A Two-Stage Three-Machine Flow Shop Assembly Problem Mixed with a Controllable Number and Sum-of-Processing Times-Based Learning Effect by Simulated Annealing Algorithms

Shang-Chia Liu

Department of Business Administration, FuJen Catholic University, New Taipei City 242062, Taiwan

Correspondence should be addressed to Shang-Chia Liu; 056298@mail.fju.edu.tw

Received 31 July 2020; Revised 2 September 2020; Accepted 23 September 2020; Published 10 October 2020

Academic Editor: Dujuan Wang

Copyright © 2020 Shang-Chia Liu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The two-stage assembly scheduling problem is widely used in industrial and service industries. This study focuses on the two-stage three-machine flow shop assembly problem mixed with a controllable number and sum-of-processing times-based learning effect, in which the job processing time is considered to be a function of the control of the truncation parameters and learning based on the sum of the processing time. However, the truncation function is very limited in the two-stage flow shop assembly scheduling settings. Thus, this study explores a two-stage three-machine flow shop assembly problem with truncated learning to minimize the makespan criterion. To solve the proposed model, we derive several dominance rules, lemmas, and lower bounds applied in the branch-and-bound method. On the other hand, three simulated annealing algorithms are proposed for finding approximate solutions. In both the small and large size number of job situations, the SA algorithm is better than the JS algorithm in this study. All the experimental results of the proposed algorithm are presented on small and large job sizes, respectively.

1. Introduction

For decades, scheduling models usually assumed that the processing time of the job is known and fixed [1]. But, in a real production system, both the efficiency of the machine and the skill of the worker can increase with the working time, thereby reducing the actual work processing time with the development of production. Many researchers indicated certain types of learning effects, such as [2] in the position-based learning effect and [3] in the time-dependent learning effect. This topic continues to attract a lot of research interest [4–6].

On the other hand, Biskup, Kuo and Yang [2, 3], and Kuolamas and Kyparisis [7] introduced models that involve a learning effect on two-machine scheduling or flow shop scheduling settings following the same or different learning ideas. There are a multitude of studies related to two-machine scheduling or flow shop scheduling settings with the learning effect consideration, including [8–11] and [12]. Besides, there are some two-stage models also with the learning effect consideration. Wu et al. [13] adopted the learning model developed by Biskup [2] to solve a two-stage flow shop scheduling problem with three machines to minimize the makespan. They devised three simulated annealing (SA) algorithms and three cloud theory-based SA algorithms. Adopting the learning model in [3, 14], a branch-and-bound (B&B) algorithm is devised incorporating six (hybrid) particle swarm optimization (PSO) methods to solve the two-stage flow shop assembly scheduling problem to minimize the total job completion time.

Most studies using the learning model are applied in the single-machine or flow shop setting. However, the two-stage assembly scheduling problem is relatively unexplored. Recently, Wu et al. and Zou et al. [15, 16] considered position-based learning in connection with a two-stage assembly scheduling problem. They proposed different versions of simulated annealing and cloud theory-based simulated annealing to solve this problem. Moreover, Wu et al. [17] considered this problem with general learning effects. They developed and evaluated the performances of six hybrid...
particle swarm optimizations (PSO). Wu et al. [18] have conducted the research about a combined approach for two-agent scheduling with a sum-of-processing-times-based learning effect. To the best of our knowledge, no other research considers the two-stage assembly scheduling problem with learning effects. Table 1 presents a comparative study of existing models to solve the considered scheduling problem. Many studies claim that most of the productive items in manufacturing systems may be formulated in a two-stage assembly scheduling model. However, the literature neglects accumulated learning experience in solving a two-stage assembly scheduling problem. In fact, the sum-of-processing times-based learning model is pertinent to process manufacturing in which an initial setup is often followed by a lengthy uninterrupted production process. Motivated by this observation, this study introduces the 2-stage 3-machine assembly problem with a sum-of-processing times of already processed jobs learning to minimize the makespan criterion. This model assumes that there are two machines in the first stage and an assembly machine in the second stage. This study first provides some dominances and a lower bound applied in the branch-and-bound method for an optimal solution. Three heuristics modified from Johnson’s rule with and without improving by an interchange pairwise method are, then, separately applied in three simulated annealing algorithms for finding near-optimal solutions. Finally, the statistical results of the three proposed algorithms are evaluated and reported.

2. Problem Statement

The considered two-stage assembly scheduling problem is formally described in this section. We assume a series of $n$ given jobs to be processed on three machines. The main idea of the studied problem is executing $n$ given jobs on three machines in two stages. Each job has strictly more than two operations. For the operations of the $n$ job, the first stage is performed in a parallel way in two machines. The operation processing of the second stage only begins when the operations of the first stage are completed. All jobs are available at time zero. Each job $j$ is decomposed on three tasks as follows: the ordinary processing times (without learning effect) of job $j$ are $a_j$, $b_j$, and $c_j$ on machines $M_1$, $M_2$, and $M_3$, respectively. Other assumptions are no machine can process more than one job at a time; no idle time is allowed on machines $M_1$ and $M_2$; and once a job has begun processing, it cannot be interrupted.

Following the works of Kuo and Yang [3] and Liu et al. [19], the real processing times of $J_j$, if $J_j$ scheduling on the position $r$ of a job sequence $S$, are considered as $a_j \max \{a, (1 + \sum_{k=1}^{r-1} a_{[k]}) \alpha\}$, $b_j \max \{a, (1 + \sum_{k=1}^{r-1} b_{[k]}) \beta\}$, and $c_j \max \{a, (1 + \sum_{k=1}^{r-1} c_{[k]}) \gamma\}$ on machines $M_1$, $M_2$, and $M_3$, respectively, where $\alpha$ denotes the controllable parameter with $\alpha < 0$, and $a_j$, $a_k$, $\alpha_k$ denote the learning indices for $M_1$, $M_2$, and $M_3$. For a schedule $S$, the finishing time of a job, say $J_i$, to be scheduled on the $r$th position (in $S$) on machine $M_3$ is defined, and denoted, as $C_{3[r]}(S)$:

$$
C_{3[r]}(S) = \max \left\{ t_1 + a_j \max \left\{ a, \left(1 + \sum_{k=1}^{r-1} a_{[k]} \right) \alpha \right\}, t_2 + b_j \max \left\{ a, \left(1 + \sum_{k=1}^{r-1} b_{[k]} \right) \beta \right\}, t_3 + c_j \max \left\{ a, \left(1 + \sum_{k=1}^{r-1} c_{[k]} \right) \gamma \right\} \right\},
$$

(1)

where $t_1$, $t_2$, and $t_3$ are denoted as the beginning times of $J_i$ on machines $M_1$, $M_2$, and $M_3$ in $S$, respectively. The optimal criterion of this study is to find a scheduled $S$ to minimize the makespan (or $C_{3[n]}(S)$).

3. Lower Bounds and Some Lemmas

This section proposes some useful lower bounds used in a branch-and-bound method. Before deriving the lower bounds, we define some notations. Suppose $t_1$, $t_2$, and $t_3$ denote the starting time of the first job in $S_{U3}$ on all three machines. Then, we assume that $s_{PS}$ are the partially scheduled $k$ jobs and $s_{U3}$ are the remaining unscheduled $(n-k)$ jobs. $a_{(1)} \leq a_{(2)} \leq \cdots \leq a_{(n-k)}$; $b_{(1)} \leq b_{(2)} \leq \cdots \leq b_{(n-k)}$; and $c_{(1)} \leq c_{(2)} \leq \cdots \leq c_{(n-k)}$ denote increasing sequences of $a_1, a_2, \ldots, a_{n-k}$; $b_1, b_2, \ldots, b_{n-k}$; and $c_1, c_2, \ldots, c_{n-k}$ in $s_{U3}$, respectively. Notably, $a_i, b_i, c_i$ are not necessarily from the same job, and $s_{P1}, s_{P2}$, and $s_{P3}$ are three subsequences of $n$ jobs $n$ increasing sequences of $a_1, a_2, \ldots, a_{n-k}$; $b_1, b_2, \ldots, b_{n-k}$; and $c_1, c_2, \ldots, c_{n-k}$ in $s_{U3}$, respectively.

Additionally, by using the ideas of Kuo and Yang [3] and Liu et al. [19], a lower bound for a subsequent can be yielded as follows:

| Parameter | $n=10$, $m=3$ |
|-----------|----------------|
| $T_i^*$ | $10^{-3}$ |
| $T_f^*$ | $10^{-8}$ |
| $e_i$ | 0.95 |
| $N_i$ | 10 |
| $\beta$ | 0.2 |
| $\alpha_1$ | -0.1 |
| $\alpha_2$ | -0.1 |
| $\alpha_3$ | -0.2 |

Table 1: Small size number of jobs parameters.
Lemma 1. If \( y \geq x > 0, \ T > 0, \) and \( e < 0, \) then \( y(T + x)^e \geq x(T + y)^e. \)

Lemma 2. If \( \lambda \geq 1, \ 0 \leq t, \) and \( e < 0, \) then \( \lambda [1 - (1 + t)^e] \geq 1 - (1 + \lambda t)^e. \)

Lemmas 1 and 2 have been proved by Kuo and Yang and Liu et al. [3, 19], respectively.

Lemma 3. If \( y \geq x > 0, \ T > 0, \ e < 0, \) and \( 0 < \alpha < 1, \) then \( y \cdot \max[a, (T + x)^e] \geq x \cdot \max[a, (T + y)^e]. \)

Proof. Note that \( (T + x)^e \geq (T + y)^e. \)

In a similar way,

\[
C_{(k+1)}(s) = \max \left\{ t_1 + a_{(k+1)} \max \left[ a, \left(1 + \sum_{j=1}^{k} a[j]\right)^{a_1} \right], t_2 + b_{(k+1)} \max \left[ a, \left(1 + \sum_{j=1}^{k} b[j]\right)^{a_2} \right] \right\}
\]

\[+ c_{(k+1)} \max \left[ a, \left(1 + \sum_{j=1}^{k} c[j]\right)^{a_3} \right],
\]

\[\geq t_1 + a_{(k+1)} \frac{\alpha + \left(1 + \sum_{j=1}^{k} a[j]\right)^{a_1}}{2} + c_{(k+1)} \frac{\alpha + \left(1 + \sum_{j=1}^{k} c[j]\right)^{a_3}}{2},
\]

\[C_{(k+2)}(s) \geq t_1 + a_{(k+1)} \frac{\alpha + \left(1 + \sum_{j=1}^{k} a[j]\right)^{a_1}}{2} + a_{(k+2)} \frac{\alpha + \left(1 + \sum_{j=1}^{k} a[j]\right)^{a_2}}{2} + c_{(k+2)} \frac{\alpha + \left(1 + \sum_{j=1}^{k} c[j]\right)^{a_3}}{2}
\]

\[
\vdots,
\]

\[C_{[k+n]}(s) \geq t_1 + \sum_{j=1}^{n} a_{(n-j+1)} \frac{\alpha + \left(1 + \sum_{j=1}^{n-j+1} a[j]\right)^{a_1}}{2} + c_{[k+n]} \frac{\alpha + \left(1 + \sum_{j=1}^{n-j+1} c[j]\right)^{a_3}}{2},
\]

\[\text{lower } bd_1 = t_1 + \sum_{j=1}^{n} a_{(n-j+1)} \frac{\alpha + \left(1 + \sum_{j=1}^{n-j+1} a[j]\right)^{a_1}}{2} + c_{(i)} \frac{\alpha + \left(1 + \sum_{j=1}^{n-j+1} c[j]\right)^{a_3}}{2},
\]

\[\text{lower } bd = \max\{\text{lower } bd_1, \text{lower } bd_2, \text{lower } bd_3\}.
\]

In what follows, we will propose four properties used in a branch-and-bound to improve its search power. Before deriving our properties, we first give four useful lemmas which will be used in the proof of Property 1.

Case 1. As \( \alpha > (T + x)^e, \)

\[
y \cdot \max[a, (T + x)^e] = y \cdot \alpha,
\]

\[
x \cdot \max[a, (T + y)^e] = x \cdot \alpha.
\]

So, the inequality holds.

Case 2. As \( (T + x)^e \leq \alpha > (T + y)^e, \)

\[
y \cdot \max[a, (T + x)^e] = y \cdot \alpha,
\]

\[
x \cdot \max[a, (T + y)^e] = x \cdot \alpha.
\]

The inequality holds.

Case 3. As \( (T + y)^e \geq \alpha, \)

\[
y \cdot \max[a, (T + x)^e] = y \cdot \alpha,
\]

\[
x \cdot \max[a, (T + y)^e] = x \cdot \alpha.
\]

From Lemma 1, the inequality holds.

Lemma 4. If \( \lambda \geq 1, \ t \geq 0, \ e < 0, \) and \( 0 < \beta < 1, \) then \( \lambda [\max\{\beta, 1\} - \max[\beta, (1 + t)^e]\} \geq \max[\beta, 1] - \max[\beta, (1 + \lambda t)^e]. \)

Proof. Note that \( (1 + \lambda t)^e \geq 1 + t > 0; \) thus, \( (1 + \lambda t)^e \leq (1 + t)^e \leq 1 \) because \( e < 0. \)

Case 1. As \( \beta \leq (1 + \lambda t)^e \leq 1, \)

\[
\lambda [\max[\beta, 1] - \max[\beta, (1 + t)^e]] = \lambda [1 - (1 + t)^e],
\]

\[
\max[\beta, 1] - \max[\beta, (1 + \lambda t)^e] = 1 - (1 + \lambda t)^e.
\]

From Lemma 2, the inequality holds.
Case 2. As $(1 + \lambda t)^\circ < \beta < (1 + t)^\circ \leq 1$,
\[
\lambda \left[ \max \{ \beta, 1 \} - \max \{ \beta, (1 + t)^\circ \} \right] = \lambda \left[ 1 - (1 + t)^\circ \right],
\]
max\{\beta, 1\} - max\{\beta, (1 + t)^\circ\} = 1 - \beta,

which is less than $1 - (1 + \lambda t)^\circ$. From Lemma 2, the equality holds.

Case 3. As $(1 + t)^\circ \leq \beta < 1$,
\[
\lambda \left[ \max \{ \beta, 1 \} - \max \{ \beta, (1 + t)^\circ \} \right] = \lambda \left[ 1 - \beta \right],
\]
max\{\beta, 1\} - max\{\beta, (1 + t)^\circ\} = 1 - \beta.

The equality holds because $\lambda \geq 1$.

Case 4. As $\beta \geq 1$,
\[
\lambda \left[ \max \{ \beta, 1 \} - \max \{ \beta, (1 + t)^\circ \} \right] = \lambda \left[ \beta - \beta \right] = 0,
\]
max\{\beta, 1\} - max\{\beta, (1 + t)^\circ\} = \beta - \beta = 0.

In summary, the inequality always holds for the given conditions.
To show max\{\beta, 1\} - max\{\beta, (1 + \lambda t)^\circ\} = \beta - \beta = 0 is no less than $S_2$, it suffices to show $C_{3j}(S_1) \leq C_{3j}(S_2)$.

Property 1. If \( a_j < a_j, b_j < b_j, c_j > c_j, \) \( t_j > \max\{t_1 + a_j, \max\{a_1, (1 + \sum_{i=1}^{k-1} a_i)^r\}, \max\{a_1, (1 + \sum_{i=1}^{k-1} b_i)^r\}\} \), and \( t_1 + a_j \max\{a_1, (1 + \sum_{i=1}^{k-1} a_i)^r\} + a_j \max\{a_1, (1 + \sum_{i=1}^{k-1} a_i)^r + a_j)^r\} > \max\{t_2 + b_j \max\{a_1, (1 + \sum_{i=1}^{k-1} b_i)^r\} + b_j \max\{a_1, (1 + \sum_{i=1}^{k-1} b_i)^r\}\}, \) then \( S_1 \) is no less than \( S_2 \).

Proof. For a job sequence \( S \), if the job \( j \) is assigned to the \( r \)-th position of \( S \), then according to the definition of the completion time of a job,

\[
C_{3j}(S) = \max \left\{ t_1 + a_j \max\{a_1, (TA)^r\}, t_2 + b_j \max\{a_1, (TB)^r, c_j \max\{a_1, (TC)^r\} \right\}.
\]

To simplify the proof, let \( TA = (1 + \sum_{i=1}^{k-1} a_i)^r \), \( TB = (1 + \sum_{i=1}^{k-1} b_i)^r \), and \( TC = (1 + \sum_{i=1}^{k-1} c_i)^r \). Then,

\[
C_{3j}(S_1) = \max\{t_1 + a_j \max\{a_1, (TA)^r\} + a_j \max\{a_1, (TA + a_j)^r\}, t_2 + b_j \max\{a_1, (TB)^r\} + b_j \max\{a_1, (TB + b_j)^r\},
\]
max\{t_3, t_1 + a_j \max\{a_1, (TA)^r\}, b_j \max\{a_1, (TB)^r\} + c_j \max\{a_1, (TC)^r\} \},

C_{3j}(S_2) = \max\{t_1 + a_j \max\{a_1, (TA)^r\} + a_j \max\{a_1, (TA + a_j)^r\}, t_2 + b_j \max\{a_1, (TB)^r\} + b_j \max\{a_1, (TB + b_j)^r\},
\]
max\{t_3, t_1 + a_j \max\{a_1, (TA)^r\}, b_j \max\{a_1, (TB)^r\} + c_j \max\{a_1, (TC)^r\} \},

Using the given conditions \( a_j < a_j, b_j < b_j, \) and \( t_j > \max\{t_1 + a_j \max\{a_1, (TA)^r\}, t_2 + b_j \max\{a_1, (TB)^r\}\} \),

one has

\[
C_{3j}(S_1) = \max\{t_1 + a_j \max\{a_1, (TA)^r\} + a_j \max\{a_1, (TA + a_j)^r\}, t_2 + b_j \max\{a_1, (TB)^r\} + b_j \max\{a_1, (TB + b_j)^r\},
\]
max\{t_3, t_1 + a_j \max\{a_1, (TA)^r\}, c_j \max\{a_1, (TC)^r\} \},

C_{3j}(S_2) = \max\{t_1 + a_j \max\{a_1, (TA)^r\} + a_j \max\{a_1, (TA + a_j)^r\}, t_2 + b_j \max\{a_1, (TB)^r\} + b_j \max\{a_1, (TB + b_j)^r\},
\]
max\{t_3, c_j \max\{a_1, (TC)^r\} \} + c_j \max\{a_1, (TC + c_j)^r\} \}.
Applying the given condition,

\[
t_1 + a_i \max\{a, (TA)^c_i\} + a_j \max\{a, (TA + a_j)^{c_i}\} > \max\{t_2 + b_i \max\{a, (TB)^{c_i}\} + b_j \max\{a, (TB + b_j)^{c_i}\},
\]
\[
t_3 + c_i \max\{a, (TC)^{c_i}\}\right),
\]

and then,

\[
C_{j_i} (S_i) = t_1 + a_j \max\{a, (TA)^{c_i}\} + a_i \max\{a, (TA + a_j)^{c_i}\} + c_i \max\{a, (TC + c_j)^{c_i}\},
\]

\[
C_{j_2} (S_2) = t_1 + a_j \max\{a, (TA)^{c_i}\} + a_i \max\{a, (TA + a_j)^{c_i}\} + c_i \max\{a, (TC + c_j)^{c_i}\}.
\]

Equation (16) holds by showing \(t_1 + a_j \max\{a, (TA)^{c_i}\} + a_i \max\{a, (TA + a_j)^{c_i}\} > [t_1 + a_j \max\{a, (TA)^{c_i}\} + a_i \max\{a, (TA + a_j)^{c_i}\}].\)

\[
a_j \max\{a, (TA)^{c_i}\} + a_i \max\{a, (TA + a_j)^{c_i}\} - a_i \max\{a, (TA + a_j)^{c_i}\} = a_i \left[\max\{a, (TA)^{c_i}\} - \max\{a, (TA + a_j)^{c_i}\}\right] - a_i \left[\max\{a, (TA + a_j)^{c_i}\} - \max\{a, (TA + a_j)^{c_i}\}\right],
\]

\[
-\max\left\{\frac{\alpha}{(TA)^{c_i}} \left(1 + \frac{a_j}{a_i} \frac{1}{(TA)^{c_i}}\right)^{c_i}\right\},
\]

\[
a_j \max\{a, (TA)^{c_i}\} + a_i \max\{a, (TA + a_j)^{c_i}\} = a_i \left[\max\{a, (TA)^{c_i}\} - \max\{a, (TA + a_j)^{c_i}\}\right] - \max\{a, (TA)^{c_i}\} - \max\{a, (TA + a_j)^{c_i}\}\right],
\]

where \(\lambda = (a_j/a_i) > 1\), \(\beta = (a_0/(TA)^{c_i}) > 0\), and \(t = (a_j/(TA)^{c_i}) > 0\). Applying Lemma 4, \(t_1 + a_j \max\{a, (TA)^{c_i}\} + a_i \max\{a, (TA + a_j)^{c_i}\} > [t_1 + a_i \max\{a, (TA + a_j)^{c_i}\} + a_i \max\{a, (TA + a_j)^{c_i}\}]\) follows and, thus, completes the claim.

Finally, one needs to show that \(C_{j_3} (S_3) - C_{j_3} (S_3) > 0\) to fulfill the proof. From equations (15) and (16), one obtains, applying \(a_i < a_j, c_j\) and the fact \((TA + a_j)^{c_i} < (TA + a_j)^{c_i}\) and \((TC + c_j)^{c_i} > (TC + c_j)^{c_i}\), that all three terms of \(C_{j_3} (S_3) - C_{j_3} (S_3)\) are greater than or equal to 0, i.e., \(C_{j_3} (S_3) - C_{j_3} (S_3) \geq 0\).

The details of the proofs of Property 2–4 are omitted because they can be obtained in a similar way to Property 1. \(\Box\)

**Property 2.** If \(a_i < a_j, b_j < b_j, c_j > c_j, \)
\[t_3 > \max\{t_1 + a_j \max\{a, (1 + \frac{a_j}{a_i})^a_k\}, t_2 + b_j \max\{a, (1 + \frac{b_j}{a_i})^b_j\}\}\right), \text{and} \ t_2 + b_j \max\{a, (1 + \frac{b_j}{a_i})^b_j\}] + b_j \max\{a, (1 + \frac{b_j}{a_i})^b_j\} + c_j \max\{a, (1 + \frac{c_j}{a_i})^c_j\} > t_3 + a_j \max\{a, (1 + \frac{b_j}{a_i})^b_j\} + c_j \max\{a, (1 + \frac{c_j}{a_i})^c_j\}\right), \text{and} \ t_2 + b_j \max\{a, (1 + \frac{b_j}{a_i})^b_j\}] + b_j \max\{a, (1 + \frac{b_j}{a_i})^b_j\} + c_j \max\{a, (1 + \frac{c_j}{a_i})^c_j\}\right), \text{and} \ t_2 + b_j \max\{a, (1 + \frac{b_j}{a_i})^b_j\}] + b_j \max\{a, (1 + \frac{b_j}{a_i})^b_j\} + c_j \max\{a, (1 + \frac{c_j}{a_i})^c_j\}\right), \text{and} \ t_2 + b_j \max\{a, (1 + \frac{b_j}{a_i})^b_j\}] + b_j \max\{a, (1 + \frac{b_j}{a_i})^b_j\} + c_j \max\{a, (1 + \frac{c_j}{a_i})^c_j\}\right), \text{and} \ t_2 + b_j \max\{a, (1 + \frac{b_j}{a_i})^b_j\} + c_j \max\{a, (1 + \frac{c_j}{a_i})^c_j\}\right). \]
4. Heuristics and Metaheuristics

This section validates the performance levels of nine algorithms, namely, JS_max, JS_min, JS_mean, JS_maxpi, JS_minpi, JS_meanpi, SA_max, SA_min, and SA_mean. Note that all algorithms were developed in FORTRAN and executed on a computer with a 3.00-GHz Intel® Core™ i9-10980XE CPU and 16-GB RAM. Our experiments are divided into two parts. The first part is dedicated to the small size instances, and the second is devoted to the large size instances. For the two cases, we randomly generated the normal processing times from uniform distribution U (1, 100) on M1, M2, and M3 by fixing an individual seed for each case. The factor β would be set at 0.2, 0.4, and 0.6. The learning effect would be set at −0.1, −0.1, and −0.2, and the computational results for each small and large size instances are shown. This section describes the considered two-stage assembly scheduling problem.

4.1. Small Size Number of Jobs. The parameters of the small size number of jobs are selected as the number of work pieces (n) at 10, number of machines (m) of 3, and the final temperature (Tf) is fixed to 10−8, β is 0.2, α1 is −0.1, α2 is −0.1, and α3 is −0.2. The selections of other parameters are shown in Table 1.

As a result of parameter selection, the starting temperature (Ti) is 10−3, cooling coefficient is 0.95, and the number of tests (Nr) of Johnson’s algorithm is 10.

Next, the parameter selection process is explained in detail.

We select the parameter starting temperature (Ti). The range is from 10−4 to 104 at 10 times the interval, and the cooling coefficient (cf) is fixed at 0.95. The number of tests of Johnson’s algorithm (Nr) is 5, and there are also generated data and parameters defined before selecting parameters.

By observing Figure 1, it is obvious that only the initial temperature (Tf) is changed in this range. It can be observed that the AEP is relatively low at 10−3, so the initial temperature (Tf) is 10−3.

After fixing the starting temperature (Tf) to 10−3, we select the parameter cooling coefficient (cf), ranging from 0.05 to 0.95 with 0.05 intervals, and fix the number of tests (Nr) of the Johnson’s algorithm as 5. There are also generated data and parameters defined before selecting the parameters.

By selecting the parameter of the cooling coefficient (cf), the overall decline in its value is obvious. It becomes flat after reaching 0.80, so we choose a lower value in this part. The cooling coefficient (cf) is fixed at 0.95. After fixing the starting temperature (Tf) to 10−3 and the cooling coefficient (cf) to 0.95 (please see Figure 2), we select the parameter Johnson’s algorithm test times (Nr) with a range of 1 to 20 and the interval of 1 and see the difference. The generated data and parameters are fixed and defined before selecting the parameters.

When selecting the parameter Johnson’s algorithm test times (Nr), we can see from Figure 3 that although it fluctuates up and down, its AEP value is actually very low, so we choose a lower value in this part. Johnson’s algorithm test times (Nr) is 10. It shows when the number of work pieces of the B&B algorithm is 12. The results of finding the best solution in a few cases cannot be found, so only a few values (in the Fs column) are recorded. From the viewpoint of the number of nodes and CPU time, the difference in the number of jobs can also be clearly seen.

Table 2 uses the β values of 0.2, 0.4, and 0.6 to distinguish the table. It can be seen regardless of whether the number of work pieces is 8, 10, or 12, the smaller the β value, the greater the number of nodes. Also, there is no best solution for the CPU time.

Table 3 is a table distinguished by three different situations of α1, α2, α3. It can be observed that regardless of whether the number of work pieces is 8, 10, or 12, the
Figure 3: Parameter selection of Johnson’s algorithm test times (N_r) change.

Table 2: Different β for the mean of nodes and CPU time.

| Parameter | Node Mean | CPU time Mean |
|-----------|-----------|---------------|
| 8         |           |               |
| 0.2       | 9871      | 0.11          |
| 0.4       | 4474      | 0.05          |
| 0.6       | 2825      | 0.03          |
| 10        |           |               |
| 0.2       | 751491    | 10.33         |
| 0.4       | 262445    | 3.89          |
| 0.6       | 196723    | 3.13          |
| 12        |           |               |
| 0.2       | 51984213  | 978.76        |
| 0.4       | 27751022  | 585.76        |
| 0.6       | 16943700  | 327.17        |

Table 3: Different β for the mean of nodes and CPU time.

| Parameter | Node Mean | CPU time Mean |
|-----------|-----------|---------------|
| n         | α_1       | α_2          | α_3 |
| 8         |           |               |     |
| −0.1      | −0.1      | −0.1         | 4448 | 0.06 |
| −0.2      | −0.2      | −0.1         | 4861 | 0.05 |
| 10        |           |               |     |
| −0.1      | −0.2      | −0.1         | 608118 | 8.73 |
| 12        |           |               |     |
| −0.2      | −0.2      | −0.1         | 32651016 | 649.04 |

The simulation result for the small size number of jobs from the total mean at the bottom shows that the SA-related algorithm performs better in the small size number of jobs.

Table 4 has β values of 0.2, 0.4, and 0.6, and no obvious features are noted.

4.2 Large Size Number of Jobs. The parameters of the large size number of jobs are selected when the number of work pieces (n) is 10, the number of machines (m) is 3, and the final temperature (T_f) is fixed to 10^{-8}. β is 0.2, α_1 is −0.1, α_2 is −0.1, and α_3 is −0.2. The selections of other parameters are shown in Table 10.

From the summary of Tables 11 and 12, the SA algorithm is the best in calculating the RPD. However, the CPU_time shows that the SA algorithm takes a relatively long time. With pi or without pi, the JS algorithm clearly shows that the solution with pi to the run out of RPD is slightly inferior to the solution without pi. The JS algorithm with pi for the CPU_time is the fastest.

Table 13 shows the residual error in the large size number of jobs is not in line with the normal state, so we also...
use the same method as the small size number of jobs to compare the relationship between the algorithms.

In the Kruskal–Wallis test, the null hypothesis is rejected as with the small size number of jobs, meaning the median of each group is not the same and multiple comparison analysis

### Table 4: Simulation result for $\beta$ values.

| Parameter | JS_max Mean | JS_min Mean | JS_mean Mean | JS_maxpi Mean | JS_minpi Mean | JS_meanpi Mean | SA_max Mean | SA_min Mean | SA_mean Mean |
|-----------|-------------|-------------|--------------|---------------|---------------|---------------|-------------|-------------|--------------|
| $N$       |             |             |              |               |               |               |             |             |              |
| 8         | 2.92        | 6.69        | 1.96         | 1.81          | 4.43          | 1.15          | 0.10        | 0.09        | 0.08         |
| 0.2       | 1.95        | 5.94        | 1.67         | 1.21          | 4.18          | 1.12          | 0.05        | 0.06        | 0.05         |
| 0.4       | 2.09        | 6.70        | 1.60         | 1.33          | 4.95          | 1.01          | 0.06        | 0.06        | 0.06         |
| 0.6       | 1.59        | 6.11        | 1.38         | 1.09          | 4.40          | 0.93          | 0.07        | 0.10        | 0.08         |
| 0.2       | 1.70        | 5.99        | 1.36         | 1.04          | 4.75          | 0.88          | 0.11        | 0.08        | 0.10         |
| 0.4       | 1.59        | 5.93        | 1.40         | 0.97          | 4.59          | 0.93          | 0.05        | 0.09        | 0.08         |
| 0.6       | 1.32        | 6.24        | 1.26         | 0.92          | 4.88          | 0.95          | 0.10        | 0.09        | 0.10         |
| Mean      | 2.03        | 6.51        | 1.68         | 1.28          | 4.75          | 1.10          | 0.08        | 0.08        | 0.07         |

### Table 5: Simulation result for $\alpha$ values.

| Parameter | JS_max Mean | JS_min Mean | JS_mean Mean | JS_maxpi Mean | JS_minpi Mean | JS_meanpi Mean | SA_max Mean | SA_min Mean | SA_mean Mean |
|-----------|-------------|-------------|--------------|---------------|---------------|---------------|-------------|-------------|--------------|
| $n$       |             |             |              |               |               |               |             |             |              |
| 8         | 1.71        | 5.14        | 1.60         | 1.34          | 4.53          | 0.98          | 0.05        | 0.06        | 0.06         |
| 0.2       | 1.48        | 5.52        | 1.37         | 1.04          | 3.20          | 0.93          | 0.06        | 0.06        | 0.06         |
| 0.6       | 2.58        | 9.01        | 2.43         | 1.80          | 6.04          | 1.88          | 0.13        | 0.11        | 0.10         |
| Mean      | 2.03        | 6.51        | 1.68         | 1.28          | 4.75          | 1.10          | 0.08        | 0.08        | 0.07         |

### Table 6: Small size number of jobs analysis variable.

| Class level information | Values |
|-------------------------|--------|
| Algorithm               | JS_max, JS_maxπ, JS_mean, JS_meanπ, JS_min, JS_minπ, SA_max, SA_mean, SA_min, and SA_mean |
| $N$                     | 3, 8, 10, 12 |
| $\beta$                | 0.2, 0.4, 0.6 |
| $\alpha_1$             | 2, -0.1, -0.2 |
| $\alpha_2$             | 2, -0.1, -0.2 |
| $\alpha_3$             | 2, -0.1, -0.2 |

### Table 7: Normality test of residual error of the GLM model in the small size number of jobs.

| Normality test | Statistics | $P$ value |
|----------------|------------|-----------|
| Shapiro–Wilk   | $W$        | 0.936652  | <0.0001   |
| Kolmogorov–Smirnov | $D$     | 0.086882  | <0.0000   |
| Cramer–von Mises | $W$-Sq   | 0.532874  | <0.0050   |
| Anderson–Darling | $A$-Sq   | 3.734124  | <0.0050   |

### Table 8: Kruskal–Wallis verification of the small size number of jobs.

| Kruskal–Wallis test | $Pr > ChiSq$ |
|---------------------|--------------|
| Chi-square          | 214.5511     | <0.0001    |
| DF                  | 8            |            |
can be performed to show the differences between each group (please see Table 14).

Figure 5 is a box diagram of nine algorithms scored by nonparametric statistics. The figure shows clear differences among SAs and others under the large size number of jobs.

Table 15 shows the results of the pairwise comparison. If the $P$ value value is greater than 0.05, it means the effect between the pair is not significant, but they are correlative. In

| Algorithm      | Statistic | $P$ value |
|----------------|-----------|-----------|
| $JS_{\text{max}}$ vs. $JS_{\text{min}}$ | 8.5999 | <0.0001 |
| $JS_{\text{max}}$ vs. $JS_{\text{mean}}$ | 1.8843 | 0.9218 |
| $JS_{\text{max}}$ vs. $JS_{\text{maxpi}}$ | 5.3095 | 0.0054 |
| $JS_{\text{max}}$ vs. $JS_{\text{minpi}}$ | 7.7925 | <0.0001 |
| $JS_{\text{max}}$ vs. $SA_{\text{max}}$ | 6.9494 | <0.0001 |
| $JS_{\text{max}}$ vs. $SA_{\text{min}}$ | 8.9241 | <0.0001 |
| $JS_{\text{max}}$ vs. $SA_{\text{mean}}$ | 8.9268 | <0.0001 |
| $JS_{\text{max}}$ vs. $SA_{\text{max}}$ | 8.9231 | <0.0001 |
| $JS_{\text{max}}$ vs. $SA_{\text{mean}}$ | 8.7717 | <0.0001 |
| $JS_{\text{min}}$ vs. $JS_{\text{maxpi}}$ | 7.7925 | <0.0001 |
| $JS_{\text{min}}$ vs. $JS_{\text{minpi}}$ | 6.9494 | <0.0001 |
| $JS_{\text{min}}$ vs. $JS_{\text{meanpi}}$ | 8.9241 | <0.0001 |
| $JS_{\text{min}}$ vs. $SA_{\text{max}}$ | 8.9268 | <0.0001 |
| $JS_{\text{min}}$ vs. $SA_{\text{min}}$ | 8.9231 | <0.0001 |
| $JS_{\text{min}}$ vs. $SA_{\text{mean}}$ | 8.7717 | <0.0001 |
| $JS_{\text{min}}$ vs. $SA_{\text{max}}$ | 8.9246 | <0.0001 |
| $JS_{\text{min}}$ vs. $SA_{\text{min}}$ | 8.9273 | <0.0001 |
| $JS_{\text{min}}$ vs. $SA_{\text{mean}}$ | 8.9236 | <0.0001 |
| $JS_{\text{maxpi}}$ vs. $JS_{\text{minpi}}$ | 8.7224 | <0.0001 |
| $JS_{\text{maxpi}}$ vs. $JS_{\text{meanpi}}$ | 7.8666 | 0.9418 |
| $JS_{\text{maxpi}}$ vs. $SA_{\text{max}}$ | 8.9243 | <0.0001 |
| $JS_{\text{maxpi}}$ vs. $SA_{\text{min}}$ | 8.9270 | <0.0001 |
| $JS_{\text{maxpi}}$ vs. $SA_{\text{mean}}$ | 8.9233 | <0.0001 |
| $JS_{\text{minpi}}$ vs. $JS_{\text{meanpi}}$ | 8.8449 | <0.0001 |
| $JS_{\text{minpi}}$ vs. $SA_{\text{max}}$ | 8.9239 | <0.0001 |
| $JS_{\text{minpi}}$ vs. $SA_{\text{min}}$ | 8.9267 | <0.0001 |

Table 9: Pairwise comparison of nine algorithms for the small size number of jobs.

| Algorithm      | Statistic | $P$ value |
|----------------|-----------|-----------|
| $JS_{\text{minpi}}$ vs. $SA_{\text{mean}}$ | 8.9229 | <0.0001 |
| $JS_{\text{meanpi}}$ vs. $SA_{\text{max}}$ | 8.9244 | <0.0001 |
| $JS_{\text{meanpi}}$ vs. $SA_{\text{min}}$ | 8.9272 | <0.0001 |
| $JS_{\text{meanpi}}$ vs. $SA_{\text{mean}}$ | 8.9234 | <0.0001 |
| $SA_{\text{max}}$ vs. $SA_{\text{min}}$ | 0.6278 | 1.0000 |
| $SA_{\text{max}}$ vs. $SA_{\text{mean}}$ | 0.4790 | 1.0000 |
| $SA_{\text{min}}$ vs. $SA_{\text{mean}}$ | 1.1795 | 0.9999 |

Table 9: Continued.

| Algorithm      | Statistic | $P$ value |
|----------------|-----------|-----------|
| $JS_{\text{minpi}}$ vs. $SA_{\text{mean}}$ | 8.9229 | <0.0001 |
| $JS_{\text{meanpi}}$ vs. $SA_{\text{max}}$ | 8.9244 | <0.0001 |
| $JS_{\text{meanpi}}$ vs. $SA_{\text{min}}$ | 8.9272 | <0.0001 |
| $JS_{\text{meanpi}}$ vs. $SA_{\text{mean}}$ | 8.9234 | <0.0001 |
| $SA_{\text{max}}$ vs. $SA_{\text{min}}$ | 0.6278 | 1.0000 |
| $SA_{\text{max}}$ vs. $SA_{\text{mean}}$ | 0.4790 | 1.0000 |
| $SA_{\text{min}}$ vs. $SA_{\text{mean}}$ | 1.1795 | 0.9999 |

Table 10: Parameter selection of the large size number of jobs.

| Parameter | $n = 90, m = 3$ |
|-----------|-----------------|
| $T_f$ | $10^{-3}$ |
| $T_f$ | $10^{-8}$ |
| $\alpha_1$ | 0.95 |
| $N_f$ | 90 |
| $\beta$ | 0.2 |
| $\alpha_1$ | 0.1 |
| $\alpha_2$ | 0.1 |
| $\alpha_3$ | 0.2 |

Figure 4: Scoring box of the nine algorithms.
Table 11: RPD summary of the large size number of jobs.

| Parameter | $\beta$ | $\alpha$ | Mean | Max | Mean | Max | Mean | Max | Mean | Max | Mean | Max | Mean | Max | Mean | Max |
|-----------|---------|---------|------|-----|------|-----|------|-----|------|-----|------|-----|------|-----|------|-----|
| N         | -0.1    | -0.1   | 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37|
| $\alpha_1$| -0.1    | -0.1   | 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37|
| $\alpha_2$| -0.1    | -0.1   | 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37|
| $\alpha_3$| -0.1    | -0.1   | 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37| 0.20 | 0.37|

Mean: 0.13 1.40 0.18 0.23 1.58 0.26 0.12 0.16 0.09 0.12

Total mean: 0.13 1.40 0.18 0.23 1.58 0.26 0.12 0.16 0.09 0.12
| Parameter | JS_max | JS_min | JS_mean | JS_maxpi | JS_minpi | JS_meanpi | SA_max | SA_min | SA_mean |
|-----------|--------|--------|---------|----------|----------|-----------|--------|--------|---------|
| n         | β      | α      | α      | Mean     | Max      | Mean      | Max    | Mean    | Max     |
| 60        | -0.1   | -0.1   | -0.1   | 0.02     | 0.03     | 0.02      | 0.03   | 0.00    | 0.00    | 0.19    | 0.20    | 0.19    | 0.22    |
| 0.2       | -0.1   | -0.1   | -0.2   | 0.02     | 0.03     | 0.02      | 0.03   | 0.00    | 0.00    | 0.19    | 0.20    | 0.19    | 0.22    |
| 0.6       | -0.1   | -0.1   | -0.2   | 0.02     | 0.03     | 0.02      | 0.03   | 0.00    | 0.00    | 0.19    | 0.20    | 0.19    | 0.22    |
| Mean      | -0.1   | -0.1   | -0.1   | 0.06     | 0.08     | 0.06      | 0.08   | 0.00    | 0.00    | 0.28    | 0.30    | 0.28    | 0.30    |
| 0.2       | -0.1   | -0.1   | -0.2   | 0.06     | 0.06     | 0.06      | 0.06   | 0.00    | 0.00    | 0.28    | 0.30    | 0.29    | 0.31    |
| 0.6       | -0.1   | -0.1   | -0.2   | 0.06     | 0.06     | 0.06      | 0.06   | 0.00    | 0.00    | 0.28    | 0.30    | 0.28    | 0.30    |
| Mean      | -0.1   | -0.1   | -0.1   | 0.13     | 0.14     | 0.13      | 0.14   | 0.00    | 0.00    | 0.38    | 0.39    | 0.38    | 0.41    |
| 0.2       | -0.1   | -0.1   | -0.2   | 0.13     | 0.14     | 0.13      | 0.14   | 0.00    | 0.00    | 0.38    | 0.41    | 0.39    | 0.39    |
| 0.6       | -0.1   | -0.1   | -0.2   | 0.14     | 0.14     | 0.14      | 0.14   | 0.00    | 0.02    | 0.39    | 0.41    | 0.39    | 0.42    |
| Mean      | -0.1   | -0.1   | -0.1   | 0.14     | 0.13     | 0.14      | 0.14   | 0.00    | 0.00    | 0.39    | 0.42    | 0.39    | 0.41    |
| 0.2       | -0.1   | -0.1   | -0.2   | 0.13     | 0.14     | 0.13      | 0.14   | 0.00    | 0.02    | 0.39    | 0.41    | 0.39    | 0.42    |
| 0.6       | -0.1   | -0.1   | -0.2   | 0.13     | 0.13     | 0.13      | 0.13   | 0.00    | 0.02    | 0.39    | 0.41    | 0.39    | 0.42    |
| Mean      | -0.1   | -0.1   | -0.1   | 0.14     | 0.13     | 0.14      | 0.14   | 0.00    | 0.00    | 0.38    | 0.38    | 0.38    | 0.38    |
| Total mean| 0.14   | 0.13   | 0.13   | 0.00     | 0.00     | 0.00      | 0.00   | 0.38    | 0.38    | 0.38    | 0.38    | 0.38    | 0.38    |
Table 13: Normality test of residual error of the GLM model with the large size number of jobs.

| Hypothesis test          | Statistics | P value |
|--------------------------|------------|---------|
| Shapiro–Wilk             | W          | 0.916964| Pr < W     | <0.0001 |
| Kolmogorov–Smirnov       | D          | 0.120337| Pr < D     | <0.0100 |
| Cramer–von Mises         | W-Sq       | 0.862822| Pr < W-Sq  | <0.0050 |
| Anderson–Darling         | A-Sq       | 5.519845| Pr < A-Sq  | <0.0050 |

Table 14: Kruskal–Wallis test of the small size number of jobs.

| Kruskal–Wallis test | Chi-square | DF | Pr > ChiSq |
|---------------------|------------|----|------------|
|                     | 204.1873   | 8  | <0.0001    |

Figure 5: Scoring box of nine algorithms.

Table 15: Pairwise comparison of nine algorithms for the large size number of jobs.

| Algorithm    | Statistic | P value |
|--------------|-----------|---------|
| JS_max vs. JS_min | 8.9219   | <0.0001 |
| JS_max JS_mean | 1.5462   | 0.9754  |
| JS_max JS_maxpi | 2.6857  | 0.6144  |
| JS_max JS_minpi | 8.9231  | <0.0001 |
| JS_max JS_meanpi | 3.0313  | 0.4434  |
| JS_max SA_max  | 8.2129   | <0.0001 |
| JS_max vs. SA_min | 6.3166 | 0.0003  |
| JS_max SA_mean | 8.4541   | <0.0001 |
| JS_min JS_mean | 8.9231   | <0.0001 |
the pairwise comparison, JS_max and JS_mean are in the same group, JS_max and JS_maxpi are in the same group, JS_max and JS_meanpi are in the same group, JS_min and JS_minpi are in the same group, JS_mean and JS_meanpi are in the same group, JS_maxpi and JS_meanpi are in the same group, and SA_max and the SA_mean are in the same group. These results of the SA algorithm and JS algorithms are also significant, so they would not be divided into the same group. The RPD summary of the large size number of jobs shows the solution of the SA algorithm is comparatively better. The SA algorithm is better than the JS algorithm in this case.

5. Conclusions

This study focuses on two-stage three-machine flow shop assembly problems mixed with a controllable number and sum-of-processing times-based learning effect, in which job processing time is considered to be a function of the control of the truncation parameter and learning based on the sum of the processing time. We derive several dominance rules, lemmas, and lower bounds applied in the branch-and-bound method. On the other hand, three simulated annealing algorithms are proposed for finding approximate solutions. Computational results show that the SA algorithm is relatively better for the small size number of jobs. In the large size number of jobs, the solution of the SA algorithm is comparatively better. Both in the small and large size number of jobs situation, the SA algorithm is better than the JS algorithm in this study. For future research, it would be interesting to develop more powerful dominance rules and a sharper lower bound on the optimal solution for medium-size instances.

Data Availability

All data are available on request.

Conflicts of Interest

The author declares that there are no conflicts of interest regarding the publication of this paper.

References

[1] M. Pinedo, Scheduling: Theory, Algorithms and Systems, Prentice-Hall, Upper Saddle River, NJ, USA, third edition, 2008.
[2] D. Biskup, “Single-machine scheduling with learning considerations,” European Journal of Operational Research, vol. 115, no. 1, pp. 173–178, 1999.
[3] W.-H. Kuo and D.-L. Yang, “Minimizing the total completion time in a single-machine scheduling problem with a time-dependent learning effect,” European Journal of Operational Research, vol. 174, no. 2, pp. 1184–1190, 2006.
[4] A. Azzouz, M. Ennigrou, and L. Ben Said, “Scheduling problems under learning effects: classification and cartography,” International Journal of Production Research, vol. 56, no. 4, pp. 1642–1661, 2018.
[5] D. Biskup, “A state-of-the-art review on scheduling with learning effects,” *European Journal of Operational Research*, vol. 188, no. 2, pp. 315–329, 2008.

[6] Y. Yin, T. C. E. Cheng, and C.-C. Wu, “Scheduling with time-depending processing times,” *Mathematical Problems in Engineering*, vol. 2015, Article ID 367585, 2 pages, 2015.

[7] C. Kuolamas and G. J. Kyparisis, “Single-machine and two-machine flowshop scheduling with general learning functions,” *European Journal of Operational Research*, vol. 178, no. 2, pp. 402–407, 2007.

[8] P. Chen, C.-C. Wu, and W.-C. Lee, “A bi-criteria two-machine flowshop scheduling problem with a learning effect,” *Journal of the Operational Research Society*, vol. 57, no. 9, pp. 1113–1125, 2006.

[9] M. C. Isler, B. Toklu, and V. Celik, “Scheduling in a two-machine flow-shop for earliness/tardiness under learning effect.” *The International Journal of Advanced Manufacturing Technology*, vol. 61, no. 9-12, pp. 1129–1137, 2012.

[10] W. Lee and C.-C. Wu, “Minimizing total completion time in a two-machine flowshop with a learning effect,” *International Journal of Production Economics*, vol. 88, no. 1, pp. 85–93, 2004.

[11] J.-B. Wang and Z.-Q. Xia, “Flow-shop scheduling with a learning effect,” *Journal of the Operational Research Society*, vol. 56, no. 11, pp. 1325–1330, 2005.

[12] C.-C. Wu, S.-C. Liu, T. C. E. Cheng, Y. Cheng, S.-Y. Liu, and W.-C. Lin, “Re-entrant flowshop scheduling with learning considerations to minimize the makespan,” *Iranian Journal of Science and Technology, Transactions A: Science*, vol. 42, no. 2, pp. 727–744, 2018a.

[13] C.-C. Wu, D.-J. Wang, S.-R. Cheng, I.-H. Chung, and W.-C. Lin, “A two-stage three-machine assembly scheduling problem with a position-based learning effect,” *International Journal of Production Research*, vol. 56, no. 9, pp. 3064–3079, 2018b.

[14] C.-C. Wu, J.-Y. Chen, W.-C. Lin, K. Lai, S.-C. Liu, and P.-W. Yu, “A two-stage three-machine assembly flow shop scheduling with learning consideration to minimize the flowtime by six hybrids of particle swarm optimization,” *Swarm and Evolutionary Computation*, vol. 41, pp. 97–110, 2018c.

[15] W.-H. Wu, Y. Yin, T. C. E. Cheng et al., “A combined approach for two-agent scheduling with sum-of-processing-times-based learning effect,” *Journal of the Operational Research Society*, vol. 68, no. 2, pp. 111–120, 2017.

[16] Y. Zou, W.-C. Wang, J.-Y. Lin et al., “Two-stage three-machine assembly scheduling problem with sum-of-processing-times-based learning effect,” *Soft Computing*, vol. 24, no. 7, pp. 5445–5462, 2020.

[17] C.-C. Wu, J.-Y. Chen, W.-C. Lin, K. Lai, D. Bai, and S.-Y. Lai, “A two-stage three-machine assembly scheduling flowshop problem with both two-agent and learning phenomenon,” *Computers & Industrial Engineering*, vol. 130, pp. 485–499, 2019a.

[18] C.-C. Wu, D. Bai, A. Azzouz et al., “A branch-and-bound algorithm and four metaheuristics for minimizing total completion time for atwo-stage assembly flow-shop scheduling problem with learning consideration,” *Engineering Optimization*, vol. 52, no. 6, pp. 1009–1103, 2020.

[19] S.-C. Liu, J. Duan, W.-C. Lin et al., “A branch-and-bound algorithm for two-agent scheduling with learning effect and late work criterion,” *Asia-Pacific Journal of Operational Research*, vol. 35, no. 5, pp. 1–24, 2018.