Journal of Industrial and Management Optimization

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

JIMO, 22(7): 3219–3251.
DOI:10.3934/jimo.2026118
Received: 25 March 2026
Revised: 02 June 2026
Accepted: 11 June 2026
Published: 18 June 2026

Research article

Total weighted completion time scheduling under resource allocation and time-dependent learning effects

Xinfeng Wu, Lin Lin* and Ji-Bo Wang*

School of Computer, Shenyang Aerospace University, Shenyang, 110136, P.R. China

* **Correspondence:** Email: linlin@sau.edu.cn; wangjibo75@163.com.

Abstract: This paper investigates a single-machine scheduling problem that integrates cumulative time-dependent learning effects with non-renewable resource allocation, where job processing times are controllable through resource investment. The actual processing time of each job is modeled as a nonlinear function of the allocated resource amount and the cumulative normal processing times of previously processed jobs, thereby reflecting proficiency gains in modern manufacturing environments. The objective is to minimize a weighted aggregate cost consisting of total weighted completion time and total resource consumption. By establishing convexity with respect to the resource variables, we derive an analytical optimal resource allocation policy for any fixed job sequence. This result transforms the original joint continuous-discrete optimization problem into an equivalent combinatorial sequencing problem. To solve the resulting problem, we develop an exact branch-and-bound (B&B) algorithm with a lower-bound scheme based on the monotonicity of the learning term. For larger instances, simulated annealing, tabu search, and genetic algorithms are developed to obtain high-quality feasible solutions within limited computational time. The computational results evaluate the pruning efficiency of the B&B algorithm and the stability of the proposed metaheuristics. The analysis also shows that, under strong learning effects, the classical WSPT rule is no longer universally dominant; in some cases, SPT-type sequences may be preferable. These findings provide useful insights for scheduling decisions in labor-intensive and high-precision manufacturing environments.

Keywords: single-machine scheduling; learning effect; non-renewable resource allocation; total weighted completion time; combinatorial optimization
Mathematics Subject Classification: 90B35

1. Introduction

Modern smart manufacturing is increasingly characterized by dynamic and resource-controllable production environments. In high-precision and labor-intensive sectors, the classical assumption of

fixed job processing times is often restrictive. In practice, processing efficiency may improve through learning, while job durations can also be compressed by allocating non-renewable resources such as energy, specialized tools, or manpower. Balancing proficiency gains with physical resource investment within a unified framework is therefore an important issue for intelligent production systems.

Although learning effects and resource allocation have been studied separately, their integration under a weighted completion-time objective leads to substantial modeling and algorithmic challenges. In many practical settings, jobs have heterogeneous weights, which makes total weighted completion time a relevant performance measure. The combination of a nonlinear resource allocation function and cumulative time-dependent learning creates a strong coupling between continuous resource decisions and discrete sequencing decisions. In particular, a job's actual processing time depends not only on its resource input but also on the cumulative normal processing times of all preceding jobs. This sequence dependence weakens the applicability of classical priority rules because the marginal effect of resource investment is jointly determined by the job position and the accumulated system state.

To address this integrated problem, this paper studies a single-machine scheduling model that incorporates both cumulative time-dependent learning and non-renewable resource allocation. Using the standard three-field notation, the problem is denoted as $1 \mid p_j^A(u_j) = \left(\frac{p'_j(1+\sum_{l=1}^{h-1} p'_{[l]})^m}{u_j}\right)^{\eta} \mid \alpha \sum w_{[j]}C_{[j]} + \beta \sum u_{[j]}$. Detailed notation is provided in Section 3.1. All jobs are available at time zero, and preemption is not allowed. The objective is to minimize a weighted aggregate of total weighted completion time and total resource consumption cost by jointly optimizing the discrete job sequence and the continuous resource-allocation vector.

The primary contributions of this work are as follows:

- **Unified Framework with Heterogeneous Weights:** We establish an integrated scheduling framework that captures the sequence-dependent marginal efficiency of resource investment under a weighted completion-time objective. The model addresses the interaction between cumulative learning effects and physical resource allocation in job environments with heterogeneous weights.
- **Analytical Decoupling and Theoretical Reduction:** The core theoretical result (Theorem 1) establishes convexity and derives a closed-form optimal resource allocation policy for any fixed sequence. This analytical result separates the continuous resource variables from the discrete sequencing decisions and reduces the original joint optimization problem to an equivalent combinatorial sequencing model.
- **Structure-Driven Exact Methodology:** We develop an exact branch-and-bound (B&B) procedure with a lower-bounding scheme derived from the monotonic properties of the cumulative learning term. This procedure provides an exact benchmark for small- and medium-scale instances and supports the evaluation of heuristic solution quality.

The remainder of this paper is organized as follows. Section 2 reviews the relevant literature. Section 3 formally defines the problem. Section 4 derives the structural properties and the optimal resource allocation policy. Section 5 presents the B&B algorithm and metaheuristics. Section 6 evaluates the performance through computational experiments, and Section 7 concludes the paper.

2. Literature review

This paper lies at the intersection of scheduling with controllable processing times and scheduling with learning effects. To position our contributions within the broader scheduling literature (Pinedo [1], Karger et al. [2]), we review three related research streams: resource allocation, time-dependent processing times, and integrated models.

2.1. Resource allocation and controllable processing times

Classical scheduling models often assume static processing times. Vickson [3] was among the first to relax this assumption by allowing processing times to be compressed through continuous resource allocation. Subsequent research systematically classified resource functions into linear, convex, and concave models (Shabtay and Steiner [4]), and expanded into non-renewable resource optimization (Yin and Wang [5], Wang et al. [6], Li et al. [7], Abdeljaouad et al. [8]). Recent extensions have addressed resource allocations, Wang et al. [9] considered due-window assignment scheduling with resource allocation. The general earliness-tardiness cost minimization subject to total resource consumption cost is bounded, they proposed some algorithms. Zhang and Wang [10], Lv and Wang [11], Zhang and Yin [12], and Zhang et al. [13] studied resource allocation scheduling problems with group technology and due-window assignments. Huang et al. [14] considered resource allocation scheduling with learning effects and group technology. The total resource consumption minimization subject to makespan is bounded, they proposed heuristic and branch-and-bound algorithms. Kovalevs et al. [15] studied scheduling problems with resource allocation. They proved that some single-machine scheduling problems are equivalent to two-machine flow shop problems. Sun et al. [16] considered no-wait flow shop scheduling with learning effects and resource allocation. They proved that some due window assignment problems are polynomially solvable. Shabtay [17] studied resource allocation proportionate flow shop scheduling problems. Although these studies demonstrate the flexibility of time-cost trade-offs, they typically assume static machine efficiency and do not explicitly model dynamic proficiency improvements of operators.

2.2. Scheduling with time-dependent learning and deterioration effects

The learning effect captures reductions in processing time due to accumulated experience (Biskup [18, 19], Wu et al. [20]). Conversely, many practical settings involve processing time expansions, modeled as deteriorating jobs (Mosheiov [21], Oron [22]), often subject to release times (Lee et al. [23], Sun et al. [24], Sun et al. [25]) or delivery times (Miao et al. [26], Lu et al. [27], Kong et al. [28]). These dynamic phenomena fall under the broader umbrella of time-dependent scheduling, an area that has been extensively formalized over the past decades (Gawiejnowicz [29–31], Yin et al. [32]). Early learning models primarily focused on "position-dependent" learning (Cheng and Wang [33], Wang et al. [34]), where durations depend solely on a job's sequence index. For labor-intensive environments, Kuo and Yang [35] introduced the more realistic "time-dependent" model, linking processing time to the cumulative normal processing times of all preceding jobs. Recent works have further enriched these models, Lv et al. [36] considered single-machine scheduling with deteriorating effects. For the total weighted completion time minimization, they proposed some solution algorithms. Li et al. [37] examined flow shop scheduling with truncated learning effects. For the makespan minimization, they

proposed some heuristics and a branch-and-bound algorithm. Lv and Wang [38] considered two-machine flow shop scheduling with release dates and truncated learning effects. For the total completion time minimization, they proposed a branch-and-bound algorithm and some heuristics. Song et al. [39] conducted single-machine multiple due-windows assignment scheduling with learning and deteriorating effects. Liu et al. [40] studied single-machine scheduling with deterioration effects and maintenance activity. They proved that the makespan minimization is NP-hard.

2.3. Integrated models and research gaps

Recent studies have integrated resource allocation with learning or deterioration effects to overcome the limitations of isolated modeling perspectives (Wang and Wang [41], Zhang et al. [42], Lu et al. [43], Ren et al. [44], Yin and Gao [45], Yin et al. [46]). However, these integrated models predominantly focus on unweighted criteria (e.g., makespan or unweighted total completion time).

Introducing heterogeneous job weights changes the structure of the problem. The combination of a weighted objective, nonlinear convex resource functions, and cumulative time-dependent learning creates a challenging continuous-discrete optimization problem (Hartmanis [47]). The nonlinear nature of time-dependent learning can substantially alter the problem structure and may invalidate classical priority rules. In this weighted setting, it is nontrivial to separate continuous resource variables from discrete sequencing decisions. To address this gap, we establish a convexity result (Theorem 1) and derive a closed-form optimal resource policy for any fixed sequence. This analytical reduction transforms the original joint optimization problem into an equivalent sequencing framework suitable for exact and heuristic algorithm design.

3. Problem description

3.1. Notations

For clarity, the mathematical notation and indices used throughout this paper are summarized in Table 1.

Remark. In this model, resource allocation is restricted to $u_j > 0$. Mathematically, as shown in Eq. (3.1), u_j appears in the denominator; if $u_j \to 0$, the actual processing time $p_j^A(u_j)$ tends to infinity, yielding an infinite objective value. From a practical perspective, $u_j = 0$ represents the absence of resource investment and would halt the production process. It is therefore excluded from the feasible region.

For these reasons, the decision variable is explicitly restricted to the domain $u_j > 0$, which is consistent with both mathematical validity and practical interpretation.

Table 1. Mathematical notations and indices.

Notation	Description
Indices	
j	General index representing a specific job or a position in the sequence ($j = 1, 2, \ldots, n$).
k, l	Dummy summation indices denoting sequence positions in cumulative functions.
$[h]$	The index of some job scheduled at the hth position in sequence π.
Parameters and Variables	
n	Total number of jobs to be scheduled, $n \in \mathbb{Z}^+$.
$\mathcal{J}$	Set of jobs, $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$.
p'_j	Normal (basic) processing time of job J_j ($p'_j > 0$).
w_j	Weight of job J_j, representing its relative importance ($w_j > 0$).
m	Learning index, capturing the effect of cumulative experience ($m \leq 0$).
η	Convexity parameter of the resource-time relationship ($\eta > 0$).
α, β	Weighting factors for completion-time cost and resource cost ($\alpha, \beta > 0$).
u_j	Amount of non-renewable resource allocated to job J_j ($u_j > 0$).
π	A job sequence (permutation), $\pi = [J_{[1]}, J_{[2]}, \ldots, J_{[n]}]$.
$p_j^A(u_j)$	Actual processing time of job J_j given resource allocation u_j.
$C_{[h]}$	Completion time of the job scheduled at the hth position.
$f(\pi, u)$	Joint objective function considering sequence and resource allocation.

3.2. Problem statement

Consider a set of n jobs $\mathcal{J}$ to be processed on a single machine. All jobs are available at time zero, preemption is forbidden, and the machine can process at most one job at a time. The actual processing time of a job is influenced by both a time-dependent learning effect and the allocation of a non-renewable resource.

Following the model in Wang and Wang [41], if job J_j is processed at the hth position of a sequence π, its actual processing time $p_j^A(u_j)$ is

$$p_j^A(u_j) = \left(\frac{p'_j \left(1 + \sum_{l=1}^{h-1} p'_{[l]}\right)^m}{u_j} \right)^\eta, \tag{3.1}$$

where $m \leq 0$ is the learning index and $\eta > 0$ is the convexity parameter. The term $\sum_{l=1}^{h-1} p'_{[l]}$ represents the cumulative normal processing times of all jobs preceding position h.

Given a sequence $\pi = [J_{[1]}, J_{[2]}, \ldots, J_{[n]}]$, the completion time of the job at position h is

$$C_{[h]} = \sum_{k=1}^{h} p_{[k]}^A(u_{[k]}), \qquad h = 1, \ldots, n. \tag{3.2}$$

The objective is to determine the optimal sequence π and resource-allocation vector $u = (u_{[1]}, \ldots, u_{[n]})$

that minimize a weighted aggregate of total weighted completion time and total resource consumption cost.

The joint optimization problem is formally formulated as follows:

$$\text{Minimize} \quad f(\pi, u) = \alpha \sum_{j=1}^{n} w_{[j]} C_{[j]} + \beta \sum_{j=1}^{n} u_{[j]} \tag{3.3}$$

$$\text{Subject to:} \quad p^A_{[h]}(u_{[h]}) = \left(\frac{p'_{[h]} \left(1 + \sum_{l=1}^{h-1} p'_{[l]}\right)^m}{u_{[h]}} \right)^\eta, \quad h = 1, \ldots, n, \tag{3.4}$$

$$C_{[h]} = \sum_{k=1}^{h} p^A_{[k]}(u_{[k]}), \quad h = 1, \ldots, n, \tag{3.5}$$

$$\pi \in \Pi, \tag{3.6}$$

$$u_{[h]} > 0, \quad h = 1, \ldots, n. \tag{3.7}$$

In this formulation:

- Eq. (3.3) is the joint objective function to be minimized. The decision variables are the discrete job sequence π and the continuous resource-allocation vector u.
- Eq. (3.4) defines the sequence-dependent actual processing time.
- Eq. (3.5) specifies the single-machine completion-time relation, where preemption is not allowed and jobs are processed continuously.
- Eq. (3.6) restricts the sequence π to be a valid permutation within the set Π, where Π represents all $n!$ possible permutations of the job set $\mathcal{J}$.
- Eq. (3.7) is the domain constraint for the resource variables, guaranteeing mathematical validity and preventing the actual processing time from reaching infinity.

Using the standard three-field notation (Graham et al. [48]), this problem is denoted by: $1 \mid p^A_j(u_j) = \left(\frac{p'_j(1+\sum p'_{[l]})^m}{u_j}\right)^\eta \mid \alpha \sum w_{[j]} C_{[j]} + \beta \sum u_{[j]}$. Wang and Wang [41] considered the problem $1 \mid p^A_j(u_j) = \left(\frac{p'_j(1+\sum p'_{[l]})^m}{u_j}\right)^\eta \mid \alpha \sum \xi_j p^A_{[j]}(u_{[j]}) + \beta \sum g_{[j]} u_{[j]}$, where ξ_j denotes the position-dependent weight of jth position, i.e., the weight ξ_j does not correspond with the job J_j but with the jth position (see Wang et al. [49] and Wang [50]), and $g_{[j]}$ is the cost when allocating unit resource to job $J_{[j]}$. Wang et al. [6] studied the following model: $p^A_j(u_j) = \left(\frac{p'_j}{u_j}\right)^\eta + b_j s_j$, where $b_j \geq 0$ (resp. $s_j \geq 0$) denotes the deterioration rate (resp. starting time) of job J_j. For the problem $1 \mid p^A_j(u_j) = \left(\frac{p'_j}{u_j}\right)^\eta + b_j s_j \mid \alpha \sum w_{[j]} C_{[j]} + \beta \sum g_{[j]} u_{[j]}$, they proposed some solution algorithms.

3.3. Practical Applicability and Model Assumptions

Clarify the industrial and economic rationale underlying the main modeling assumptions:

- **Zero Release Time and No Preemption:** These assumptions are standard in batch-processing environments and are suitable for high-setup-cost manufacturing (e.g., CNC machining,

semiconductor baking) where interrupting a task may cause workpiece damage or require costly recalibration.

- **Cumulative Time-Dependent Learning:** Unlike position-dependent models that represent experience only through the number of completed jobs, our model links learning to cumulative normal processing time. In customized manufacturing (e.g., aerospace assembly), operators gain substantially more experience from a complex 10-hour task than a 1-hour task. Aggregating standard processing times therefore provides a practical proxy for proficiency accumulated through sustained workload exposure.
- **Diminishing Marginal Returns in Resource Allocation:** The resource-time compression function reflects the economic principle of diminishing returns, where η controls the elasticity of processing time with respect to resource investment. Physically, while additional non-renewable resources (e.g., energy, premium tooling) compress durations, inherent machine limitations imply that successive investments may yield progressively smaller time reductions.
- **Soft Constraints via Cost Penalties:** Rather than imposing hard capacity limits, we penalize resource consumption directly in the objective function. This is consistent with flexible manufacturing settings, where resources (e.g., energy, overtime) are constrained by economic budgets rather than strict physical ceilings, and supports a cost-benefit optimization perspective.

4. Structural results

In this section, we analyze the optimal resource-allocation structure for a fixed sequence.

4.1. Optimal resource allocation and equivalent model

Theorem 1. *For a given sequence π, the optimal resource allocation for the problem* $1\left|p_j^A(u_j)=\left(\frac{p'_j(1+\sum p'_{[l]})^m}{u_j}\right)^{\eta}\right|\alpha\sum w_{[j]}C_{[j]}+\beta\sum u_{[j]}$ *is*

$$u^*_{[h]}=\left(\frac{\alpha\eta}{\beta}\right)^{\frac{1}{\eta+1}}\left[p'_{[h]}\left(1+\sum_{l=1}^{h-1}p'_{[l]}\right)^m\left(\sum_{j=h}^{n}w_{[j]}\right)^{1/\eta}\right]^{\frac{\eta}{\eta+1}},\quad h=1,\ldots,n. \tag{4.1}$$

Proof. For a fixed job sequence $\pi=[J_{[1]},J_{[2]},\ldots,J_{[n]}]$, substituting the processing time (3.1) and completion time (3.2) into the objective function (3.3) gives

$$f(\pi,u)=\alpha\sum_{j=1}^{n}w_{[j]}\left(\sum_{h=1}^{j}p^A_{[h]}(u_{[h]})\right)+\beta\sum_{h=1}^{n}u_{[h]}. \tag{4.2}$$

Step 1: Summation Rearrangement. By changing the order of the double summation, where $1\le h\le j\le n$, the terms associated with each position h can be isolated:

$$\sum_{j=1}^{n}w_{[j]}\sum_{h=1}^{j}p^A_{[h]}(u_{[h]})=\sum_{h=1}^{n}p^A_{[h]}(u_{[h]})\left(\sum_{j=h}^{n}w_{[j]}\right). \tag{4.3}$$

Let $W_{[h]} = \sum_{j=h}^{n} w_{[j]}$ be the cumulative weight. Defining the position-dependent constant $G_{[h]} = p'_{[h]}(1 + \sum_{l=1}^{h-1} p'_{[l]})^m$, the objective function is rewritten as a separable function:

$$f(\pi, u) = \sum_{h=1}^{n} \left[\alpha W_{[h]} G_{[h]}^{\eta} u_{[h]}^{-\eta} + \beta u_{[h]} \right]. \tag{4.4}$$

Step 2: Convexity Proof. To establish the uniqueness of the optimal resource allocation, we examine the convexity of $f(\pi, u)$ with respect to $u_{[h]}$. Let $\phi(u_{[h]}) = \alpha W_{[h]} G_{[h]}^{\eta} u_{[h]}^{-\eta} + \beta u_{[h]}$. Its first and second derivatives are

$$\frac{d\phi}{du_{[h]}} = -\eta \alpha W_{[h]} G_{[h]}^{\eta} u_{[h]}^{-(\eta+1)} + \beta, \tag{4.5}$$

$$\frac{d^2\phi}{du_{[h]}^2} = \eta(\eta + 1) \alpha W_{[h]} G_{[h]}^{\eta} u_{[h]}^{-(\eta+2)}. \tag{4.6}$$

Since $\alpha, \eta, W_{[h]}, G_{[h]}, u_{[h]} > 0$, we have $\frac{d^2\phi}{du_{[h]}^2} > 0$. Thus, $f(\pi, u)$ is strictly convex in u, and the optimal allocation is obtained by setting the first derivative to zero.

Step 3: Derivation of $u^*_{[h]}$. Setting $\frac{d\phi}{du_{[h]}} = 0$ yields $\beta = \eta \alpha W_{[h]} G_{[h]}^{\eta} u_{[h]}^{-(\eta+1)}$. Rearranging the terms to isolate $u_{[h]}$, we obtain $u_{[h]}^{\eta+1} = \frac{\eta \alpha W_{[h]} G_{[h]}^{\eta}}{\beta}$. Taking the $(\eta + 1)$-th root on both sides provides the intermediate form:

$$u^*_{[h]} = \left(\frac{\alpha \eta}{\beta} \right)^{\frac{1}{\eta+1}} G_{[h]}^{\frac{\eta}{\eta+1}} W_{[h]}^{\frac{1}{\eta+1}}. \tag{4.7}$$

To obtain a unified expression, we factor out the common exponent $\frac{\eta}{\eta+1}$ from $G_{[h]}$ and $W_{[h]}$. Since $W_{[h]}^{\frac{1}{\eta+1}} = \left(W_{[h]}^{1/\eta} \right)^{\frac{\eta}{\eta+1}}$, the optimal resource allocation policy can be written as

$$u^*_{[h]} = \left(\frac{\alpha \eta}{\beta} \right)^{\frac{1}{\eta+1}} \left[G_{[h]} W_{[h]}^{1/\eta} \right]^{\frac{\eta}{\eta+1}}. \tag{4.8}$$

□

4.2. Equivalent sequencing model

Substituting $u^*_{[h]}$ back into (4.4), each term in the summation becomes

$$\begin{aligned} f_h(u^*_{[h]}) &= \alpha W_{[h]} G_{[h]}^{\eta} \left(\frac{\alpha \eta W_{[h]} G_{[h]}^{\eta}}{\beta} \right)^{-\frac{\eta}{\eta+1}} + \beta \left(\frac{\alpha \eta W_{[h]} G_{[h]}^{\eta}}{\beta} \right)^{\frac{1}{\eta+1}} \\ &= \alpha^{\frac{1}{\eta+1}} \beta^{\frac{\eta}{\eta+1}} (\eta^{-\frac{\eta}{\eta+1}} + \eta^{\frac{1}{\eta+1}}) \left(G_{[h]}^{\eta} W_{[h]} \right)^{\frac{1}{\eta+1}}. \end{aligned} \tag{4.9}$$

Since the coefficient is a constant independent of the sequence, the original problem reduces to

$$\min_{\pi} g(\pi) = \sum_{h=1}^{n} \left[p'_{[h]} \left(1 + \sum_{l=1}^{h-1} p'_{[l]} \right)^m \left(\sum_{j=h}^{n} w_{[j]} \right)^{1/\eta} \right]^{\frac{\eta}{\eta+1}}. \tag{4.10}$$

Remark. Equation (4.10) reduces the joint optimization problem to a combinatorial sequencing problem through the closed-form resource allocation policy. As shown in (4.1), the optimal resource $u^*_{[h]}$ increases monotonically with both the normal processing time $p'_{[h]}$ and the cumulative weight $W_{[h]}$. In particular, the nonlinear cumulative learning term $\left(1 + \sum_{l=1}^{h-1} p'_{[l]}\right)^m$ breaks the standard of the objective function $g(\pi)$. This sequence dependence may invalidate classical static priority rules such as WSPT, motivating the exact branch-and-bound procedure and metaheuristics developed in the following sections.

4.3. Structural properties and dominance relationships

The reduction to the combinatorial sequencing problem $g(\pi)$ highlights the nonlinear role of the cumulative learning effect $(1 + \sum p'_{[l]})^m$. This sequence dependence may invalidate classical priority rules such as WSPT. To characterize the problem structure and analyze conditions for preferred local job orders, we employ the adjacent pairwise interchange (API) method to establish dominance relationships. Let $E = \frac{\eta}{\eta+1}$.

Lemma 1. *(Adjacent Dominance Condition) Consider two sequences π_1 and π_2 differing only in the order of adjacent jobs J_i and J_j at positions h and $h + 1$:*

- $\pi_1 = (J_{[1]}, \ldots, J_{[h-1]}, J_i, J_j, J_{[h+2]}, \ldots, J_{[n]})$
- $\pi_2 = (J_{[1]}, \ldots, J_{[h-1]}, J_j, J_i, J_{[h+2]}, \ldots, J_{[n]})$

Let $A(h) = 1 + \sum_{l=1}^{h-1} p'_{[l]}$ be the cumulative normal processing time before position h, and $W(h) = \sum_{k=h+2}^{n} w_{[k]}$ be the residual weight after position $h + 1$. π_1 strictly dominates π_2 (i.e., $g(\pi_1) \le g(\pi_2)$) if $[p'_i A(h)^m (w_i + w_j + W(h))]^E - [p'_i (A(h) + p'_j)^m (w_i + W(h))]^E \le [p'_j A(h)^m (w_i + w_j + W(h))]^E - [p'_j (A(h) + p'_i)^m (w_j + W(h))]^E$.

Proof. Because π_1 and π_2 share identical job sets before position h and after position $h + 1$, their cost contributions for $k < h$ and $k > h + 1$ are identical and therefore cancel out. Evaluating the localized costs for positions h and $h + 1$ yields

$$C(\pi_1) = [p'_i A(h)^m (w_i + w_j + W(h))]^E + [p'_j (A(h) + p'_i)^m (w_j + W(h))]^E.$$

$$C(\pi_2) = [p'_j A(h)^m (w_i + w_j + W(h))]^E + [p'_i (A(h) + p'_j)^m (w_i + W(h))]^E.$$

Imposing the condition $C(\pi_1) \le C(\pi_2)$ and grouping terms by p'_i and p'_j yields the inequality in Lemma 1. □

Theoretical Insight: Lemma 1 formalizes the sequence dependence of the transformed model. Unlike classical scheduling, where optimal priority relies on a static ratio p'_j/w_j, the dominance between J_i and J_j here depends dynamically on both past states (cumulative learning $A(h)^m$) and future states (residual weight $W(h)$). This bidirectional dependency explains why static greedy rules cannot guarantee optimality and illustrates the combinatorial complexity of the sequencing problem.

4.4. Numerical examples

To illustrate the behavior of classical sequencing rules under the proposed model, we compare the shortest processing time (SPT) rule and the weighted shortest processing time (WSPT) rule using two examples.

Example 1 Consider a case with $n = 4$ jobs. The normal processing-time vector is $P' = (3, 6, 4, 8)$, the weight vector is $w = (1, 2, 3, 1)$, the learning index is $m = -0.5$, and the convexity parameter is $\eta = 3$.

(1) Sequencing rules

- SPT rule: Jobs are sorted as $\pi_{\mathrm{SPT}} = (J_1, J_3, J_2, J_4)$, where $P' = (3, 4, 6, 8)$.
- WSPT rule: Ratios p'_j/w_j are $(3, 3, 1.33, 8)$. Sorting gives $\pi_{\mathrm{WSPT}} = (J_3, J_1, J_2, J_4)$, where $P' = (4, 3, 6, 8)$.

(2) Detailed calculation steps The objective is calculated by $g(\pi) = \sum_{h=1}^{n} [G_{[h]} W_{[h]}^{1/\eta}]^{\frac{\eta}{\eta+1}}$. For $\eta = 3$, the exponent is 0.75.

Table 2. Results of SPT and WSPT.

Rule	h	$p'_{[h]}$	$1 + \sum p'_{[l]}$	$W_{[h]}$	$[p'_{[h]}(1 + \sum p'_{[l]})^m W_{[h]}]^{0.75}$	$g(\pi)$
SPT	1	3	1	7	$(3 \times 1^{-0.5} \times 7)^{0.75} = 9.8143$	
	2	4	4	6	$(4 \times 4^{-0.5} \times 6)^{0.75} = 6.4474$	**22.0322**
	3	6	8	3	$(6 \times 8^{-0.5} \times 3)^{0.75} = 3.9023$	
	4	8	14	1	$(8 \times 14^{-0.5} \times 1)^{0.75} = 1.8682$	
WSPT	1	4	1	7	$(4 \times 1^{-0.5} \times 7)^{0.75} = 12.1278$	
	2	3	5	3	$(3 \times 5^{-0.5} \times 3)^{0.75} = 3.0188$	**21.4730**
	3	6	8	2	$(6 \times 8^{-0.5} \times 2)^{0.75} = 2.8488$	
	4	8	14	1	$(8 \times 14^{-0.5} \times 1)^{0.75} = 3.4776$	

In this instance, $g(\pi_{\mathrm{WSPT}}) < g(\pi_{\mathrm{SPT}})$, so WSPT performs better (see Table 2).

Example 2 Consider $n = 3$ jobs with $P' = (2, 3, 5)$, $w = (1, 3, 2)$, $m = -2$, and $\eta = 1$.

(1) Sequencing rules

- SPT rule: $\pi_{\mathrm{SPT}} = (J_1, J_2, J_3)$, where $P' = (2, 3, 5)$ and $w = (1, 3, 2)$.
- WSPT rule: Ratios p'_j/w_j are $(2, 1, 2.5)$. Sorting gives $\pi_{\mathrm{WSPT}} = (J_2, J_1, J_3)$, where $P' = (3, 2, 5)$ and $w = (3, 1, 2)$.

(2) Detailed calculation steps For $\eta = 1$, the exponent is 0.5.

Table 3. Results of SPT and WSPT.

Rule	h	$p'_{[h]}$	$1 + \sum p'_{[l]}$	$W_{[h]}$	$[p'_{[h]}(1 + \sum p'_{[l]})^m W_{[h]}]^{0.5}$	$g(\pi)$
SPT	1	2	1	6	$(2 \times 1^{-2} \times 6)^{0.5} = 3.4641$	
	2	3	3	5	$(3 \times 3^{-2} \times 5)^{0.5} = 1.2910$	**5.2821**
	3	5	6	2	$(5 \times 6^{-2} \times 2)^{0.5} = 0.5270$	
WSPT	1	3	1	6	$(3 \times 1^{-2} \times 6)^{0.5} = 4.2426$	
	2	2	4	3	$(2 \times 4^{-2} \times 3)^{0.5} = 0.6124$	**5.3820**
	3	5	6	2	$(5 \times 6^{-2} \times 2)^{0.5} = 0.5270$	

Here, $g(\pi_{\mathrm{SPT}}) < g(\pi_{\mathrm{WSPT}})$, showing that the classical WSPT intuition for weighted completion-time problems does not always carry over to the present setting (see Table 3).

Discussion on Parameter Settings The parameters in the examples are selected to reflect representative industrial scenarios:

1. **Learning Index (m)**: Values such as -0.5 and -2 represent different levels of proficiency improvement. In labor-intensive assembly lines (e.g., electronics), as more workload is completed, time-dependent learning can reduce the actual processing times of subsequent tasks.
2. **Convexity Parameter (η)**: Setting $\eta = 3$ or $\eta = 1$ reflects the efficiency of resource investment. A larger η (e.g., 3) represents a setting in which additional resources, such as high-precision tools or energy, can reduce processing time, while the resulting marginal reductions may diminish as investment increases.
3. **Rule Performance**: Example 2 illustrates that under strong learning effects ($m = -2$), the benefit of cumulative experience gained from finishing shorter jobs (SPT) can outweigh the immediate priority of high-weight jobs (WSPT).

5. Solution algorithms

After the resource allocation variables are eliminated through the fixed-sequence optimization in Section 4, the joint optimization problem reduces to a nonlinear combinatorial sequencing problem [21, 23, 47]. Although the fixed-sequence resource allocation admits a closed-form solution, the remaining sequencing problem involves an $n!$ permutation space and is computationally challenging. This difficulty motivates the development of an exact branch-and-bound (B&B) algorithm and several heuristic algorithms (HA). In the revised implementation, the three metaheuristics are compared under the same wall-clock CPU-time budget, so that the comparison is not affected by different iteration definitions or different per-iteration computational costs.

5.1. Heuristic algorithms (HA)

The HA framework consists of a constructive baseline (HA-1) and three metaheuristics: simulated annealing (HA-2), tabu search (HA-3), and genetic algorithm (HA-4). HA-1 provides a fast initial upper bound, while HA-2–HA-4 are designed to further improve the incumbent sequence before B&B starts. To address the fairness issue in the original version, SA, TS, and GA are terminated by the same CPU-time limit rather than by different maximum iteration counts. This unified stopping rule assigns the same computational budget to each metaheuristic.

5.1.1. HA-1: Simple constructive heuristic

HA-1 generates a small pool of deterministic and random sequences and selects the best one as the initial solution. Specifically, we evaluate the following candidate sequences:

1. the SPT sequence obtained by sorting jobs in non-decreasing order of p'_j;
2. the LPT sequence obtained by sorting jobs in nonincreasing order of p'_j;
3. five random permutations generated independently.

The sequence with the smallest value of $g(\pi)$ among these candidates is denoted by π_H and is used as the initial upper bound. The computational burden of HA-1 is negligible because only a constant number of sequences are evaluated.

5.1.2. HA-2: Simulated annealing (SA)

HA-2 applies SA to improve the HA-1 solution through stochastic neighborhood search.

1. **Initialization:** Set the current and best-found solutions to π_H.
2. **Temperature calibration:** Before the main search, sample 200 random moves and collect the relative cost increases of uphill moves. The scale parameter λ is determined from the median positive relative increase so that the initial acceptance probability is approximately $p_0 = 0.7$.
3. **Neighborhood structure:** At each iteration, a neighboring sequence is generated by either a swap move or an insertion move. In the implementation, insertion is selected with probability 0.40 and swap with probability 0.60.
4. **Acceptance criterion:** Let $\Delta_r = (g(\pi') - g(\pi))/\max\{g(\pi), \epsilon\}$ be the relative objective change. If $\Delta_r \leq 0$, the neighbor is accepted. Otherwise, it is accepted with probability

$$\exp(-\psi_k \Delta_r),$$

   where $\psi_k = k/\lambda$ is the inverse temperature at iteration k.
5. **Termination:** The search stops when the common CPU-time budget is exhausted, and the best sequence found during the search is returned.

Because each move requires only one objective evaluation, SA provides fast neighborhood-based improvement within the given time budget.

5.1.3. HA-3: Tabu search (TS)

HA-3 performs an intensified local search and uses a tabu mechanism to avoid cycling.

1. **Initialization:** Construct an initial pool consisting of the SPT sequence, the LPT sequence, and 10 random permutations. The best candidate is selected as the initial current solution and global best.
2. **Neighborhood exploration:** At each iteration, evaluate the full swap neighborhood generated by exchanging any two positions in the current sequence.
3. **Tabu mechanism:** The tabu attribute is the swapped position pair. The tabu tenure is set to $\max(5, \lfloor 2\sqrt{n} \rfloor) + U(0, 2)$, where $U(0, 2)$ is a small random jitter used to reduce cycling.
4. **Aspiration criterion:** A tabu move is allowed if it produces a new global best solution.
5. **Termination:** The search stops when the common CPU-time budget is exhausted or when no admissible move is available.

Although one TS iteration is more expensive than one SA iteration because it evaluates the full swap neighborhood, the unified CPU-time limit provides a fairer comparison with SA and GA.

5.1.4. HA-4: Genetic algorithm (GA)

HA-4 uses population-based evolutionary search to maintain diversity and explore multiple promising regions.

1. **Encoding and fitness:** Each individual is represented by a job permutation π, and its fitness is evaluated by the objective value $g(\pi)$.

2. **Initial population:** The population contains the HA-1 best sequence, the SPT sequence, the LPT sequence, and randomly generated permutations until the population size reaches $P = 80$.
3. **Selection:** Tournament selection with tournament size $k = 3$ is used to select parents.
4. **Crossover:** Order crossover (OX) is applied with probability $p_c = 0.90$ to preserve relative ordering information while generating feasible permutations.
5. **Mutation:** Mutation is applied with probability $p_m = 0.35$. Three mutation operators are used: swap, insertion, and inversion, with probabilities 0.50, 0.35, and 0.15, respectively.
6. **Elitism and replacement:** The best two individuals are copied directly to the next generation, and the remaining population is filled by offspring.
7. **Termination:** GA evolves continuously until the same CPU-time budget used by SA and TS is exhausted. Thus, GA is no longer constrained by an implicit or unclear generation limit.

This revised setting allows GA to perform population evolution under the same computational budget and avoids bias caused by an unclear or overly restrictive generation limit.

5.2. Branch-and-Bound (B&B)

B&B systematically explores the permutation space and uses lower bounds to prune subtrees, thereby reducing the need for exhaustive enumeration of all $n!$ permutations. Each node corresponds to a partial sequence

$$\pi = [\pi_p, \pi_s],$$

where π_p contains the first k fixed jobs and π_s is the set of the remaining $n - k$ unscheduled jobs.

5.2.1. Branching strategy

We adopt a memory-efficient depth-first search (DFS) strategy. At a node, child nodes are generated by appending one unscheduled job from π_s to the end of the partial sequence π_p. To obtain high-quality incumbents early, candidate jobs are examined in LPT order; that is, jobs with larger normal processing times are branched first.

5.2.2. Lower-bound construction

The objective value of a partial sequence decomposes into the scheduled prefix and the unscheduled suffix:

$$g(\pi) = \sum_{h=1}^{k} \left[p'_{[h]} \left(1 + \sum_{l=1}^{h-1} p'_{[l]} \right)^m \sum_{j=h}^{n} w_{[j]} \right]^{\frac{\eta}{\eta+1}} + \sum_{h=k+1}^{n} \left[p'_{[h]} \left(1 + \sum_{l=1}^{k} p'_{[l]} + \sum_{l=k+1}^{h-1} p'_{[l]} \right)^m \sum_{j=h}^{n} w_{[j]} \right]^{\frac{\eta}{\eta+1}}.$$

The prefix contribution is computed exactly. For the unscheduled suffix, relaxed estimates are computed by replacing the unknown accumulated normal processing times with aggregate values. Because $(1 + x)^m$ is monotonically nonincreasing for $m \leq 0$ [19, 21], using larger surrogate accumulated processing times yields an underestimation of the learning-adjusted suffix cost. Let $p_{\max} = \max\{p'_j \mid J_j \in \pi_s\}$ and let p_{avg} be the average normal processing time of the unscheduled jobs. The first estimate,

denoted by LB_A, uses $(h-k-1)p_{\max}$ as the surrogate accumulated suffix processing time, while the second estimate, denoted by LB_C, uses $(h-k-1)p_{\text{avg}}$. In both estimates, the remaining jobs are sorted by non-decreasing normal processing time to obtain the relaxed suffix contribution.

The final lower bound used for pruning is

$$LB = LB_{\text{prefix}} + \min\{LB_A, LB_C\}.$$

Taking the smaller relaxed estimate preserves the validity of the lower bound and avoids unsafe pruning. A node is pruned if $LB \geq UB$, where UB is the incumbent objective value obtained from the best heuristic or complete B&B solution found so far.

5.2.3. B&B algorithm outline

1. Generate an initial sequence using HA-1, HA-2, HA-3, or HA-4, and set its objective value as the initial upper bound UB.
2. Initialize the DFS search from the root node with an empty prefix.
3. At each node, generate child nodes by appending remaining jobs according to the LPT branching order with symmetry breaking for identical processing times.
4. Compute the lower bound LB for each child node; prune the node if $LB \geq UB$.
5. Whenever a complete sequence with objective value smaller than UB is obtained, update the incumbent sequence and UB.
6. Continue until the search tree is exhausted or the 3600-second B&B time limit is reached.

The B&B procedure is used as an exact benchmark for small- and medium-scale instances and as a truncated exact search for larger instances. The metaheuristics provide initial upper bounds that can improve pruning efficiency.

6. Computational experiments and results

This section evaluates the proposed model and solution methods using both a small illustrative instance and randomized simulations. The small instance demonstrates the search and pruning behavior of B&B, while the randomized simulations compare heuristic performance and B&B across different problem sizes and learning indices.

6.1. Small instance: An illustrative B&B search

Consider $n = 4$ jobs with basic processing times

$$P' = (4, 1, 3, 2) \quad \text{(corresponding to } J_1, J_2, J_3, J_4),$$

weights

$$W = (2, 1, 3, 2),$$

learning index $m = -0.5$, and $\eta = 1$.

6.1.1. Initial upper bound UB (from HA)

For illustration, HA-1 is used to obtain an initial upper bound by comparing the SPT and LPT sequences:

- SPT: $\pi_{\mathrm{SPT}} = [J_2, J_4, J_3, J_1]$, yielding

$$g(\pi_{\mathrm{SPT}}) = 10.4523.$$

- LPT: $\pi_{\mathrm{LPT}} = [J_1, J_3, J_4, J_2]$, yielding

$$g(\pi_{\mathrm{LPT}}) = 10.5129.$$

Hence,

$$UB = 10.4523.$$

(HA-2, HA-3, and HA-4 can further improve this upper bound through neighborhood search; the B&B procedure remains unchanged.)

6.1.2. Lower-bound computation and pruning example

A depth-first search strategy is adopted in B&B. For visual clarity, child nodes in the illustrative search tree (Figure 1) are displayed in job-index order; in the actual implementation, branching follows the LPT order described in Section 5 to obtain high-quality incumbents early. Each node corresponds to a partial sequence $\pi = [\pi_p, \pi_s]$, where π_p contains the first k fixed jobs and π_s is the remaining job set. Since $(1 + x)^m$ is nonincreasing for $m \leq 0$, the lower bound LB is computed using the proposed bounding scheme, and a node is pruned when $LB \geq UB$.

Consider the leftmost branch at level 1, where the first job is fixed to J_1. $\pi_p = [J_1]$ and $\pi_s = \{J_2, J_3, J_4\}$. Sort the unscheduled jobs by SPT to obtain $[J_2, J_4, J_3]$, and note that $p_{\max} = \max\{1, 2, 3\} = 3$. With $\eta = 1$ (exponent $1/2$), the lower bound is

$$\begin{aligned} LB(J_1) = {} & \sqrt{4 \cdot (1+0)^{-0.5} \cdot (2+1+2+3)} + \sqrt{1 \cdot (1+4)^{-0.5} \cdot (1+2+3)} \\ & + \sqrt{2 \cdot (1+4+3)^{-0.5} \cdot (2+3)} + \sqrt{3 \cdot (1+4+2 \cdot 3)^{-0.5} \cdot 3} = 10.8225. \end{aligned}$$

Because $LB(J_1) = 10.8225 > UB = 10.4523$, this branch is pruned.

6.1.3. Optimal solution and search-tree illustration

The resulting search process is illustrated in Figure 1, where the value in parentheses is the node lower bound LB, and "×" indicates pruning. The optimal sequence is

$$\pi^* = [J_3, J_4, J_1, J_2],$$

with

$$g(\pi^*) = 9.9108,$$

which improves the initial upper bound $UB = 10.4523$.

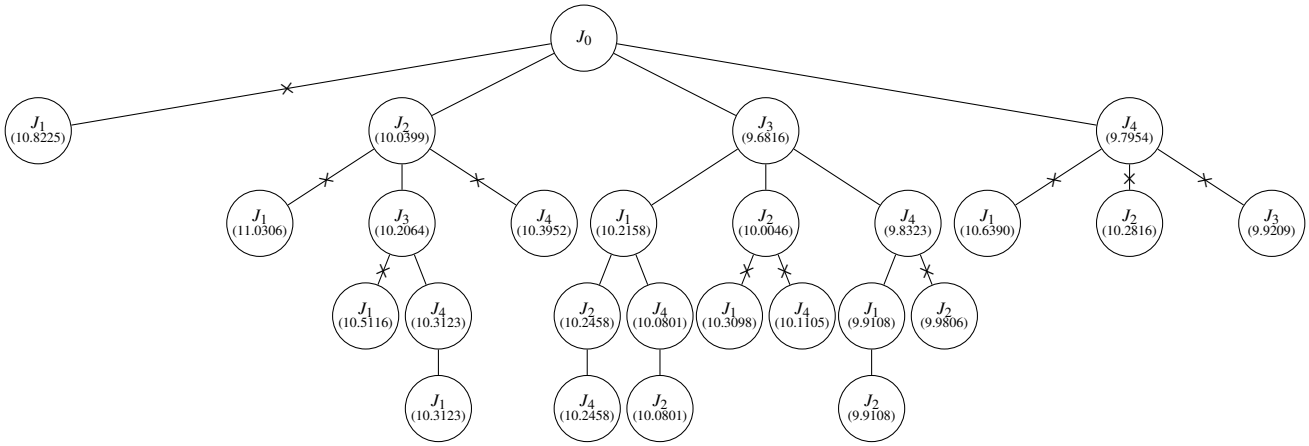

Figure 1. Illustration of the B&B search tree.

6.2. Randomized experiment design and metrics

To further evaluate HA and B&B under different problem sizes and learning indices, we conduct simulations on randomly generated instances.

We consider

$$n \in \{12, 13, 14, 15, 16, 17, 18, 19, 20\}, \qquad m \in \{-0.1, -0.25, -0.4, -0.55, -0.7\},$$

and fix $\eta = 1$. For each parameter pair (n, m), we generate 20 independent instances with

$$p'_j \sim U[1, 100], \qquad w_j \sim U[1, 10],$$

using fixed random seeds to ensure reproducibility.

For each instance, we run HA and B&B and record:

- CPU time of the heuristic procedure (seconds);
- CPU time of B&B (seconds);
- number of nodes visited by B&B (nodes).

We also define the relative error of the heuristic with respect to the optimal solution as

$$\text{error} = \frac{g(\pi_H) - g(\pi^*)}{g(\pi^*)},$$

where π_H is the heuristic solution and π^* is the B&B optimal sequence.

For each (n, m), both mean and maximum values are reported to reflect typical and difficult cases. Note that the B&B search effort is instance-dependent; even under the same parameter settings, CPU time and node counts may vary. Reporting both statistics provides a more complete view of stability and robustness.

6.3. Parameter settings for metaheuristics

To ensure a fair comparison among the metaheuristics, the revised experiments use a unified wall-clock CPU-time limit rather than different iteration-based stopping rules. Specifically, HA-2 (SA),

HA-3 (TS), and HA-4 (GA) are all run for the same time budget of 1.0 second per instance, as shown in Table 4. This design avoids bias caused by different computational costs per iteration: SA evaluates one neighbor per iteration, TS evaluates a full swap neighborhood, and GA evaluates an entire population in each generation. Thus, equal CPU time provides a more balanced basis for comparing solution quality.

Table 4. Revised parameter settings for the proposed metaheuristic algorithms.

Algorithm	Parameter	Value/Strategy
HA-2 (SA)	Time budget	1.0 s per instance
	Initial solution	Best sequence from HA-1
	Temperature calibration	200 sampled moves, target acceptance $p_0 = 0.7$
	Neighborhood structure	40% insertion, 60% swap
	Cooling schedule	Auto-calibrated inverse-temperature schedule
HA-3 (TS)	Time budget	1.0 s per instance
	Initial solution	Best of SPT, LPT, and 10 random sequences
	Tabu tenure	$\max(5, \lfloor 2\sqrt{n} \rfloor) + U(0, 2)$
	Neighborhood structure	Full swap neighborhood
	Aspiration criterion	Accept tabu move if it improves the global best
HA-4 (GA)	Time budget	1.0 s per instance
	Population size (P)	80
	Elitism size (E)	2
	Crossover strategy	Order crossover (OX), $p_c = 0.90$
	Mutation strategy	Swap/insertion/inversion, $p_m = 0.35$
	Mutation ratio	50% swap, 35% insertion, 15% inversion
	Selection mechanism	Tournament selection ($k = 3$)

All algorithms were implemented in C++. In the revised experiments, SA, TS, and GA are controlled by the same 1.0-second CPU-time cap, while the B&B algorithm is capped at 3600 seconds per instance.

All experiments were conducted on the same machine: Windows 10, AMD Ryzen 7 5700X (3.40 GHz), and 32 GB RAM. The algorithms were implemented in C++ and compiled using Visual Studio Code (version 1.108.0) with g++ (C++17).

6.4. Comparative results of algorithmic performance

Tables 5–8 summarize the computational performance of the exact branch-and-bound (B&B) algorithm and the relative errors of the corresponding heuristic solutions under the revised algorithmic protocol. Because the pruning effectiveness of B&B is sensitive to the quality of the initial upper bound (UB), the B&B time and number of explored nodes are reported separately when initialized with HA-1, HA-2 (SA), HA-3 (TS), and HA-4 (GA). For HA-2, HA-3, and HA-4, the reported heuristic solution quality is obtained under the same 1.0-second CPU-time budget, so the comparison no longer depends on unequal iteration limits or an implicit GA generation limit.

6.5. Early-stage pruning acceleration and synergistic mechanism

The relaxed lower bound relies on surrogate estimates of the unknown suffix contribution. As the learning effect intensifies (i.e., when the absolute value of m is large), such bounds may become looser because the cumulative learning term becomes more sensitive to the estimated accumulated processing time. In the revised implementation, the B&B procedure is combined with stronger heuristic upper bounds generated under the unified CPU-time budget. These upper bounds can compensate for looseness in the lower bound and help trigger the pruning condition $LB \geq UB$ more frequently. Therefore, the metaheuristics and B&B are used as complementary components: the former provides high-quality incumbents, while the latter verifies or improves them through systematic enumeration and pruning.

Table 5. Performance comparison between HA-1 and B&B under different parameter settings.

Parameters		HA-1 time (s)		B&B time (s)		B&B nodes		error	
n	m	mean	max	mean	max	mean	max	mean	max
12	−0.1	0.000009	0.000026	0.729418	3.402485	614453	3033178	0.121590	0.217895
12	−0.25	0.000008	0.000017	0.866185	3.977869	705797	3337144	0.115330	0.195294
12	−0.4	0.000007	0.000009	0.889641	3.794606	729029	3151005	0.108858	0.171412
12	−0.55	0.000007	0.000010	0.730308	3.192779	592983	2673096	0.098463	0.167079
12	−0.7	0.000007	0.000013	0.553759	2.085315	452848	1697743	0.084765	0.138448
13	−0.1	0.000008	0.000011	2.592505	8.041556	2035267	6468566	0.132502	0.217458
13	−0.25	0.000008	0.000010	3.235748	10.202840	2529850	8198754	0.124932	0.208274
13	−0.4	0.000008	0.000012	3.503554	11.043658	2734430	8825165	0.118410	0.202176
13	−0.55	0.000008	0.000023	3.520268	11.626691	2745389	9130125	0.106663	0.185310
13	−0.7	0.000008	0.000011	2.618875	9.713395	2011000	7543601	0.094474	0.172242
14	−0.1	0.000008	0.000010	13.378978	51.920282	10021577	39355563	0.129063	0.214557
14	−0.25	0.000008	0.000010	17.468197	68.428848	13075344	52296153	0.124509	0.196201
14	−0.4	0.000008	0.000010	20.048102	75.356698	15061435	57700676	0.120656	0.180400
14	−0.55	0.000008	0.000010	19.220595	69.281327	14456968	52996252	0.111844	0.176609
14	−0.7	0.000008	0.000013	14.349473	63.337602	10782196	48416388	0.098127	0.150144
15	−0.1	0.000008	0.000013	58.892677	349.630472	42139699	253529675	0.130680	0.215604
15	−0.25	0.000008	0.000010	77.819300	411.970060	56110027	296916121	0.124990	0.203725
15	−0.4	0.000009	0.000011	95.340767	501.312540	69072468	360500370	0.118655	0.192927
15	−0.55	0.000008	0.000012	86.269263	447.498336	63962464	336337912	0.108671	0.177774
15	−0.7	0.000008	0.000010	66.584734	253.458372	48841625	194784025	0.098505	0.160142
16	−0.1	0.000009	0.000012	383.159404	3362.541331	261041600	2302008323	0.136964	0.203444
16	−0.25	0.000009	0.000012	474.869666	3600	330148572	2529783791	0.131002	0.194655
16	−0.4	0.000009	0.000014	560.079041	3600	400163260	2607174830	0.123003	0.183278
16	−0.55	0.000008	0.000012	549.724384	3600	391001936	2609162373	0.114848	0.169773

Parameters		HA-1 time (s)		B&B time (s)		B&B nodes		error	
n	m	mean	max	mean	max	mean	max	mean	max
16	−0.7	0.000008	0.000011	430.287118	2205.149509	306170148	1608125065	0.104828	0.162070
17	−0.1	0.000009	0.000012	1274.554944	3600	877797112	2504465890	0.139756	0.228041
17	−0.25	0.000009	0.000012	1600.570548	3600	1092349886	2504156302	0.131264	0.208499
17	−0.4	0.000009	0.000012	1804.935663	3600	1225319399	2514474189	0.113068	0.194398
17	−0.55	0.000009	0.000015	1865.926332	3600	1240467267	2451199334	0.104383	0.174776
17	−0.7	0.000009	0.000011	1653.990803	3600	1111184067	2488239346	0.092941	0.151916
18	−0.1	0.000010	0.000014	2293.015090	3600	1412204075	2442076874	0.118510	0.196754
18	−0.25	0.000010	0.000012	2464.516240	3600	1587648118	2368308390	0.102457	0.180253
18	−0.4	0.000012	0.000015	2780.525934	3600	1525023809	2170585694	0.056905	0.130246
18	−0.55	0.000012	0.000015	2682.459585	3600	1629680114	2235601667	0.044697	0.116065
18	−0.7	0.000012	0.000015	2716.058779	3600	1410302247	2232834101	0.044770	0.101068
19	−0.1	0.000010	0.000011	2693.515637	3600	1651614905	2275271478	0.069767	0.144849
19	−0.25	0.000011	0.000013	3342.487124	3600	2016520973	2247937677	0.056840	0.134581
19	−0.4	0.000013	0.000014	3600	3600	2127843104	2226016127	0.031032	0.092885
19	−0.55	0.000012	0.000015	3600	3600	2093659439	2165721567	0.025640	0.103702
19	−0.7	0.000011	0.000013	3424.544627	3600	2013848142	2202780947	0.025335	0.096995
20	−0.1	0.000010	0.000012	3600	3600	2171491428	2234194961	0.053114	0.117709
20	−0.25	0.000011	0.000013	3600	3600	2194072864	2250228138	0.032092	0.065340
20	−0.4	0.000011	0.000013	3600	3600	2183172129	2253331603	0.007690	0.021062
20	−0.55	0.000012	0.000014	3600	3600	2179345262	2248086455	0.006257	0.018927
20	−0.7	0.000012	0.000014	3600	3600	2175752534	2243198462	0.006195	0.018281

Table 6. Performance comparison between HA-2 (SA) and B&B under different parameter settings.

Parameters		HA time (s)		B&B time (s)		B&B nodes		error	
n	m	mean	max	mean	max	mean	max	mean	max
12	−0.1	1.000139	1.000183	0.314781	1.944843	256798	1597247	0.000000	0.000000
12	−0.25	1.000137	1.000205	0.257920	1.339310	208210	1059578	0.000000	0.000000
12	−0.4	1.000131	1.000176	0.212775	1.126049	168818	871466	0.000000	0.000000
12	−0.55	1.000131	1.000179	0.168434	0.954269	137328	786301	0.000000	0.000000
12	−0.7	1.000128	1.000167	0.113640	0.450822	93525	375136	0.000000	0.000000
13	−0.1	1.000156	1.000280	1.015428	3.656707	783966	2772537	0.000000	0.000000

Parameters		HA time (s)		B&B time (s)		B&B nodes		error	
n	m	mean	max	mean	max	mean	max	mean	max
13	−0.25	1.000146	1.000190	0.841245	2.911783	645982	2274881	0.000000	0.000000
13	−0.4	1.000144	1.000188	0.640466	2.314212	490709	1770562	0.000000	0.000000
13	−0.55	1.000142	1.000185	0.475522	1.807482	361276	1399535	0.000000	0.000000
13	−0.7	1.000144	1.000201	0.315955	0.959509	239654	754980	0.000000	0.000000
14	−0.1	1.000163	1.000230	5.206569	19.983959	3833138	14735593	0.000000	0.000000
14	−0.25	1.000151	1.000199	4.341932	17.387174	3184842	12973734	0.000000	0.000000
14	−0.4	1.000167	1.000200	3.312911	13.552692	2380514	9798184	0.000000	0.000000
14	−0.55	1.000146	1.000198	2.353887	11.792209	1709419	8632852	0.000000	0.000000
14	−0.7	1.000153	1.000199	1.469688	5.741868	1068292	4300771	0.000000	0.000000
15	−0.1	1.000162	1.000208	22.497072	157.703088	15840216	111011215	0.000000	0.000000
15	−0.25	1.000157	1.000211	18.037973	115.324330	12750187	81960144	0.000000	0.000000
15	−0.4	1.000153	1.000214	14.116764	100.070185	9889668	71099673	0.000000	0.000000
15	−0.55	1.000155	1.000214	10.684687	85.903452	7536890	61995960	0.000000	0.000000
15	−0.7	1.000155	1.000208	6.521275	41.885710	4580476	30026654	0.000000	0.000000
16	−0.1	1.000182	1.000252	148.603743	1511.061249	101352340	1034548853	0.000000	0.000000
16	−0.25	1.000172	1.000227	114.434307	1055.255499	79319055	733647889	0.000000	0.000000
16	−0.4	1.000165	1.000229	88.315677	858.071824	61943472	613148487	0.000000	0.000000
16	−0.55	1.000180	1.000259	70.685930	752.280100	47175254	515642474	0.000000	0.000000
16	−0.7	1.000178	1.000234	38.543274	330.685472	25377734	223549639	0.000000	0.000000
17	−0.1	1.000191	1.000412	584.525431	3600	377160723	2299938989	0.000000	0.000000
17	−0.25	1.000183	1.000249	515.757488	3600	341610964	2368053870	0.000000	0.000000
17	−0.4	1.000177	1.000236	403.501664	3178.353083	269823670	2166576772	0.000000	0.000000
17	−0.55	1.000179	1.000239	261.469095	1959.622806	174180996	1336353065	0.000000	0.000000
17	−0.7	1.000184	1.000279	179.399061	1449.933236	119622252	986607280	0.000000	0.000000
18	−0.1	1.000191	1.000266	1626.030436	3600	1040599452	2334414690	0.000000	0.000000
18	−0.25	1.000196	1.000261	1567.254346	3600	988834837	2274457743	0.000000	0.000000
18	−0.4	1.000186	1.000247	1369.788697	3600	865198196	2278409854	0.000000	0.000000
18	−0.55	1.000189	1.000248	1040.804592	3600	656227046	2323818775	0.000000	0.000000
18	−0.7	1.000190	1.000255	778.103475	3600	488366329	2334796464	0.000000	0.000000
19	−0.1	1.000201	1.000307	2452.375278	3600	1460263490	2203668812	0.000000	0.000000
19	−0.25	1.000216	1.000293	2474.926069	3600	1382941999	2179892804	0.000000	0.000000
19	−0.4	1.000195	1.000260	2204.737268	3600	1293913056	2170292998	0.000000	0.000000
19	−0.55	1.000196	1.000261	2391.864366	3600	1282760835	2110912997	0.000000	0.000000
19	−0.7	1.000190	1.000255	2186.525278	3600	1306334011	2159111046	0.000000	0.000000
20	−0.1	1.000172	1.000227	2909.630359	3600	1657284520	2106338190	0.000000	0.000000

Parameters		HA time (s)		B&B time (s)		B&B nodes		error	
n	m	mean	max	mean	max	mean	max	mean	max
20	−0.25	1.000057	1.000142	3079.819807	3600	1728411852	2066188207	0.000000	0.000000
20	−0.4	1.000142	1.000264	3059.029250	3600	1701402448	2035776102	0.000000	0.000000
20	−0.55	1.000091	1.000181	3021.316229	3600	1691015551	2046618359	0.000000	0.000000
20	−0.7	1.000140	1.000261	2961.235229	3600	1695339208	2078303929	0.000000	0.000000

Table 7. Performance comparison between HA-3 (TS) and B&B under different parameter settings.

Parameters		HA time (s)		B&B time (s)		B&B nodes		error	
n	m	mean	max	mean	max	mean	max	mean	max
12	−0.1	1.000027	1.000053	0.310585	1.911929	256799	1597247	0.000000	0.000000
12	−0.25	1.000039	1.000275	0.258722	1.294315	208211	1059578	0.000000	0.000000
12	−0.4	1.000025	1.000047	0.218535	1.289570	178811	1071325	0.000429	0.008587
12	−0.55	1.000020	1.000038	0.170733	0.965962	137328	786301	0.000000	0.000000
12	−0.7	1.000020	1.000036	0.115587	0.453260	93526	375136	0.000000	0.000000
13	−0.1	1.000024	1.000047	1.015212	3.651761	783967	2772537	0.000000	0.000000
13	−0.25	1.000027	1.000067	0.831707	2.882173	645983	2274881	0.000000	0.000000
13	−0.4	1.000028	1.000066	0.658444	2.542127	507952	1952526	0.000386	0.007723
13	−0.55	1.000053	1.000503	0.469844	1.762374	361276	1399535	0.000000	0.000000
13	−0.7	1.000043	1.000152	0.316039	0.959208	239655	754980	0.000000	0.000000
14	−0.1	1.000039	1.000092	5.226676	19.881813	3833138	14735593	0.000000	0.000000
14	−0.25	1.000037	1.000077	4.354819	17.303401	3184842	12973734	0.000000	0.000000
14	−0.4	1.000039	1.000083	3.274036	13.161861	2380514	9798184	0.000000	0.000000
14	−0.55	1.000034	1.000064	2.325248	11.496744	1712818	8632852	0.000026	0.000527
14	−0.7	1.000030	1.000078	1.453073	5.855987	1068292	4300771	0.000000	0.000000
15	−0.1	1.000046	1.000082	22.219185	155.335038	15840217	111011215	0.000000	0.000000
15	−0.25	1.000049	1.000100	18.060218	115.043913	12750188	81960144	0.000000	0.000000
15	−0.4	1.000047	1.000088	14.021344	99.249085	9889669	71099673	0.000000	0.000000
15	−0.55	1.000038	1.000102	10.701167	85.999072	7536891	61995960	0.000000	0.000000
15	−0.7	1.000041	1.000099	6.589024	41.851297	4580477	30026654	0.000000	0.000000
16	−0.1	1.000051	1.000123	149.019429	1506.068221	101352341	1034548853	0.000000	0.000000
16	−0.25	1.000052	1.000099	113.608692	1037.183944	79319055	733647889	0.000000	0.000000
16	−0.4	1.000043	1.000116	89.349539	853.685925	61943472	613148487	0.000000	0.000000
16	−0.55	1.000060	1.000127	71.112735	760.814681	47175254	515642474	0.000000	0.000000

Parameters		HA time (s)		B&B time (s)		B&B nodes		error	
n	m	mean	max	mean	max	mean	max	mean	max
16	−0.7	1.000050	1.000094	38.356714	333.867724	25515734	223549639	0.000092	0.001836
17	−0.1	1.000071	1.000154	583.897549	3600	376149405	2279712621	0.000000	0.000000
17	−0.25	1.000060	1.000149	515.361676	3600	342280565	2381445890	0.000000	0.000000
17	−0.4	1.000061	1.000097	403.324591	3185.480608	269823670	2166576772	0.000000	0.000000
17	−0.55	1.000054	1.000126	260.910151	1949.259689	174280474	1336353065	0.000012	0.000243
17	−0.7	1.000069	1.000147	179.803293	1447.766918	120090216	986607280	0.000592	0.011843
18	−0.1	1.000073	1.000129	1623.833893	3600	1044103256	2351119567	0.000000	0.000000
18	−0.25	1.000069	1.000128	1562.831685	3600	992056296	2289153107	0.000000	0.000000
18	−0.4	1.000074	1.000226	1369.443339	3600	865314817	2279104842	0.000000	0.000000
18	−0.55	1.000053	1.000131	1042.745084	3600	656016333	2324083188	0.000000	0.000000
18	−0.7	1.000056	1.000122	786.640793	3600	491939345	2326361924	0.000485	0.007416
19	−0.1	1.000070	1.000132	2451.227923	3600	1465919788	2217043707	0.000000	0.000000
19	−0.25	1.000086	1.000152	2474.223183	3600	1384746581	2197845777	0.000000	0.000000
19	−0.4	1.000114	1.000646	2205.636923	3600	1293503495	2168427389	0.000001	0.000011
19	−0.55	1.000172	1.000227	2378.573972	3600	1422001644	2198977746	0.000000	0.000000
19	−0.7	1.000053	1.000131	2180.478883	3600	1312208122	2174343187	0.000000	0.000000
20	−0.1	1.000049	1.000100	2904.203166	3600	1665672406	2122024491	0.000000	0.000000
20	−0.25	1.000050	1.000094	3078.248419	3600	1735165438	2081091292	0.000000	0.000000
20	−0.4	1.000049	1.000100	3058.407144	3600	1705005429	2044827136	0.000000	0.000000
20	−0.55	1.000043	1.000152	3018.757386	3600	1723519857	2081224600	0.000085	0.000181
20	−0.7	1.000027	1.000067	2960.950498	3600	1720650655	2116300462	0.000000	0.000000

Table 8. Performance comparison between HA-4 (GA) and B&B under different parameter settings.

Parameters		HA time (s)		B&B time (s)		B&B nodes		error	
n	m	mean	max	mean	max	mean	max	mean	max
12	−0.1	1.000039	1.000064	0.317065	1.948440	256798	1597247	0.000000	0.000000
12	−0.25	1.000034	1.000093	0.259233	1.312902	208210	1059578	0.000000	0.000000
12	−0.4	1.000035	1.000066	0.212988	1.052645	168818	871466	0.000000	0.000000
12	−0.55	1.000033	1.000074	0.170416	0.966488	137328	786301	0.000000	0.000000
12	−0.7	1.000036	1.000069	0.115546	0.449512	93525	375136	0.000000	0.000000
13	−0.1	1.000036	1.000074	1.019230	3.637797	783966	2772537	0.000000	0.000000

Parameters		HA time (s)		B&B time (s)		B&B nodes		error	
n	m	mean	max	mean	max	mean	max	mean	max
13	−0.25	1.000035	1.000078	0.841901	2.824273	645982	2274881	0.000000	0.000000
13	−0.4	1.000044	1.000077	0.653147	2.367304	490709	1770562	0.000000	0.000000
13	−0.55	1.000044	1.000076	0.483009	1.831756	361276	1399535	0.000000	0.000000
13	−0.7	1.000041	1.000074	0.318094	0.988232	239654	754980	0.000000	0.000000
14	−0.1	1.000042	1.000080	5.262016	20.182313	3833138	14735593	0.000000	0.000000
14	−0.25	1.000043	1.000081	4.364062	17.399408	3184842	12973734	0.000000	0.000000
14	−0.4	1.000044	1.000092	3.328082	13.573577	2380514	9798184	0.000000	0.000000
14	−0.55	1.000031	1.000066	2.346287	11.608978	1709419	8632852	0.000000	0.000000
14	−0.7	1.000040	1.000084	1.469408	5.811633	1068292	4300771	0.000000	0.000000
15	−0.1	1.000049	1.000104	22.570592	158.356723	15840216	111011215	0.000000	0.000000
15	−0.25	1.000042	1.000073	18.113041	116.220950	12750187	81960144	0.000000	0.000000
15	−0.4	1.000047	1.000097	14.165936	100.527058	9889668	71099673	0.000000	0.000000
15	−0.55	1.000050	1.000114	10.740559	86.697865	7536890	61995960	0.000000	0.000000
15	−0.7	1.000042	1.000079	6.547685	41.937744	4580476	30026654	0.000000	0.000000
16	−0.1	1.000052	1.000084	149.196700	1516.062556	101352340	1034548853	0.000000	0.000000
16	−0.25	1.000041	1.000090	114.941893	1060.104368	79319055	733647889	0.000000	0.000000
16	−0.4	1.000044	1.000086	88.801318	861.996614	61943472	613148487	0.000000	0.000000
16	−0.55	1.000048	1.000079	70.885154	754.559235	47175254	515642474	0.000000	0.000000
16	−0.7	1.000048	1.000156	38.670486	333.461908	25377734	223549639	0.000000	0.000000
17	−0.1	1.000042	1.000071	586.411270	3600	376639811	2289520754	0.000000	0.000000
17	−0.25	1.000055	1.000233	517.812573	3600	341202533	2359885253	0.000000	0.000000
17	−0.4	1.000052	1.000112	406.252850	3200.313403	269823670	2166576772	0.000000	0.000000
17	−0.55	1.000043	1.000083	263.692500	1975.559281	174180996	1336353065	0.000000	0.000000
17	−0.7	1.000040	1.000071	180.454514	1457.964527	119622252	986607280	0.000000	0.000000
18	−0.1	1.000045	1.000075	1629.091231	3600	1037185505	2327877671	0.000000	0.000000
18	−0.25	1.000047	1.000102	1566.870015	3600	988847611	2275462084	0.000000	0.000000
18	−0.4	1.000048	1.000084	1370.991784	3600	862788890	2267092916	0.000000	0.000000
18	−0.55	1.000050	1.000102	1043.441526	3600	655551694	2317414673	0.000000	0.000000
18	−0.7	1.000052	1.000100	780.813905	3600	486789511	2317146939	0.000000	0.000000
19	−0.1	1.000057	1.000112	2452.256239	3600	1458968250	2200424201	0.000000	0.000000
19	−0.25	1.000053	1.000111	2476.427841	3600	1375389636	2181597775	0.000000	0.000000
19	−0.4	1.000045	1.000079	2206.185701	3600	1289544464	2154937092	0.000000	0.000000
19	−0.55	1.000045	1.000075	2380.222552	3600	1414553265	2193408641	0.000000	0.000000
19	−0.7	1.000041	1.000090	2186.961509	3600	1304119974	2149799601	0.000000	0.000000
20	−0.1	1.000047	1.000097	2909.335616	3600	1654261550	2098636460	0.000000	0.000000

Parameters		HA time (s)		B&B time (s)		B&B nodes		error	
n	m	mean	max	mean	max	mean	max	mean	max
20	−0.25	1.000042	1.000080	3079.182976	3600	1722903622	2060911454	0.000000	0.000000
20	−0.4	1.000042	1.000073	3059.568338	3600	1694397071	2023624038	0.000000	0.000000
20	−0.55	1.000041	1.000074	3019.646855	3600	1714167608	2067135470	0.000000	0.000000
20	−0.7	1.000050	1.000114	2959.990014	3600	1707168134	2109715149	0.000000	0.000000

Tables 5–8 quantify how different heuristic upper bounds dictate B&B search efficiency and overall solution quality.

Computationally, HA-1, HA-2 (SA), and HA-4 (GA) incur very small overhead in the reported tests, with running times that are much less sensitive to job size n and learning factor m than those of B&B. While HA-3 (TS) exhibits the expected runtime growth due to intensive neighborhood evaluations, it remains tractable for $n \leq 20$ in the reported experiments. By contrast, the exact B&B algorithm exhibits rapid growth in computational effort and frequently reaches the time limit for larger instances. This contrast illustrates the computational difficulty of the sequencing search space [47].

Regarding solution quality, stronger learning effects (smaller m) tend to reduce relative errors across the tested heuristics. Compared with the constructive baseline HA-1, the metaheuristics (HA-2, HA-3, and HA-4) are designed to exploit additional search mechanisms and can provide improved or more stable incumbents in many tested cases. These heuristic solutions also affect B&B efficiency: initializing B&B with a better upper bound can reduce runtime and node exploration compared with HA-1. Thus, while HA-1 provides immediate feasibility, the metaheuristics can supply stronger incumbents for accelerating exact or truncated B&B search, yielding a flexible trade-off between computational cost and solution quality.

These performance patterns can be interpreted through three structural mechanisms:

- **Impact of Learning Effects:** Stronger learning effects (smaller m) increase the value of early workload accumulation. This can shift the objective landscape toward sequences that emphasize shorter processing times early in the schedule. By incorporating SPT-related logic, the heuristics can capture this tendency and reduce relative errors in relevant instances.
- **Algorithmic Discrepancies:** The continuous-discrete coupling creates a rugged search landscape with local optima. While the constructive HA-1 and single-trajectory HA-2 (SA) may be affected by local search limitations, the short-term memory of HA-3 (TS) and the population diversity of HA-4 (GA) provide additional mechanisms for exploring the search space.
- **Early-Stage Pruning Acceleration:** Pruning a node at depth k eliminates a subtree of up to $(n - k)!$ permutations. Providing the B&B algorithm with a strong metaheuristic upper bound (UB) can cause the lower bound (LB) to meet the pruning condition closer to the root node. This early-stage pruning helps mitigate the combinatorial explosion.

6.6. Large-scale heuristic performance comparison

At the tested small-to-medium scales ($n \leq 16$), all three metaheuristics (HA-2, HA-3, HA-4) consistently achieved error rates near zero relative to the B&B optimum, indicating that they frequently

located the global optimum. This makes their relative performance difficult to distinguish from Tables 6–8 alone. To further differentiate the metaheuristics at scales where B&B becomes computationally infeasible, we extend the experiments to larger instances with $n \in \{30, 50, 80, 100\}$.

In this large-scale experiment, the B&B algorithm is omitted due to its exponential growth in computational cost. Instead, HA-1 serves as the baseline reference, and the three metaheuristics are compared by the percentage improvement over HA-1 under a unified 2.0-second CPU-time budget:

$$\text{Improvement\%} = \frac{g(\pi_{\text{HA-1}}) - g(\pi_{\text{HA}})}{g(\pi_{\text{HA-1}})} \times 100\%.$$

For each parameter pair (n, m), 20 independent instances were generated using the same random seeds as in the small-scale experiment. The same algorithmic parameters listed in Table 4 were used, except that the time budget was extended to 2.0 seconds to accommodate the larger instance sizes. The results are summarized in Table 9.

Table 9. Performance comparison of metaheuristics against HA-1 baseline under a unified 2.0-second CPU-time budget.

Parameters		HA-1	SA imp%		TS imp%		GA imp%	
n	m	$g(\pi_{\text{HA-1}})$	mean	max	mean	max	mean	max
30	−0.1	1211.64	13.33	17.55	13.33	17.55	13.33	17.55
30	−0.25	812.37	13.15	16.62	13.14	16.62	13.15	16.62
30	−0.4	554.38	13.09	16.44	13.06	16.44	13.09	16.44
30	−0.55	386.10	13.03	16.16	12.95	16.16	13.04	16.16
30	−0.7	274.60	12.66	15.73	12.61	15.73	12.67	15.73
50	−0.1	2507.71	14.31	17.15	14.31	17.15	14.31	17.15
50	−0.25	1615.45	14.56	16.70	14.56	16.70	14.56	16.70
50	−0.4	1056.73	14.97	16.85	14.96	16.83	14.97	16.83
50	−0.55	704.05	15.32	17.12	15.31	17.12	15.32	17.12
50	−0.7	478.67	15.32	17.57	15.27	17.57	15.32	17.57
80	−0.1	4863.32	14.68	17.72	14.68	17.72	14.68	17.72
80	−0.25	3030.06	15.38	18.30	15.38	18.30	15.38	18.30
80	−0.4	1916.13	16.38	18.46	16.37	18.39	16.37	18.46
80	−0.55	1222.92	16.68	18.51	16.66	18.51	16.68	18.51
80	−0.7	792.90	16.33	19.08	16.32	19.08	16.34	19.08
100	−0.1	6725.94	14.87	17.42	14.85	17.40	14.87	17.42
100	−0.25	4120.15	15.71	18.25	15.67	18.20	15.71	18.22
100	−0.4	2562.84	16.95	19.07	16.90	18.97	16.94	19.07
100	−0.55	1604.35	17.33	19.70	17.26	19.65	17.33	19.70
100	−0.7	1015.77	16.77	20.30	16.70	20.27	16.77	20.30

As shown in Table 9, all three metaheuristics consistently and significantly improve over the HA-1 baseline across all tested scales, with improvement rates ranging from approximately 12% to 20%.

Two clear patterns emerge. First, the improvement over HA-1 increases with both the problem size n and the absolute value of the learning index $|m|$, suggesting that larger and more learning-intensive instances offer greater room for sequence optimization beyond simple constructive rules. Second, under the same 2.0-second budget, SA and GA exhibit nearly identical performance, while TS consistently trails by a small but systematic margin. This ordering is consistent with the per-iteration computational cost: TS exhaustively evaluates the full $O(n^2)$ swap neighborhood in each iteration, which limits the number of iterations achievable within the fixed time budget, whereas SA and GA spend their budget on more frequent, lighter objective evaluations. The results also confirm that, despite being the simplest metaheuristic, SA performs competitively with the population-based GA under equalized computational resources for the tested instances.

6.7. Cost-benefit trade-off analysis under multigradient time constraints

To quantify the trade-off between solution improvement and computational time from a practical perspective, we conducted an extended numerical analysis under multiple time limits. We selected the largest tested instance size ($n = 20$) with a strong learning index ($m = -0.55$) to represent a challenging computational scenario. To capture the trajectory of the exact search and isolate the effect of heuristic upper bounds, the branch-and-bound (B&B) algorithm was initialized with a random sequence. The incumbent solutions and their relative errors were recorded at selected time milestones over a duration.

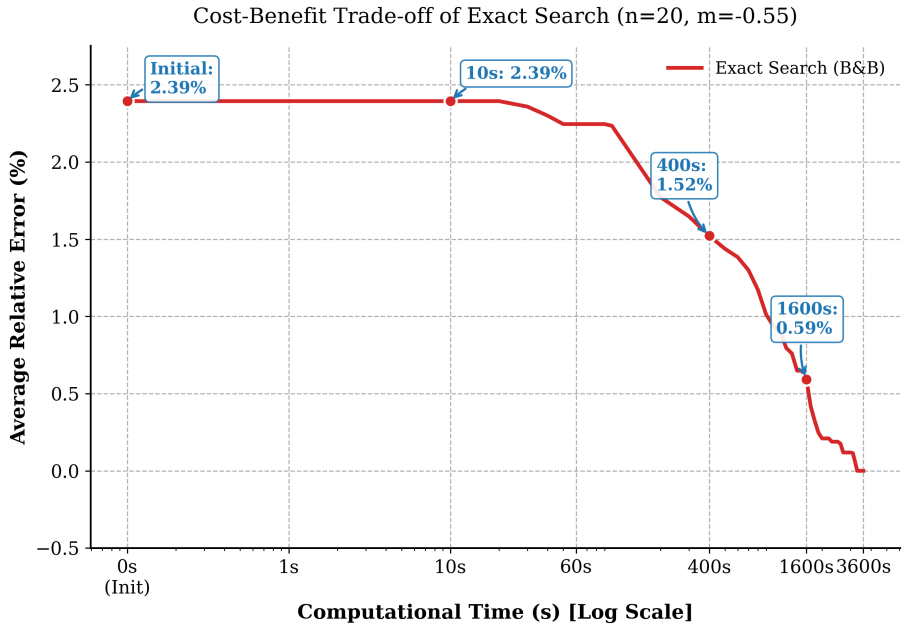

Figure 2. Cost-benefit trade-off analysis of the B&B search under multiple time limits ($n = 20, m = -0.55$). The incumbent illustrates the gradual convergence of exact search over time.

As illustrated by the empirical data in Figure 2, the exact search process exhibits a gradual improvement trajectory over time. Starting from a random sequence with an initial relative error of 2.39%, the B&B algorithm progressively reduces the error as the search explores deeper into the solution space. At the 400-second mark, the relative error has decreased to 1.52%, and it further

declines to 0.59% by 1600 seconds. The final incumbent solution is discovered at approximately 3300 seconds, after which no further improvement is found within the 3600-second time budget. Across the full run, approximately 2.1×10^9 nodes are explored, underscoring the substantial computational effort required for deep exhaustive search at this problem scale.

This quantitative evidence highlights the practical trade-off between computational investment and solution quality. While the B&B algorithm ultimately identifies a near-optimal or optimal solution, the improvement over a simple random starting point narrows gradually rather than abruptly. For the tested instance, the marginal return per unit of computing time diminishes as the search progresses, suggesting that time-limited exact search can serve as a practical offline benchmark, with metaheuristics providing a computationally efficient alternative for online decision-making.

6.8. Managerial insights and practical implications

Beyond algorithmic contributions, the structural and computational results provide several model-consistent implications for decision-makers in labor-intensive and high-precision manufacturing. These implications should be interpreted within the assumptions of the proposed cumulative normal-processing-time learning model.

- **Learning as workload accumulation:** In this study, the learning effect is driven by cumulative normal processing time, which represents accumulated standard workload exposure before each job, rather than the actual compressed processing time after resource allocation. Therefore, operator experience should not be interpreted as a cost-free virtual resource or as a direct substitute for physical resource investment. The results also do not imply that short jobs should always be scheduled first. Instead, the benefit of an SPT-like sequence depends on the distribution of normal processing times, job weights, the learning index m, and the resource-allocation cost.
- **Caution with static dispatching rules:** Traditional production software, such as ERP or APS systems, often relies on static priority rules such as SPT or WSPT. Our analysis shows that such rules are not universally reliable in the proposed sequence-dependent setting. The preferred sequence depends jointly on accumulated workload-based learning and residual cumulative job weights. Hence, managers should use the proposed model as a decision-support tool to evaluate candidate schedules rather than applying a single fixed priority rule.
- **Tailored algorithm deployment:** For small- and medium-scale instances where exact verification is required, the B&B algorithm can be used to obtain benchmark optimal solutions. However, its scalability is limited by the conservative lower bound and the factorial growth of the sequencing space. For larger or real-time scheduling instances, the proposed metaheuristics provide practical high-quality solutions under limited computational budgets.

Overall, the proposed framework helps managers quantitatively evaluate the trade-off among workload-accumulation-based learning, job-dependent urgency, and physical resource-allocation cost. It should be viewed as a parameter-dependent decision-support model, rather than as a source of universal managerial rules.

7. Conclusion

This study investigates a single-machine scheduling problem that simultaneously considers time-dependent learning effects and non-renewable resource allocation. The objective is to minimize a weighted aggregate of total weighted completion time and total resource consumption cost. To address the nonlinear and combinatorial complexity arising from sequence-dependent processing times, we conduct a systematic study covering model formulation, structural analysis, algorithm design, and numerical experiments.

From a theoretical perspective, the optimal resource-allocation structure is derived for any given job sequence, which enables the original joint optimization problem to be reduced to an equivalent sequencing model. The resulting sequencing problem remains highly nonlinear and computationally challenging.

On the algorithmic side, several heuristic methods are developed to construct high-quality feasible solutions rapidly. Among them, HA-1 has a simple structure and very low computational cost, and it can generate feasible solutions almost instantaneously. Building upon this baseline, HA-2 (SA), HA-3 (TS), and HA-4 (GA) incorporate randomized search, local improvement, and population-based mechanisms, respectively, to improve solution quality and robustness. Moreover, these heuristics can provide tighter initial upper bounds for the branch-and-bound procedure, thereby improving pruning efficiency and overall search performance.

Overall, for single-machine scheduling problems with learning effects and resource-allocation decisions, combining heuristic approaches with exact methods offers an effective solution strategy that balances solution quality and computational efficiency. Future research may extend the proposed model to settings with release dates, flow shop settings (Shabtay and Oron [51], Pan et al. [52], Jin et al. [53]), parallel-machine environments (Wang [54], Ben Abdellafou et al. [55], Hu et al. [56]), or more general learning and deterioration mechanisms, to further broaden its applicability.

Author contributions

Xinfeng Wu, Lin Lin, and Ji-Bo Wang designed the methodology; Lin Lin and Ji-Bo Wang conducted the investigation; Xinfeng Wu wrote the original draft; and Xinfeng Wu, Lin Lin, and Ji-Bo Wang reviewed and edited the manuscript. All authors have read and agreed to the published version of the manuscript.

Use of Generative-AI tools declaration

During the preparation of this manuscript, the authors used ChatGPT solely for the purpose of language editing, grammatical correction, and improving the overall readability of the text. The AI tool was not used to generate original ideas, collect data, or perform analysis. Following the use of this tool, the authors manually reviewed and edited the content as necessary and take full responsibility for the final integrity and accuracy of the work.

Acknowledgments

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

Conflict of interest

The authors declare no conflict of interest.

Data Availability

The experimental data, source code, and parameter settings (including random seeds) that support the findings of this study are available from the corresponding author upon reasonable request.

References

1. M. L. Pinedo, *Scheduling-Theory, Algorithms, and Systems*, Vol. 29, Springer, 2012. https://doi.org/10.1007/978-1-4614-2361-4
2. D. R. Karger, C. Stein, J. Wein, Scheduling algorithms, *Handbook of Algorithms and Theory of Computation,* (1999).
3. R. G. Vickson, Choosing the job sequence and processing times to minimize total processing plus flow cost on a single machine, *Oper. Res.,* **28** (1980), 1155–1167. https://doi.org/10.1287/opre.28.5.1155
4. D. Shabtay, G. Steiner, A survey of scheduling with controllable processing times, *Discret. Appl. Math.,* **155** (2007), 1643–1666. https://doi.org/10.1016/j.dam.2007.02.003
5. N. Yin, X. Y. Wang, Single-machine scheduling with controllable processing times and learning effect, *Int. J. Adv. Manuf. Technol.,* **54** (2011), 743–748. https://doi.org/10.1007/s00170-010-2973-z
6. 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://dx.doi.org/10.1142/S0217595923500264
7. 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
8. M. A. Abdeljaouad, Z. Bahroun, N. E. H. Saadani, R. Sheiko, K. Al-Assaf, A resource-constrained optimization model for parallel machine scheduling with constraint programming, *Decis. Anal. J.,* **15** (2025), 100585. https://doi.org/10.1016/j.dajour.2025.100585
9. 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
10. 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

11. 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

12. 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

13. 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

14. 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

15. S. Kovalev, I. Chalamon, A. Becuwe, Single machine scheduling with resource constraints: Equivalence to two-machine flow-shop scheduling for regular objectives, *J. Oper. Res. Soc.,* **75** (2024), 1343–1346. https://doi.org/10.1080/01605682.2023.2244529

16. 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

17. D. Shabtay, Coordinating scheduling and resource-allocation decisions in a proportionate flow-shop scheduling environment, *Optim. Lett.,* **20** (2026), 727–745. https://doi.org/10.1007/s11590-025-02274-6

18. D. Biskup, Single-machine scheduling with learning considerations, *Eur. J. Oper. Res.,* **115** (1999), 173–178. https://doi.org/10.1016/S0377-2217(98)00246-X

19. D. Biskup, A state-of-the-art review on scheduling with learning effects, *Eur. J. Oper. Res.,* **188** (2008), 315–329. https://doi.org/10.1016/j.ejor.2007.05.040

20. C. C. Wu, W. C. Lin, X. Zhang, I. H. Chung, T. H. Yang, K. Lai, Tardiness minimisation for a customer order scheduling problem with sum-of-processing-time-based learning effect, *J. Oper. Res. Soc.,* **70** (2019), 487–501. https://doi.org/10.1080/01605682.2018.1447249

21. G. Mosheiov, Scheduling jobs under simple linear deterioration, *Comput. Oper. Res.,* **21** (1994), 653–659. https://doi.org/10.1016/0305-0548(94)90080-9

22. D. Oron, Single machine scheduling with simple linear deterioration to minimize total absolute deviation of completion times, *Comput. Oper. Res.,* **35** (2008), 2071–2078. https://doi.org/10.1016/j.cor.2006.10.010

23. W. C. Lee, C. C. Wu, Y. H. Chung, Scheduling deteriorating jobs on a single machine with release times, *Comput. Ind. Eng.,* **54** (2008), 441–452. https://doi.org/10.1016/j.cie.2007.08.006

24. 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

25. 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

26. 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

27. 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

28. 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

29. S. Gawiejnowicz, *Time-dependent scheduling*, Berlin: Springer Science & Business Media, 2008. https://doi.org/10.1007/978-3-540-69446-5

30. S. Gawiejnowicz, *Models and algorithms of time-dependent scheduling*, Vol. 2, Springer, 2020. https://doi.org/10.1007/978-3-662-59362-2

31. S. Gawiejnowicz, A review of four decades of time-dependent scheduling: Main results, new topics, and open problems, *J. Sched.,* **23** (2020), 3–47. https://doi.org/10.1007/s10951-019-00630-w

32. Y. Yin, T. C. E. Cheng, C. C. Wu, Scheduling with time-dependent processing times, *Math. Probl. Eng.,* **2015** (2015), 704543. http://doi.org/10.1155/2015/367585

33. T. C. E. Cheng, G. Wang, Single machine scheduling with learning effect considerations, *Ann. Oper. Res.,* **98** (2000), 273–290. https://doi.org/10.1023/A:1019216726076

34. J. B. Wang, C. T. Ng, T. C. E. Cheng, L. L. Liu, Single-machine scheduling with a time-dependent learning effect, *Int. J. Prod. Econ.,* **111** (2008), 802–811. https://doi.org/10.1016/j.ijpe.2007.03.013

35. W. H. Kuo, D. L. Yang, Single-machine group scheduling with a time-dependent learning effect, *Comput. Oper. Res.,* **33** (2006), 2099–2112. https://doi.org/10.1016/j.cor.2004.11.024

36. 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

37. 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

38. 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

39. 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. http://doi.org/10.1142/S0217595926500016

40. T. Liu, X. Y. Sun, L. H. Zhang, Y. H. Fan, Coordinating single-machine scheduling with the general linear deterioration function and maintenance activity: minimizing the makespan, *Eng. Optim.,* (2026). https://doi.org/10.1080/0305215X.2026.2645254

41. Y. C. Wang, J. B. Wang, Study on convex resource allocation scheduling with a time-dependent learning effect, *Mathematics,* **11** (2023), 3179. https://doi.org/10.3390/math11143179

42. 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

43. 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

44. N. Ren, D. Y. Lv, J. B. Wang, X. Y. Wang, Solution algorithms for single-machine scheduling with learning effects and exponential past-sequence-dependent delivery times, *J. Ind. Manag. Optim.,* **19** (2023), 8429–8450. https://doi.org/10.3934/jimo.2023045

45. 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

46. 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

47. J. Hartmanis, Computers and intractability: a guide to the theory of NP-completeness, *SIAM Rev.,* **24** (1982), 90. https://doi.org/10.1137/1024022

48. 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

49. 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

50. 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

51. D. Shabtay, D. Oron, Proportionate flow-shop scheduling with rejection, *J. Oper. Res. Soc.,* **67** (2016), 752–769. https://doi.org/10.1057/jors.2015.95

52. L. Pan, X. Y. Sun, X. N. Geng, Dynamic programming for proportionate flow-shop scheduling with due-date assignment, *Asia-Pacific J. Oper. Res.,* (2026), 2650002. https://doi.org/10.1142/S0217595926500028

53. R. Jin, W. Tong, Y. Xu, T. Dong, Two-stage flexible flow shop scheduling in GPU systems, *J. Oper. Res. Soc.,* **77** (2026), 793–804. https://doi.org/10.1080/01605682.2025.2504435

54. J. Y. Wang, Minimizing the total weighted tardiness of overlapping jobs on parallel machines with a learning effect, *J. Oper. Res. Soc.,* **71** (2020), 910–927. https://doi.org/10.1080/01605682.2019.1590511

55. K. Ben Abdellafou, K. Zidi, W. Ghaban, Scheduling intrees with unavailability constraints on two parallel machines, *Symmetry,* **18** (2026), 103. https://doi.org/10.3390/sym18010103

56. C. Hu, Y. Zhai, S. Lu, X. Liu, Parallel machine group scheduling with time-dependent setup time and combined effects of deterioration and learning, *Memet. Comput.,* **18** (2026), 8. https://doi.org/10.1007/s12293-025-00487-x

AIMS Press