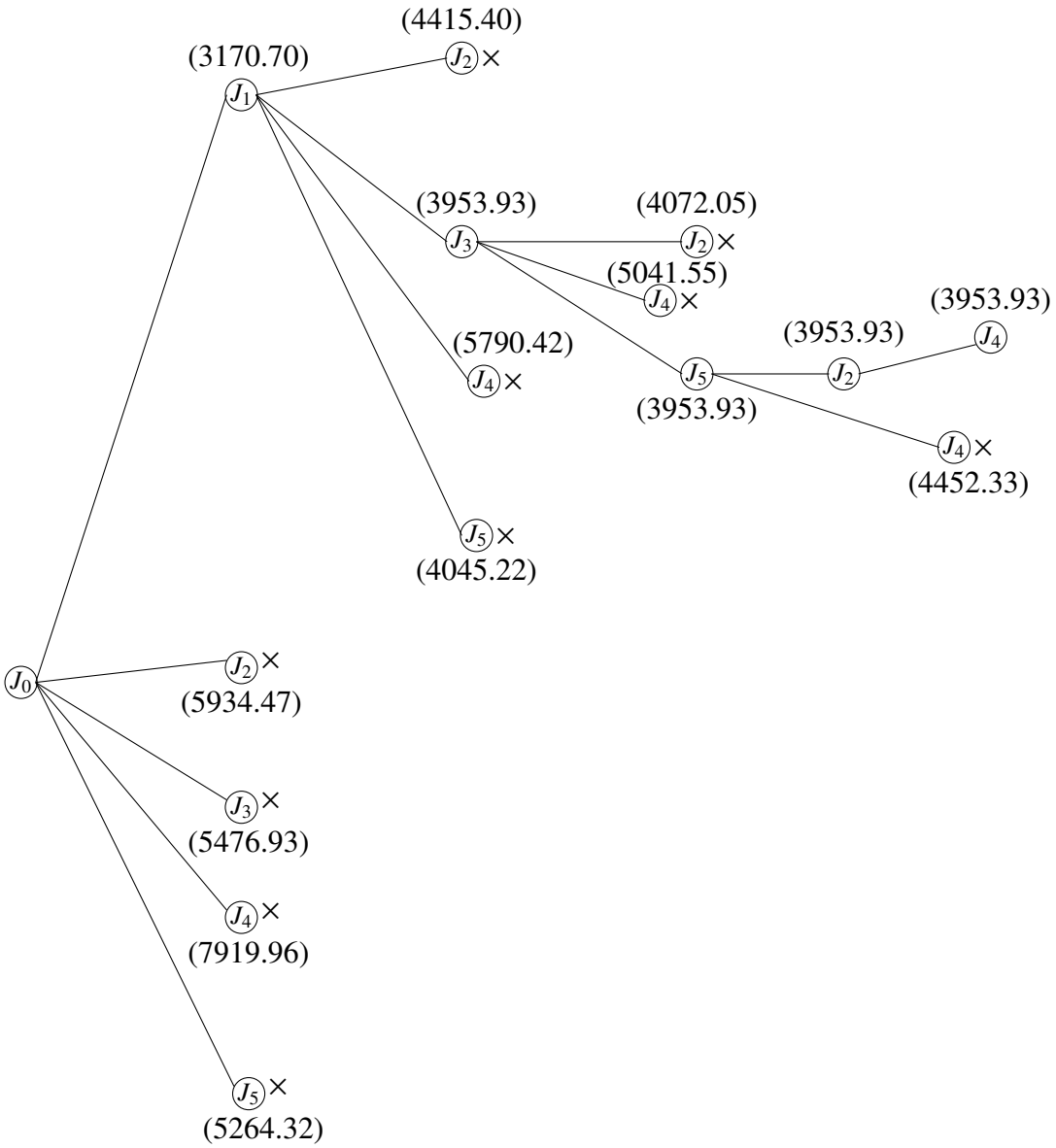

Figure 1. Search tree of the bb algorithm for Example 1 (× denotes pruning).

Based on the branch-bounding tree (i.e., Figure 1) obtained after executing the bb algorithm, it can be concluded that the optimal sequence in this case is $J_1 \rightarrow J_3 \rightarrow J_5 \rightarrow J_2 \rightarrow J_4$, and its corresponding objective function value is 3953.93.

4. Other algorithms

In this section, other algorithms are given to solve $1\,|p_i = a_i + bt, r_i|\sum_{i=1}^{n} w_i C_i$. First, the tabu search (ts) is used, the initial sequence is chosen by using Algorithm 1, and the ts algorithm terminates after at most $1000n$.

Algorithm 3: ts

Input: a_i, r_i, w_i, b

Output: The best job sequence S_{best} and $\sum w_i C_i(S_{best})$

Phase 1: Initialization
Initialize the tabu list as empty and the iteration counter as zero;
Generate the initial sequence using Algorithm 1, denoted as S_{best};
Compute $\sum w_i C_i(S_{best})$;

Phase 2: Tabu search iteration
while *iteration counter* < *maximum iterations* **do**
    Explore the neighborhood of the current sequence;
    Find the non-tabu neighboring sequence S' with minimum $\sum w_i C_i$;
    if $\sum w_i C_i(S') < \sum w_i C_i(S_{best})$ **then**
        $S_{best} = S'$;
    end
    Update the tabu list;
    Increment the iteration counter;
end
return S_{best} *and its objective value*.

On the other hand, simulated annealing (sa) can also be proposed for $1\,|p_i = a_i + bt, r_i|\sum_{i=1}^{n} w_i C_i$.

Algorithm 4: sa

Input: a_i, r_i, w_i, b

Output: The best job sequence S_{best} and $\sum w_i C_i(S_{best})$

Phase 1: Initialization

Generate the initial sequence (S_{best}) using Algorithm 1 (ub);

Phase 2: Simulated annealing iteration

Set the iteration counter $k = 0$;

while $k < 1000n$ **do**

  Generate a neighboring sequence using pairwise interchange, denoted as S_{new};

  Compute $\Delta = \sum w_i C_i(S_{new}) - \sum w_i C_i(S_{best})$;

  if $\Delta < 0$ **then**

    Accept S_{new} unconditionally;

  end

  else

    Compute the acceptance probability

$$P(\text{accept}) = \min\{1, \exp(-\alpha\Delta)\},$$

    where $\alpha = k/\delta$ and δ is a control parameter ($\delta = 1$ in our experiments);

    Generate a random number $\theta \sim U(0, 1)$;

    if $P(accept) > \theta$ **then**

      Accept S_{new};

    end

  end

  Update the current sequence if a new sequence was accepted;

  $k = k + 1$;

end

return *S_{best} and its objective value.*

We also adapt the neh (Nawaz–Enscore–Ham) heuristic (Nawaz et al. [53]) to our problem.

Algorithm 5: neh

Input: a_i, r_i, w_i, b

Output: The best job sequence S_{best} and $\sum w_i C_i(S_{best})$

Phase 1: Initialization

Generate the initial sequence using Algorithm 1 (ub);

Phase 2: NEH insertion procedure

Construct a partial sequence with the first two jobs from the initial sequence, considering both possible orders, and select the one with minimum $\sum w_i C_i$;

$j = 3$;

while $j \leq n$ **do**

  Take the j-th job from the initial sequence;

  Insert it into all j possible positions of the current partial sequence;

  Select the sequence with minimum $\sum w_i C_i$ as the new partial sequence;

  $j = j + 1$;

end

return S_{best} *and its objective value*.

Obviously, from Nawaz et al. [53], the time complexity associated with Algorithm 5 (i.e., the neh) is $O(n^2)$.

5. Mathematical programming model

Within this section, we construct a mathematical programming model aimed at addressing the scheduling problem $1\,|p_i = a_i + bt, r_i|\sum_{i=1}^{n} w_i C_i$. The detailed formulation is constructed as follows:

$$\min \sum_{j=1}^{n}\sum_{k=1}^{n} w_j Z_{jk} \tag{10}$$

s.t.

$$\sum_{j=1}^{n} Y_{jk} = 1, \quad k = 1, \ldots, n, \tag{11}$$

$$\sum_{k=1}^{n} Y_{jk} = 1, \quad j = 1, \ldots, n, \tag{12}$$

$$S_k \geq \sum_{j=1}^{n} r_j Y_{jk}, \quad k = 1, \ldots, n, \tag{13}$$

$$S_k \geq C_{k-1}, \quad k = 2, \ldots, n, \tag{14}$$

$$C_k = (1 + b)S_k + \sum_{j=1}^{n} a_j Y_{jk}, \quad k = 1, \ldots, n, \tag{15}$$

$$Z_{jk} \leq MY_{jk}, \quad j, k = 1, \ldots, n, \tag{16}$$

$$Z_{jk} \le C_k, \quad j,k = 1,\ldots,n, \tag{17}$$

$$Z_{jk} \ge C_k - M(1 - Y_{jk}), \quad j,k = 1,\ldots,n, \tag{18}$$

$$Y_{jk} \in \{0,1\}, \quad S_k, C_k, Z_{jk} \ge 0, \quad j,k = 1,\ldots,n. \tag{19}$$

Y_{jk} is a binary variable that equals 1 if job J_j is assigned to position k, and 0 otherwise. S_k (resp, C_k) denotes the start time (resp, completion time) of the job scheduled in the k-th position. Z_{jk} is an auxiliary variable introduced to linearize the product term $w_j C_k Y_{jk}$. Equation (10) minimizes the total weighted completion time. Constraints (11) and (12) are assignment constraints ensuring that each job occupies exactly one position and each position is taken by exactly one job. Inequality (13) forces the start time of the job in position k to be no earlier than its release time. Inequality (14) enforces the machine–idleness restriction, i.e., the job in position k can start only after the job in position $k-1$ has been completed. Equation (15) defines the completion time of the job in position k by incorporating the linear deterioration effect $p_i = a_i + bt$: the term $\sum_{j=1}^{n} a_j Y_{jk}$ selects the basic processing time of the job placed in position k, and the factor $(1+b)S_k$ accounts for the deteroration incurred from its start time. Constraints (16)–(18) are the big–M linearization constraints that replace the nonlinear expression $w_j C_k Y_{jk}$ with the auxiliary variable Z_{jk}, thereby converting the model into a solvable mixed-integer linear program. Constraint (19) specifies the domain of each variable.

The constant M is chosen as a sufficiently large upper bound on the latest possible completion time, for instance, $M = \max_j r_j + \sum_{j=1}^{n} a_j (1+b)^n$. The initial condition $C_0 = 0$ ensures that the first job can start at time zero or later, depending on its release time.

6. Computational experiments

In this section, the ub, neh, ts, sa, and bb algorithms are tested. All algorithms were programmed in C++ and tested under the following configuration: development platform - Visual Studio 2022 (version 17.1.0); memory limit - 64GB maximum allocation; hardware specifications - a Windows 10-based desktop workstation with an Intel(R) Core(TM) i5-10500 processor (3.10GHz base/boost frequency) and 8GB RAM.

In the experimental setup, the subsequent parameters are utilized to randomly generate experimental datasets:

(1) A small-scale set of jobs n =12, 13, 14, 15, 16, 17;

(2) A large-scale set of job n =120, 130, 140, 150, 160, 170;

(3) Deterioration factor $b = 0.05, 0.15, 0.25$;

(4) Normal processing time a_i is drawn from a uniform discrete set distribution spanning the interval [1, 100];

(5) Release time r_i is sampled from a uniform discrete set distributed over [1, 50], [50, 100], [1, 100];

(6) Weight w_i is sampled from a uniform discrete set distributed over [1, 10].

For each combination (i.e., n, a_i, b, r_i, and w_i), 30 instances are randomly generated, and then the generated instances are experimentally simulated to analyze the performance of each algorithm (i.e., ub, neh, ts, sa, and bb), and the criteria of running time and error bound are defined. For the running time, i.e., CPU time, the time unit of each instance is millisecond (ms). In the small-scale computation experiments, for the heuristics, the error bound is defined as $\frac{\sum_{i=1}^{n} w_i C_i(G)}{\sum_{i=1}^{n} w_i C_i(G^*)}$, where G is a solution obtained by the ub, neh, ts, and sa, and G^* is the optimal solution obtained by the bb. The simulation results are

shown in Tables 2–7, where Tables 2, 4, and 6 show the CPU times, and Tables 3, 5, and 7 are the error bounds. From Tables 2, 4, and 6, it can be seen that although the CPU time for bb grows rapidly as the number of jobs increases, the problem can still be solved in a reasonable time ($n \leq 17$), and for larger b, there are more smaller CPU times. From Tables 3, 5, 7, and Figure 2, we can conclude that neh and sa are more accurate than ub and ts, and their mean errors are 1.026 and 1.012, respectively. In addition, the neh also has a relatively short runtime, with its maximum runtime being only 2 ms.

In the large-scale computation experiments, the bb algorithm is disabled in this case, and for the heuristics, the error bound is modified to $\frac{\sum_{i=1}^{n} w_i C_i(G)}{\sum_{i=1}^{n} w_i C_i(G\_best)}$, where G is a solution obtained by the ub, neh, ts, and sa, and G_best is the best solution obtained by the ub, neh, ts, or sa. From Tables 8–13, the results show that sa outperforms the other algorithms in terms of both accuracy and efficiency for large-scale instances. From Figure 3, it can be visualized more clearly that the accuracy of the sa algorithm outperforms the other algorithms, that the change of the release time has a relatively small impact on the accuracy of the algorithms, and the performance of these algorithms are relatively stable.

To validate the performance of the bb algorithm, comparative trials were carried out between the newly proposed bb method and the mathematical programming formulation (Eqs. (10)–(19)) presented in Section 5, where the mathematical model was solved using the Gurobi optimizer (v11.0.3). The evaluation focused on small-scale instances with job counts $n = \{12, 13, 14, 15, 16, 17\}$, generating 10 randomized test cases for each parameter configuration. Computational outcomes are summarized in Tables 14–16, detailing: (1) execution times (in milliseconds) for both Gurobi and bb, and (2) node exploration counts for the bb algorithm. Statistical analysis reveals that for $n \leq 14$, the bb algorithm outperformed Gurobi in 77.8% of cases (21 out of 27 instances) based on average CPU time metrics.

Table 2. CPU time for $r_i \in U[1, 50]$.

		ub		neh		ts		sa		bb	
n	b	mean	max	mean	max	mean	max	mean	max	mean	max
12	0.05	1.35	2	0.6	1	307.75	334	6.1	9	3546.45	14903
	0.15	1.1	2	0.55	1	309.15	335	6.55	9	788.05	5065
	0.25	1.4	2	0.65	1	287.4	306	5.7	7	163.4	573
13	0.05	1.4	2	0.45	1	3278.15	58719	7.35	9	9939.4	39330
	0.15	1.35	3	0.7	1	3249.45	58009	6.85	9	2804.25	16595
	0.25	1.25	2	0.75	1	5586.05	52855	6.5	9	418.65	1580
14	0.05	1.55	2	0.85	1	426.15	452	7.7	10	52770.3	291875
	0.15	1.25	2	0.7	2	3957.1	70901	7.65	11	4901.7	16934
	0.25	1.3	2	0.9	2	420.45	517	7.45	11	773.6	2670
15	0.05	1.55	2	0.8	1	4709.2	84691	8	11	231988.7	1602539
	0.15	1.4	2	0.85	1	4859.9	87277	8.45	11	23396.25	240910
	0.25	1.3	3	0.8	2	8768	83520	8.65	14	4601.4	46116
16	0.05	1.4375	3	0.875	2	12582.56	96688	8.689	12	703236.8	3464028
	0.15	1.45	2	1	2	10762.25	102746	9.45	14	88967.2	468986
	0.25	1.45	2	0.9	2	571.85	617	8.7	12	5231.65	26585
17	0.05	1.48	2	1.037	2	681.48	731	9.4	15	37454.3	20086163
	0.15	1.55	2	1.2	2	698.6	743	10.05	13	292310	1891917
	0.25	1.6	2	1	1	6638.2	120087	8.55	11	16781	76431

Table 3. Error results for $r_i \in U[1, 50]$.

		ub		neh		ts		sa	
n	b	mean	max	mean	max	mean	max	mean	max
12	0.05	1.009	1.06	1.001	1.013	1.008	1.06	1.005	1.06
	0.15	1.01	1.065	1.002	1.028	1.005	1.035	1.002	1.028
	0.25	1.008	1.083	1.001	1.02	1.006	1.072	1.001	1.005
13	0.05	1.005	1.025	1.001	1.011	1.003	1.022	1.001	1.002
	0.15	1.013	1.077	1.002	1.01	1.008	1.048	1.007	1.048
	0.25	1.011	1.117	1.002	1.017	1.006	1.053	1.003	1.025
14	0.05	1.008	1.038	1.001	1.01	1.006	1.038	1.003	1.031
	0.15	1.006	1.039	1.004	1.031	1.004	1.027	1.003	1.027
	0.25	1.009	1.07	1.003	1.049	1.005	1.05	1.001	1.003
15	0.05	1.006	1.033	1.002	1.013	1.005	1.033	1.003	1.017
	0.15	1.009	1.053	1.001	1.005	1.004	1.037	1.002	1.037
	0.25	1.021	1.016	1.002	1.033	1.013	1.15	1.001	1.006
16	0.05	1.01	1.067	1.003	1.023	1.007	1.064	1.003	1.014
	0.15	1.013	1.059	1.003	1.015	1.007	1.036	1.002	1.016
	0.25	1.01	1.076	1.008	1.064	1.005	1.065	1.002	1.04
17	0.05	1.007	1.036	1.002	1.022	1.006	1.03	1.003	1.022
	0.15	1.009	1.035	1.003	1.014	1.005	1.022	1.001	1.01
	0.25	1.016	1.08	1.007	1.046	1.008	1.056	1.001	1.01

Table 4. CPU time for $r_i \in U[50, 100]$.

		ub		neh		ts		sa		bb	
n	b	mean	max	mean	max	mean	max	mean	max	mean	max
12	0.05	1.25	2	0.7	1	6709.55	43284	6	9	1805.8	7063
	0.15	1.35	3	0.25	1	2266.75	40228	6.15	9	339.15	1285
	0.25	1.3	2	0.6	1	288.9	315	5.85	8	196.75	1313
13	0.05	1.25	2	0.8	2	3089.85	55198	6.5	8	13698.35	46164
	0.15	1.05	2	0.55	1	2880.85	51441	6.35	9	1661.15	8932
	0.25	1.1	2	0.95	1	3060.7	54524	6.45	10	513.8	2079
14	0.05	1.4	2	0.65	1	3827.2	68578	7.55	11	88970.25	348790
	0.15	1.35	2	0.7	1	6707.05	63722	6.7	8	4345.3	15520
	0.25	1.25	2	1	2	3830.8	68606	8.4	12	1183.7	3354
15	0.05	1.2	2	0.85	1	4666.5	83905	8.5	11	69328.6	298545
	0.15	1.35	2	0.7	1	8259.75	78448	7.75	11	13652.75	117384
	0.25	1.55	2	0.9	2	499.75	532	8.45	11	3792.3	14410
16	0.05	1.5	2	1	1	11433.9	108941	8.9	11	561076.1	1906206
	0.15	1.3	3	1.1	2	14780.45	97809	8.25	11	61209	386763
	0.25	1.35	2	1.05	2	585.45	625	8.55	12	14334.2	70283
17	0.05	1.7	2	1.26	2	664.15	728	9.48	13	2690400	12638277
	0.15	1.6	2	1	1	11960.25	114200	9.05	12	151655.6	852499
	0.25	1.45	3	1.2	2	19185.7	127224	9.55	13	29950.5	293267

Table 5. Error results for $r_i \in U[50, 100]$.

		ub		neh		ts		sa	
n	b	mean	max	mean	max	mean	max	mean	max
12	0.05	1.006	1.062	1.001	1.003	1.004	1.054	1.001	1.004
	0.15	1.015	1.069	1.004	1.03	1.012	1.064	1.005	1.03
	0.25	1.006	1.026	1.002	1.026	1.005	1.04	1.002	1.04
13	0.05	1.007	1.03	1.001	1.003	1.005	1.028	1.003	1.028
	0.15	1.007	1.042	1.002	1.023	1.004	1.043	1.003	1.043
	0.25	1.01	1.043	1.004	1.035	1.006	1.035	1.004	1.035
14	0.05	1.009	1.041	1.001	1.016	1.006	1.041	1.005	1.041
	0.15	1.005	1.042	1.002	1.022	1.002	1.022	1.001	1.022
	0.25	1.008	1.052	1.002	1.023	1.004	1.034	1.002	1.034
15	0.05	1.008	1.034	1.002	1.02	1.005	1.034	1.002	1.021
	0.15	1.007	1.049	1.003	1.022	1.005	1.039	1.001	1.018
	0.25	1.011	1.054	1.004	1.036	1.01	1.054	1.008	1.054
16	0.05	1.005	1.024	1.001	1.004	1.003	1.021	1.001	1.003
	0.15	1.019	1.069	1.003	1.031	1.009	1.061	1.001	1.011
	0.25	1.02	1.136	1.01	1.077	1.015	1.077	1.009	1.077
17	0.05	1.005	1.027	1.001	1.011	1.002	1.022	1.001	1.01
	0.15	1.01	1.067	1.005	1.026	1.008	1.047	1.001	1.011
	0.25	1.011	1.039	1.006	1.03	1.008	1.03	1.004	1.03

Table 6. CPU time for $r_i \in U[1, 100]$.

		ub		neh		ts		sa		bb	
n	b	mean	max	mean	max	mean	max	mean	max	mean	max
12	0.05	1.25	2	0.35	1	2424	43016	6.6	13	1607.7	8246
	0.15	1.35	2	0.5	1	4262.05	40288	5.5	8	360	1732
	0.25	1.4	2	0.45	1	2268.4	40207	5.55	6	252.95	1241
13	0.05	1.2	2	0.45	1	3065.5	54676	6.95	10	3683.65	16245
	0.15	1.4	2	0.65	1	2871.15	51191	6.25	8	921.1	4434
	0.25	1.2	2	0.55	1	2868	51218	6.5	9	498.75	2926
14	0.05	1.3	2	0.9	1	7156.95	67990	7.7	10	48286.65	164838
	0.15	1.4	2	0.6	1	3562.6	63894	7	9	3771.95	12397
	0.25	1.2	2	0.8	1	6845.75	66257	7.1	10	926.8	3836
15	0.05	1.5	2	0.75	1	12986.65	84114	7.95	12	171297.3	1706856
	0.15	1.65	2	0.9	1	8337.3	79271	7.95	11	13364.05	46608
	0.25	1.4	2	0.85	1	8250.05	78579	7.25	10	2068.25	11176
16	0.05	1.45	2	0.95	1	26145.25	106849	9	14	414092.55	1278991
	0.15	1.5	2	0.95	2	5289.55	95410	8.25	12	40471.3	232524
	0.25	1.5	2	0.9	1	19604.2	96639	7.95	12	5103.8	18153
17	0.05	1.42	2	1.28	2	680.95	715	9.38	16	3314912.9	26756202
	0.15	1.55	3	1.2	2	646.2	675	9.35	12	116254	1000509
	0.25	1.45	2	1.1	2	12438.1	120699	9.1	11	16860.15	62927

Table 7. Error results for $r_i \in U[1, 100]$.

		ub		neh		ts		sa	
n	b	mean	max	mean	max	mean	max	mean	max
	0.05	1.018	1.067	1.006	1.036	1.012	1.063	1.001	1.008
12	0.15	1.029	1.161	1.008	1.08	1.021	1.0132	1.001	1.011
	0.25	1.029	1.1	1.012	1.069	1.026	1.095	1.005	1.095
	0.05	1.014	1.085	1.005	1.037	1.01	1.085	1.008	1.085
13	0.15	1.032	1.268	1.007	1.119	1.022	1.253	1.005	1.055
	0.25	1.035	1.142	1.012	1.1	1.016	1.089	1.009	1.078
	0.05	1.026	1.102	1.008	1.08	1.016	1.087	1.002	1.011
14	0.15	1.049	1.187	1.026	1.137	1.035	1.187	1.012	1.187
	0.25	1.042	1.177	1.014	1.128	1.035	1.171	1.005	1.051
	0.05	1.024	1.087	1.008	1.045	1.017	1.085	1.002	1.01
15	0.15	1.032	1.137	1.007	1.035	1.019	1.062	1.002	1.017
	0.25	1.027	1.217	1.012	1.186	1.022	1.191	1.006	1.046
	0.05	1.027	1.104	1.013	1.055	1.021	1.104	1.005	1.053
16	0.15	1.029	1.1	1.012	1.087	1.017	1.093	1.002	1.044
	0.25	1.042	1.121	1.013	1.098	1.023	1.097	1.006	1.075
	0.05	1.026	1.089	1.007	1.066	1.019	1.088	1.004	1.021
17	0.15	1.029	1.116	1.012	1.081	1.023	1.113	1.001	1.012
	0.25	1.038	1.245	1.011	1.053	1.029	1.186	1.004	1.042

Table 8. CPU time for $r_i \in U[1, 50]$.

		ub		neh		ts		sa	
n	b	mean	max	mean	max	mean	max	mean	max
	0.05	36.4	52	342.1	412	44255.55	151240	58.2	78
120	0.15	32.25	61	359.5	404	41904.3	130299	65	85
	0.25	30.55	42	358.35	384	41173.35	120142	64.75	84
	0.05	39.4	54	501.1	546	49342.05	60135	92.1	114
130	0.15	37.55	57	516.85	544	48185.25	60115	98.35	121
	0.25	39.15	50	499.4	536	59923.6	190143	97.9	116
	0.05	46.4	61	660.15	736	67018.05	235078	78.65	95
140	0.15	45.45	54	585.15	615	68583.85	250015	71.25	82
	0.25	44.05	53	583.3	595	50043.8	50473	70.65	77
	0.05	56.1	75	805	897	78804.95	351456	87.3	110
150	0.15	54.5	61	873.05	910	100815.5	329165	91.8	112
	0.25	56	70	873.3	949	69112.15	71289	89	110
	0.05	68.85	88	1118.4	1154	103971.1	289651	98.75	115
160	0.15	68	88	1125.05	1189	125902.7	386579	103.9	125
	0.25	65.55	83	1062.35	1130	100278.35	328514	96.75	112
	0.05	80.3	91	1329.95	1421	138768.6	495142	107.2	122
170	0.15	78.9	93	1327.75	1391	125071.6	456812	102.3	110
	0.25	76.4	89	1320.3	1370	142442.6	564238	102.9	112

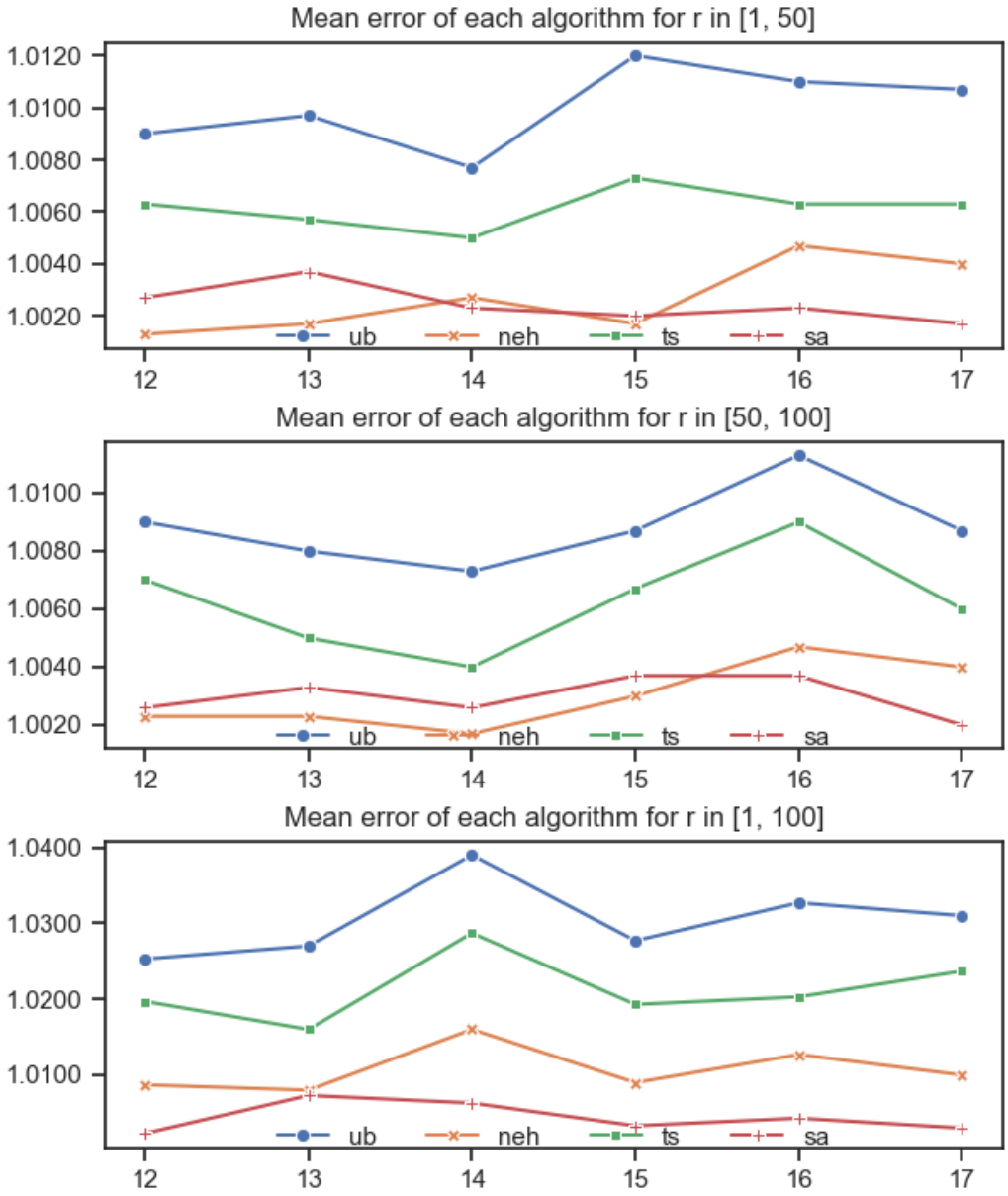

Figure 2. Algorithm-specific mean error in small-scale scenarios.

Table 9. Error results for $r_i \in U[1, 50]$.

		ub		neh		ts		sa	
n	b	mean	max	mean	max	mean	max	mean	max
120	0.05	1.040	1.070	1.027	1.069	1.032	1.067	1.000	1.001
	0.15	1.188	1.465	1.124	1.367	1.133	1.349	1.000	1.000
	0.25	1.141	1.851	1.067	1.474	1.097	1.680	1.000	1.001
130	0.05	1.047	1.031	1.083	1.039	1.039	1.077	1.000	1.000
	0.15	1.175	1.446	1.125	1.409	1.131	1.404	1.000	1.000
	0.25	1.125	1.416	1.073	1.414	1.077	1.288	1.000	1.000
140	0.05	1.055	1.090	1.035	1.079	1.046	1.088	1.000	1.000
	0.15	1.254	1.512	1.117	1.379	1.197	1.439	1.000	1.002
	0.25	1.281	2.097	1.134	1.682	1.195	1.914	1.000	1.000
150	0.05	1.052	1.095	1.041	1.084	1.040	1.090	1.001	1.023
	0.15	1.269	1.730	1.132	1.357	1.188	1.478	1.008	1.163
	0.25	1.131	1.598	1.070	1.386	1.071	1.281	1.003	1.048
160	0.05	1.061	1.087	1.041	1.072	1.051	1.078	1.001	1.027
	0.15	1.283	1.806	1.156	1.493	1.213	1.581	1.001	1.013
	0.25	1.144	2.134	1.083	1.757	1.107	2.090	1.000	1.000
170	0.05	1.059	1.107	1.045	1.100	1.050	1.105	1.000	1.000
	0.15	1.269	2.245	1.105	1.280	1.205	1.697	1.000	1.007
	0.25	1.206	1.927	1.115	1.633	1.125	1.652	1.000	1.000

Table 10. CPU time for $r_i \in U[50, 100]$.

		ub		neh		ts		sa	
n	b	mean	max	mean	max	mean	max	mean	max
120	0.05	32.15	44	338.1	375	37661.65	96281	57.55	68
	0.15	33.35	45	334.95	348	38217.5	125374	55.8	67
	0.25	35.2	50	339.6	372	38631.4	117345	57	78
130	0.05	42.35	53	463.85	495	50065.85	196325	63.8	71
	0.15	42.1	51	467.75	531	65512.9	245817	64.45	81
	0.25	42.4	55	464.5	495	51968.9	221389	65.55	77
140	0.05	48.25	56	620.7	656	73097.95	278123	73.9	89
	0.15	50.75	64	627.3	671	60256.3	203118	73.25	80
	0.25	51	68	624.25	662	83256.1	318475	74.4	93
150	0.05	59.75	73	817.3	857	87931.3	352743	84.4	102
	0.15	59.1	51	808.05	856	91408.45	372133	83.95	95
	0.25	57	76	804.45	868	90943.7	340125	85	117
160	0.05	68.15	82	1043.25	1082	77128.15	78058	95.6	114
	0.15	70.35	101	1032.8	1065	76941.15	77772	93.15	111
	0.25	69.6	88	1045	1102	115077	391154	92.85	105
170	0.05	81.8	95	1385	1514	138760.4	300139	114.4	139
	0.15	82.95	106	1431.4	1503	119586.55	420071	117	148
	0.25	82.05	100	1464.7	1515	103751.9	105025	115.95	136

Table 11. Error results for $r_i \in U[50, 100]$.

		ub		neh		ts		sa	
n	b	mean	max	mean	max	mean	max	mean	max
120	0.05	1.027	1.073	1.020	1.052	1.020	1.046	1.000	1.005
	0.15	1.117	1.252	1.063	1.209	1.069	1.154	1.000	1.000
	0.25	1.046	1.283	1.021	1.076	1.012	1.062	1.000	1.000
130	0.05	1.031	1.056	1.022	1.055	1.026	1.050	1.001	1.022
	0.15	1.143	1.463	1.057	1.218	1.089	1.351	1.004	1.048
	0.25	1.047	1.199	1.029	1.169	1.018	1.123	1.000	1.000
140	0.05	1.050	1.093	1.030	1.069	1.040	1.079	1.000	1.008
	0.15	1.141	1.314	1.092	1.187	1.100	1.276	1.000	1.000
	0.25	1.068	1.293	1.027	1.102	1.032	1.136	1.000	1.004
150	0.05	1.040	1.070	1.030	1.062	1.029	1.064	1.000	1.000
	0.15	1.092	1.288	1.046	1.186	1.054	1.169	1.003	1.050
	0.25	1.073	1.295	1.033	1.132	1.031	1.188	1.000	1.000
160	0.05	1.047	1.102	1.036	1.096	1.040	1.086	1.000	1.008
	0.15	1.134	1.322	1.062	1.180	1.081	1.235	1.007	1.140
	0.25	1.061	1.378	1.023	1.090	1.025	1.126	1.000	1.000
170	0.05	1.052	1.092	1.038	1.079	1.042	1.083	1.000	1.000
	0.15	1.133	1.279	1.070	1.196	1.088	1.235	1.000	1.000
	0.25	1.059	1.329	1.026	1.135	1.018	1.165	1.000	1.006

Table 12. CPU time for $r_i \in U[1, 100]$.

		ub		neh		ts		sa	
n	b	mean	max	mean	max	mean	max	mean	max
120	0.05	32.5	45	353.25	381	45499.7	125069	61.15	80
	0.15	37	47	352.6	388	47079.2	98539	59.9	84
	0.25	34.65	46	350.45	368	62954.8	150695	60.45	73
130	0.05	42.7	54	469.4	517	51936	195167	67.55	82
	0.15	42.1	53	472.8	493	72659.9	209153	69.95	94
	0.25	41.8	54	477.05	516	75430.15	235873	70.95	86
140	0.05	48.3	64	633.25	701	71857.15	200096	75.65	93
	0.15	47.2	56	666.1	746	116685.3	257103	82.3	98
	0.25	49.35	66	673.4	741	76520.4	236578	83.65	108
150	0.05	56.95	64	876.6	979	94762.2	296220	92.6	126
	0.15	53.45	61	847.6	959	69815.25	72360	90.25	103
	0.25	55.95	70	849	900	96124.75	245329	92.6	116
160	0.05	70.85	85	1096.4	1153	105212.1	312453	104.9	130
	0.15	64.45	83	1090.9	1176	106522.3	343655	100.05	116
	0.25	64.45	74	1104.1	1179	93961.9	300112	102.95	121
170	0.05	77.65	89	1392.7	1440	111538.2	352131	114.5	132
	0.15	78.1	95	1438.1	1520	144361.1	321174	121.55	142
	0.25	78.7	91	1389.95	1473	132893	385620	111.7	142

Table 13. Error results for $r_i \in U[1, 100]$.

		ub		neh		ts		sa	
n	b	mean	max	mean	max	mean	max	mean	max
120	0.05	1.027	1.073	1.020	1.052	1.020	1.046	1.000	1.005
	0.15	1.117	1.252	1.063	1.209	1.069	1.154	1.000	1.000
	0.25	1.046	1.283	1.021	1.076	1.012	1.062	1.000	1.000
130	0.05	1.031	1.056	1.022	1.055	1.026	1.050	1.001	1.022
	0.15	1.143	1.463	1.057	1.218	1.089	1.351	1.004	1.048
	0.25	1.047	1.199	1.029	1.169	1.018	1.123	1.000	1.000
140	0.05	1.050	1.093	1.030	1.069	1.040	1.079	1.000	1.008
	0.15	1.141	1.314	1.092	1.187	1.100	1.276	1.000	1.000
	0.25	1.068	1.293	1.027	1.102	1.032	1.136	1.000	1.004
150	0.05	1.040	1.070	1.030	1.062	1.029	1.064	1.000	1.000
	0.15	1.092	1.288	1.046	1.186	1.054	1.169	1.003	1.050
	0.25	1.073	1.295	1.033	1.132	1.031	1.188	1.000	1.000
160	0.05	1.047	1.102	1.036	1.096	1.040	1.086	1.000	1.008
	0.15	1.134	1.322	1.062	1.180	1.081	1.235	1.007	1.140
	0.25	1.061	1.378	1.023	1.090	1.025	1.126	1.000	1.000
170	0.05	1.052	1.092	1.038	1.079	1.042	1.083	1.000	1.000
	0.15	1.133	1.279	1.070	1.196	1.088	1.235	1.000	1.000
	0.25	1.059	1.329	1.026	1.135	1.018	1.165	1.000	1.006

Table 14. Results of Gurobi and bb for $r_i \in U[1, 50]$.

		cpu time (ms) of Gurobi		cpu time (ms) of bb		number of search nodes for bb	
n	b	mean	max	mean	max	mean	max
12	0.05	839.30	1622.00	466.90	1128.00	57146.30	191929.00
	0.15	556.60	1112.00	96.00	228.00	8283.80	20403.00
	0.25	443.00	691.00	59.90	277.00	5611.10	30277.00
13	0.05	1466.90	3552.00	2098.10	7137.00	181630.20	721553.00
	0.15	643.40	1109.00	305.70	722.00	22880.10	57885.00
	0.25	631.30	1176.00	73.70	182.00	4219.00	8378.00
14	0.05	3215.30	6858.00	9258.40	25762.00	766984.60	2665687.00
	0.15	1971.00	5091.00	1515.00	3132.00	102289.60	211674.00
	0.25	881.60	1975.00	174.90	507.00	9828.20	30106.00
15	0.05	6756.70	20899.00	60958.80	362078.00	4642975.00	28580580.00
	0.15	1886.40	3843.00	3839.80	12611.00	217766.70	969154.00
	0.25	991.50	2660.00	959.30	2653.00	43959.70	134313.00
16	0.05	11798.30	49887.00	256517.60	1411956.00	19773616.10	120741937.00
	0.15	1466.00	4721.00	5228.30	21342.00	258575.80	1098862.00
	0.25	943.30	1932.00	2684.10	7699.00	113817.40	336486.00
17	0.05	10419.60	36942.00	560097.00	1905439.00	32625807.50	139698218.00
	0.15	5226.30	19151.00	74303.80	385118.00	3503886.20	19431464.00
	0.25	2170.80	6979.00	2722.80	11554.00	84500.30	379562.00

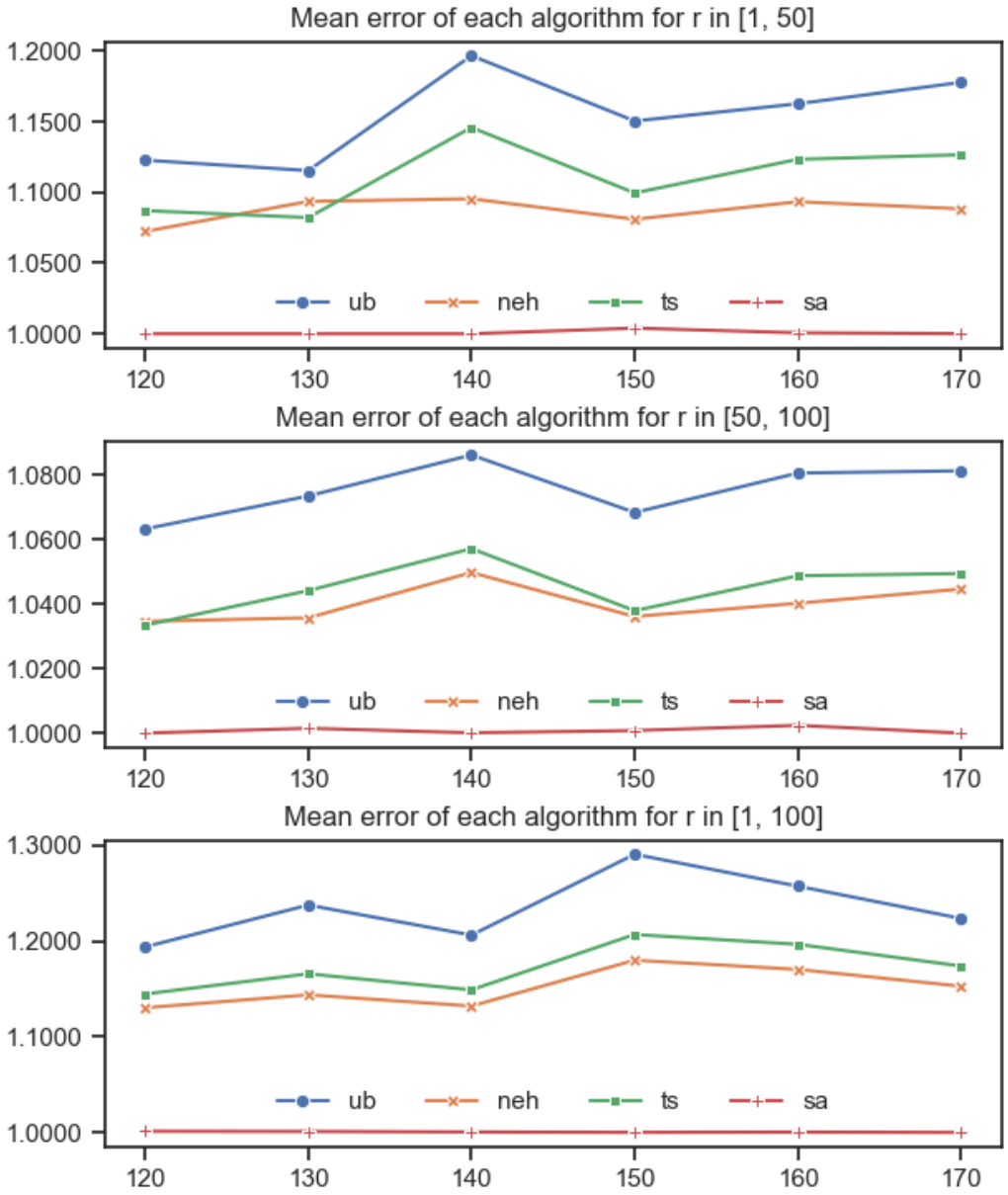

Figure 3. Algorithm-specific mean error in large-scale scenarios.

Table 15. Results of Gurobi and bb for $r_i \in U[50, 100]$.

		cpu time (ms) of Gurobi		cpu time (ms) of bb		number of search nodes for bb	
n	b	mean	max	mean	max	mean	max
12	0.05	1127.00	1801.00	735.40	1725.00	82030.60	214164.00
	0.15	727.70	1211.00	241.20	734.00	19208.90	55736.00
	0.25	424.00	704.00	48.60	216.00	3603.50	17363.00
13	0.05	1253.30	1776.00	2623.80	9072.00	247807.20	1014509.00
	0.15	986.20	1370.00	262.40	521.00	17494.40	37367.00
	0.25	587.30	1044.00	157.00	347.00	10770.40	21907.00
14	0.05	2132.50	8830.00	10736.30	53940.00	1115646.30	6026836.00
	0.15	1800.70	7052.00	504.80	1317.00	24173.50	61015.00
	0.25	706.20	1506.00	359.50	712.00	18569.60	43169.00
15	0.05	8104.80	45554.00	66536.20	426773.00	4708261.30	29241183.00
	0.15	3375.40	26618.00	5790.50	39104.00	319156.50	2178729.00
	0.25	739.90	1055.00	841.30	3419.00	38702.30	168416.00
16	0.05	10973.40	25919.00	346112.70	2493062.00	27823947.30	216657510.00
	0.15	2252.00	8108.00	21689.80	105569.00	1132409.70	5926768.00
	0.25	1293.40	3560.00	1957.00	5451.00	58789.70	141473.00
17	0.05	12990.30	45687.00	403835.90	2052096.00	24202019.60	133440606.00
	0.15	3365.20	11266.00	27164.50	102913.00	1037359.10	4244437.00
	0.25	1920.90	5352.00	4316.90	12813.00	134388.30	369327.00

Table 16. Results of Gurobi and bb for $r_i \in U[1, 100]$.

		cpu time (ms) of Gurobi		cpu time (ms) of bb		number of search nodes for bb	
n	b	mean	max	mean	max	mean	max
12	0.05	1083.40	2037.00	667.00	2830.00	79013.50	388067.00
	0.15	561.40	1484.00	62.20	266.00	5307.30	21922.00
	0.25	492.90	1009.00	21.80	87.00	1885.90	7397.00
13	0.05	1776.10	6932.00	2529.30	8318.00	273094.30	824317.00
	0.15	776.00	1045.00	270.50	886.00	21717.10	67595.00
	0.25	593.50	1448.00	49.80	235.00	3905.90	17980.00
14	0.05	4516.30	24367.00	4697.00	15481.00	357474.70	1322988.00
	0.15	1625.80	4307.00	907.10	2937.00	53600.90	159180.00
	0.25	630.50	989.00	205.60	568.00	10947.20	27312.00
15	0.05	7688.40	33123.00	22584.20	81725.00	1586999.40	6308583.00
	0.15	1944.50	9306.00	2839.20	18381.00	149422.40	986315.00
	0.25	1027.60	2604.00	701.60	2813.00	32037.20	114062.00
16	0.05	11978.40	50081.00	100874.00	455826.00	6394577.20	26504167.00
	0.15	2762.40	8438.00	8983.50	33769.00	483458.10	2160956.00
	0.25	1394.60	2894.00	1537.50	5505.00	60306.30	264115.00
17	0.05	65102.70	495188.00	267329.50	781756.00	13718723.00	37833418.00
	0.15	22575.00	165222.00	56100.70	387076.00	2321179.20	16589785.00
	0.25	2180.50	4585.00	3411.80	19972.00	127447.50	718967.00

7. Conclusions

This article studied the single-machine total weighted completion time minimization with linear deterioration effects and release times. For this NP-hard problem, several heuristic and branch-and-bound algorithms were proposed. The computational outcomes indicated that the bb method is capable of addressing problem instances involving 17 jobs within an acceptable timeframe. The neh and sa algorithms are more accurate than ub and ts for small-scale instances, while sa outperforms the other algorithms for large-scale instances. For future research, the problems with m ($m > 1$) parallel machines (see Chen et al. [54]) or m-machine flow (job, open) shops (see Li et al. [55], Wang et al. [56], Sun et al. [57], Azerine et al. [58], Lin and Wang [59], Abreu and Nagano [60]) and the problems with

job-rejection (see Liu et al. [61], Geng et al. [62], Geng et al. [63]) will be considered. Other interesting topics might concern the scheduling with learning effects (see Zhang et al. [64]) or resource allocations (see Lv and Wang [65], Huang et al. [66], Zhang and Wang [67], Zhang and Yin [68]).

Author contributions

Li-Han Zhang, Xiao-Yuan Wang, Na Yin, Zheng-Guo Lv, and Ji-Bo Wang designed the methodology; Xiao-Yuan Wang, Na Yin, and Ji-Bo Wang made the investigation; Li-Han Zhang wrote the original draft; and Li-Han Zhang, Xiao-Yuan Wang, Na Yin, and Ji-Bo Wang reviewed and edited the manuscript. All authors have read and agreed to the published version of the manuscript.

Acknowledgments

The authors acknowledge the helpful comments from the editor and anonymous reviewers.

Conflict of interest

The authors declare that there is no conflict of interest.

Funding

This work was supported by the Science Research Foundation of the Educational Department of Liaoning Province (LJ222510143003).

References

1. G. Mosheiov, ∧-shaped policies to schedule deteriorating jobs, *J. Oper. Res. Soc.,* **47** (1996), 1184–1191. https://doi.org/10.1057/jors.1996.146
2. A. Kononov, Single machine scheduling problems with processing times proportional to an arbitrary function, *Discret. Analysis Oper. Res.* **5** (1998), 17–37 (in Russian). http://old.math.nsc.ru/publishing/DAOR/content/1998/03/17.pdf
3. S. Browne, U. Yechiali, Scheduling deteriorating jobs on a single processor. *Oper. Res.* **38** (1990), 495–498. https://doi.org/10.1287/opre.38.3.495
4. S. Gawiejnowicz, *Models and algorithms of time-dependent scheduling*, Vol. 2, Springer, 2020. https://doi.org/10.1007/978-3-662-59362-2
5. T. C. E. Cheng, L. Kang, C. T. Ng, Due-date assignment and single machine scheduling with deteriorating jobs, *J. Oper. Res. Soc.,* **55** (2004), 198–203. https://doi.org/10.1057/palgrave.jors.2601681
6. V. A. Strusevich, K. Rustogi, *Scheduling with Time-Changing Effects and Rate-Modifying Activities*. Springer, Berlin-Heidelberg (2017). https://link.springer.com/book/10.1007/978-3-319-39574-6
7. J. Labetoulle, E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, Preemptive scheduling of uniform machines subject to release dates, In: W. R. Pulleyblank, (ed.) *Progress in Combinatorial Optimization*, 245–261. Academic Press, New York (1984). https://doi.org/10.1016/B978-0-12-566780-7.50020-9

8. J. K. Lenstra, A. H. G. R. Kan, P. Brucker, Complexity of machine scheduling problems, *Ann. Discret. Math.,* **1** (1977), 343–362. https://doi.org/10.1016/S0167-5060(08)70743-X
9. C. X. Miao, Complexity of scheduling with proportional deterioration and release dates, *Iran. J. Sci. Tech., Trans. A: Sci.,* **42** (2018), 1337–1342. https://doi.org/10.1007/s40995-017-0466-8
10. Z. G. Lv, L. H. Zhang, X. Y. Wang, J. B. Wang, Single machine scheduling proportionally deteriorating jobs with ready times subject to the total weighted completion time minimization, *Mathematics* **12**(2024), 610. https://doi.org/10.3390/math12040610
11. M. Atsmony, G. Mosheiov, Minimizing total completion time with linear deterioration: A new lower bound, *Comput. Ind. Eng.,* **163** (2022), 107867. https://doi.org/10.1016/j.cie.2021.107867
12. Q. F. Kong, C. M. Wei, J. B. Wang, Minmax delivery completion time scheduling with delivery times and deteriorating jobs, *Comput. Appl. Math.,* **45** (2026), 276. https://doi.org/10.1007/s40314-026-03684-7
13. H. He, Y. Hu, W. W. Liu, Scheduling with deteriorating effect and maintenance activities under parallel processors, *Eng. Optim.,* **53** (2021), 2070–2087. https://doi.org/10.1080/0305215X.2020.1844194
14. W. Liu, X. Wang, L. Li, P. Zhao, A maintenance activity scheduling with time-and-position dependent deteriorating effects, *Math. Biosci. Eng.,* **19** (2022), 11756–11767. https://doi.org/10.3934/mbe.2022547
15. X. Sun, X. N. Geng, Single-machine scheduling with deteriorating effects and machine maintenance, *Int. J. Prod. Res.,* **57** (2019), 3186–3199. https://doi.org/10.1080/00207543.2019.1566675
16. J. Qian, Y. Zhan, The due window assignment problems with deteriorating job and delivery time, *Mathematics,* **10** (2022), 1672. https://doi.org/10.3390/math10101672
17. D. Y. Lv, J. Xue, J. B. Wang, Minmax common due-window assignment scheduling with deteriorating jobs, *J. Oper. Res. Soc. China,* **12** (2024), 681–693. https://doi.org/10.1007/s40305-023-00511-2
18. L. H. Zhang, X. N. Geng, J. Xue, J. B. Wang, Single machine slack due window assignment and deteriorating jobs, *J. Ind. Manag. Optim.,* **20** (2024), 1593–1614. https://doi.org/10.3934/jimo.2023136
19. F. Liu, J. Yang, Y. Y. Lu, Solution algorithms for single-machine group scheduling with ready times and deteriorating jobs, *Eng. Optim.,* **51** (2019), 862–874. https://doi.org/10.1080/0305215X.2018.1500562
20. N. Yin, M. Gao, Single-machine group scheduling with general linear deterioration and truncated learning effects, *Comput. Appl. Math.,* **43** (2024), 386. https://doi.org/10.1007/s40314-024-02881-6
21. N. Yin, H. He, Y. Zhao, Y. Chang, N. Wang, Integrating group setup time deterioration effects and job processing time learning effects with group technology in single-machine green scheduling, *Axioms,* **14** (2025), 480. https://doi.org/10.3390/axioms14070480
22. D. Y. Lv, J. B. Wang, No-idle flow shop scheduling with deteriorating jobs and common due date under dominating machines, *Asia-Pacific J. Oper. Res.,* **41** (2024), 2450003. https://doi.org/10.1142/S0217595924500039
23. D. Y. Lv, J. B. Wang, Research on two-machine flow shop scheduling problem with release dates and truncated learning effects, *Eng. Optim.,* **57** (2025), 1828–1848. https://doi.org/10.1080/0305215X.2024.2372633

24. C. X. Miao, J. Zou, Parallel-machine scheduling with time-dependent and machine availability constraints, *Math. Prob. Eng.,* **2015** (2015), 956158. https://doi.org/10.1155/2015/956158

25. D. W. Li, X. W. Lu, Parallel-batch scheduling with deterioration and rejection on a single machine, *Appl. Math. J. Chinese Univ.,* **35** (2020), 141–156. https://doi.org/10.1007/s11766-020-3624-2

26. D. Y. Bai, H. Xue, L. Wang, C. C. Wu, W. C. Lin, D. H. Abdulkadir, Effective algorithms for single-machine learning-effect scheduling to minimize completion-time-based criteria with release dates, *Expert Syst. Appl.,* **156** (2020), 113445. https://doi.org/10.1016/j.eswa.2020.113445

27. J. Qian, H. X. Lin, Y. F. Kong, Y. S. Wang, Tri-criteria single machine scheduling model with release times and learning factor, *Appl. Math. Comput.* **387** (2020), 124543. https://doi.org/10.1016/j.amc.2019.06.057

28. X. Sun, X. N. Geng, F. Liu, Flow shop scheduling with general position weighted learning effects to minimise total weighted completion time, *J. Oper. Res. Soc.,* **72** (2021), 2674–2689. https://doi.org/10.1080/01605682.2020.1806746

29. L. Y. Wang, Due-window assignment scheduling problems with position-dependent weights, truncated learning effects and past-sequence-dependent setup-times, *Symmetry,* **18** (2026), 396. https://doi.org/10.3390/sym18030396

30. H. R. Song, J. K. Yang, J. B. Wang, Study on multiple due-windows assignment scheduling with learning and deteriorating effects, *Asia-Pacific J. Oper. Res.,* (2026), 2650001. https://doi.org/10.1142/S0217595926500016

31. Y. Y. Lu, M. H. Li, A unified analysis for single-machine scheduling with resource allocations, learning and deteriorating effects simultaneously, *Comput. Appl. Math.,* **45** (2026), 388. https://doi.org/10.1007/s40314-026-03784-4

32. J. B. Wang, Y. C. Wang, C. Wan, D. Y. Lv, L. Zhang, Controllable processing time scheduling with total weighted completion time objective and deteriorating jobs, *Asia-Pacific J. Oper. Res.,* **41** (2024), 2350026. http://doi.org/10.1142/S0217595923500264

33. L. H. Zhang, M. H. Li, L. Lin, Study on controllable processing time and minmax group scheduling with common due-window assignment, *Symmetry,* **18** (2026), 358. https://doi.org/10.3390/sym18020358

34. X. F. Wu, L. Lin, J. B. Wang, Total weighted completion time scheduling cost under resource-allocation and time-dependent learning effects, *J. Ind. Manag. Optim.,* **22** (2026), 3219–3251. https://doi.org/10.3934/jimo.2026118

35. C. C. Wu, T. H. Yang, X. G. Zhang, C. C. Kang, I. H. Chung, W. C. Lin, Using heuristic and iterative greedy algorithms for the total weighted completion time order scheduling with release times, *Swarm Evol. Comput.,* **44** (2019), 913–926. https://doi.org/10.1016/j.swevo.2018.10.003

36. D. Bai, Y. Li, H. Xue, J. Yang, R. Chen, C. C. Wu, et al., A bi-agent single-processor scheduling to minimize the sum of maximum lateness with release dates, *J. Oper. Res. Soc.,* **77** (2026), 1049–1067. https://doi.org/10.1080/01605682.2025.2516106

37. R. L. Graham, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, *Ann. Discret. Math.,* **5** (1979), 287–326. https://doi.org/10.1016/s0167-5060(08)70356-x

38. A. Bachman, A. Janiak, M. Y. Kovalyov, Minimizing the total weighted completion time of deteriorating jobs, *Inform. Process. Lett.* **81** (2002), 81–84. https://doi.org/10.1016/S0020-0190(01)00196-X

39. J. Pei, X. M. Wang, W. J. Fan, P. M. Pardalos, X. B. Liu, Scheduling step-deteriorating jobs on bounded parallel-batching machines to maximise the total net revenue, *J. Oper. Res. Soc.,* **70** (2019), 1830–1847. https://doi.org/10.1080/01605682.2018.1464428

40. T. C. E. Cheng, S. A. Kravchenko, B. M. T. Lin, Scheduling step-deteriorating jobs to minimize the total completion time. *Comput. Ind. Eng.,* **144** (2020), 106329. https://doi.org/10.1016/j.cie.2020.106329

41. H. J. Kim, E. S. Kim, J. H. Lee, Scheduling of step-improving jobs with an identical improving rate, *J. Oper. Res. Soc.,* **73** (2022), 1127–1136. https://doi.org/10.1080/01605682.2021.1886616

42. C. C. Wu, W. C. Lin, A. Azzouz, J. Y. Xu, Y. L. Chiu, Y. W. Tsai, P. Y. Shen, A bicriterion single-machine scheduling problem with step-improving processing times, *Comput. Ind. Eng.,* **171** (2022), 108469. https://doi.org/10.1016/j.cie.2022.108469

43. C. X. Miao, J. X. Song, Y. Z. Zhang, Single-machine time-dependent scheduling with proportional and delivery times, *Asia-Pacific J. Oper. Res.,* **40** (2023), 2240015. http://doi.org/10.1142/S0217595922400152

44. R. R. Mao, D. Y. Lv, N. Ren, J. B. Wang, Supply chain scheduling with deteriorating jobs and delivery times, *J. Appl. Math. Comput.,* **70** (2026), 2285–2312. https://doi.org/10.1007/s12190-024-02052-0

45. Y. Y. Lu, S. Zhang, J. Y. Tao, Earliness-tardiness scheduling with delivery times and deteriorating jobs, *Asia-Pacific J. Oper. Res.,* **42** (2025), 2450009. https://doi.org/10.1142/S021759592450009X

46. M. H. Li, D. Y. Lv, Z. G. Lv, L. H. Zhang, J. B. Wang, A two-agent resource allocation scheduling problem with slack due-date assignment and general deterioration function, *Comput. Appl. Math.,* **43** (2024), 229. https://doi.org/10.1007/s40314-024-02753-z

47. X. Y. Wang, W. G. Liu, Delivery scheduling with variable processing times and due date assignments, *Bull. Malays. Math. Sci. Soc.,* **48** (2025), 76. https://doi.org/10.1007/s40840-025-01856-y

48. Y. Sun, Y. Y. Lu, D. Y. Lv, J. B. Wang, Single-machine setup time scheduling with general linear deterioration subject to makespan and total completion time minimization, *J. Oper. Res. Soc.,* (2025). https://doi.org/10.1080/01605682.2025.2560511

49. Z. W. Sun, D. Y. Lv, P. Ji, J. B. Wang, Minimizing total completion time scheduling problem with ready times and linear deterioration functions of processing times, *J. Appl. Math. Comput.,* **72** (2026), 42. https://doi.org/10.1007/s12190-025-02694-8

50. G. H. Hardy, J. E. Littlewood, G. Polya, *Inequalities*, Cambridge University Press, 1988. https://doi.org/10.1017/s0025557200143451

51. X. Y. Wang, W. G. Liu, Single machine group scheduling jobs with resource allocations subject to unrestricted due date assignments, *J. Appl. Math. Comput.,* **70** (2024), 6283–6308. https://doi.org/10.1007/s12190-024-02216-y

52. J. B. Wang, Z. W. Sun, M. Gao, Research on single-machine scheduling with due-window assignment and resource allocation under total resource consumption cost is bounded, *J. Appl. Math. Comput.,* **71** (2025), 7905–7927. https://doi.org/10.1007/s12190-025-02599-6

53. M. Nawaz, J. E. E. Enscore, I. Ham, A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, *Omega,* **11** (1983), 91–95. https://doi.org/10.1016/0305-0483(83)90088-9

54. K. Chen, X. Y. Shi, D. L. Yao, T. C. E. Cheng, M. Ji, Parallel-machine scheduling with machine unavailability to maximize total early work, *J. Oper. Res. Soc.,* (2025). https://doi.org/10.1080/01605682.2025.2565464

55. M. H. Li, D. Y. Lv, L. H. Zhang, J. B. Wang, Permutation flow shop scheduling with makespan objective and truncated learning effects, *J. Appl. Math. Comput.,* **70** (2024), 2907–2939. https://doi.org/10.1007/s12190-024-02080-w

56. J. B. Wang, D. Y. Lv, C. Wan, Proportionate flow shop scheduling with job-dependent due windows and position-dependent weights, *Asia-Pacific J. Oper. Res.,* **42** (2025), 2450011. https://doi.org/10.1142/S0217595924500118

57. Y. Sun, D. Y. Lv, X. Huang, Properties for due window assignment scheduling on a two-machine no-wait proportionate flow shop with learning effects and resource allocation, *J. Oper. Res. Soc.,* **77** (2026), 599–615. https://doi.org/10.1080/01605682.2025.2492817

58. A. Azerine, M. Boudhar, D. Rebaine, Minimising the makespan in a two-machine open shop scheduling with competing agents, *J. Oper. Res. Soc.,* (2026). https://doi.org/10.1080/01605682.2026.2651218

59. L. F. Lin, J. Y. Wang, Strategic delay in job shop scheduling: non-zero start time optimisation under multi-constraint environments, *J. Oper. Res. Soc.,* (2026). https://doi.org/10.1080/01605682.2026.2682267

60. L. R. Abreu, M. S. Nagano, Mixed-integer and constraint programming formulations for distributed open shop scheduling problem with makespan minimization, *J. Oper. Res. Soc.,* (2026). https://doi.org/10.1080/01605682.2026.2682260

61. W. Liu, X. Wang, L. Li, W. Dai, Due-window assignment scheduling with job-rejection, truncated learning effects and setup times, *J. Ind. Manag. Optim.,* **20** (2024), 313–324. https://doi.org/10.3934/jimo.2023079

62. X. N. Geng, X. Sun, J. Wang, L. Pan, Scheduling on proportionate flow shop with job rejection and common due date assignment, *Comput. Ind. Eng.,* **181** (2023), 109317. https://doi.org/10.1016/j.cie.2023.109317

63. X. N. Geng, X. Sun, J. Wang, B. Mor, Scheduling on proportionate flowshop with total late work and job rejection, *Oper. Res.,* **25** (2025), 82. https://doi.org/10.1007/s12351-025-00951-z

64. Y. Zhang, X. Sun, T. Liu, J. Y. Wang, X. N. Geng, Single-machine scheduling simultaneous consideration of resource allocations and exponential time-dependent learning effects, *J. Oper. Res. Soc.,* **76** (2025), 528–540. https://doi.org/10.1080/01605682.2024.2371527

65. D. Y. Lv, J. B. Wang, Single-machine group technology scheduling with resource allocation and slack due window assignment including minmax criterion, *J. Oper. Res. Soc.,* **76** (2025), 1696–1712. https://doi.org/10.1080/01605682.2024.2430351

66. X. Huang, H. He, H. Bei, Y. Zhao, N. Wang, Y. Chang, Group-scheduling with simultaneous learning effects and convex resource allocations, *Oper. Res. Perspect.,* **15** (2025), 100370. https://doi.org/10.1016/j.orp.2025.100370

67. L. H. Zhang, J. B. Wang, Resource allocation and minmax scheduling under group technology and different due-window assignments, *Axioms,* **14** (2025), 827. https://doi.org/10.3390/axioms14110827

68. L. H. Zhang, N. Yin, Study on single-machine group scheduling with convex resource allocations and different due-date assignments, *Bull. Malays. Math. Sci. Soc.,* **49** (2026), 33. https://doi.org/10.1007/s40840-025-02031-z

AIMS Press