



Research article

Single-machine linear resource-dependent consumption scheduling with DeJong's time-dependent learning effect

Kai-Fang Zhang, Jia-Fan Zhang and Ji-Bo Wang*

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

* **Correspondence:** Email: wangjibo75@163.com.

Abstract: This article addresses a single-machine scheduling problem with resource-consumption and DeJong's time-dependent learning effect simultaneously. Under linear resource-consumption, the objective is to find the optimal sequence and optimal resource-consumption such that the weighted sum of makespan, total completion time, and total resource-consumption cost is minimized. To solve the problem, some optimal properties are given, then some solution algorithms (including a branch-and-bound algorithm and some heuristic algorithms) are proposed.

Keywords: scheduling; single-machine; branch-and-bound; linear resource-consumption; learning effect

Mathematics Subject Classification: 90B35

1. Introduction

In many real-world production processes, the processing times of jobs (tasks or products) tend to decrease due to learning effects and/or resource allocation. Learning effects play a crucial role when workers perform repetitive tasks in scheduling environments (see Jiang et al. [1]; Azzouz et al. [2]; Wu et al. [3]; Liao et al. [4]; Liu et al. [5]; Sun et al. [6]; Zhao [7]; Liu and Wang [8]; Lv and Wang [9]; Yin and Gao [10]; Gao et al. [11]; Lv and Wang [12]; Wang and Liu [13]; Yin et al. [14]; Song et al. [15]; Wang [16]). In resource-dependent scheduling, the actual processing times are controllable by allocating additional amounts of resources such as fuel, manpower, or energy (see Nowicki and Zdrzalka [17]; Guan et al. [18]; Shabtay and Steiner [19]; Oron [20]; Yin et al. [21]; Wang et al. [22]; Gao et al. [23]; Ren and Yang [24]; Geng and Yuan [25]; Geurtsen et al. [26]; Pang and Meng [27]; Li et al. [28]; Sun et al. [29]; Wang and Liu [30]; Wang et al. [31]; Weng et al. [32]; Lv and Wang [33]; Wang et al. [34]; Zhang and Wang [35]; Zhang et al. [36]; Zhang et al. [37]; Zhang and Yin [38]). Some studies have addressed scheduling problems that simultaneously consider learning effects and resource allocation (see Wang et al. [39]; Lu et al. [40]; Yin [41]). More recently, Zhao [42] investigated due-window

assignment in flow shop scheduling with learning effects and resource allocation. Wang et al. [43] studied a single-machine resource allocation scheduling problem with truncated learning effects and provided a bicriteria analysis for the total weighted completion time and the total resource consumption cost. Qian et al. [44] and Qian and Guo [45] considered common due-window assignment scheduling with learning effects and resource allocations under delivery time constraints, and proved that some problem variants are polynomially solvable. Zhang et al. [46] examined resource allocation scheduling with exponential time-dependent learning effects. Huang et al. [47] focused on group scheduling with simultaneous learning effects and convex resource allocations. Sun et al. [48] studied a no-wait flow shop with learning effects and resource allocation under due-window assignment and demonstrated the polynomial solvability of certain cases. Lu and Li [49] considered resource allocation scheduling problems with learning and deteriorating effects. Wu et al. [50] studied resource allocation scheduling with time-dependent learning effects.

In the years 2025–2026, research on learning-based resource allocation scheduling has achieved significant advancements in three dimensions: model formulation, scheduling environment, and solution algorithms. With respect to learning effects, Zhang et al. [46] overcame the limitation of traditional log-linear or power-function forms by proposing an exponential time-dependent learning effect. Under a controllable resource input, they derived an analytical solution for the optimal resource allocation and proved polynomial solvability in several cases. Huang et al. [47] considered single-machine group scheduling with learning effects and convex resource consumption function. Sun et al. [48] introduced the position-dependent learning effect and convex resource allocation into the no-wait proportionate flow shop problem.

Regarding resource consumption functions, the linear compression model (i.e., the actual processing time of J_g is $Q_g = a_g - b_g u_g$, where a_g is the normal processing time of J_g , u_g is the resource allocated to J_g , and b_g is the positive compression rate of J_g) remains popular for its simplicity, but nonlinear forms are gaining attention. Wang et al. [34] introduced the convex resource function model into the due-window assignment. Zhang et al. [36] proposed an additive resource allocation model and applied it to multiple parallel-batch machines with different available time windows. Meanwhile, the scheduling environment has rapidly expanded from the classical single-machine to group scheduling, proportionate and no-wait flow shops, distributed multi-agent systems, and comprehensive settings with various practical constraints. Bai et al. [51] studied a distributed two-agent flow shop where two competing agents share a production line, each possessing independent job sets and objectives; the learning effect can spill over between agents, and resource allocation becomes a bi-level optimization problem involving both game-theoretic and cooperative aspects. Kong et al. [52] and Lu et al. [53] respectively investigated single-machine scheduling with deteriorating jobs and delivery times; the former minimizes the maximum delivery completion time, while the latter minimizes total earliness–tardiness penalties, with job processing times deteriorating linearly as their start times increase. In the context of green manufacturing, Yin et al. [14] incorporated carbon emission costs into the objective function, simultaneously optimizing completion time, resource consumption, and carbon emissions in group scheduling, and built a learning–resource integration model aimed at minimizing the total green cost. Furthermore, Zhang et al. [37] employed the minimax earliness–tardiness criterion with a common due-window in group scheduling, and Lv and Wang [33] investigated convex resource allocation under different due-window assignment strategies, further enriching the problem spectrum and optimization objectives in this field.

Despite these advances, none of the above studies integrates the DeJong time-dependent learning effect with a linear resource consumption model on a single machine. The DeJong model, characterized by the convex combination parameter $M \in [0, 1]$, offers a flexible mechanism that interpolates between no learning ($M = 1$) and pure cumulative-time learning ($M = 0$). In this paper, we fill this gap by considering the problem with linear resource-consumption and DeJong's time-dependent learning effect. We derive optimal resource allocation rules, sequencing properties, and design a branch-and-bound algorithm with two lower bounds, complemented by tabu search and simulated annealing heuristics. Our computational experiments show that the simulated annealing algorithm consistently produces near-optimal solutions within seconds for instances up to 300 jobs, thereby providing an effective tool for this novel problem class. The related studies and highlighting differences between them are presented in Table 1.

The paper is organized as follows: Section 2 formulates the problem. Section 3 presents the basic properties of the problem. Section 4 proposes the branch-and-bound algorithm to solve the problem. Section 5 presents the tabu search algorithm and simulated annealing algorithm to solve the problem. Section 6 gives the computational experiments. Section 7 presents the conclusions.

Table 1. Comparative literature review on learning effects and resource allocation.

Literature	Learning Effect & Resource Consumption Model	Environment	Objective	Identified gap
Wang et al. [39]	Log-linear learning; Linear compression	Single machine	$\sum C_j + \text{resource cost}$	No DeJong time-dependent learning; learning and resource are independent
Lu et al. [40]	Log-linear learning; Convex function	Single machine group	$C_{\max} + \text{resource cost}$	Position-dependent only; no cumulative processing-time learning
Yin [41]	Job-dependent; Linear compression	Single machine + due-window	Earliness/tardiness + resource cost	No DeJong learning; separate optimization
Zhao [42]	Position-dependent learning; Linear compression	Flow shop	Earliness/ tardiness + window cost+ resource cost	Restricted to two-machine flow shop; no time-dependent learning
Wang et al. [43]	Truncated log-linear; Convex function	Single machine	Bi-criteria: $\sum C_j + \text{resource cost}$	No time-dependent learning; no weighted single-objective
Qian et al. [44]	Job-dependent; Linear compression	Single machine + due-window	Earliness/tardiness + resource cost	No DeJong time-dependent learning; no weighted time cost
Zhang et al. [46]	Exponential time-dependent learning; Convex function	Single machine	$\sum C_j + \text{resource cost}$	Exponential form, not DeJong; no M parameter
Huang et al. [47]	Log-linear learning; Convex function	Group scheduling	Completion time + resource cost	No DeJong structure; non-single-machine; nonlinear resource
Sun et al. [48]	Position-dependent; Linear compression	Two-machine no-wait flow shop	Earliness/tardiness + window + resource	Multi-machine; no time-dependent learning; no DeJong
This study	DeJong time-dependent learning; Linear compression	Single machine	Weighted sum: $\alpha C_{\max} + \beta \sum C_j + \gamma \sum v_g u_g$	First integration of DeJong time-dependent learning and linear resource consumption on a single machine

2. Problem description

2.1. Problem statement

A set $M = \{J_1, J_2, \dots, J_n\}$ is to be processed on a single-machine, and the jobs ready for processing at time 0. If J_g is scheduled at the h th position, Biskup [54] proposed the following model, i.e., a log-linear learning effect, that is, the actual processing time of J_g is $Q_{gh} = a_g h^\xi$, where a_g is the normal processing time of J_g , and $\xi \leq 0$ is a learning factor. Kuo and Yang [55] proposed the following model, i.e., a time-dependent learning effect: $Q_{gh} = a_g \left(1 + \sum_{l=1}^{h-1} a_{<l>} \right)^\varrho$, where $\varrho \leq 0$ is a learning factor, and $J_{<l>}$ denotes some job scheduled at the l th position. Ji et al. [56] and Zhang et al. [57] proposed DeJong's learning effect, i.e., $Q_{gh} = a_g \left(M + (1 - M)h^\xi\right)$, where $0 \leq M \leq 1$ is a given constant. Wang et al. [39] studied the following model: $Q_{gh} = a_g h^\xi - b_g u_g$, where u_g is the resource allocated to J_g , and b_g is the positive compression rate of J_g . From Wang et al. [39], Kuo and Yang [55], Ji et al. [56], and Zhang et al. [57], we consider DeJong's time-dependent learning effect, i.e.,

$$Q_{gh} = a_g \left(M + (1 - M) \left(1 + \sum_{l=1}^{h-1} a_{<l>} \right)^\varrho \right) - b_g u_g, \quad (1)$$

where $0 \leq u_g \leq \bar{u}_g \leq \frac{a_g(1+P-a_{\max})^\varrho}{b_g}$, $P = \sum_{g=1}^n a_g$, $a_{\max} = \max\{a_g, g = 1, 2, \dots, n\}$, and $\bar{u}_g$ is the upper bound on the resource u_g .

For a given sequence κ , let $C_g = C_g(\kappa)$ be the completion time. In this paper, we focus on the optimal sequence and resource allocation for minimizing

$$G = \alpha C_{\max} + \beta \sum_{g=1}^n C_g + \gamma \sum_{g=1}^n v_g u_g, \quad (2)$$

where $\alpha \geq 0, \beta \geq 0$, and $\gamma \geq 0$ are given constants, $C_{\max} = C_{<n>}$ is the makespan, $\sum_{g=1}^n C_g$ is the total completion time, and v_g is the cost of allocating unit resource to J_g . As in Pinedo [58] and Agnetis et al. [59], using standard scheduling notation, our problem is denoted as

$$1 \left| Q_{gh} = a_g \left(M + (1 - M) \left(1 + \sum_{l=1}^{h-1} a_{<l>} \right)^\varrho \right) - b_g u_g \right| \alpha C_{\max} + \beta \sum_{g=1}^n C_g + \gamma \sum_{g=1}^n v_g u_g. \quad (3)$$

where “1” denotes the single-machine setting, “ $Q_{gh} = a_g \left(M + (1 - M) \left(1 + \sum_{l=1}^{h-1} a_{<l>} \right)^\varrho \right) - b_g u_g$ ” denotes the job characteristic, and “ $\alpha C_{\max} + \beta \sum_{g=1}^n C_g + \gamma \sum_{g=1}^n v_g u_g$ ” denotes the objective function. The comprehensive notations are provided in Table 2.

Table 2. Notation definitions of the scheduling model.

Symbol	Description
Indices	
g	Job index, $g = 1, 2, \dots, n$.
h	Position index in the processing sequence, $h = 1, 2, \dots, n$.
$\langle l \rangle$	The job scheduled at the l th position in a sequence.
Parameters	
n	Total number of jobs.
a_g	Normal (standard) processing time of job J_g ($a_g > 0$).
b_g	Positive compression rate of job J_g ($b_g > 0$).
v_g	Unit resource cost of job J_g .
$\bar{u}_g$	Upper bound on the resource allocated to job J_g , $\bar{u}_g \leq \frac{a_g(1+P-a_{\max})^\rho}{b_g}$.
P	Sum of all normal processing times, $P = \sum_{g=1}^n a_g$.
$a_{\max}$	Maximum normal processing time, $a_{\max} = \max\{a_g \mid g = 1, \dots, n\}$.
M	Incompressibility factor in DeJong's learning model ($0 \leq M \leq 1$).
ϱ	Time-dependent learning index ($\varrho \leq 0$).
α, β, γ	Non-negative weight coefficients in the objective function.
Decision Variables	
u_g	Amount of resource allocated to job J_g , with $0 \leq u_g \leq \bar{u}_g$.
κ	A complete sequence, $\kappa = (J_{\langle 1 \rangle}, J_{\langle 2 \rangle}, \dots, J_{\langle n \rangle})$.
Derived Quantities	
Q_{gh}	Actual processing time of job J_g scheduled at position h , defined in Eq. (1).
C_g	Completion time of job J_g under a given sequence.
$C_{\max}$	Makespan (maximum completion time), $C_{\max} = C_{\langle n \rangle}$.
$\sum_{g=1}^n C_g$	Total completion time of all jobs.
Objective Function	
G	Minimized total cost: $G = \alpha C_{\max} + \beta \sum_{g=1}^n C_g + \gamma \sum_{g=1}^n v_g u_g$.

2.2. An example

Consider a single-machine scheduling problem with three jobs $\{J_1, J_2, J_3\}$. The parameters are set as follows:

- Normal processing times: $a_1 = 2, a_2 = 3, a_3 = 1.5$
- Unit resource costs: $v_1 = 5, v_2 = 8, v_3 = 6$
- Positive compression rates: $b_1 = 0.5, b_2 = 0.7, b_3 = 0.4$
- Upper bounds on resource allocation: $\bar{u}_1 = 4, \bar{u}_2 = 2, \bar{u}_3 = 5$
- DeJong's learning effect parameters: $M = 0.5, \varrho = -0.2$
- Objective weights: $\alpha = 1, \beta = 1, \gamma = 1$

Suppose the jobs are processed in the order (J_1, J_2, J_3) . According to the actual processing time model in Eq (1),

$$Q_{[1]} = a_1[M + (1 - M)(1 + 0)^\varrho] - b_1 u_1 = 2[0.5 + 0.5 \times 1] - 0.5 u_1 = 2 - 0.5 u_1,$$

$$Q_{[2]} = a_2 \left[M + (1 - M)(1 + a_1)^g \right] - b_2 u_2 = 3[0.5 + 0.5(3)^{-0.2}] - 0.7u_2 \approx 2.704 - 0.7u_2,$$

$$Q_{[3]} = a_3 \left[M + (1 - M)(1 + a_1 + a_2)^g \right] - b_3 u_3 = 1.5[0.5 + 0.5(6)^{-0.2}] - 0.4u_3 \approx 1.274 - 0.4u_3.$$

If no resource is allocated, i.e., $u_1 = u_2 = u_3 = 0$, the completion times are

$$C_1 = 2, \quad C_2 \approx 4.704, \quad C_{\max} = C_3 \approx 5.978,$$

and the total completion time is $\sum C_g \approx 12.682$. The resulting objective value is

$$G = \alpha C_{\max} + \beta \sum_{g=1}^3 C_g + \gamma \sum_{g=1}^3 v_g u_g \approx 1 \times 5.978 + 1 \times 12.682 + 0 = 18.660.$$

This example illustrates how the resource allocation and the DeJong's time-dependent learning effect interact to determine the optimal schedule.

3. Basic properties

Lemma 1. For a given sequence of

$$1 \left| Q_{gh} = a_g \left(M + (1 - M) \left(1 + \sum_{l=1}^{h-1} a_{<l>} \right)^g \right) - b_g u_g \right| \alpha C_{\max} + \beta \sum_{g=1}^n C_g + \gamma \sum_{g=1}^n v_g u_g,$$

the optimal resource allocation is

$$u_{<g>}^* = \begin{cases} 0, & \text{if } \gamma v_{<g>} - [\alpha + \beta(n - g + 1)]b_{<g>} > 0, \\ u_{<g>} \in [0, \bar{u}_{<g>}], & \text{if } \gamma v_{<g>} - [\alpha + \beta(n - g + 1)]b_{<g>} = 0, \\ \bar{u}_{<g>}, & \text{if } \gamma v_{<g>} - [\alpha + \beta(n - g + 1)]b_{<g>} < 0. \end{cases} \quad (4)$$

Proof. Obviously, $C_{\max} = \sum_{g=1}^n Q_{<g>}^A$, $\sum_{g=1}^n C_g = \sum_{g=1}^n (n - g + 1)Q_{<g>}^A$. For a given sequence $\kappa = (J_{<1>}, J_{<2>}, \dots, J_{<n>})$, it follows that

$$\begin{aligned} G(u_{<g>}) &= \alpha C_{\max} + \beta \sum_{g=1}^n C_g + \gamma \sum_{g=1}^n v_g u_g \\ &= \alpha C_{<n>} + \beta \sum_{g=1}^n C_{<g>} + \gamma \sum_{g=1}^n v_{<g>} u_{<g>} \\ &= \sum_{g=1}^n [\alpha + \beta(n - g + 1)] Q_{<g>}^A + \gamma \sum_{g=1}^n v_{<g>} u_{<g>} \\ &= \sum_{g=1}^n [\alpha + \beta(n - g + 1)] \left[a_{<g>} \left(M + (1 - M) \left(1 + \sum_{l=1}^{g-1} a_{<l>} \right)^g \right) - b_{<g>} u_{<g>} \right] \\ &\quad + \gamma \sum_{g=1}^n v_{<g>} u_{<g>} \end{aligned}$$

$$\begin{aligned}
&= \sum_{g=1}^n [\alpha + \beta(n - g + 1)] a_{<g>} \left(M + (1 - M) \left(1 + \sum_{l=1}^{g-1} a_{<l>} \right)^g \right) \\
&\quad + \sum_{g=1}^n \left\{ \gamma v_{<g>} - [\alpha + \beta(n - g + 1)] b_{<g>} \right\} u_{<g>}.
\end{aligned} \tag{5}$$

Taking the derivative for (5), it follows that

$$\frac{\partial G(u_{<g>})}{\partial u_{<g>}} = \gamma v_{<g>} - [\alpha + \beta(n - g + 1)] b_{<g>}. \tag{6}$$

From (6), if $\gamma v_{<g>} - [\alpha + \beta(n - g + 1)] b_{<g>} > 0$, then $u_{<g>}^* = 0$; similarly, the other cases of (4) hold. $\square$

Lemma 2. The term $\sum_{g=1}^n [\alpha + \beta(n - g + 1)] a_{<g>} \left(M + (1 - M) \left(1 + \sum_{l=1}^{g-1} a_{<l>} \right)^g \right)$ is minimized by arranging the non-decreasing order of a_g .

Proof. Let $\kappa = (J_{<1>}, J_{<2>}, \dots, J_i, J_j, \dots, J_{<n>})$ and $\kappa' = (J_{<1>}, J_{<2>}, \dots, J_j, J_i, \dots, J_{<n>})$, where $a_i \leq a_j$ and J_i is scheduled at the h th position in κ .

$$\begin{aligned}
&[\alpha + \beta(n - h + 1)] a_i \left(M + (1 - M) \left(1 + \sum_{l=1}^{h-1} a_{<l>} \right)^g \right) \\
&+ [\alpha + \beta(n - (h + 1) + 1)] a_j \left(M + (1 - M) \left(1 + \sum_{l=1}^{h-1} a_{<l>} + a_i \right)^g \right) \\
&- [\alpha + \beta(n - h + 1)] a_j \left(M + (1 - M) \left(1 + \sum_{l=1}^{h-1} a_{<l>} \right)^g \right) \\
&- [\alpha + \beta(n - (h + 1) + 1)] a_i \left(M + (1 - M) \left(1 + \sum_{l=1}^{h-1} a_{<l>} + a_j \right)^g \right) \\
&= \beta(a_i - a_j) \left(M + (1 - M) \left(1 + \sum_{l=1}^{h-1} a_{<l>} \right)^g \right) \\
&\quad + [\alpha + \beta(n - h)] (1 - M) \left(1 + \sum_{l=1}^{h-1} a_{<l>} \right)^g \\
&\quad \times \left(a_i - a_j + a_j \left(1 + \frac{a_i}{\left(1 + \sum_{l=1}^{h-1} a_{<l>} \right)} \right)^g - a_i \left(1 + \frac{a_j}{\left(1 + \sum_{l=1}^{h-1} a_{<l>} \right)} \right)^g \right) \\
&\leq 0,
\end{aligned}$$

where

$$\left(a_i - a_j + a_j \left(1 + \frac{a_i}{\left(1 + \sum_{l=1}^{h-1} a_{<l>} \right)} \right)^g - a_i \left(1 + \frac{a_j}{\left(1 + \sum_{l=1}^{h-1} a_{<l>} \right)} \right)^g \right) \leq 0$$

can be obtained by Kuo and Yang [55] and $a_i \leq a_j$. $\square$

Lemma 3. Let $x_{g,h} = 1$ if job J_g is scheduled in the h th position, and $x_{g,h} = 0$, otherwise, the term $\sum_{g=1}^n \left\{ \gamma v_{<g>} - [\alpha + \beta(n - g + 1)] b_{<g>} \right\} u_{<g>}$ is minimized by the assignment problem (AP_G):

$$\min \sum_{g=1}^n \sum_{h=1}^n \lambda_{g,h} x_{g,h} \quad (7)$$

subject to

$$\sum_{g=1}^n x_{g,h} = 1, \quad h = 1, 2, \dots, n, \quad (8)$$

$$\sum_{h=1}^n x_{g,h} = 1, \quad g = 1, 2, \dots, n, \quad (9)$$

$$x_{g,h} = 0 \text{ or } 1, \quad g, h = 1, 2, \dots, n, \quad (10)$$

where

$$\lambda_{g,h} = \begin{cases} 0, & \text{if } \gamma v_g - [\alpha + \beta(n - h + 1)] b_g \geq 0, \\ \left\{ \gamma v_g - [\alpha + \beta(n - h + 1)] b_g \right\} \bar{u}_g, & \text{if } \gamma v_g - [\alpha + \beta(n - h + 1)] b_g < 0. \end{cases} \quad (11)$$

Proof. See the process of Wang et al. [43]. $\square$

Lemma 4. If $a_g = a$, then an optimal sequence of

$$1 \left| Q_{gh} = a_g \left(M + (1 - M) \left(1 + \sum_{l=1}^{h-1} a_{<l>} \right)^e \right) - b_g u_g \right| \alpha C_{\max} + \beta \sum_{g=1}^n C_g + \gamma \sum_{g=1}^n v_g u_g$$

is obtained by the AP_G :

$$\min \sum_{g=1}^n \sum_{h=1}^n \vartheta_{g,h} x_{g,h} \quad (12)$$

subject to (8), (9), (10),

where

$$\vartheta_{g,h} = \begin{cases} a[\alpha + \beta(n - h + 1)] [M + (1 - M) (1 + (h - 1)a)^e], & \text{if } \gamma v_g - [\alpha + \beta(n - h + 1)] b_g \geq 0, \\ a[\alpha + \beta(n - h + 1)] [M + (1 - M) (1 + (h - 1)a)^e] \\ + \left\{ \gamma v_g - [\alpha + \beta(n - h + 1)] b_g \right\} \bar{u}_g, & \text{if } \gamma v_g - [\alpha + \beta(n - h + 1)] b_g < 0. \end{cases} \quad (13)$$

Proof. See Lemma 3. $\square$

4. Branch-and-bound algorithm

4.1. Lower bound

Let $\kappa = (\kappa_{seq}, \kappa_{unseq})$ be a sequence where κ_{seq} (resp. κ_{unseq}) is the sequenced (resp. un-sequenced) part, and there are q jobs in κ_{seq} . From (5), it follows that

$$\begin{aligned}
 G(\kappa_{seq}, \kappa_{unseq}) = & \sum_{g=1}^q [\alpha + \beta(n - g + 1)] a_{<g>} \left(M + (1 - M) \left(1 + \sum_{l=1}^{g-1} a_{<l>} \right)^g \right) \\
 & + \sum_{g=1}^q \left\{ \gamma v_{<g>} - [\alpha + \beta(n - g + 1)] b_{<g>} \right\} u_{<g>} \\
 & + \sum_{g=q+1}^n [\alpha + \beta(n - g + 1)] a_{<g>} \left(M + (1 - M) \left(1 + \sum_{l=1}^q a_{<l>} + \sum_{l=1}^{g-1} a_{<l>} \right)^g \right) \\
 & + \sum_{g=q+1}^n \left\{ \gamma v_{<g>} - [\alpha + \beta(n - g + 1)] b_{<g>} \right\} u_{<g>}. \quad (14)
 \end{aligned}$$

From (14), the terms $\sum_{g=1}^q [\alpha + \beta(n - g + 1)] a_{<g>} \left(M + (1 - M) \left(1 + \sum_{l=1}^{g-1} a_{<l>} \right)^g \right)$, $\sum_{l=1}^q a_{<l>}$, and $\sum_{g=1}^q \left\{ \gamma v_{<g>} - [\alpha + \beta(n - g + 1)] b_{<g>} \right\} u_{<g>}$ are known. The term $\sum_{g=q+1}^n [\alpha + \beta(n - g + 1)] a_{<g>} \left(M + (1 - M) \left(1 + \sum_{l=1}^q a_{<l>} + \sum_{l=1}^{g-1} a_{<l>} \right)^g \right)$ is minimized by Lemma 2. The term $\sum_{g=q+1}^n \left\{ \gamma v_{<g>} - [\alpha + \beta(n - g + 1)] b_{<g>} \right\} u_{<g>}$ is minimized by Lemma 3. Hence, the lower bound of

$$1 \left| Q_{gh} = a_g \left(M + (1 - M) \left(1 + \sum_{l=1}^{h-1} a_{<l>} \right)^g \right) - b_g u_g \right| \alpha C_{\max} + \beta \sum_{g=1}^n C_g + \gamma \sum_{g=1}^n v_g u_g$$

is

$$\begin{aligned}
 LB'_G = & \sum_{g=1}^q [\alpha + \beta(n - g + 1)] a_{<g>} \left(M + (1 - M) \left(1 + \sum_{l=1}^{g-1} a_{<l>} \right)^g \right) \\
 & + \sum_{g=1}^q \left\{ \gamma v_{<g>} - [\alpha + \beta(n - g + 1)] b_{<g>} \right\} u_{<g>} \\
 & + \sum_{g=q+1}^n [\alpha + \beta(n - g + 1)] a_{(g)} \left(M + (1 - M) \left(1 + \sum_{l=1}^q a_{<l>} + \sum_{l=q+1}^{g-1} a_{(l)} \right)^g \right) \\
 & + \sum_{g=q+1}^n \left\{ \gamma v_{<g>}^* - [\alpha + \beta(n - g + 1)] b_{<g>}^* \right\} u_{<g>}^*, \quad (15)
 \end{aligned}$$

where $a_{(q+1)} \leq a_{(q+2)} \leq \dots \leq a_{(n)}$, $v_{<g>}^*$, $b_{<g>}^*$, and $u_{<g>}^*$ are the optimal solutions by Lemmas 1 and 3.

Similarly, let $a_{\min} = \min\{a_g | g \in \kappa_{unseq}\}$, and from Lemma 4, we have another lower bound:

$$LB''_G = \sum_{g=1}^q [\alpha + \beta(n - g + 1)] a_{<g>} \left(M + (1 - M) \left(1 + \sum_{l=1}^{g-1} a_{<l>} \right)^g \right)$$

$$\begin{aligned}
& + \sum_{g=1}^q \left\{ \gamma v_{<g>} - [\alpha + \beta(n - g + 1)] b_{<g>} \right\} u_{<g>} \\
& + \sum_{g=q+1}^n [\alpha + \beta(n - g + 1)] a_{\min} \left(M + (1 - M) \left(1 + \sum_{l=1}^q a_{<l>} + (g - 1 - q) a_{\min} \right)^{\varrho} \right) \\
& + \sum_{g=q+1}^n \left\{ \gamma v_{<g>}^{**} - [\alpha + \beta(n - g + 1)] b_{<g>}^{**} \right\} u_{<g>}^{**}, \tag{16}
\end{aligned}$$

where $v_{<g>}^{**}$, $b_{<g>}^{**}$, and $u_{<g>}^{**}$ are the optimal solutions by Lemmas 1 and 4.

Combining Eqs (15) and (16), the lower bound of

$$1 \left| Q_{gh} = a_g \left(M + (1 - M) \left(1 + \sum_{l=1}^{h-1} a_{<l>} \right)^{\varrho} \right) - b_g u_g \right| \alpha C_{\max} + \beta \sum_{g=1}^n C_g + \gamma \sum_{g=1}^n v_g u_g$$

is

$$LB_G = \max\{LB'_G, LB''_G\}. \tag{17}$$

4.2. Upper bound

From Section 3 and Nawaz et al. [60], the following heuristic algorithm is proposed as an upper bound ($\widetilde{UB}_G$) for

$$1 \left| Q_{gh} = a_g \left(M + (1 - M) \left(1 + \sum_{l=1}^{h-1} a_{<l>} \right)^{\varrho} \right) - b_g u_g \right| \alpha C_{\max} + \beta \sum_{g=1}^n C_g + \gamma \sum_{g=1}^n v_g u_g.$$

Algorithm 1 ($\widetilde{UB}_G$)

Input: Learning factor ϱ , $\alpha = \beta = \gamma = 1$, M , a_g , b_g , v_g , $\bar{u}_g$ for $g = 1, 2, \dots, n$.

Output: An approximate sequence κ .

Step 1.

Step 1.1. Arrange the jobs by the non-decreasing order of a_g .

Step 1.2. Arrange the jobs by the AP_G (7)–(11).

Step 1.3. Arrange the jobs by the AP_G (12), (8)–(10).

Step 2. Choose a sequence with the minimal value (see (5), and corresponding resources are given by Lemma 1) from Steps 1.1–1.3.

Step 3. Pick the two jobs from the first and second positions of the list of Step 2, and find the best sequence for these two jobs by calculating the objective cost (see (5), and corresponding resources are given by Lemma 1) for the two possible sequences. Do not change the relative positions of these two jobs with respect to each other in the remaining steps of the algorithm. Set $r = 3$.

Step 4. Pick the job in the r th position of the list generated in Step 2 and find the best schedule by placing it at all possible r positions in the partial sequence found in the previous step, without changing the positions relative to each other of the already assigned jobs. The number of enumerations at this step equals r .

Step 5. If $n = r$, STOP, otherwise set $r = r + 1$ and go to Step 4.

From LB_G and $\widetilde{UB}_G$, the normal branch-and-bound method ($\widetilde{BB}_G$) can be proposed, where an initial $\widetilde{UB}_G$ is obtained by Algorithm 1, LB_G is obtained by (17), and the depth-first search strategy is used.

5. Complex algorithms

In addition, the tabu search ($\widetilde{TS}$) algorithm and simulated annealing ($\widetilde{SA}$) algorithm can be proposed for

$$1 \left| Q_{gh} = a_g \left(M + (1 - M) \left(1 + \sum_{l=1}^{h-1} a_{<l>} \right)^{\varrho} \right) - b_g u_g \right| \alpha C_{\max} + \beta \sum_{g=1}^n C_g + \gamma \sum_{g=1}^n v_g u_g.$$

The initial schedule used in the $\widetilde{TS}$ algorithm and $\widetilde{SA}$ algorithm is chosen by Algorithm 1, the maximum number of iterations for the $\widetilde{TS}$ algorithm is set at $500n$, and the $\widetilde{SA}$ algorithm is set at $500n$.

5.1. Simulated annealing algorithm

This section provides a detailed account of the implementation of the $\widetilde{SA}$ algorithm. As a meta-heuristic algorithm inspired by the annealing process in metallurgy, $\widetilde{SA}$ possesses robust global search capabilities and the ability to find high-quality optimal solutions.

Algorithm 2 ($\widetilde{SA}$)

Input: Maximum iterations $I_{\max} = 500n$, learning factor ϱ , $\alpha = \beta = \gamma = 1$, M , a_g , b_g , v_g , $\bar{u}_g$ for $g = 1, 2, \dots, n$.

Output: An approximate sequence κ .

Step 1. First, call Algorithm 1 ($\widetilde{UB}_G$ Algorithm) to determine an initial job sequence. Let $\widehat{F}$ be the objective function.

Step 2. On the basis of the current solution, generate a new neighborhood solution by randomly performing a pairwise swap operation (i.e., randomly select two different jobs in the sequence and swap their positions).

Step 3. Acceptance probability: When a new schedule is generated, it is accepted if its objective value is smaller than that of the original schedule; otherwise, it is accepted with some probability that decreases as the process evolves. The probability of acceptance is generated from an exponential distribution,

$$P(\text{accept}) = \exp(-\alpha \times \Delta \widehat{F} / L),$$

where α is a control parameter and $\Delta \widehat{F} / L$ is the change in $\widehat{F}$. In addition, the method was adopted to change α in the k th iteration as follows:

$$\alpha = \frac{k}{\delta}$$

where δ is an experimental constant. After preliminary trials, $\delta = 1$ was used in our experiments. If $\widehat{F}$ increases as a result of a random pairwise interchange, the new schedule is accepted when $P(\text{accept}) > \theta$, where θ is randomly sampled from the uniform distribution $U(0, 1)$.

Step 4. The algorithm terminates when the total number of iterations reaches $500 \times n$ or the temperature drops to a preset final temperature.

5.2. Tabu search algorithm

In this subsection, the $\widetilde{TS}$ algorithm is used to find a near-optimal solution. The initial sequence used in the $\widetilde{TS}$ algorithm is chosen by arranging the jobs by the algorithms $\widetilde{UB}_G$, and the maximum number of iterations for the $\widetilde{TS}$ algorithm is set at $500n$. The implementation of the $\widetilde{TS}$ algorithm is given below.

Algorithm 3 ($\widetilde{TS}$)

Input: Maximum iterations $I_{max} = 500n$, learning factor ϱ , $\alpha = \beta = \gamma = 1$, M , a_g , b_g , v_g , $\bar{u}_g$ for $g = 1, 2, \dots, n$.

Output: An approximate sequence κ .

Step 1. Let the tabu list be empty and the iteration number be zero.

Step 2. Set the initial sequence of the $\widetilde{TS}$ algorithm by using the algorithms $\widetilde{UB}_G$, calculate its objective function, and set the current schedule as the best solution ϱ^* .

Step 3. Search the associated neighborhood of the current schedule and resolve if there is a schedule ϱ^{**} with the smallest objective function in associated neighborhoods and it is not in the tabu list.

Step 4. If ϱ^{**} is better than ϱ^* , then let $\varrho^* = \varrho^{**}$. Update the tabu list and the iteration number.

Step 5. If there is not a schedule in associated neighborhoods but it is not in the tabu list or the maximum number of iterations is reached, then output the local optimal sequence ϱ and objective function value. Otherwise, update the tabu list and go to Step 3.

6. Computational experiments and results

This section tests the accuracy and efficiency of the proposed algorithms $\widetilde{UB}_G$ (Algorithm 1), $\widetilde{SA}$, $\widetilde{TS}$, and $\widetilde{BB}_G$. We coded these algorithms in PyCharm 408 2025.2.1.1 and use the Python 3.12.10 interpreter and ran the computational experiments on a 409 desktop work station with a CPU with an Intel(R) Core(TM) i7-10710U CPU @ 1.10 GHz, 16.00 410 GB RAM.

We assume that the following parameters are given:

- (1) $\alpha = \beta = \gamma = 1$;
- (2) $\varrho = -0.2, -0.3, -0.4, -0.5$;
- (3) $M = 0.2, 0.5, 0.8$;
- (4) a_g and v_g are uniformly distributed over $[1, 100]$ respectively;
- (5) b_g and $\bar{u}_g$ are uniformly distributed over $[1, 30]$ respectively (such that $Q_{gh} > 0$);
- (6) $n = 9, 10, 11, 12, 13$ (for small-scale instances and global optimum can be obtained by the $\widetilde{BB}_G$).
- (7) $n = 100, 150, 200, 250, 300$ (for larger-scale instances and $\widetilde{BB}_G$ was disabled).

For (1), the symmetric selection of weight coefficients is to establish a benchmark model. For (2), the multiple-level values of the learning effect parameter ϱ are intended to systematically examine the robustness of the algorithm under different learning environments. For (3), the multi-level settings of the learning effect parameter M are intended to systematically examine the robustness of the algorithm under different learning environments. For (4) and (5), the uniform distribution range and boundary settings of the job parameters are meant to ensure that the model constraints (positive processing time) are met and to simulate a variety of scenarios. For (6) and (7), the division of problem scale is for the

different purposes of accurately evaluating the algorithm's accuracy (small scale) and demonstrating the algorithm's practical application ability (large scale).

To evaluate the performance of algorithms including the $Heu \in \{\widetilde{UB}_G, \widetilde{SA}, \widetilde{TS}\}$ and $\widetilde{BB}_G$ algorithms for small-scale problems, the random test instance generation mechanism in this study is as follows. For learning effect parameters, the time-dependent learning factor ϱ takes four levels of -0.2 , -0.3 , -0.4 , and -0.5 , while the interpolation parameter M of DeJong's learning effect takes three levels of 0.2 , 0.5 , and 0.8 , systematically covering learning environments of varying intensities. The objective weights α , β , and γ are all fixed at 1 to form a symmetric benchmark model. The normal processing time a_g and unit resource cost v_g of each job are independently drawn from the uniform distribution $[1, 100]$ to ensure sufficient diversity among jobs. The resource compression rate b_g and resource upper bound $\bar{u}_g$ are generated from $[1, 30]$, with constraints ensuring the actual processing time remains nonnegative. The problem scales are divided into two groups: small-scale ($n = 9, 10, 11, 12, 13$) and large-scale ($n = 100, 150, 200, 250, 300$), where the former is used to accurately compare algorithm errors against the optimal solution, and the latter to test practical solving efficiency. For each parameter combination, 30 random instances are independently generated to eliminate the randomness of a single sample and ensure statistical reliability of the experimental conclusions. For branch-and-bound algorithms, average and maximum runtime (in seconds) were recorded. For heuristic algorithms, average and maximum error percentages were recorded. The calculation method for these error percentages is

$$\frac{G(Heu) - G(Opt)}{G(Opt)} 100\%$$

in which $Heu \in \{\widetilde{UB}_G, \widetilde{SA}, \widetilde{TS}\}$. For small-scale jobs experiments, Opt denotes the optimal schedule by the $\widetilde{BB}_G$ algorithm.

In small-scale instance computation experiments, Tables 3, 4, and 5 present the CPU running times (unit: seconds) of various algorithms, while Tables 6, 7, and 8 display the error rate data. As shown in Tables 3, 4, and 5, the study demonstrates that the $\widetilde{BB}_G$ algorithm can handle computational tasks involving up to 13 nodes within a reasonable time frame. However, since this problem belongs to the NP-hard category, both the execution time and the number of node traversals increase exponentially with the size of the task. The maximum execution time of the $\widetilde{BB}_G$ algorithm is approximately 1681.6960 seconds, with a peak node traversal count reaching 160,507,24 nodes. Although the algorithm can still generate optimal solutions in large-scale scenarios, its significant computational overhead poses challenges for practical applications. Regarding error rates, intelligent optimization algorithms can typically find sequences that approximate the optimal solution within shorter CPU run times. As shown in Tables 6, 7, and 8, in most cases, the performance of the $\widetilde{TS}$ and $\widetilde{SA}$ algorithms significantly outperforms that of the $\widetilde{UB}_G$ algorithm, consistently providing optimal solutions in less time. As the value of n increases, the CPU runtime of the $\widetilde{TS}$ algorithm exceeds that of both the $\widetilde{SA}$ and $\widetilde{UB}_G$ algorithms, reaching a maximum of 0.5647 seconds for the $\widetilde{TS}$ algorithm, 0.0019 seconds for the $\widetilde{UB}_G$ algorithm, and 0.2713 seconds for the $\widetilde{SA}$ algorithm, yet all algorithms' execution times remain within acceptable limits. Therefore, in small-scale experiments, the $\widetilde{TS}$ and $\widetilde{SA}$ algorithms achieve a better balance between CPU runtime and solution quality, making them more efficient.

In large-scale computational experiments ($n = 100, 150, 200, 250, 300$), Tables 9, 10, and 11 present the CPU time of each algorithm, while Tables 12, 13, and 14 display the error rates, where the error percentage is

$$\frac{G(Heu) - G(Best)}{G(Best)} 100\%$$

in which $Heu \in \{\widetilde{UB}_G, \widetilde{SA}, \widetilde{TS}\}$. For large-scale jobs experiments, $Best$ denotes the best sequence by the algorithms $\widetilde{UB}_G$, $\widetilde{SA}$, and $\widetilde{TS}$.

In computational experiments conducted on large-scale instances, Tables 9, 10, and 11 present the CPU execution times of each algorithm (units: seconds), while Tables 12, 13, and 14 display the error rate data for each algorithm. As shown in these tables, all algorithms exhibit relatively low error rates. Specifically, the error rates of the $\widetilde{TS}$ and $\widetilde{SA}$ algorithms are similar, but the $\widetilde{SA}$ algorithm generally has the lowest error rate among all algorithms; conversely, the $\widetilde{UB}_G$ algorithm has a higher error rate, often being the highest among them. However, Tables 9, 10, and 11 also indicate that the CPU execution time of the $\widetilde{TS}$ algorithm is significantly longer than that of the $\widetilde{SA}$ and $\widetilde{UB}_G$ algorithms, reaching up to 235.0246 seconds, whereas the maximum CPU runtime of the $\widetilde{SA}$ algorithm is only 51.9478 seconds; nonetheless, the CPU runtime of all algorithms remains within an acceptable range. Therefore, in large-scale experiments, both the $\widetilde{TS}$ and $\widetilde{SA}$ algorithms achieve a better balance between CPU efficiency and solution quality, making them more efficient.

Table 3. CPU time (s) of algorithms for $M = 0.2$ (Small-Scale).

		$\widetilde{BB}_G$		$\widetilde{UB}_G$		$\widetilde{SA}$		$\widetilde{TS}$			
		CPU Time(s)		Node Number		CPU Time(s)		CPU Time(s)		CPU Time(s)	
n	ϱ	mean	max	mean	max	mean	max	mean	max	mean	max
9	-0.2	0.3607	0.7824	5307.00	11728.00	0.0002	0.0008	0.0558	0.0563	0.0812	0.0865
	-0.3	0.2745	0.4729	4200.60	7124.00	0.0004	0.0011	0.0569	0.0590	0.0819	0.0850
	-0.4	0.9621	2.4730	13182.60	32477.00	0.0006	0.0011	0.0581	0.0617	0.0841	0.0882
	-0.5	0.2104	0.4323	3219.20	6557.00	0.0004	0.0010	0.0616	0.0641	0.0853	0.0921
10	-0.2	2.7069	10.7840	17881.20	58223.00	0.0006	0.0010	0.0855	0.0982	0.1617	0.1798
	-0.3	6.3092	20.5142	53526.40	147511.00	0.0004	0.0010	0.0945	0.1426	0.1544	0.1822
	-0.4	2.9369	11.7102	31122.00	123041.00	0.0004	0.0010	0.0795	0.0923	0.1645	0.1851
	-0.5	6.2627	14.8897	63065.80	159840.00	0.0002	0.0010	0.0784	0.0862	0.1508	0.1720
11	-0.2	12.3867	37.9443	134034.00	389462.00	0.0004	0.0011	0.0888	0.1002	0.1982	0.2182
	-0.3	14.6582	47.1920	172896.60	553285.00	0.0003	0.0010	0.0874	0.0946	0.1964	0.2014
	-0.4	17.9943	33.9420	215834.40	384533.00	0.0006	0.0010	0.0830	0.0852	0.1895	0.1949
	-0.5	8.4031	23.9742	100007.20	271884.00	0.0004	0.0010	0.0835	0.0849	0.1898	0.1929
12	-0.2	345.6180	662.8780	3780505.00	7309485.00	0.0010	0.0010	0.0971	0.1009	0.2623	0.2723
	-0.3	146.0589	425.0653	1540884.67	4456984.00	0.0010	0.0010	0.0961	0.1002	0.2686	0.2807
	-0.4	284.0188	516.7235	3003596.33	5520523.00	0.0010	0.0010	0.0971	0.1054	0.2731	0.2999
	-0.5	127.6224	246.7136	1387657.00	2602992.00	0.0007	0.0010	0.0994	0.1010	0.2792	0.2934
13	-0.2	1061.1135	1401.3740	10901669.00	15686130.00	0.0010	0.0011	0.1264	0.1492	0.4346	0.5647
	-0.3	750.7272	1102.0001	7410484.00	10466784.00	0.0007	0.0010	0.1228	0.1367	0.3851	0.3858
	-0.4	1002.4937	1302.0002	10484563.00	10763651.00	0.0011	0.0012	0.1100	0.1126	0.3593	0.3631
	-0.5	1161.8779	1681.6960	12643972.67	16050724.00	0.0008	0.0011	0.1108	0.1151	0.3593	0.3627

Table 4. CPU time (s) of algorithms for $M = 0.5$ (Small-Scale).

		$\widetilde{BB}_G$				$\widetilde{UB}_G$		$\widetilde{SA}$		$\widetilde{TS}$	
n	ϱ	CPU Time(s)		Node Number		CPU Time(s)		CPU Time(s)		CPU Time(s)	
		mean	max	mean	max	mean	max	mean	max	mean	max
9	-0.2	0.3607	0.7824	5300.00	11781.00	0.0002	0.0008	0.0558	0.0563	0.0812	0.0865
	-0.3	0.2745	0.4729	4203.60	7144.00	0.0004	0.0011	0.0569	0.0590	0.0819	0.0850
	-0.4	0.9621	2.4730	13827.60	32475.00	0.0006	0.0011	0.0581	0.0617	0.0841	0.0882
	-0.5	0.2104	0.4323	3194.20	6572.00	0.0004	0.0010	0.0616	0.0641	0.0853	0.0921
10	-0.2	2.7069	10.7840	18813.20	52235.00	0.0006	0.0011	0.0319	0.0338	0.1346	0.2194
	-0.3	6.3092	20.5142	55261.40	175118.00	0.0004	0.0010	0.0945	0.1426	0.1544	0.1822
	-0.4	2.9369	11.7102	31224.00	130418.00	0.0004	0.0012	0.0795	0.0923	0.1645	0.1851
	-0.5	6.2627	14.8897	60654.80	198401.00	0.0002	0.0010	0.0784	0.0862	0.1508	0.1720
11	-0.2	12.3867	37.9443	140349.00	394626.00	0.0004	0.0011	0.0888	0.1002	0.1982	0.2182
	-0.3	14.6582	47.1920	128962.60	532857.00	0.0003	0.0010	0.0874	0.0946	0.1964	0.2014
	-0.4	17.9943	33.9420	258345.40	345331.00	0.0006	0.0010	0.0830	0.0852	0.1895	0.1949
	-0.5	8.4031	23.9742	100079.20	218842.00	0.0004	0.0010	0.0835	0.0849	0.1898	0.1929
12	-0.2	388.8892	1102.0010	3471571.00	9558582.00	0.0004	0.0011	0.1097	0.1175	0.3164	0.3603
	-0.3	169.2462	322.3118	1030280.33	2038389.00	0.0011	0.0012	0.1074	0.1095	0.3101	0.3184
	-0.4	142.1627	336.5996	1229394.67	3817346.00	0.0010	0.0010	0.1082	0.1228	0.2948	0.3192
	-0.5	549.9126	1102.0005	3601222.33	7929176.00	0.0010	0.0010	0.1631	0.2713	0.3657	0.5406
13	-0.2	521.6007	1102.0003	5091508.67	11921071.00	0.0007	0.0010	0.1144	0.1148	0.3798	0.3813
	-0.3	420.6651	643.8195	4565029.67	6233817.00	0.0006	0.0010	0.1092	0.1117	0.3597	0.3611
	-0.4	795.1791	1102.0008	7251792.00	11888362.00	0.0003	0.0010	0.1162	0.1250	0.3842	0.4205
	-0.5	967.5826	1102.0004	9653149.00	12039545.00	0.0005	0.0010	0.1141	0.1275	0.3709	0.3910

Table 5. CPU time (s) of algorithms for $M = 0.8$ (Small-Scale).

		$\widetilde{BB}_G$				$\widetilde{UB}_G$		$\widetilde{SA}$		$\widetilde{TS}$	
		CPU Time(s)		Node Number		CPU Time(s)		CPU Time(s)		CPU Time(s)	
n	ϱ	mean	max	mean	max	mean	max	mean	max	mean	max
9	-0.2	0.4530	0.9452	6367.80	12714.00	0.0002	0.0010	0.0573	0.0589	0.0816	0.0828
	-0.3	0.8027	2.2440	11421.40	31343.00	0.0004	0.0010	0.0562	0.0569	0.0804	0.0819
	-0.4	0.5680	1.2616	8087.20	17033.00	0.0006	0.0010	0.0557	0.0569	0.0807	0.0819
	-0.5	0.2846	0.4103	4170.60	6039.00	0.0009	0.0019	0.0569	0.0589	0.0808	0.0829
10	-0.2	7.0574	23.4201	89432.80	286008.00	0.0004	0.0010	0.0683	0.0713	0.1266	0.1288
	-0.3	4.9563	13.6872	65795.60	179552.00	0.0002	0.0010	0.0688	0.0701	0.1245	0.1252
	-0.4	3.2548	7.0584	42962.20	95168.00	0.0006	0.0010	0.0682	0.0698	0.1253	0.1268
	-0.5	6.2011	14.3451	78224.00	179635.00	0.0004	0.0010	0.0691	0.0698	0.1247	0.1261
11	-0.2	27.3935	101.0913	318182.40	1137215.00	0.0006	0.0011	0.0826	0.0864	0.1860	0.1888
	-0.3	38.8517	48.2872	455652.80	551309.00	0.0003	0.0010	0.0829	0.0839	0.1865	0.1898
	-0.4	10.8728	18.2250	132843.60	230482.00	0.0008	0.0010	0.0827	0.0836	0.1874	0.1901
	-0.5	63.3144	253.2307	712844.20	2824853.00	0.0006	0.0010	0.0831	0.0840	0.1888	0.1922
12	-0.2	158.3018	341.8801	1788668.00	3874948.00	0.0009	0.0010	0.0962	0.0978	0.2703	0.2781
	-0.3	225.6573	516.4513	2497078.33	5689840.00	0.0011	0.0012	0.0967	0.0983	0.2683	0.2744
	-0.4	432.6346	879.4942	4827816.00	9755191.00	0.0007	0.0010	0.0950	0.0957	0.2612	0.2702
	-0.5	124.1963	289.5522	1374960.00	3108865.00	0.0006	0.0010	0.0920	0.0925	0.2616	0.2665
13	-0.2	1000.0009	1100.0010	10100506.00	10723155.00	0.0003	0.0010	0.1075	0.1090	0.3586	0.3626
	-0.3	875.6720	1120.0010	9062671.00	11240218.00	0.0000	0.0000	0.1102	0.1133	0.3599	0.3677
	-0.4	898.8594	1022.0004	9470854.00	10549803.00	0.0003	0.0010	0.1085	0.1093	0.3576	0.3616
	-0.5	674.7852	1031.0008	7485857.33	10605271.00	0.0007	0.0011	0.1112	0.1187	0.3576	0.3665

Table 6. Error of algorithms for $M = 0.2$ (Small-Scale).

n	ϱ	$\widetilde{UB}_G$		$\widetilde{SA}$		$\widetilde{TS}$	
		Error		Error		Error	
		mean(%)	max(%)	mean(%)	max(%)	mean(%)	max(%)
9	-0.2	0.7310	1.6683	0.0000	0.0000	0.0300	0.0700
	-0.3	0.8656	1.6846	0.0000	0.0000	0.0000	0.0000
	-0.4	0.8236	1.6355	0.0000	0.0000	0.0000	0.0000
	-0.5	1.0271	1.6863	0.0000	0.0000	0.0000	0.0000
10	-0.2	0.8204	1.4875	0.0000	0.0000	0.0000	0.0000
	-0.3	0.7195	1.3741	0.0000	0.0000	0.0000	0.0000
	-0.4	0.8924	1.3749	0.0000	0.0000	0.0200	0.0600
	-0.5	0.9263	1.4734	0.0000	0.0000	0.0000	0.0000
11	-0.2	0.8745	1.5867	0.0000	0.0000	0.0000	0.0000
	-0.3	0.7181	1.5766	0.0000	0.0000	0.0000	0.0000
	-0.4	0.9073	1.5047	0.0000	0.0000	0.0000	0.0000
	-0.5	0.9788	1.5880	0.0000	0.0000	0.0000	0.0000
12	-0.2	0.8833	1.5458	0.0000	0.0000	0.0000	0.0000
	-0.3	0.9102	1.5402	0.0000	0.0000	0.0000	0.0000
	-0.4	0.7777	1.5285	0.0000	0.0000	0.0800	0.1700
	-0.5	0.9183	1.3479	0.0000	0.0000	0.0000	0.0000
13	-0.2	0.7812	1.3709	0.0000	0.0000	0.0000	0.0000
	-0.3	0.7598	1.3370	0.0000	0.0000	0.0000	0.0000
	-0.4	0.6914	1.3020	0.0000	0.0000	0.0000	0.0000
	-0.5	0.6829	1.3804	0.0000	0.0000	0.0200	0.0500

Table 7. Error of algorithms for $M = 0.5$ (Small-Scale).

n	ϱ	$\widetilde{UB}_G$		$\widetilde{SA}$		$\widetilde{TS}$	
		Error		Error		Error	
		mean(%)	max(%)	mean(%)	max(%)	mean(%)	max(%)
9	-0.2	0.9364	1.6887	0.0000	0.0000	0.0300	0.0800
	-0.3	0.9204	1.6658	0.0000	0.0000	0.0000	0.0000
	-0.4	0.8004	1.6814	0.0000	0.0000	0.0000	0.0000
	-0.5	0.8959	1.6486	0.0000	0.0000	0.0000	0.0000
10	-0.2	0.8243	1.4751	0.0000	0.0000	0.0000	0.0000
	-0.3	0.6792	1.4308	0.0000	0.0000	0.0000	0.0000
	-0.4	0.8279	1.4600	0.0000	0.0000	0.0000	0.0000
	-0.5	0.8999	1.4788	0.0000	0.0000	0.0000	0.0000
11	-0.2	0.8560	1.5667	0.0000	0.0000	0.0000	0.0000
	-0.3	0.9051	1.5690	0.0000	0.0000	0.0000	0.0000
	-0.4	0.9143	1.5428	0.0000	0.0000	0.0000	0.0000
	-0.5	0.8861	1.5389	0.0000	0.0000	0.0000	0.0000
12	-0.2	0.9632	1.5137	0.0000	0.0000	0.0000	0.0000
	-0.3	0.9065	1.4437	0.0000	0.0000	0.0000	0.0000
	-0.4	0.7880	1.5347	0.0000	0.0000	0.0000	0.0000
	-0.5	0.8143	1.4899	0.0000	0.0000	0.0200	0.0500
13	-0.2	0.7462	1.3866	0.0000	0.0000	0.0500	0.1200
	-0.3	0.7118	1.3567	0.0000	0.0000	0.0000	0.0000
	-0.4	0.6856	1.3599	0.0000	0.0000	0.0000	0.0000
	-0.5	0.6610	1.3454	0.0000	0.0000	0.0000	0.0000

Table 8. Error of algorithms for $M = 0.8$ (Small-Scale).

n	ϱ	$\widetilde{UB}_G$		$\widetilde{SA}$		$\widetilde{TS}$	
		Error		Error		Error	
		mean(%)	max(%)	mean(%)	max(%)	mean(%)	max(%)
9	-0.2	0.9066	1.6753	0.0000	0.0000	0.0000	0.0000
	-0.3	0.8312	1.6704	0.0000	0.0000	0.0000	0.0000
	-0.4	0.9978	1.6433	0.0000	0.0000	0.0100	0.0400
	-0.5	0.8370	1.6493	0.0000	0.0000	0.0000	0.0000
10	-0.2	0.7894	1.4861	0.0000	0.0000	0.0000	0.0000
	-0.3	0.9615	1.4841	0.0000	0.0000	0.0200	0.0600
	-0.4	0.7610	1.4575	0.0000	0.0000	0.0000	0.0000
	-0.5	0.7980	1.4779	0.0000	0.0000	0.0400	0.1100
11	-0.2	0.7521	1.4189	0.0000	0.0000	0.0000	0.0000
	-0.3	0.9558	1.5879	0.0000	0.0000	0.0000	0.0000
	-0.4	1.0057	1.5870	0.0000	0.0000	0.0000	0.0000
	-0.5	0.8132	1.5575	0.0000	0.0000	0.0000	0.0000
12	-0.2	0.9268	1.5247	0.0000	0.0000	0.0000	0.0000
	-0.3	0.8287	1.5149	0.0000	0.0000	0.0600	0.1400
	-0.4	0.9586	1.5342	0.0000	0.0000	0.0000	0.0000
	-0.5	0.8653	1.4718	0.0000	0.0000	0.0000	0.0000
13	-0.2	0.8296	1.3756	0.0000	0.0000	0.0000	0.0000
	-0.3	0.8922	1.3873	0.0000	0.0000	0.0000	0.0000
	-0.4	0.7230	1.3479	0.0000	0.0000	0.0000	0.0000
	-0.5	0.7354	1.3485	0.0000	0.0000	0.0000	0.0000

Table 9. CPU time (s) of algorithms for $M = 0.2$ (Large-Scale).

n	ϱ	$\widetilde{UB}_G$		$\widetilde{SA}$		$\widetilde{TS}$	
		CPU Time(s)		CPU Time(s)		CPU Time(s)	
		mean	max	mean	max	mean	max
100	-0.2	0.3368	0.3986	5.4654	5.9566	7.9835	8.5091
	-0.3	0.3477	0.3995	5.5537	5.9879	8.0726	8.6021
	-0.4	0.3531	0.3999	5.5104	5.9766	8.0219	8.4035
	-0.5	0.3468	0.3959	5.5707	5.9866	7.9335	8.3919
150	-0.2	1.0033	1.1024	12.0291	12.9656	26.0203	29.9932
	-0.3	0.9905	1.0999	11.9335	12.8922	26.1273	29.9368
	-0.4	0.9900	1.0876	11.9197	12.9672	25.8793	29.4627
	-0.5	0.9770	1.0970	12.0810	12.9985	25.6883	29.7675
200	-0.2	2.3517	2.4979	22.4042	23.9326	64.9570	69.9980
	-0.3	2.3435	2.4981	22.5855	23.9493	65.3723	69.5045
	-0.4	2.3832	2.4982	22.5030	23.8884	65.1875	69.3399
	-0.5	2.3534	2.4957	22.7265	23.8888	64.1703	68.7113
250	-0.2	4.4298	4.5555	31.9046	32.9918	127.6046	134.0026
	-0.3	4.4052	4.5043	31.9674	32.7788	127.3991	134.9590
	-0.4	4.4422	4.5579	31.9788	32.8491	127.9170	134.5863
	-0.5	4.4225	4.5317	32.0508	32.9097	128.1704	135.0075
300	-0.2	7.9464	8.4618	49.9186	51.9478	229.8504	234.9708
	-0.3	7.9223	8.4324	50.1242	51.8558	230.1951	234.9614
	-0.4	7.9706	8.4985	50.2040	51.8504	230.0030	234.4680
	-0.5	8.0413	8.4777	49.6500	51.9439	230.2067	234.4430

Table 10. CPU time (s) of algorithms for $M = 0.5$ (Large-Scale).

n	ϱ	$\widetilde{UB}_G$		$\widetilde{SA}$		$\widetilde{TS}$	
		CPU Time(s)		CPU Time(s)		CPU Time(s)	
		mean	max	mean	max	mean	max
100	-0.2	0.3485	0.3928	5.5836	5.9931	8.0477	8.4479
	-0.3	0.3430	0.3992	5.4453	5.9861	7.9866	8.5053
	-0.4	0.3402	0.3998	5.4323	5.9639	8.0075	8.4810
	-0.5	0.3377	0.3947	5.4989	5.9938	7.9943	8.4466
150	-0.2	0.9825	1.1041	12.0647	12.9792	26.1724	29.6123
	-0.3	1.0226	1.1013	12.0552	12.8950	24.7249	29.1526
	-0.4	1.0071	1.1082	12.0095	12.9880	25.5411	29.9097
	-0.5	1.0143	1.1086	12.0681	12.8655	25.8615	29.7833
200	-0.2	2.3757	2.4983	22.5101	23.9091	65.5818	69.5423
	-0.3	2.3363	2.4699	22.3446	23.8115	64.6411	69.1108
	-0.4	2.3742	2.4980	22.5860	23.8298	64.7764	69.9651
	-0.5	2.3590	2.4889	22.6147	23.9145	65.6192	69.8076
250	-0.2	4.4001	4.5483	32.0329	32.9168	126.9069	134.7619
	-0.3	4.4363	4.5525	32.0522	32.9999	127.6819	132.8637
	-0.4	4.4236	4.5440	31.9378	32.8709	127.3116	132.5223
	-0.5	4.4301	4.5523	31.9339	32.9833	128.5002	134.9198
300	-0.2	7.9672	8.4909	50.0317	51.8851	230.4622	234.8236
	-0.3	8.0684	8.4759	49.7543	51.8934	230.5359	234.9023
	-0.4	8.0708	8.4789	50.4463	51.9091	230.9405	234.9925
	-0.5	8.0004	8.3941	50.1058	51.8220	229.7375	234.6347

Table 11. CPU time (s) of algorithms for $M = 0.8$ (Large-Scale).

n	ϱ	$\widetilde{UB}_G$		$\widetilde{SA}$		$\widetilde{TS}$	
		CPU Time(s)		CPU Time(s)		CPU Time(s)	
		mean	max	mean	max	mean	max
100	-0.2	0.3371	0.3976	5.4573	5.9507	8.0242	8.4630
	-0.3	0.3401	0.3981	5.4489	5.9225	7.9591	8.5013
	-0.4	0.3462	0.3977	5.4927	5.9920	8.0103	8.4157
	-0.5	0.3506	0.4000	5.5382	5.9985	8.0244	8.5056
150	-0.2	0.9950	1.1064	11.9999	12.9471	25.5428	29.9825
	-0.3	1.0067	1.1060	11.9457	12.8457	24.7435	29.2596
	-0.4	0.9988	1.1084	12.1129	12.9742	25.6515	29.9795
	-0.5	0.9856	1.0899	12.0807	12.9765	25.4959	29.9399
200	-0.2	2.3260	2.4824	22.6180	23.9432	64.5590	69.4368
	-0.3	2.3356	2.4939	22.3998	23.9193	64.4255	69.5309
	-0.4	2.3232	2.4573	22.6955	23.9936	64.6205	70.0065
	-0.5	2.3426	2.4991	22.5372	23.9666	64.2216	69.9974
250	-0.2	4.4115	4.5515	32.0265	32.9563	127.5509	134.2925
	-0.3	4.4164	4.5517	32.2161	32.9829	127.8442	135.0128
	-0.4	4.4198	4.5420	32.0138	32.9763	126.9724	134.7507
	-0.5	4.4358	4.5457	32.0902	32.9955	127.5025	134.9609
300	-0.2	8.0591	8.4968	49.9152	51.6635	229.6729	234.4150
	-0.3	7.9876	8.4731	50.1998	51.8977	230.3331	235.0246
	-0.4	7.9619	8.4970	50.6420	51.9026	229.7306	234.8962
	-0.5	7.9565	8.4766	50.0738	51.8864	230.3527	234.6065

Table 12. Error of algorithms for $M = 0.2$ (Large-Scale).

n	ϱ	$\widetilde{UB}_G$		$\widetilde{SA}$		$\widetilde{TS}$	
		Error		Error		Error	
		mean(%)	max(%)	mean(%)	max(%)	mean(%)	max(%)
100	-0.2	0.3352	0.3925	0.0000	0.0000	0.0090	0.0158
	-0.3	0.3409	0.3939	0.0000	0.0000	0.0088	0.0160
	-0.4	0.3378	0.3956	0.0000	0.0000	0.0083	0.0158
	-0.5	0.3360	0.3919	0.0000	0.0000	0.0086	0.0149
150	-0.2	0.2713	0.3088	0.0000	0.0000	0.0170	0.0303
	-0.3	0.2711	0.3077	0.0000	0.0000	0.0150	0.0306
	-0.4	0.2657	0.3093	0.0000	0.0000	0.0193	0.0311
	-0.5	0.2616	0.3077	0.0000	0.0000	0.0173	0.0310
200	-0.2	0.2520	0.2978	0.0000	0.0000	0.0078	0.0130
	-0.3	0.2433	0.3000	0.0000	0.0000	0.0083	0.0130
	-0.4	0.2359	0.2835	0.0000	0.0000	0.0060	0.0112
	-0.5	0.2427	0.2998	0.0000	0.0000	0.0080	0.0125
250	-0.2	0.2321	0.2594	0.0000	0.0000	0.0165	0.0297
	-0.3	0.2210	0.2598	0.0000	0.0000	0.0194	0.0325
	-0.4	0.2262	0.2591	0.0000	0.0000	0.0158	0.0294
	-0.5	0.2235	0.2585	0.0000	0.0000	0.0191	0.0323
300	-0.2	0.1785	0.2199	0.0000	0.0000	0.0386	0.0552
	-0.3	0.1904	0.2197	0.0000	0.0000	0.0396	0.0560
	-0.4	0.1872	0.2192	0.0000	0.0000	0.0429	0.0556
	-0.5	0.1818	0.2166	0.0000	0.0000	0.0403	0.0555

Table 13. Error of algorithms for $M = 0.5$ (Large-Scale).

n	ϱ	$\widetilde{UB}_G$		$\widetilde{SA}$		$\widetilde{TS}$	
		Error		Error		Error	
		mean(%)	max(%)	mean(%)	max(%)	mean(%)	max(%)
100	-0.2	0.3327	0.3942	0.0000	0.0000	0.0085	0.0156
	-0.3	0.2646	0.3084	0.0000	0.0000	0.0196	0.0303
	-0.4	0.3449	0.3981	0.0000	0.0000	0.0103	0.0156
	-0.5	0.3381	0.3926	0.0000	0.0000	0.0091	0.0157
150	-0.2	0.3327	0.3942	0.0000	0.0000	0.0085	0.0156
	-0.3	0.2432	0.2926	0.0000	0.0000	0.0077	0.0128
	-0.4	0.2689	0.3083	0.0000	0.0000	0.0184	0.0317
	-0.5	0.2689	0.3079	0.0000	0.0000	0.0186	0.0313
200	-0.2	0.2445	0.3000	0.0000	0.0000	0.0065	0.0130
	-0.3	0.2236	0.2589	0.0000	0.0000	0.0190	0.0320
	-0.4	0.2505	0.2994	0.0000	0.0000	0.0080	0.0128
	-0.5	0.2465	0.2869	0.0000	0.0000	0.0076	0.0125
250	-0.2	0.2233	0.2550	0.0000	0.0000	0.0166	0.0289
	-0.3	0.3439	0.3935	0.0000	0.0000	0.0084	0.0155
	-0.4	0.2236	0.2577	0.0000	0.0000	0.0180	0.0307
	-0.5	0.2269	0.2589	0.0000	0.0000	0.0165	0.0314
300	-0.2	0.1816	0.2184	0.0000	0.0000	0.0416	0.0559
	-0.3	0.1816	0.2166	0.0000	0.0000	0.0395	0.0558
	-0.4	0.1806	0.2167	0.0000	0.0000	0.0406	0.0548
	-0.5	0.1924	0.2192	0.0000	0.0000	0.0381	0.0518

Table 14. Error of algorithms for $M = 0.8$ (Large-Scale).

n	ϱ	$\widetilde{UB}_G$		$\widetilde{SA}$		$\widetilde{TS}$	
		Error		Error		Error	
		mean(%)	max(%)	mean(%)	max(%)	mean(%)	max(%)
100	-0.2	0.3360	0.3970	0.0000	0.0000	0.0082	0.0155
	-0.3	0.3384	0.3994	0.0000	0.0000	0.0091	0.0152
	-0.4	0.3505	0.3960	0.0000	0.0000	0.0084	0.0145
	-0.5	0.3360	0.3988	0.0000	0.0000	0.0094	0.0151
150	-0.2	0.2696	0.3095	0.0000	0.0000	0.0170	0.0297
	-0.3	0.2733	0.3085	0.0000	0.0000	0.0181	0.0311
	-0.4	0.2724	0.3094	0.0000	0.0000	0.0207	0.0319
	-0.5	0.2715	0.3074	0.0000	0.0000	0.0203	0.0317
200	-0.2	0.2343	0.2860	0.0000	0.0000	0.0081	0.0130
	-0.3	0.2392	0.2977	0.0000	0.0000	0.0078	0.0128
	-0.4	0.2461	0.2980	0.0000	0.0000	0.0079	0.0128
	-0.5	0.2394	0.2987	0.0000	0.0000	0.0073	0.0129
250	-0.2	0.2189	0.2570	0.0000	0.0000	0.0168	0.0312
	-0.3	0.2267	0.2549	0.0000	0.0000	0.0169	0.0316
	-0.4	0.2247	0.2569	0.0000	0.0000	0.0150	0.0287
	-0.5	0.2215	0.2575	0.0000	0.0000	0.0164	0.0322
300	-0.2	0.1854	0.2191	0.0000	0.0000	0.0389	0.0559
	-0.3	0.1857	0.2164	0.0000	0.0000	0.0390	0.0506
	-0.4	0.1877	0.2176	0.0000	0.0000	0.0396	0.0520
	-0.5	0.1845	0.2199	0.0000	0.0000	0.0399	0.0524

7. Summary and future research

This study investigated a single-machine problem with the resource-consumption and DeJong's time-dependent learning effect simultaneously. For this problem of small instance sizes, a branch-and-bound algorithm ($\widetilde{BB}_G$) capable of accurately obtaining optimal solutions was proposed. For large-scale instances, we introduced the heuristic algorithm ($\widetilde{UB}_G$), simulated annealing ($\widetilde{SA}$), and tabu search ($\widetilde{TS}$). Extensive computational experiments demonstrated that the $\widetilde{SA}$ can find near-optimal solutions within short time frames. At the level of management insights, the study reveals the dynamic trade-off relationship between resource investment and cumulative learning effects, indicating that managers should implement differentiated resource allocation based on the position of jobs in the sequence, concentrating limited resources on key jobs that have the greatest impact on overall goals, especially on later-stage tasks, while proactively stimulating learning effects by prioritizing jobs with shorter basic processing times, thereby achieving the objective of shortening the total cycle time.

This study still has several limitations: At the model level, it only considers a single-machine deterministic problem and does not involve multi-machine environments, stochastic disturbances, or more complex forms of learning effects, which differs from real-world production; at the algorithm level, the solution scale of exact algorithms is limited; at the validation level, the experiments are entirely based on simulation data, lacking support and verification from industrial cases.

Further research should further consider the scheduling model with deteriorating jobs (see Kong et al. [52]; Lu et al. [53]; Liu et al. [61]; Qiu and Wang [62]; Sun et al. [63]), the scheduling model with job rejection (i.e., outsourcing, see Zhang et al. [64]; Shabtay et al. [65]; Geng et al. [66]; Bai et al. [67]; Song et al. [68]; Choi and Park [69]), or the scheduling model in flow shop setting (see Bai et al. [51];

Wu et al. [70]; Lv and Wang [71]; Li et al. [72]; Miao et al. [73]; Wang et al. [74]).

Further research can also be systematically deepened from three dimensions: In terms of job characteristics, the deterioration of job processing and outsourcing/rejection decisions can be introduced to more realistically depict time-loss in production and the balance between in-house and outsourced manufacturing; in terms of the production environment, the limitation of single machines can be overcome by extending the model to complex layouts such as no-wait flow shops and distributed multi-agent systems, integrating hybrid modes like collective scheduling and batch processing; in terms of optimization objectives and uncertainties, green manufacturing considerations such as carbon emission costs or peak energy constraints can be included, and robust optimization, distributed robust optimization, or multi-objective evolutionary algorithms can be used to effectively cope with parameter fluctuations and multi-objective conflicts, thereby continuously enhancing the practical value and theoretical depth of this scheduling model.

Author contributions

Kai-Fang Zhang, Jia-Fan Zhang, and Ji-Bo Wang designed the methodology; Jia-Fan Zhang and Ji-Bo Wang conducted the investigation; Kai-Fang Zhang wrote the original draft; and Kai-Fang Zhang, Jia-Fan Zhang, 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. This work was supported by the Basic Scientific Research Project of Liaoning Provincial Department of Education [Year 2026; Project leader: Li Wang].

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. Z. Y. Jiang, F. F. Chen, X. D. Zhang, Single-machine scheduling with times-based and job-dependent learning effect, *J. Oper. Res. Soc.*, **68** (2017), 809–815. <https://doi.org/10.1057/jors.2016.40>
2. A. Azzouz, M. Ennigrou, L. B. Said, Scheduling problems under learning effects: Classification and cartography, *Int. J. Prod. Res.*, **56** (2018), 1642–1661. <https://doi.org/10.1080/00207543.2017.1355576>
3. C. C. Wu, W. C. Lin, X. G. Zhang, I. H. Chung, T. H. Yang, K. J. 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>
4. B. Liao, X. Wang, X. Zhu, S. Yang, P. M. Pardalos, Less is more approach for competing groups scheduling with different learning effects, *J. Comb. Optim.*, **39** (2020), 33–54. <https://doi.org/10.1007/s10878-019-00452-8>
5. S. W. Liu, X. B. Liu, J. Pei, P. M. Pardalos, Q. R. Song, Parallel-batching machines scheduling problem with a truncated time-dependent learning effect via a hybrid CS-JADE algorithm, *Optim. Methods Softw.*, **35** (2020), 116–141. <https://doi.org/10.1080/10556788.2019.1577415>
6. 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>
7. S. Zhao, Scheduling jobs with general truncated learning effects including proportional setup times, *Comput. Appl. Math.*, **41** (2022), 146. <https://doi.org/10.1007/s40314-022-01851-0>
8. Z. Liu, J. B. Wang, Single-machine scheduling with simultaneous learning effects and delivery times, *Mathematics*, **12** (2024), 2522. <https://doi.org/10.3390/math12162522>
9. D. Y. Lv, J. B. Wang, Considering the peak power consumption problem with learning and deterioration effect in flow shop scheduling, *Comput. Ind. Eng.*, **197** (2024), 110599. <https://doi.org/10.1016/j.cie.2024.110599>
10. 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>
11. J. Gao, J. Zou, Y. Zhang, Single-machine scheduling with a deteriorating maintenance activity and DeJong's learning effect, *Int. J. Found. Comput. Sci.*, **36** (2025), 635–648. <https://doi.org/10.1142/S0129054122460030>
12. 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>
13. X. Wang, W. 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>
14. 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>

15. H. R. Song, J. K. Yang, J. B. Wang, Study on multiple due-windows assignment scheduling with learning and deteriorating effects, *Asia-Pac. J. Oper. Res.*, (2026), 2650001. <https://doi.org/10.1142/S0217595926500016>
16. 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>
17. E. Nowicki, S. Zdrzalka, A survey of results for sequencing problems with controllable processing times, *Discret. Appl. Math.*, **26** (1990), 271–287. [https://doi.org/10.1016/0166-218X\(90\)90105-L](https://doi.org/10.1016/0166-218X(90)90105-L)
18. X. H. Guan, Q. Z. Zhai, F. Lai, New lagrangian relaxation based algorithm for resource scheduling with homogeneous subproblems, *J. Optim. Theory Appl.*, **113** (2002), 65–82. <https://doi.org/10.1023/A:1014805213554>
19. 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>
20. D. Oron, Scheduling controllable processing time jobs in a deteriorating environment, *J. Oper. Res. Soc.*, **64** (2014), 49–56. <https://doi.org/10.1057/jors.2013.5>
21. Y. Q. Yin, D. J. Wang, T. C. E. Cheng, C. C. Wu, Bi-criterion single-machine scheduling and due-window assignment with common flow allowances and resource-dependent processing times, *J. Oper. Res. Soc.*, **67** (2016), 1169–1183. <https://doi.org/10.1057/jors.2016.14>
22. D. J. Wang, Y. G. Yu, H. X. Qiu, Y. Q. Yin, T. C. E. Cheng, Two-agent scheduling with linear resource-dependent processing times, *Nav. Res. Logist.*, **67** (2020), 573–591. <https://doi.org/10.1002/nav.21936>
23. J. S. Gao, X. M. Zhu, K. Y. Bai, R. T. Zhang, New controllable processing time scheduling with subcontracting strategy for no-wait job shop problem, *Int. J. Prod. Res.*, **60** (2022), 2254–2274. <https://doi.org/10.1080/00207543.2021.1886368>
24. J. F. Ren, Y. Yang, Common due-window assignment and minmax scheduling with resource allocation and group technology on a single machine, *Eng. Optim.*, **54** (2022), 1819–1834. <https://doi.org/10.1080/0305215X.2021.1961761>
25. Z. C. Geng, J. J. Yuan, Single-machine scheduling of multiple projects with controllable processing times, *Eur. J. Oper. Res.*, **308** (2023), 1074–1090. <https://doi.org/10.1016/j.ejor.2023.01.026>
26. M. Geurtsen, J. B. H. C. Didden, J. Adan, Z. Atan, I. Adan, Production, maintenance and resource scheduling: A review, *Eur. J. Oper. Res.*, **305** (2023), 501–529. <https://doi.org/10.1016/j.ejor.2022.03.045>
27. N. S. Pang, Q. C. Meng, Resource allocation in robust scheduling, *J. Oper. Res. Soc.*, **74** (2023), 125–142. <https://doi.org/10.1080/01605682.2022.2029593>
28. 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>
29. X. Sun, X. N. Geng, J. Y. Wang, L. Pan, Slack due window assignment scheduling in the single-machine with controllable processing times, *J. Ind. Manag. Optim.*, **20** (2024), 15–35. <https://doi.org/10.3934/jimo.2023067>

30. X. Wang, W. 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>
31. 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-Pac. J. Oper. Res.*, **41** (2024), 2350026. <https://doi.org/10.1142/S0217595923500264>
32. W. Weng, C. Chu, P. Wu, Resource allocation to minimize the makespan with multi-resource operations, *J. Syst. Sci. Complex.*, **37** (2024), 2054–2070. <https://doi.org/10.1007/s11424-024-3284-z>
33. 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>
34. 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>
35. 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>
36. L. Zhang, S. S. Li, Z. C. Geng, R. X. Chen, J. R. Xu, Multiple parallel-batch machines scheduling with additive resource assignment and machine available times, *Discret. Appl. Math.*, **379** (2026), 675–684. <https://doi.org/10.1016/j.dam.2025.09.035>
37. 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>
38. 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>
39. D. Wang, M. Z. Wang, J. B. Wang, Single-Machine scheduling with learning effect and resource-dependent processing times, *Comput. Ind. Eng.*, **59** (2010), 458–462. <https://doi.org/10.1016/j.cie.2010.06.002>
40. Y. Y. Lu, J. B. Wang, P. Ji, H. He, A note on resource allocation scheduling with group technology and learning effects on a single machine, *Eng. Optim.*, **49** (2017), 1621–1632. <https://doi.org/10.1080/0305215X.2016.1265305>
41. N. Yin, Single-machine due-window assignment resource allocation scheduling with job-dependent learning effect, *J. Appl. Math. Comput.*, **56** (2018), 715–725. <https://doi.org/10.1007/s12190-017-1091-6>
42. S. Zhao, Resource allocation flowshop scheduling with learning effect and slack due window assignment, *J. Ind. Manag. Optim.*, **17** (2021), 2817–2835. <https://doi.org/10.3934/jimo.2020096>
43. J. B. Wang, D. Y. Lv, J. Xu, P. Ji, F. Li, Bicriterion scheduling with truncated learning effects and convex controllable processing times, *Int. Trans. Oper. Res.*, **28** (2021), 1573–1593. <https://doi.org/10.1111/itor.12888>

44. J. Qian, G. Chang, X. Zhang, Single-machine common due-window assignment and scheduling with position-dependent weights, delivery time, learning effect and resource allocations, *J. Appl. Math. Comput.*, **70** (2024), 1965–1994. <https://doi.org/10.1007/s12190-024-02023-5>
45. J. Qian, Z. Y. Guo, Common due window assignment and single machine scheduling with delivery time, resource allocation, and job-dependent learning effect, *J. Appl. Math. Comput.*, **70** (2024), 4441–4471. <https://doi.org/10.1007/s12190-024-02090-8>
46. Y. Zhang, X. Sun, T. Liu, J. 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>
47. 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>
48. 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>
49. 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>
50. X. F. Wu, L. Lin, J. B. Wang, Total weighted completion time scheduling under resource allocation and time-dependent learning effects, *J. Ind. Manag. Optim.*, **22** (2026), 3219–3251. <https://doi.org/10.3934/jimo.2026118>
51. D. Y. Bai, W. J. Zheng, C. B. Zang, J. Yang, C. C. Wu, H. Qin, Discrete optimization algorithms for distributed bi-agent flowshop scheduling with release dates, *Swarm Evol. Comput.*, **98** (2025), 102101. <https://doi.org/10.1016/j.swevo.2025.102101>
52. 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>
53. Y. Y. Lu, S. Zhang, J. Y. Tao, Earliness-tardiness scheduling with delivery times and deteriorating jobs, *Asia-Pac. J. Oper. Res.*, **42** (2025), 1–17. <https://doi.org/10.1142/S021759592450009X>
54. 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](https://doi.org/10.1016/S0377-2217(98)00246-X)
55. W. H. Kuo, D. L. Yang, Minimizing the total completion time in a single-machine scheduling problem with a time-dependent learning effect, *Eur. J. Oper. Res.*, **174** (2006), 1184–1190. <https://doi.org/10.1016/j.ejor.2005.03.020>
56. M. Ji, D. Yao, Q. Yang, T. C. E. Cheng, Machine scheduling with Dejong's learning effect, *Comput. Ind. Eng.*, **80** (2015), 195–200. <https://doi.org/10.1016/j.cie.2014.12.009>
57. K. F. Zhang, J. B. Wang, Y. Y. Lu, On flowshop scheduling with the DeJong's learning effect, *Asia-Pac. J. Oper. Res.*, (2026), 2650029. <https://doi.org/10.1142/S0217595926500296>
58. M. L. Pinedo, Scheduling Theory, Algorithms, and Systems, *Springer Cham.* <https://doi.org/10.1007/978-3-031-05921-6>

59. A. Agnetis, J. C. Billaut, M. Pinedo, D. Shabtay, Fifty years of research in scheduling-Theory and applications, *Eur. J. Oper. Res.*, **327** (2025), 367–393. <https://doi.org/10.1016/j.ejor.2025.01.034>
60. M. Nawaz, E. E. Ensore Jr, 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](https://doi.org/10.1016/0305-0483(83)90088-9)
61. 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>
62. X. Y. Qiu, J. B. Wang, Single-machine scheduling with mixed due-windows and deterioration effects, *J. Appl. Math. Comput.*, **71** (2025), 2527–2542. <https://doi.org/10.1007/s12190-024-02306-x>
63. 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>
64. L. Q. Zhang, L. F. Lu, J. J. Yuan, Single machine scheduling with release dates and rejection, *Eur. J. Oper. Res.*, **198** (2009), 975–978. <https://doi.org/10.1016/j.ejor.2008.10.006>
65. D. Shabtay, N. Gaspar, M. Kaspi, A survey on offline scheduling with rejection, *J. Sched.*, **16** (2013), 3–28. <https://doi.org/10.1007/s10951-012-0303-z>
66. X. N. Geng, X. Y. 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>
67. B. Bai, C. M. Wei, H. Y. He, J. B. Wang, Study on single-machine common/slack due-window assignment scheduling with delivery times, variable processing times and outsourcing, *Mathematics*, **12** (2024), 2883. <https://doi.org/10.3390/math12182883>
68. J. X. Song, C. X. Miao, F. Y. Kong, Scheduling with step learning and job rejection, *Oper. Res.*, **25** (2025), 6. <https://doi.org/10.1007/s12351-024-00887-w>
69. B. C. Choi, M. J. Park, Minimizing total completion time in single-machine scheduling with convex resource consumption and job rejection, *Theor. Comput. Sci.*, **1064** (2026), 115708. <https://doi.org/10.1016/j.tcs.2025.115708>
70. C. C. Wu, J. Y. Chen, W. C. Lin, K. Lai, D. Bai, S. Y. Lai, A two-stage three-machine assembly scheduling flowshop problem with both two-agent and learning phenomenon, *Comput. Ind. Eng.*, **130** (2019), 485–499. <https://doi.org/10.1016/j.cie.2019.02.047>
71. D. Y. Lv, J. B. Wang, No-idle flow shop scheduling with deteriorating jobs and common due date under dominating machines, *Asia-Pac. J. Oper. Res.*, **41** (2024), 1–19. <https://doi.org/10.1142/S0217595924500039>
72. N. Li, R. Ma, Y. Z. Zhang, Single-machine scheduling with the learning effect of processing time and the deterioration effect of delivery time for prefabricated components, *J. Comb. Optim.*, **49** (2025), 37. <https://doi.org/10.1007/s10878-025-01271-w>
73. C. X. Miao, H. R. Wu, J. M. Hu, Two-agent scheduling with time-dependent and position-dependent in a flow-shop system, *RAIRO-Oper. Res.*, **59** (2025), 3253–3265. <https://doi.org/10.1051/ro/2025126>

-
74. J. B. Wang, D. Y. Lv, C. Wan, Proportionate flow shop scheduling with job-dependent due windows and position-dependent weights, *Asia-Pac. J. Oper. Res.*, **42** (2025), 1–17. <https://doi.org/10.1142/S0217595924500118>



AIMS Press

©2026 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (<https://creativecommons.org/licenses/by/4.0>)