Hindawi
Discrete Dynamics in Nature and Society
Volume 2018, Article ID 8170294, 15 pages
https://doi.org/10.1155/2018/8170294

Research Article

Minimizing the Makespan for a Two-Stage Three-Machine Assembly Flow Shop Problem with the Sum-of-Processing-Time Based Learning Effect

Win-Chin Lin

Department of Statistics, Feng Chia University, Taichung 40724, Taiwan

Correspondence should be addressed to Win-Chin Lin; linwc@fcu.edu.tw

Received 30 January 2018; Accepted 8 April 2018; Published 23 May 2018

Academic Editor: Manuel De la Sen

Copyright © 2018 Win-Chin Lin. 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.

Two-stage production process and its applications appear in many production environments. Job processing times are usually assumed to be constant throughout the process. In fact, the learning effect accrued from repetitive work experiences, which leads to the reduction of actual job processing times, indeed exists in many production environments. However, the issue of learning effect is rarely addressed in solving a two-stage assembly scheduling problem. Motivated by this observation, the author studies a two-stage three-machine assembly flow shop problem with a learning effect based on sum of the processing times of already processed jobs to minimize the makespan criterion. Because this problem is proved to be NP-hard, a branch-and-bound method embedded with some developed dominance propositions and a lower bound is employed to search for optimal solutions. A cloud theory-based simulated annealing (CSA) algorithm and an iterated greedy (IG) algorithm with four different local search methods are used to find near-optimal solutions for small and large number of jobs. The performances of adopted algorithms are subsequently compared through computational experiments and nonparametric statistical analyses, including the Kruskal–Wallis test and a multiple comparison procedure.

1. Introduction

This study considers a two-stage assembly scheduling problem consisting of n jobs. Each of the n jobs has two parts (components) to be processed first, and the parts are then assembled on one assembly operation into an ordered product. At the first stage, two distinct parts can be fabricated at the same time in two dedicated machines, M_1 and M_2. Once the parts are finished, they are transmitted to the second stage, the assembly stage. There is a single assembly machine, M_3, in this stage to assemble parts into the ordered product. The assembly work can only start after the two parts are completed and delivered to the assembly line. The objective of the production plan is to minimize the maximum completion time among all the jobs, i.e., to minimize the makespan.

Wu et al. [1] studied the makespan minimization in the two-stage three-machine flow shop problem with position-based learning consideration based on Biskup [2]. Wu etal. [3] addressed a two-stage three-machine flow shop scheduling problem with a cumulated learning function to minimize the flowtime (or total completion time). They developed a branch-and-bound algorithm and proposed six versions of hybrid particle swam optimization (PSO) to solve it. However, the sum-of-processing-times-based learning model (based on [4]) has yet to be considered in two-stage assembly flow shop environment. In light of the lack of studies on the sum-of-processing-times-based learning model, this research studies a two-stage three-machine assembly problem with a learning effect based on sum of the processing times of already processed jobs to minimize the makespan criterion.

The paper proceeds as follows. Section 2 presents some notations and problem definition. Section 3 provides some dominant properties and a lower bound to be used in a branch-and-bound method and then introduces a cloud theory-based simulated annealing (CSA) and four versions of the iterated greedy (IG) algorithm. Section 4 provides computational results. Section 5 is the conclusion.

1.1. Literature Survey. Invoking a real-world example of fire engine manufacturing and assembly procedure, Lee et al. [5] studied the issue of minimizing the makespan in the three-machine assembly-type flow shop scheduling problem. A fire engine consists of two parts, a body and a chassis, which are manufactured separately in two production lines in the plant; an engine, bought from outside, can be considered ready. Once the body and chassis are completed, they will be transmitted to the assembly line, where the end product, a fire engine, is built up. They proved the problem is strongly NP-complete, discussed some special polynomial-time solvable cases, and constructed exact and heuristics for them. Potts et al. [6] also demonstrated the complexity of the proposed problem with several machines in the first stage. They introduced several heuristic algorithms and presented the worst-case analysis of performances of these algorithms. Hariri and Potts [7] developed a branch-and-bound method, embedded with some dominance properties, for the considered problem with several machines in the first stage. Sun et al. [8] proposed 14 heuristics algorithms to solve the problem.

Allahverdi and Al-Anzi [9] proposed two evolutionary algorithms, a particle swarm optimization (PSO) and a tabu search, and a simple efficient algorithm to solve the considered problem. Koulamas and Kyparisis [10] studied a two-stage assembly flow shop scheduling problem with uniform parallel machines on one or both stages. They presented an asymptotically optimal heuristic to minimize the makespan, as the number of jobs increase. Their results improved earlier results in some situations. Fattahi et al. [11] considered the same problem but all machines in the first (possibly including two manufacturing stages) stage(s) can process all kinds of parts, named a hybrid flow shop in the first stage(s). They proposed several heuristics based on Johnson's rule and evaluated the performance of them. Fattahi et al. [12] also considered the hybrid flow shop problem but with setup time and presented a hierarchical branch-and-bound algorithm with several lower bounds and upper bounds. Hatami et al. [13] addressed a distributed assembly permutation scheduling problem in which the first stage comprised several identical flow shop factories. These factories, each with a permutation flow shop scheduling problem, produced parts which were later assembled by a single machine in the assembly stage. Komaki et al. [14] addressed a two-manufacturing-stage hybrid flow shop scheduling problem and one assembly stage to minimize the makespan, where the hybrid meant identical parallel machines or dedicated machines could present in the fabrication stages. They presented a lower bound, several heuristics, and two metaheuristic algorithms based on an artificial immune system. Their computational results showed the outperformance of the proposed lower bound and heuristic algorithms. Jung et al. [15] considered a slightly different two-stage assembly scheduling problem. A single manufacturing machine was used in the first stage to produce various types of parts which were assembled into various products (ordered by customers) in the second stage. The aforementioned studies addressed the problem with respect to the makespan criterion.

With regard to performance criteria other than the makespan for two stages or more stages assembly flow shop problems, Allahverdi and Aydilek [16] provided research works focusing on the following criteria: mean (or total) completion time, a weighted sum of mean completion time and makespan, maximum lateness, total tardiness, and bicriteria. Some of these studies were together with other machine settings, for example, setup times often treated as separate ones from processing times or due dates required by customers. For instance, Maleki-Darounkolaei et al. [17] minimized the weighted mean completion time and makespan for the proposed problem with sequence-dependent setup times and blocking times. For more studies on different criteria and different machine settings for two-stage or three-stage assembly flow shop problem, readers can refer to Al-Anzi and Allahverdi [18], Allahverdi and Al-Anzi [19], Al-Anzi and Allahverdi [20], Allahverdi and Al-Anzi [21], Fattahi et al. [11], Allahverdi and Aydilek [16], and Jung et al. [15], among others.

Several real-life examples or applications for two-stage and three-stage assembly problem can be found in Lee et al. [5], Potts et al. [6], Tozkapan et al. [22], Allahverdi and Al-Anzi [19], Al-Anzi and Allahverdi [20], Sung and Kim [23], Allahverdi and Al-Anzi [24], Cheng et al. [25], Fattahi et al. [11], Mozdgir et al. [26], Fattahi et al. [12], Hatami et al. [13], and Jung et al. [15], among others.

On the other hand, the learning concept has been widely addressed in different branches of industry, for example, product development, job rotation, and job scheduling problem [27]. The first application in scheduling literature appeared in airplane's manufacturing, where the assembly costs reduced as the times of repetitions increased [28]. Yelle [29] gave a comprehensive survey on learning curves applied in many industries including manufacturing sector. In addition to the excellent review on scheduling with learning effect by Biskup [30], Anzanello and Fogliatto [27] also gave a review on the wide applicability of learning curve models in production systems. The work of Azzouz et al. [31] is a more recent survey paper in the scheduling literature providing applications of learning effects.

In scheduling techniques, Biskup [2] and Cheng and Wang [32] were some of the early researchers who proposed modeling a learning effect by a learning curve (function). Since then, many other learning curves (functions) in connection with a single machine, flow shop, or other scheduling problems, have been developed and modified.

Kuo and Yang work [4], one of the extensively discussed studies on the scheduling research community, proposed the sum-of-processing-times-based learning model, in which the real processing time of a job is relevant to the sum of the previous work times. Actually, the sum-of-processing-times based learning model is relevant to a long-winded and uninterrupted production process in which initial steps are set up. In manufacturing steel plates or bars by a foundry, the products are required to be inspected for surface defects. A single inspector is regarded as a processor or a system to work on those steel bars (jobs). After the processor finishes some work units (jobs), he will spend less time on processing follow-up work units as his experiences grow. Thus, Dondeti and Mohanty [33] modeled the real processing time of a job as a function of the sum of the processing times of the already processed jobs.

Further modifications of the sum-of-processing-times based learning effect in scheduling problems were proposed by Cheng et al. [34], Yin et al. [35], Rudek [36], Wu et al. [37], Lu et al. [38], Wu and Wang [39], and Pei et al. [40], among others. The readers may refer to Azzouz et al. [31] for other commonly discussed learning models: position-based learning, truncated learning, and exponential learning.

2. Notation Definition and Problem Formulation

At the outset, we define some notations and then describe the proposal problem.

2.1. Notations

n: total number of jobs

$m = 3$: total number of machines

J_i: code for the ith job, $i = 1, 2, \ldots, n$

$\omega = \{J_1, J_2, \ldots, J_n\}$: set of n jobs

M_l: machine code l, $l = 1, 2, 3$

S and $S*$: complete job schedules (sequences)

S_1, S_2, S_A, and S_U: partial job schedules (sequences)

index $[j]$: the job scheduled in the jth position (in a job sequence)

d: number of jobs removed from a sequence (S)

2.2. Input Parameters

p_i, q_i, r_i: component processing times of job J_i on machines M_1, M_2, and M_3, respectively, for $i = 1, 2, \ldots, n$

a: learning index and $a < 0$

T_0: initial temperature

T_f: final temperature

λ: annealing index

2.3. Decision Variables

T_{lk}: starting time of the kth job (in S) on machine M_l, $l = 1, 2, 3$

$C_{li}(S)$: completion time of job J_i on machine M_l, $i = 1, 2, \ldots, n$; $l = 1, 2, 3$

$C_{l[i]}(S)$: completion time of the job in position i on machine M_l, $l = 1, 2, 3$

$C_{\max}(S)$: completion time for the last job in S or makespan or $C_{3[n]}(S)$

2.4. Problem Formulation. We assume that a set of n jobs, ω, are available at time zero and will be processed on fixed three machines, M_1, M_2, and M_3. Each job J_i has three components to be processed on all three machines in three operations. Two components are first processed in parallel on machines M_1 and M_2 and then on machine M_3 for the assembly operation. Machines M_1 and M_2 have no idle times at the first stage of operation. A job to be processed on machine M_3 cannot start until it has been processed on machines M_1 and M_2. No machine can process more than one job at a time. Furthermore, preemption is not allowed; i.e., a job cannot be interrupted once it has begun processing. Sum of the component processing times on machines M_1, M_2, and M_3 are $p_i(1 + \sum_{j=1}^{k-1} p_{[j]})^a$, $q_i(1 + \sum_{j=1}^{k-1} q_{[j]})^a$, and $r_i(1 + \sum_{j=1}^{k-1} r_{[j]})^a$, respectively, where a is a learning index and $a < 0$.

For the convenience of reading, we further let $P_k = 1 + \sum_{j=1}^{k-1} p_{[j]}$, $Q_k = 1 + \sum_{j=1}^{k-1} q_{[j]}$, $R_k = 1 + \sum_{j=1}^{k-1} r_{[j]}$, $P_k^a = (1 + \sum_{j=1}^{k-1} p_{[j]})^a$, $Q_k^a = (1 + \sum_{j=1}^{k-1} q_{[j]})^a$, and $R_k^a = (1 + \sum_{j=1}^{k-1} r_{[j]})^a$.

Now, for a schedule S, the completion time $C_{3[k]}(S)$ of a job J_i to be scheduled on kth position on machine M_3 in S is calculated as $C_{3[k]}(S) = \max\{T_{1k} + p_i P_k^a, T_{2k} + q_i Q_k^a, T_{3k}\} + r_j(R_k)^a$, where T_{1k}, T_{2k}, and T_{3k} denote the starting times of a job (job J_i here) to be assigned to the kth position on machines M_1, M_2, and M_3, respectively, in S. The objective is to find a schedule $S*$ that minimizes the makespan; i.e., $C_{\max}(S^*) = \min_S\{C_{3[1]}(S), C_{3[2]}(S), \ldots, C_{3[n]}(S)\}$. For the Gantt chart of illustrating the considered two-stage three-machine environment, readers can refer to Lee et al. [5].

3. Solution Approach

A branch-and-bound (B&B) method is used to solve the proposed problem for small number of jobs with $n = 8, 9, 10, 11$. Several dominance properties and a lower bound (LB) are developed and incorporated into the branch-and-bound method to increase the search efficiency. Since the proposed problem is NP-hard, to solve the problem for large number of jobs, we utilize a cloud theory-based simulated annealing (CSA) algorithm and an iterated greedy (IG) algorithm with four different local search methods. It has been shown [7, 22, 23, 41] that the permutation schedules, in which the jobs are processed in the same order in all machines, are dominant with respect to the makespan criterion. Consequently, we limit our research to permutation schedules for the proposed problem. Using the three-field classified system in scheduling literature (see [42]), the proposed problem can be represented as AF2(2,1)/prmu, $\mathrm{LE_t/C_{max}}$.

3.1. Dominance Properties and a Lower Bound. Dominance rules are commonly used to improve search efficiency of heuristics in the scheduling research, for example, Lee et al. [5], Potts et al. [6], and Tozkapan et al. [22]. In this section, we will develop some dominance (elimination) properties and a LB embedded in a B&B method for finding optimal solutions. Let $S = (S_1, i, j, S_2)$ and $S' = (S_1, j, i, S_2)$ denote two sequences in which S_1 and S_2 are partial job sequences (or empty) and there are at most $(k - 2)$ scheduled jobs in S_1, $1 \le k \le n$. To show S dominates S', it suffices to show that $C_{3j}(S) < C_{3i}(S')$. In the following, we construct several dominance rules to assist in eliminating nodes in the B&B method. Two lemmas are provided to shorten the proof of the properties. The proofs of Lemma 1 and Properties 3 to 9 are given in the Appendix.

Lemma 1. *If* $0 < x \le y$, $0 < C$, *and* $a < 0$, *then* $x(C + y)^a \le y(C + x)^a$.

Note that if $0 < x < y$, $0 < C$, and $a < 0$, then $x(C+y)^a < y(C+x)^a$.

Lemma 2. *If* $1 \le \lambda$, $0 \le t$, *and* $a < 0$, *then* $0 \le \lambda[1-(1+t)^a] - [1-(1+\lambda t)^a]$.

Note that Lemma 2 has already been proved by Kuo and Yang [4].

Property 3. If $p_i < p_j$, $q_i < q_j$, $r_j < r_i$, $T_{3k} > \max\{T_{1k} + p_j P_k^a, T_{2k} + q_j Q_k^a\}$, $T_{1k} + p_i P_k^a + p_j (P_k + p_i)^a > \max\{T_{2k} + q_i Q_k^a + q_j (Q_k + q_i)^a, T_{3k} + r_i R_k^a\}$, and $T_{1k} + p_j P_k^a + p_i (P_k + p_j)^a > \max\{T_{2k} + q_j Q_k^a + q_i (Q_k + q_j)^a, T_{3k} + r_j R_k^a\}$, then S dominates S'.

Property 4. If $p_i < p_j$, $q_i < q_j$, $r_j < r_i$, $T_{3k} > \max\{T_{1k} + p_j P_k^a, T_{2k} + q_j Q_k^a\}$, $T_{2k} + q_i Q_k^a + q_j (Q_k + q_i)^a > \max\{T_{1k} + p_i P_k^a + p_j (P_k + p_i)^a, T_{3k} + r_i R_k^a\}$, and $T_{2k} + q_j Q_k^a + q_i (Q_k + q_j)^a > \max\{T_{1k} + p_j P_k^a + p_i (P_k + p_j)^a, T_{3k} + r_j R_k^a\}$, then S dominates S'.

Property 5. If $p_i < p_j$, $q_i < q_j$, $r_i < r_j$, $T_{3k} > \max\{T_{1k} + p_j P_k^a, T_{2k} + q_j Q_k^a\}$, and $r_i R_k^a > \max\{p_j (P_k + p_i)^a, q_j (Q_k + q_i)^a\}$, then S dominates S'.

Property 6. If $p_i < p_j$, $r_j < r_i$, $T_{1k} + p_i P_k^a > \max\{T_{2k} + q_i Q_k^a, T_{2k} + q_j Q_k^a, T_{3k}\}$, and $p_i (P_k + p)^a > \max\{q_i (Q_k + q_j)^a, q_j (Q_k + q_i)^a, r_i R_k^a\}$, then S dominates S'.

Property 7. If $q_i < q_j$, $r_j < r_i$, $T_{2k} + q_i Q_k^a > \max\{T_{1k} + p_i P_k^a, T_{1k} + p_j P_k^a, T_{3k}\}$, and $q_i (Q_k + q_j)^a > \max\{p_i (P_k + p_j)^a, p_j (P_k + p_i)^a, r_i R_k^a\}$, then S dominates S'.

Property 8. If $p_i < p_j$, $q_i < q_j$, $r_i < r_j$, $T_{1k} + p_i P_k^a > \max\{T_{2k} + q_j Q_k^a, T_{3k}\}$, and $r_i R_k^a > \max\{p_j (P_k + p_i)^a, q_j (Q_k + q_i)^a\}$, then S dominates S'.

Property 9. If $p_i < p_j$, $q_i < q_j$, $r_i < r_j$, $T_{2k} + q_i Q_k^a > \max\{T_{1k} + p_j P_k^a, T_{3k}\}$, and $r_i R_k^a > \max\{p_j (P_k + p_i)^a, q_j (Q_k + q_i)^a\}$, then S dominates S'.

The next property follows the same idea of Kuo and Yang [4], Theorem 1 thereof, but it is modified to accommodate the learning situation. Let $S = (S_A, S_U)$, where S_A has $(k-1)$ jobs, $1 \le k \le n$, and S_U has $(n-k+1)$ jobs. Let $r_{(k)} \le r_{(k+1)} \le \cdots \le r_{(n)}$ represent nondecreasing orders of processing time of jobs in S_U for Machine 3.

Property 10. If $T_{3k} \ge \max\{T_{1k} + \sum_{i=k}^{n} p_{(i)} (1 + \sum_{j=1}^{i-1} p_{[j]})^a, T_{2k} + \sum_{i=k}^{n} q_{(i)} (1 + \sum_{j=1}^{i-1} q_{[j]})^a\}$, then (S_A, S_{M3}) dominate (S_A, S_U), where S_{M3} is the partial schedule according to nondecreasing order of the processing times of S_U in Machine 3. (Note that the complexity of Property 10 is $O(n \log(n))$)

As a job being placed in a schedule on the kth position, $1 \le k \le n$, a LB, based on similar idea of Lee et al. [5], can be acquired as follows.

$$\begin{aligned} \mathrm{LB}(S) = \max \Bigg\{ & T_{3k} + \sum_{i=k}^{n} r_{(i)} \left(1 + \sum_{j=0}^{i-1} r_{[j]}\right)^a \\ & + \min_{i \in S_U} \left\{ \max \left\{ T_{3k}, T_{1k} + p_i \left(1 + \sum_{j=0}^{k-1} p_{[j]}\right)^a, T_{2k} + q_i \left(1 + \sum_{j=0}^{k-1} q_{[j]}\right)^a \right\} - T_{3k} \right\}, \\ & \max \left\{ \sum_{i=1}^{n} p_{(i)} \left(1 + \sum_{j=0}^{i-1} p_{(j)}\right)^a, \sum_{i=1}^{n} q_{(i)} \left(1 + \sum_{j=0}^{i-1} q_{(j)}\right)^a \right\} + \min_{i \in S_U} \left[r_i \left(1 + \sum_{j=1}^{n} r_j - r_i\right)^a \right] \Bigg\}. \end{aligned} \tag{1}$$

Note that $T_{11} = 0$, $T_{21} = 0$, $T_{31} = 0$, $p_{(0)} = 0$, $q_{(0)} = 0$, and $r_{(0)} = 0$.

For the branch-and-bound (B&B) method in this study, we employ the depth-first search [43] in the branching procedure: a branch is selected and explored systematically until it is pruned out or its end node is reached. The B&B method allocates jobs starting from the first position ($k = 1$) and in the forward to last position ($k = n$) allocation. To enhance the performance of the B&B method, the starting sequence (schedule) found by the algorithm IGLS3 (see more details in the next subsection) is considered as an incumbent sequence. Then, the depth-first search method is executed from the initial node and its descents from the tree, to renew the incumbent sequence. Apply Properties 3 to 9 to eliminate branches. If the LB of unsearched nodes is greater than or equal to the makespan of the incumbent sequence than eliminate the node and all nodes sprouting from it in the branch. For active nodes, Property 10 are applied to examine whether the order of unscheduled jobs can be allocated. Repeat the procedure until all nodes are explored.

The above B&B method is also used to evaluate the effectiveness of the adopted algorithms, a cloud-based annealing algorithm, and an iterated greed algorithm for small number of jobs.

For the study problem without learning consideration, AF2(2,1)//C_{max} has been shown to be NP-hard (see [5]), so does AF2(2,1)/prmu,LE_t/C_{max}. In order to search for good quality solutions and shorten the computer CPU times,

we utilize a metaheuristic cloud theory-based simulated annealing (CSA) algorithm, Li et al. [44] and Li and Du [45], and four versions of the iterated greedy (IG) algorithm, Ruiz and Stützle [46], for (near-) optimal solutions. The details are introduced in the following subsections.

3.2. A Cloud Theory-Based Simulated Annealing Algorithm. The simulated annealing algorithm (SA), a random optimization method, was first proposed by Kirkpatrick [47]. SA has been frequently and successfully adopted in combinatorial optimization problems because of its capability of escaping from local minimum in the searching process. SA imitates the process of the physical annealing to the solution of an optimization problem. It starts by constructing an initial solution, S_0, from a high initial temperature, T_0, and selects a solution S_1 randomly in the neighborhood of S_0. Whether S_1 is accepted as an incumbent solution depends on the temperature and on $C_{\max}(S_0)$ and $C_{\max}(S_1)$, values of the objective function and the makespan. If $C_{\max}(S_1) \leq C_{\max}(S_0)$, then S_1 is accepted and replaces S_0; otherwise, S_1 can also be accepted with a probability $\exp[-(C_{\max}(S_1) - C_{\max}(S_0))/T]$. As the temperature T decreases gradually, SA repeats the above process and converges to a global solution (in theory) or finds a local solution at the end the process. Allahverdi and Al-Anzi [21] successfully applied SA on an assembly flow shop problem. Although SA has been successfully applied to solve several optimization problems, it has also been found unable to solve some combinatorial optimization problems. Readers can refer to the reference listed in Boussaïd et al. [48].

The basic concept of the cloud theory has been proposed by Li et al. [44] and Li and Du [45]. Torabzadeh and Zandieh [49] proposed a cloud theory-based simulated annealing (CSA) algorithm for the 2-stage m-machine assembly flow shop scheduling problem with a bicriteria objective function. They showed that their CSA performed better than the best existing SA [21]. Studying the same scheduling problem but with learning consideration and a different criterion, we then adapt the CSA for the problem. For more details about CSA, readers may refer to Li and Du [45] and Lv et al. [50].

For the CSA to search better quality solutions, one local heuristic based on the idea of Johnson's rule [51] was constructed. The resultant solution from the heuristic was then put in CSA as an initial solution. The Johnson-based local search heuristic, coded as J_mean, is as follows.

The J_mean local search heuristic is shown as follows.

(1) For each J_i, $i = 1, 2, \ldots, n$, let $A_{1i} = (p_i + q_i)/2$ and $B_{2i} = r_i$ be the processing times M_1 and M_2. $N = \{J_1, J_2, \ldots, J_n\}$.

(2) Let $s = 1, t = n$.

(3)

(i) If $\min\{A_{11}, A_{12}, \ldots, A_{1n}, B_{21}, B_{22}, \ldots, B_{2n}\} =$ (say)A_{1i}, then assign J_i to the sth position and delete job J_i from N. Set $s = s + 1$, and go to (4).

(ii) If $\min\{A_{11}, A_{12}, \ldots, A_{1n}, B_{21}, B_{22}, \ldots, B_{2n}\} =$ (say)B_{2i}, then assign J_i to the tth position and delete J_i from N. Set $t = t - 1$, and go to (4).

(4) Delete J_i from N, if N is not empty, and then go to (3); otherwise, stop.

Note that the total computational complexity of J_mean algorithm, which is the same as that of Johnson's rule, is $O(n^2)$. The parameters aroused in CSA such as initial temperature T_0, final temperature T_f, and annealing index λ need to be determined. We referred to the values of the corresponding parameters used in Allahverdi and Aydilek [16] and performed a pilot study similar to the study in Wu et al. (2017). Finally, we adopt the $T_0 = 0.1$, $T_f = 10^{-5}$, and $\lambda = 0.98$. The procedure of CSA is as in Algorithm 1.

In the CSA procedure (Algorithm 1), E_n (the range of the cloud), H_e (the cloud drops' dispersive degree), and a given certain value u_0 are parameters from a normal distribution in the cloud theory-based simulated annealing algorithm [50]. In addition, the number of iterations N_r can be raised to enhance the searching capability of CSA [16].

3.3. Iterated Greedy Algorithm. The iterated greedy (IG) algorithm was proposed by Ruiz and Stützle [46]. The great advantages of this method are as follows: it is simple to implement and has been confirmed to yield high-quality solutions for flow shop problems [52]. The IG has been extensively employed for the last ten years in the scheduling research, especially in the (permutation or hybrid) flow shop problem, for example, Pan and Ruiz [53], Msakni et al. [54], Dubois-Lacoste et al. [55], and Pan et al. [56]. In this study, we propose an IG algorithm with four different local search methods, i.e., four versions of IG, coded as IGLS1, IGLS2, IGLS3, and IGLS4.

When performing the IG, one randomly generates an initial solution and uses it repeatedly to execute destruction and construction cycles until a stopping criterion is reached [46]. In the destruction stage, d jobs are removed from the incumbent solution S to form a new partial sequence S_D with $n - d$ jobs. Let S_R be the sequence with a size of d jobs created that contain the removed jobs according to the order of their appearances in S. In the construction stage, we choose the jobs of S_R to be reinserted in the partial solution S_D using the Nawaz-Enscore-Ham (NEH) mechanism [57]. Each job in S_R has an assigned order, and the job with the first order is inserted in all possible positions of S_D. The position with the best objective value is held. The same iteration is repeated for all remaining jobs in S_R. Note that the complexity of NEH algorithm is $O(n^3m)$.

Similar to Ruiz and Stützle [46], the IGLS algorithms utilize an acceptance criterion to determine whether a new solution should be kept or not. It adopts $W = T \cdot [\sum_i (p_i + q_i + r_i)]/(100 \cdot n \cdot m)$ as the temperature in the IG algorithm, where T is a controllable variable with $0 < T < 1$. The procedure of IG algorithm is described as in Algorithm 2.

We used four different neighborhood structures as local search paths after the construction stage in the IG algorithm. The insertion neighborhood structure, LS3, is described in step (2) of the IG algorithm. The neighborhood structures for LS1, LS2, and LS4 are described as follows. Let $S = (J_1, J_2, \ldots, J_n)$ be a solution (i.e., a sequence of n jobs) and $C_{\max}(S)$ be the objective value, i.e., the makespan, of the S.

Input initial and final temperature, (T_0, T_f), and annealing index λ.
Randomly generate a solution, improved by (Johnson's rule based) J_mean, then by a pairwise interchange, denote the solution S_0, Calculate $C_{\max}(S_0)$.
Do while $\{T < T_f\}$
  Set $S_{\text{out}} = S_0$, $r = 0$, and $T = T_0$.
  Let $E_n = T$, $H_e = T$, and $u_0 = 1 - T$.
  Let $E_n^* = \max\{E_n + H_e - 1/3 \cdot \text{rand}(0, 1), eps\}$ where $eps < 10^{-8}$.
  Let $T^* = E_n^* \cdot \sqrt{-2\ln(u_0)}$.
  Set $K = 0$.
  Do while $\{K < N_r\}$ (N_r, a predetermined number of iteration is set at 20)
    Choose i and j randomly with $i, j \in \{1, 2, \ldots, n\}$ and $i \neq j$, swap the ith job and jth job in S_0 to obtain a new solution S_1.
    Calculate the makespans $C_{\max}(S_0)$ and $C_{\max}(S_1)$.
    If $C_{\max}(S_1) < C_{\max}(S_0)$, then replace S_0 by S_1; else, replace S_0 by S_1 when $\exp[-(C_{\max}(S_1) - C_{\max}(S_0))/(C_{\max}(S_0) \cdot T^*)] < U$, where $U \sim \text{rand}(0, 1)$.
    Let $K = \mathbf{K} + 1$.
  End while
  Let $r = r + 1$ and $T = T_0 * \lambda^r$.
End while
Output a final solution S_{out}.

Algorithm 1: Procedure of the proposed CSA.

***Initialization*:**
  Randomly generate an initial solution S.
  Improve S by a local search (e.g. LS3). The LS3 is as follows.
    (1) Randomly select one job among the n jobs.
    (2) Insert the selected job to n possible positions, keep the best (the lowest $C_{\max}$) solution.
    (3) Repeat the steps until all n jobs have been selected and inserted back; update the best solution if better solution is found.
    (4) Output the final solution, S_0.
  Set $S_0 = S_1$ and $S_{\text{best}} = S_1$;
Do while $\{$***iter*** $< \mathbf{M_{max}}\}$ (a predetermined number of iterations $\mathbf{M_{max}}$ is set equal to 30)
  Destruction stage.
    Divide S_1 into S_D and S_R, where S_D is a set of d removed jobs and S_R a set of the remaining $(n - d)$ jobs;
  Construction stage
    For the removed d jobs, execute the NEH mechanism to S_R one after another until it yields the best sequence, coded as S_2;
  Improve S_2 by one of the four local search methods, LS1, LS2, LS3, LS4, coded as S_3.
  (Acceptance criterion)
    If $C_{\max}(S_3) < C_{\max}(S_1)$, then update S_1 by S_3;
    If $C_{\max}(S_3) < C_{\max}(S_{\text{best}})$, then Update S_{best} by S_3;
    Else if $q \leq \exp(\text{Cmax}(S_3) - \text{Cmax}(S_1))/W)$ then Update S_1 by S_3;
    (Note that $q \sim$ rand(0,1).)
End while
Output the final solution S_{best}

Algorithm 2: An IG algorithm, IGLS3.

LS1: swap the structures of adjacent positions.

(1) Select $k_1 = 1$, $k_2 = k_1 + 1$.

(2) Swap the job numbers between the positions k_1 and k_2 in S, and let S_0 denote the new solutions.

(3) If $C_{\max}(S_0)$ is less than $C_{\max}(S_1)$ then update S by S_0.

(4) Let $k_1 = k_1 + 1$ and $k_2 = k_2 + 1$. Repeat (2) and (3) until $k_1 = n - 1$.

(5) Repeat steps (1) to (4) until $C_{\max}(S_0)$ cannot be improved.

Table 1: Computational results of the branch-and-bound method.

n	a	Branch and Bound method				Number of Feasible Solutions
		Number of nodes		CPU time (sec.)		
		mean	max	mean	max	
8	−.100	28960.00	28960	0.35	0.41	100
	−.010	28960.67	28964	0.35	0.39	100
	−.001	28960.69	28966	0.35	0.41	100
9	−.100	260649.00	260649	3.63	3.76	100
	−.010	260650.20	260661	3.64	3.78	100
	−.001	260650.01	260654	3.62	3.73	100
10	−.100	2606502.00	2606523	47.08	48.92	100
	−.010	2606502.00	2606518	46.93	48.69	100
	−.001	2606503.00	2606561	30.56	33.18	100
11	−.100	28671517.06	28671719	422.13	485.43	100
	−.010	28671513.38	28671543	529.33	593.48	100
	−.001	28671513.06	28671530	529.24	593.19	100

(6) Return the best found solution, denoted as S_0.

LS2: complete swap and best-first neighborhood structure.

(1) Swaps each pair of positions (jobs) in S.

(2) Calculate the resultant $C_{\max}$.

(3) Compare $C_{\max}$'s for all possible $n(n+1)/2$ pairs.

(4) Keep the best one, say S_0, with the lowest value of $C_{\max}$.

(5) If $C_{\max}(S_0) < C_{\max}(S_1)$, then interchange the job numbers between the pair of positions for the best; i.e., update S by S_0.

(6) Repeat steps (1) to (5) until $C_{\max}(S_0)$ cannot be improved.

(7) Return the best found solution, S_0.

LS4: first-improvement swap neighborhood structure is as follows.

(1) Let $k_1 = 1$, $k_2 = k_1 + 1$.

(2) Swap the job numbers between the positions k_1 and k_2 in S, and let S_0 denote the new solutions.

(3) If $C_{\max}(S_0)$ is less than $C_{\max}(S_1)$ then update S by S_0.

(4) Let $k_2 = k_2 + 1$. Repeat (2) and (3) until $k_2 = n$.

(5) Let $k_1 = k_1 + 1$. Repeat steps (2) to (4) until $k_1 = n - 1$.

(6) Repeat steps (1) to (5) until $C_{\max}(S_0)$ cannot be improved.

(7) Return the best found solution, denoted as S_0.

The ideas of the LS1, LS2, LS3, and LS4 are modified from three local searches proposed by Della Croce et al. [58]. The complexities of those three local searches are $O(n^2)$. The proposed LS1, LS2, LS3, and LS4 are very fast; even their complexities are not easily found. As for other neighborhood structures and local search methods, one can refer to Li et al. [59] and Wang et al. [60, 61], among others.

4. Computational Experiments and Results

In this section, we present our experimental tests of examining the efficiency of all adopted algorithms. The branch-and-bound method was performed for small number of jobs; a CSA and an iterated greedy (IG) algorithm with 4 different local search methods (coded as IGLS1 IGLS2, IGLS3, and IGLS4) were performed for both small and large numbers of jobs. These algorithms were performed in Fortran (version 6.6, Compaq Visual Fortran) on a PC (Windows XP) with Intel(R) Core(TM) i7 Duo CPU 2.66-GHz and RAM size of 1.99 GB. The normal job processing times on machines M_1, M_2, and M_3, in line with the scheme of Lee et al. [5], were generated at random from a discrete uniform distribution $U(1, 100)$, respectively. All the parameters used in CSA adopted the settings of Wu et al. [1] while all the parameters in IG used the results of previous tests.

In the computational experiments for small number of jobs, n was set at 8, 9, 10, and 11. Following the suggestion by Liu et al. [62], we set the learning index (a) at $a = -0.1$, -0.01, and -0.001. The number of jobs removed from a sequence was set at $d = 2$, 3, and 4; no more than half of a number of jobs were removed. One hundred instances were tested for each combination of n, a, and d. A total of 3600 instances were examined. Tables 1, 2, and 3 summarized the computational results. The optimal solutions were acquired by running the B&B method for the chosen small number of jobs. We report the average error percentage (AEP) here, where AEP $= 100[(A_i - O_i^*)]\%$, and A_i was obtained by each metaheuristic and O_i^* by the B&B method. Figure 1 shows the performance results of CSA and IGLS algorithms.

As for the capability of the B&B method, we report the maximum and average number of nodes and the maximum and average execution times (in seconds) in Table 1. As the numbers of job size (n) increased, the (mean and maximum) nodes increased; the CPU times increased dramatically as the n increased (Table 1, columns 4 and 5). If the number of nodes goes beyond 10^8, the B&B method would skip to the next instance, and instances solved out with less than 10^8 nodes

Table 2: The AEP of algorithms for small number of jobs.

n	a	d	Algorithm									
			IGLS1		IGLS2		IGLS3		IGLS4		CSA	
			mean	max	mean	max	mean	max	mean	max	mean	max
8	−0.001	2	0.000	0.013	0.008	0.812	0.000	0.013	0.007	0.630	0.000	0.000
		3	0.000	0.010	0.000	0.000	0.000	0.000	0.000	0.002	0.000	0.000
		4	0.000	0.000	0.000	0.000	0.000	0.002	0.000	0.001	0.000	0.000
	−0.01	2	0.007	0.455	0.001	0.047	0.006	0.455	0.014	0.748	0.000	0.003
		3	0.002	0.105	0.000	0.002	0.000	0.002	0.001	0.047	0.000	0.003
		4	0.001	0.047	0.000	0.002	0.000	0.018	0.001	0.047	0.000	0.003
	−0.1	2	0.043	2.277	0.001	0.028	0.004	0.175	0.031	0.634	0.001	0.037
		3	0.011	0.308	0.003	0.308	0.006	0.403	0.010	0.308	0.001	0.037
		4	0.011	0.403	0.003	0.308	0.004	0.403	0.009	0.308	0.001	0.037
9	−0.001	2	0.008	0.806	0.000	0.004	0.005	0.233	0.005	0.233	0.000	0.004
		3	0.000	0.008	0.000	0.000	0.008	0.812	0.008	0.806	0.000	0.004
		4	0.002	0.215	0.000	0.001	0.000	0.000	0.000	0.002	0.000	0.004
	−0.01	2	0.009	0.815	0.020	1.932	0.004	0.342	0.010	0.877	0.001	0.038
		3	0.001	0.079	0.000	0.006	0.009	0.877	0.009	0.815	0.001	0.038
		4	0.001	0.007	0.000	0.038	0.000	0.013	0.001	0.013	0.001	0.038
	−0.1	2	0.014	0.421	0.007	0.517	0.007	0.421	0.020	0.504	0.006	0.382
		3	0.011	0.421	0.001	0.049	0.002	0.139	0.010	0.421	0.006	0.382
		4	0.009	0.421	0.006	0.406	0.000	0.009	0.006	0.131	0.006	0.382
10	−0.001	2	0.016	1.544	0.016	1.544	0.016	1.544	0.002	0.216	0.000	0.001
		3	0.017	1.544	0.016	1.544	0.000	0.000	0.028	2.757	0.000	0.001
		4	0.000	0.001	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.001
	−0.01	2	0.017	1.551	0.000	0.007	0.000	0.014	0.029	2.851	0.000	0.005
		3	0.001	0.009	0.000	0.001	0.000	0.007	0.001	0.009	0.000	0.005
		4	0.001	0.091	0.000	0.004	0.000	0.001	0.001	0.006	0.000	0.005
	−0.1	2	0.143	5.765	0.029	1.571	0.019	1.615	0.076	3.603	0.003	0.049
		3	0.063	3.603	0.011	0.879	0.012	0.878	0.052	3.603	0.003	0.049
		4	0.005	0.075	0.011	0.873	0.012	0.873	0.006	0.130	0.003	0.049
11	−0.001	2	0.030	1.609	0.000	0.016	0.021	2.056	0.016	1.580	0.000	0.000
		3	0.000	0.004	0.000	0.016	0.000	0.000	0.000	0.016	0.000	0.000
		4	0.000	0.016	0.000	0.016	0.000	0.000	0.000	0.016	0.000	0.000
	−0.01	2	0.011	0.462	0.001	0.098	0.001	0.036	0.024	2.229	0.000	0.004
		3	0.002	0.053	0.001	0.098	0.000	0.003	0.001	0.016	0.000	0.004
		4	0.002	0.098	0.001	0.098	0.000	0.002	0.002	0.098	0.000	0.004
	−0.1	2	0.070	1.548	0.004	0.113	0.012	0.276	0.042	1.273	0.008	0.257
		3	0.016	0.351	0.015	1.266	0.005	0.113	0.024	1.266	0.008	0.257
		4	0.028	1.266	0.014	1.243	0.004	0.100	0.023	1.266	0.008	0.257
	mean		**0.015**	0.733	**0.005**	0.385	**0.004**	0.329	**0.013**	0.763	**0.002**	0.065

Table 3: The mean ranks of algorithms in Kruskal–Wallis test for small and large number of jobs.

Algorithm	Small n		Big n	
	Number of obs.	Mean Rank	Number of obs.	Mean Rank
IGLS1	36	115.14	36	82.38
IGLS2	36	80.96	36	70.67
IGLS3	36	80.39	36	65.50
IGLS4	36	115.64	36	79.83
CSA	36	60.38	36	154.13
Total	180	90.50	180	90.50

*Note that average scores were used for ties.

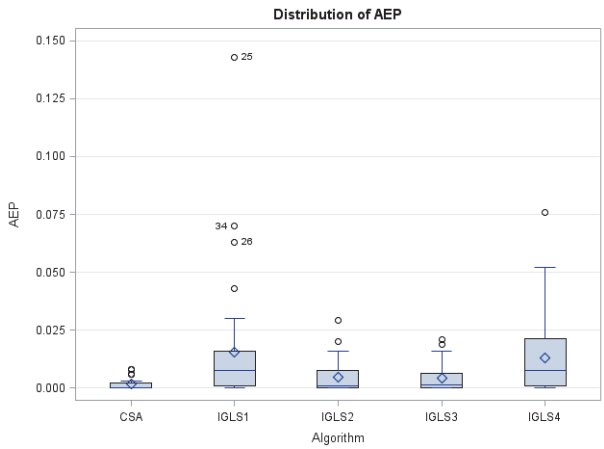


Figure 1: Boxplots of AEP for algorithms.

were recorded as feasible solution. The computational results are shown in the last column of Table 1. No matter what the number of jobs (n) or the learning index (a), all instances are solvable within 10^8 nodes. As for the impact of particular dominant properties on the efficiency of branch and bound (B&B), it can be easily done in the same way as that of Wu et al. [1].

For the proposed algorithms, CSA and IGLS, the AEPs are presented in Table 2. All AEPs of the CSA and IGLS algorithms increased slightly as the learning effect strengthened; i.e., the value of learning index (a) became larger, as shown in Table 2. As the amount of destruction (d) increased, the AEPs decreased as well; this is because the larger the value of d is, the wider the searching scope explores. Overall, the CSA (with mean AEP of 0.1%) performed the best; however, also note that the worst IGLS1 was with mean AEP of 1.5% (the last row of Table 2). The CPU times were all within 0.1 seconds; we thus omitted them here.

To compare whether the differences in the solution quality of the CSA and IGLS algorithms were statistically significant, the parametric or nonparametric statistical tests have commonly been employed [63]. There are three assumptions of the parametric analysis of variance (ANOVA), i.e., normality, homoscedasticity, and independence of the residuals. We carried out an ANOVA on AEPs and examined the assumption of normality for the residuals (using SAS 9.4). The value of the Kolmogorov–Smirnov D statistic was 0.18, and the asymptotic two-sided p value was less than 0.01; the level of significance was in general set at 0.05. Since the normality assumption was violated for the samples of AEP, we then used the nonparametric Kruskal–Wallis test on ranks of AEPs to test whether the populations of AEPs were the same. Table 3 (column 3) showed the mean ranks of the CSA and IGs algorithms.

In Table 4 (column 2), with p value (Asymp. Sig) < 0.001, it is indicated that there were significantly statistical differences among the performances of the CSA and IGLS algorithms. Consequently, a multiple comparison method, which was based on pairwise two-sample rankings, was

Table 4: Kruskal–Wallis test statistics for both small and large number of jobs.

	Kruskal-Wallis test	
	Small n	Big n
Chi-Square	31.2217	84.4702
DF	4	4
Asymp. Sig.	<0.0001	<0.0001

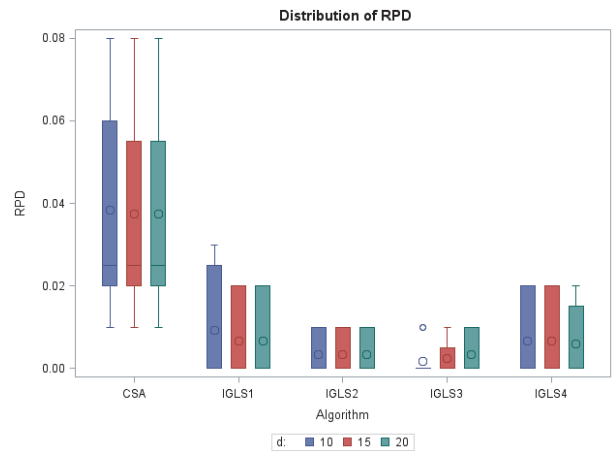


Figure 2: Boxplots of RPD for algorithms.

conducted to determine pairwise differences between algorithms. We employed the Dwass–Steel–Critchlow–Fligner (DSCF) procedure [64, 65]. As shown in Table 5, the mean ranks of AEPs could be separated significantly into two groups, at 0.05 level of significance. As illustrated in Table 5 (columns 2 and 3) of SDCF test, CSA (AEP = 0.001), IGLS2 (AEP = 0.004), and IGLS3 (AEP = 0.004) were in a better performance group and the IGLS1 (AEP = 0.015) and IGLS4 (AEP = 0.013) were in a worse performance group.

In the computational experiments for the large number of jobs, following Liu et al. [62], we also set the learning index $a = -0.1, -0.01$, and -0.001; the number of jobs removed, d, was set at 10, 15, and 20; the numbers of jobs was set at $n = 40, 50, 60$, and 70. One hundred instances were tested for each combination of n, d, and a; a total of 3600 instances were examined. We calculated the relative percent deviation (RPD), RPD = $100[(A_i - A^*)/A^*]\%$, where A_i was obtained from each algorithm and A^* was the minimum value among the CSA and four IGLS algorithms.

As shown in Table 6, the mean RPDs of CSA and IGLS algorithms increased as the learning index increased from -0.001 to -0.1 regardless of the number of jobs. As for the impact of the number of removed jobs (d) on RPDs, the average RPDs were 0.0118, 0.0113, and 0.0113, for $d = 10, 15$, and 20, respectively. Figure 2 provided the boxplots of RPDs for each number of d with these five algorithms.

The boxplots of the RPDs displayed the performances of the CSA and IG algorithms for large numbers of jobs. The CSA algorithm performed slightly worse than those of the four IGLS types. We carried out another ANOVA on RPDs and examined the normality assumption for the residuals; the

Table 5: Critchlow-Fligner procedure for differences among algorithms for small number of jobs.

Pairwise Comparison Between Algorithms	Dwass-Steel-Critchlow-Fligner procedure			
	Small n		Big n	
	DSCF value	Pr > DSCF	DSCF value	Pr > DSCF
IGLS1 vs. IGLS2	3.8926	0.0467	1.3734	0.8682
IGLS1 vs. IGLS3	3.8944	0.0465	2.1489	0.5497
IGLS1 vs. IGLS4	0.0239	1.0000	0.4338	0.9981
IGLS1 vs. CSA	6.4046	<.0001	8.1987	<.0001
IGLS2 vs. IGLS3	0.0642	1.0000	1.0924	0.9386
IGLS2 vs. IGLS4	3.9490	0.0418	1.2615	0.9000
IGLS2 vs. CSA	2.3163	0.4730	10.3262	<.0001
IGLS3 vs. IGLS4	4.1744	0.0262	2.0630	0.5894
IGLS3 vs. CSA	2.3432	0.4609	10.4627	<.0001
IGLS4 vs. CSA	6.3081	<.0001	8.8412	<.0001

p value was less than 0.01 from the Kolmogorov–Smirnov normality test. Therefore, the normality assumption was invalid for the RPD samples. A Kruskal–Wallis test on ranks of RPDs was subsequently used to check whether the RPD samples were from the same distribution. Table 3 (column 5) shows the mean ranks of the CSA and four IGLS algorithms for the large numbers of jobs. In Table 4 (column 3), the asymptotic p value is less than 0.0001. Consequently, the DSCF procedure was used to find pairwise differences between pairs of the CSA and IGLS algorithms. As displayed in Table 5 (columns 4 and 5), the IGLS algorithms were in one better performance group, whereas the CSA algorithm was the worst in large number of jobs case. Moreover, the boxplots of the RPDs (Figure 2) of the IGLS3 and IGLS2 showed less dispersion than the others. In other words, the IGLS2 and IGLS3 were both accurate and stable for solving the problem of the large numbers of jobs.

5. Conclusions and Future Studies

A makespan minimization problem for a two-stage three-machine assembly flow shop scheduling with sum-of-processed-times based processing times has been addressed. A branch-and-bound method embedded with a LB and several dominance properties was proposed for the small number of jobs ($n = 8, 9, 10, 11$). Since the problem is NP-hard, two algorithms, a cloud theory-based simulated annealing (CSA) algorithm and an iterated greedy (IG) algorithm with four local search methods, have been adopted to solve the problem for the large number of jobs ($n = 40, 50, 60, 70$). These algorithms seemed to perform well for both small number of jobs ($n = 8, 9, 10, 11$) and large number of jobs ($n = 40, 50, 60, 70$). The computational results validate the effectiveness, as claimed by researchers in the scheduling research community, of the proposed IG algorithm with four different local search methods. However, the CSA was still a competitive metaheuristic algorithm, especially for small-number-of-jobs problems in this study.

The occurrences of the study problem in many plant manufacturing environments have been exemplified by outstanding researchers such as these referred to in the introduction. Some potential research topics worthy of investigation are different learning curves including the position-based learning, an exponential learning, truncated learning, or general learning functions, especially different learning functions for different stages, i.e., at component manufacturing stage and at the assembly stage. This study considers two stages; however, there are several published papers that investigated three-stage flow shop assembly problem but all without learning effect. Therefore, extending to multistage assembly problem with learning consideration in some of these stages can be another future topic to pursue. One may generalize the properties for general nonincreasing learning curves, $p_{ik} = p_i f(\sum_{j=1}^{k-1} p_{[j]})$, where f is a nonincreasing function describing the learning effect (similarly for q_i and r_i) or job based nonincreasing learning curve, $p_{ijk} = p_{ij} f(k)$ or $p_{ijk} = p_{ij} - b_{ij} \min\{k-1, g_{ij}\}$. Readers may refer to Rudek [66–68] and Wang et al. [60, 61] for further details.

Appendix

Proofs of Lemma 1 and Properties

Proof of Lemma 1. Because $a < 0$, thus $0 < (C+y)^a/(C+x)^a \leq 1$, and $1 \leq y/x$, the result holds. □

Proof of Property 3. The completion times of job Ji and job Jj in the positions of k and $k+1$ on machines M_l, $l = 1, 2, 3$, in S are as follows.

$$C_{1i}(S) = T_{1k} + p_i \left(1 + \sum_{j=1}^{k-1} p_{[j]}\right)^a = T_{1k} + p_i P_k^a.$$

$$C_{2i}(S) = T_{2k} + q_i \left(1 + \sum_{j=1}^{k-1} q_{[j]}\right)^a = T_{2k} + q_i Q_k^a.$$

$$C_{3i}(S) = \max\{C_{1i}(S), C_{2i}(S), T_{3k}\} + r_i R_k^a.$$

$$C_{1j}(S) = C_{1i}(S) + p_j \left(1 + \sum_{j=1}^{k} p_{[j]}\right)^a = T_{1k} + p_i P_k^a + p_j (P_k + p_i)^a.$$

Table 6: The RPD of five algorithms for large number of jobs.

n	a	d	Algorithm									
			IGLS1		IGLS2		IGLS3		IGLS4		CSA	
			mean	max	mean	max	mean	max	mean	max	mean	max
40	−0.001	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	1.55
		15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	1.55
		20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	1.55
	−0.01	10	0.00	0.02	0.00	0.01	0.00	0.01	0.00	0.02	0.02	0.97
		15	0.00	0.01	0.00	0.01	0.00	0	0.00	0.01	0.02	0.97
		20	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.01	0.02	0.97
	−0.1	10	0.02	0.12	0.01	0.04	0.00	0.06	0.02	0.09	0.04	1.51
		15	0.02	0.09	0.01	0.12	0.01	0.04	0.02	0.10	0.04	1.51
		20	0.02	0.17	0.01	0.08	0.01	0.04	0.02	0.17	0.04	1.51
50	−0.001	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.31
		15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.31
		20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.31
	−0.01	10	0.00	0.32	0.00	0.01	0.00	0.01	0.00	0.02	0.01	0.30
		15	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.01	0.01	0.30
		20	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.01	0.01	0.30
	−0.1	10	0.03	0.28	0.01	0.06	0.01	0.10	0.02	0.23	0.08	1.09
		15	0.02	0.09	0.01	0.18	0.00	0.04	0.02	0.07	0.08	1.08
		20	0.02	0.14	0.01	0.28	0.01	0.07	0.01	0.12	0.08	1.08
60	−0.001	10	0.00	0.32	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.38
		15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.38
		20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.38
	−0.01	10	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.02	0.47
		15	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.01	0.02	0.47
		20	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.01	0.02	0.47
	−0.1	10	0.03	0.18	0.01	0.04	0.01	0.09	0.02	0.08	0.08	1.24
		15	0.02	0.14	0.01	0.05	0.01	0.05	0.02	0.18	0.08	1.24
		20	0.02	0.11	0.01	0.07	0.01	0.06	0.02	0.11	0.08	1.24
70	−0.001	10	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.35
		15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.35
		20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.35
	−0.01	10	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.6
		15	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.01	0.05	0.6
		20	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.05	0.6
	−0.1	10	0.03	0.17	0.01	0.08	0.00	0.11	0.02	0.17	0.07	0.74
		15	0.02	0.14	0.01	0.07	0.01	0.11	0.02	0.11	0.06	0.74
		20	0.02	0.11	0.01	0.05	0.01	0.10	0.02	0.10	0.06	0.74
		mean	**0.01**	0.07	**0.00**	0.03	**0.00**	0.03	**0.01**	0.05	**0.04**	0.79

$$
\begin{aligned}
C_{2j}(S) &= C_{2i}(S) + q_j\left(1+\sum_{j=1}^{k} q_{[j]}\right)^a = T_{2k} + q_i Q_k^a \\
&\quad + q_j\left(Q_k + q_i\right)^a. \\
C_{3j}(S) &= \max\left\{C_{1j}(S), C_{2j}(S), \max\left\{C_{1i}(S), C_{2i}(S), T_{3k}\right\}\right. \\
&\quad \left. + r_i R_k^a\right\} + r_j\left(R_k + r_i\right)^a = \max\left\{T_{1k} + p_i P_k^a\right. \\
&\quad + p_j\left(P_k + p_i\right)^a, T_{2k} + q_i Q_k^a \\
&\quad + q_j\left(Q_k + q_i\right)^a, \max\left\{T_{1k} + p_i P_k^a, T_{2k} + q_i Q_k^a, T_{3k}\right\} \\
&\quad \left. + r_i R_k^a\right\} + r_j\left(R_k + r_i\right)^a.
\end{aligned} \tag{A.1}
$$

The completion times of job Jj and job Ji in the positions of k and $k+1$ on machines M_l, $l = 1, 2, 3$, in S' are as follows.

$$
\begin{aligned}
C_{1j}(S') &= T_{1k} + p_j\left(1 + \sum_{j=1}^{k-1} p_{[j]}\right)^a = T_{1k} + p_j P_k^a.\\
C_{2j}(S') &= T_{2k} + q_i\left(1 + \sum_{j=1}^{k-1} q_{[j]}\right)^a = T_{2k} + q_j Q_k^a.\\
C_{3j}(S') &= \max\left\{C_{1j}(S'), C_{2j}(S'), T_{3k}\right\} + r_j R_k^a.\\
C_{1i}(S') &= C_{1j}(S') + p_i\left(1 + \sum_{j=1}^{k} p_{[j]}\right)^a = T_{1k}\\
&\quad + p_j P_k^a + p_i\left(P_k + p_j\right)^a.\\
C_{2i}(S') &= C_{2j}(S') + q_i\left(1 + \sum_{j=1}^{k} q_{[j]}\right)^a = T_{2k}\\
&\quad + q_j Q_k^a + q_i\left(Q_k + q_j\right)^a.\\
C_{3i}(S') &= \max\left\{C_{1i}(S'), C_{2i}(S'),\right.\\
&\quad \left.\max\left\{C_{1j}(S'), C_{2j}(S'), T_{3k}\right\} + r_j R_k^a\right\} + r_i\left(R_k\right.\\
&\quad \left.+ r_j\right)^a = \max\left\{T_{1k} + p_j P_k^a + p_i\left(P_k + p_j\right)^a, T_{2k}\right.\\
&\quad \left.+ q_j Q_k^a + q_i\left(Q_k + q_j\right)^a, \max\left\{T_{1k}+p_j P_k^a, T_{2k}\right.\right.\\
&\quad \left.\left.+ q_j Q_k^a, T_{3k}\right\} + r_j R_k^a\right\} + r_i\left(R_k + r_j\right)^a.
\end{aligned}
\tag{A.2}
$$

In the following, we claim that $C_{3i}(S') - C_{3j}(S) > 0$.

The inequality $T_{3k} > \max\{T_{1k} + p_j P_k^a, T_{2k} + q_j Q_k^a\}$ implies that

$$
\begin{aligned}
C_{3i}(S') &= \max\left\{T_{1k} + p_j P_k^a + p_i\left(P_k + p_j\right)^a, T_{2k}\right.\\
&\quad \left.+ q_j Q_k^a + q_i\left(Q_k + q_j\right)^a, T_{3k} + r_j R_k^a\right\}\\
&\quad + r_i\left(R_k + r_j\right)^a = T_{1k} + p_j P_k^a + p_i\left(P_k + p_j\right)^a\\
&\quad + r_i\left(R_k + r_j\right)^a.
\end{aligned}
\tag{A.3}
$$

The last equality holds because $T_{1k} + p_j P_k^a + p_i(P_k + p_j)^a > \max\{T_{2k} + q_j Q_k^a + q_i(Q_k + q_j)^a, T_{3k} + r_j R_k^a\}$.

Since $p_i < p_j$, $q_i < q_j$, and $T_{3k} > \max\{T_{1k} + p_j P_k^a, T_{2k} + q_j Q_k^a\}$, then $T_{3k} > \max\{T_{1k} + p_i P_k^a, T_{2k} + q_i Q_k^a\}$.

Apply the last inequality to $C_{3j}(S)$:

$$
\begin{aligned}
C_{3j}(S) &= \max\left\{T_{1k} + p_i P_k^a + p_j\left(P_k + p_i\right)^a, T_{2k}\right.\\
&\quad \left.+ q_i Q_k^a + q_j\left(Q_k + q_i\right)^a, T_{3k} + r_i R_k^a\right\}\\
&\quad + r_j\left(R_k + r_i\right)^a = T_{1k} + p_i P_k^a + p_j\left(P_k + p_i\right)^a\\
&\quad + r_j\left(R_k + r_i\right)^a.
\end{aligned}
\tag{A.4}
$$

The last equality comes from $T_{1k} + p_i P_k^a + p_j(P_k + p_i)^a > \max\{T_{2k} + q_i Q_k^a + q_j(Q_k + q_i)^a, T_{3k} + r_i R_k^a\}$.

So,

$$
\begin{aligned}
C_{3i}(S^*) - C_{3j}(S) &= \left[T_{1k} + p_j P_k^a + p_i\left(P_k + p_j\right)^a\right.\\
&\quad \left.+ r_i\left(R_k + r_j\right)^a\right] - \left[T_{1k} + p_i P_k^a + p_j\left(P_k + p_i\right)^a\right.\\
&\quad \left.+ r_j\left(R_k + r_i\right)^a\right] = p_j\left[P_k^a - \left(P_k + p_i\right)^a\right] - p_i\left[P_k^a\right.\\
&\quad \left.- \left(P_k + p_j\right)^a\right] + \left[r_i\left(R_k + r_j\right)^a - r_j\left(R_k + r_i\right)^a\right]\\
&= \left(p_i P_k^a\right)\left\{\left[\frac{p_j}{p_i}\left(1 - \left(1 + \frac{p_i}{P_k}\right)^a\right)\right.\right.\\
&\quad \left.\left.- \left(1 - \left(1 + \frac{p_j}{P_k}\right)^a\right)\right]\right\} + \left[r_i\left(R_k + r_j\right)^a - r_j\left(R_k\right.\right.\\
&\quad \left.\left.+ r_i\right)^a\right].
\end{aligned}
\tag{A.5}
$$

Letting $\lambda = p_j/p_i$ (> 1), and $t = p_i/P_k$, $t > 0$, then by applying Lemma 2, the first term of the last equation is greater than 0. Letting $y = r_i$, $x = r_j$ (i.e., $x < y$), and $C = R_k$, then Lemma 1 implies the second term of the last equation in $C_{3i}(S') - C_{3j}(S)$ is greater than 0. Consequently, $C_{3i}(S') - C_{3j}(S) > 0$, as required. □

Proof of Property 4. $C_{3i}(S') - C_{3j}(S) = (q_i Q_k^a)\{[(q_j/q_i)(1-(1+\frac{q_i}{Q_k})^a) - (1-(1+\frac{q_j}{Q_k})^a)]\} + [r_i(R_k + r_j)^a - r_j(R_k + r_i)^a]$.

Letting $\lambda = q_j/q_i$ (> 1), and $t = q_i/Q_k$, $t > 0$, then following the argument used in proof of Property 3, we have $C_{3i}(S') - C_{3j}(S) > 0$. □

Proof of Property 5. Because $p_i < p_j$, $q_i < q_j$, and $T_{3k} > \max\{T_{1k} + p_j P_k^a$, we have $T_{2k} + q_j Q_k^a\}$, $T_{3k} > \max\{T_{1k} + p_i P_k^a$, $T_{2k} + q_i Q_k^a\}$. In addition, $r_i R_k^a > \max\{p_j(P_k + p_i)^a$, $q_j(Q_k + q_i)^a\}$, together with three conditions, and it follows $C_{3i}(S') - C_{3j}(S) = [T_{3k} + r_j R_k^a + r_i(R_k + r_j)^a] - [T_{3k} + r_i R_k^a + r_j(R_k + r_i)^a] = r_i R_k^a[(r_j/r_i)(1 - (1 + r_i/R_k)^a) - (1 - (1 + (r_j/r_i)(r_i/R_k))^a)]$.

Let $\lambda = r_j/r_i$, $\lambda > 1$ and $t = r_i/R_k$, $t > 0$; then by applying Lemma 2, the last term in the above equality is greater than 0. Therefore, we have $C_{3i}(S') - C_{3j}(S) \geq 0$. □

Proof of Property 6. Note that (1) $T_{1k} + p_i P_k^a < T_{1k} + p_j P_k^a$, and $T_{2k} + q_i Q_k^a < T_{2k} + q_j Q_k^a$. This is because $p_i < p_j$ and $P_k^a > 0$, $Q_k^a > 0$. (2) $r_i(R_k + r_j)^a > r_j(R_k + r_i)^a$; this can be obtained directly from Lemma 1.

Now, apply the two longer inequalities given in the conditions of Property 6; we have

$$
\begin{aligned}
&C_{3i}(S') - C_{3j}(S)\\
&\geqq \left\{T_{1k} + p_j P_k^a + p_i\left(P_k + p_j\right)^a + r_i\left(R_k + r_j\right)^a\right\}\\
&\quad - \left\{T_{1k} + p_i P_k^a + p_j\left(P_k + p_i\right)^a + r_j\left(R_k + r_i\right)^a\right\}\\
&= p_j\left[P_k^a - \left(P_k + p_i\right)^a\right] - p_i\left[P_k^a - \left(P_k + p_j\right)^a\right]
\end{aligned}
$$

$$+\left[r_i\left(R_k+r_j\right)^a-r_j\left(R_k+r_i\right)^a\right]$$
$$=p_iP_k^a\left[\frac{p_j}{p_i}-\frac{p_j}{p_i}\left(1+\frac{p_i}{P_k}\right)^a-\left(1-\left(1+\frac{p_j}{P_k}\right)^a\right)\right]$$
$$+\left[r_i\left(R_k+r_j\right)^a-r_j\left(R_k+r_i\right)^a\right]>0. \tag{A.6}$$

Let $\lambda = p_j/p_i$, $\lambda > 1$ and $t = p_i/P_k$, $t > 0$; then by applying Lemma 2, the first term above is greater than 0. Directly applying Lemma 1, the second term above is greater than 0. □

Proof of Property 7. The proof is similar to that of Property 6 with p_i and P_k now being replaced by q_i and Q_k. □

Proof of Property 8. The value of inside max in the $C_{3j}(S)$ is equal to $T_{1k}+p_iP_k^a$ because $T_{1k}+p_iP_k^a > \max\{T_{2k}+q_jQ_k^a, T_{3k}\}$ and $q_i < q_j$. Applying the given inequality of $r_iR_k^a$ to the first two terms of the outside max, we have $C_{3j}(S) = [T_{1k}+p_iP_k^a+r_iR_k^a]+r_j(R_k+r_i)^a$.

The value of max inside the $C_{3i}(S')$ is equal to $T_{1k}+p_jP_k^a$ because $T_{1k}+p_jP_k^a > T_{1k}+p_iP_k^a > \max\{T_{2k}+q_jQ_k^a, T_{3k}\}$ and $q_i < q_j$.

Now,

$$C_{3i}\left(S'\right)=\max\left\{T_{1k}+p_jP_k^a+p_i\left(P_k+p_j\right)^a, T_{2k}\right.$$
$$\left.+q_jQ_k^a+q_i\left(Q_k+q_j\right)^a, T_{1k}+p_jP_k^a+r_jR_k^a\right\} \tag{A.7}$$
$$+r_i\left(R_k+r_j\right)^a.$$

Note that $r_iR_k^a > p_j(P_k+p_i)^a > p_i(P_k+p_j)^a$, $r_iR_k^a > q_j(Q_k+q_i)^a > q_i(Q_k+q_j)^a$, and $T_{1k}+p_jP_k^a > T_{1k}+p_iP_k^a > T_{2k}+q_jQ_k^a$. We have $C_{3i}(S') = [T_{1k}+p_jP_k^a+r_jR_k^a]+r_i(R_k+r_j)^a$.

Therefore,

$$C_{3i}\left(S'\right)-C_{3j}(S)=\left[r_iR_k^a+r_j\left(R_k+r_i\right)^a\right]-\left[r_jR_k^a\right.$$
$$\left.+r_i\left(R_k+r_j\right)^a\right]=r_iR_k^a\left[\frac{r_j}{r_i}\left(1-\left(1+\frac{r_i}{R_k}\right)^a\right)\right. \tag{A.8}$$
$$\left.-\left(1-\left(1+\frac{r_j}{r_i}\frac{r_i}{R_k}\right)^a\right)\right].$$

Let $\lambda = r_j/r_i$, $\lambda > 1$, and $t = r_i/R_k$, $t > 0$. Applying Lemma 2, we obtain the desired $C_{3i}(S') - C_{3j}(S) > 0$. □

Proof of Property 9. The proof is similar to that of Property 8 and thus omitted. □

Data Availability

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

Conflicts of Interest

The author declares that they have no conflicts of interest.

Disclosure

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Acknowledgments

The author thanks Professor K. Lai for helping in editing the English language.

References

[1] 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*, p. 2017.

[2] D. Biskup, "Single-machine scheduling with learning considerations," *European Journal of Operational Research*, vol. 115, no. 1, pp. 173–178, 1999.

[3] C. C. Wu, J. Chen, W. C. Lin, K. Lai, S. Liu, and P. 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*, 2018.

[4] 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.

[5] C.-Y. Lee, T. C. E. Cheng, and B. M. T. Lin, "Minimizing the makespan in the 3-machine assembly-type flowshop scheduling problem," *Management Science*, vol. 39, no. 5, pp. 616–625, 1993.

[6] C. N. Potts, S. V. Sevast'janov, V. A. Strusevich, L. N. Van Wassenhove, and C. M. Zwaneveld, "The two-stage assembly scheduling problem: complexity and approximation," *Operations Research*, vol. 43, no. 2, pp. 346–355, 1995.

[7] A. M. A. Hariri and C. N. Potts, "A branch and bound algorithm for the two-stage assembly scheduling problem," *European Journal of Operational Research*, vol. 103, no. 3, pp. 547–556, 1997.

[8] X. Sun, K. Morizawa, and H. Nagasawa, "Powerful heuristics to minimize makespan in fixed, 3-machine, assembly-type flowshop scheduling," *European Journal of Operational Research*, vol. 146, no. 3, pp. 498–516, 2003.

[9] A. Allahverdi and F. S. Al-Anzi, "Evolutionary heuristics and an algorithm for the two-stage assembly scheduling problem to minimize makespan with setup times," *International Journal of Production Research*, vol. 44, no. 22, pp. 4713–4735, 2006.

[10] C. Koulamas and G. J. Kyparisis, "A note on the two-stage assembly flow shop scheduling problem with uniform parallel machines," *European Journal of Operational Research*, vol. 182, no. 2, pp. 945–951, 2007.

[11] P. Fattahi, S. M. H. Hosseini, and F. Jolai, "A mathematical model and extension algorithm for assembly flexible flow shop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, pp. 1–16, 2012.

[12] P. Fattahi, S. M. Hosseini, F. Jolai, and R. Tavakkoli-Moghaddam, "A branch and bound algorithm for hybrid flow shop scheduling problem with setup time and assembly operations," *Applied Mathematical Modelling: Simulation and Computation for Engineering and Environmental Systems*, vol. 38, no. 1, pp. 119–134, 2014.

[13] S. Hatami, R. Ruiz, and C. Andrés-Romano, "Heuristics and metaheuristics for the distributed assembly permutation flowshop scheduling problem with sequence dependent setup times," *International Journal of Production Economics*, vol. 169, pp. 76–88, 2015.

[14] G. M. Komaki, E. Teymourian, and V. Kayvanfar, "Minimising makespan in the two-stage assembly hybrid flow shop scheduling problem using artificial immune systems," *International Journal of Production Research*, vol. 54, no. 4, pp. 963–983, 2015.

[15] S. Jung, Y.-B. Woo, and B. S. Kim, "Two-stage assembly scheduling problem for processing products with dynamic component-sizes and a setup time," *Computers & Industrial Engineering*, vol. 104, pp. 98–113, 2017.

[16] A. Allahverdi and H. Aydilek, "The two stage assembly flowshop scheduling problem to minimize total tardiness," *Journal of Intelligent Manufacturing*, vol. 26, no. 2, pp. 225–237, 2015.

[17] A. Maleki-Darounkolaei, M. Modiri, R. Tavakkoli-Moghaddam, and I. Seyyedi, "A three-stage assembly flow shop scheduling problem with blocking and sequence-dependent set up times," *Journal of Industrial Engineering International*, vol. 8, no. 1, article no. 26, 2012.

[18] F. S. Al-Anzi and A. Allahverdi, "A hybrid tabu search heuristic for the two-stage assembly scheduling problem," in *Proceedings of the International Journal of Operations Research*, vol. 3, pp. 109–119, 2006.

[19] A. Allahverdi and F. S. Al-Anzi, "A PSO and a Tabu search heuristics for the assembly scheduling problem of the two-stage distributed database application," *Computers & Operations Research*, vol. 33, no. 4, pp. 1056–1080, 2006.

[20] F. S. Al-Anzi and A. Allahverdi, "A self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times," *European Journal of Operational Research*, vol. 182, no. 1, pp. 80–94, 2007.

[21] A. Allahverdi and F. S. Al-Anzi, "The two-stage assembly flowshop scheduling problem with bicriteria of makespan and mean completion time," *The International Journal of Advanced Manufacturing Technology*, vol. 37, no. 1-2, pp. 166–177, 2008.

[22] A. Tozkapan, O. Kirca, and C.-S. Chung, "A branch and bound algorithm to minimize the total weighted flowtime for the two-stage assembly scheduling problem," *Computers & Operations Research*, vol. 30, no. 2, pp. 309–320, 2003.

[23] C. S. Sung and H. A. Kim, "A two-stage multiple-machine assembly scheduling problem for minimizing sum of completion times," *International Journal of Production Economics*, vol. 113, no. 2, pp. 1038–1048, 2008.

[24] A. Allahverdi and F. S. Al-Anzi, "The two-stage assembly scheduling problem to minimize total completion time with setup times," *Computers & Operations Research*, vol. 36, no. 10, pp. 2740–2747, 2009.

[25] T. C. E. Cheng, B. M. T. Lin, and Y. Tian, "Scheduling of a two-stage differentiation flowshop to minimize weighted sum of machine completion times," *Computers & Operations Research*, vol. 36, no. 11, pp. 3031–3040, 2009.

[26] A. Mozdgir, S. M. T. Fatemi Ghomi, F. Jolai, and J. Navaei, "Two-stage assembly flow-shop scheduling problem with non-identical assembly machines considering setup times," *International Journal of Production Research*, vol. 51, no. 12, pp. 3625–3642, 2013.

[27] M. J. Anzanello and F. S. Fogliatto, "Learning curve models and applications: literature review and research directions," *International Journal of Industrial Ergonomics*, vol. 41, no. 5, pp. 573–583, 2011.

[28] T. P. Wright, "Factors affecting the cost of airplanes," *Journal of the Aeronautical Sciences*, vol. 3, no. 4, pp. 122–128, 1936.

[29] L. E. Yelle, "The learning curve: historical review and comprehensive survey," *Decision Sciences*, vol. 10, no. 2, pp. 302–328, 1979.

[30] 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.

[31] A. Azzouz, M. Ennigrou, and L. B. Ben, "Scheduling problems under learning effects: classification and cartography," in *Proceedings of the International Journal of Production Research*, pp. 10–1080, 2017.

[32] T. C. E. Cheng and G. Wang, "Single machine scheduling with learning effect considerations," *Annals of Operations Research*, vol. 98, pp. 273–290, 2000.

[33] V. R. Dondeti and B. B. Mohanty, "Impact of learning and fatigue factors on single machine scheduling with penalties for tardy jobs," *European Journal of Operational Research*, vol. 105, no. 3, pp. 509–524, 1998.

[34] T. C. E. Cheng, C.-C. Wu, and W.-C. Lee, "Some scheduling problems with sum-of-processing-times-based and job-position-based learning effects," *Information Sciences*, vol. 178, no. 11, pp. 2476–2487, 2008.

[35] Y. Yin, D. Xu, and J. Wang, "Single-machine scheduling with a general sum-of-actual-processing-times-based and job-position-based learning effect," *Applied Mathematical Modelling*, vol. 34, no. 11, pp. 3623–3630, 2010.

[36] R. a. Rudek, "The single processor total weighted completion time scheduling problem with the sum-of-processing-time based learning model," *Information Sciences*, vol. 199, pp. 216–229, 2012.

[37] C.-C. Wu, Y. Yin, W.-H. Wu, and S.-R. Cheng, "Some polynomial solvable single-machine scheduling problems with a truncation sum-of-processing-times based learning effect," *European Journal of Industrial Engineering*, vol. 6, no. 4, pp. 441–453, 2012.

[38] Y.-Y. Lu, J.-J. Wang, and X. Huang, "Scheduling jobs with position and sum-of-processing-time based processing times," *Applied Mathematical Modelling*, vol. 39, no. 14, pp. 4013–4021, 2015.

[39] Y.-B. Wu and J.-J. Wang, "Single-machine scheduling with truncated sum-of-processing-times-based learning effect including proportional delivery times," *Neural Computing and Applications*, vol. 27, no. 4, pp. 937–943, 2016.

[40] J. Pei, P. M. Pardalos, X. Liu, W. Fan, and S. Yang, "Serial batching scheduling of deteriorating jobs in a two-stage supply chain to minimize the makespan," *European Journal of Operational Research*, vol. 244, no. 1, pp. 13–25, 2015.

[41] G. J. Kyparisis and C. Koulamas, "Assembly-line scheduling with concurrent operations and parallel machines," *INFORMS Journal on Computing*, vol. 14, no. 1, pp. 68–80, 2002.

[42] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.

[43] S. French, *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*, Halstead Press, New York, NY, USA, 1982.

[44] D. Y. Li, H. J. Meng, and X. M. Shi, "Membership clouds and membership cloud generators," *Journal of Computer Research and Development*, vol. 32, pp. 15–20, 1995.

[45] D. Li and Y. Du, *Artificial Intelligence with Uncertainty*, Chapman & Hall, Boca Raton, Fla, USA, 2007.

[46] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 2033–2049, 2007.

[47] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[48] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information Sciences*, vol. 237, pp. 82–117, 2013.

[49] E. Torabzadeh and M. Zandieh, "Cloud theory-based simulated annealing approach for scheduling in the two-stage assembly flowshop," *Advances in Engineering Software*, vol. 41, no. 10-11, pp. 1238–1243, 2010.

[50] P. Lv, L. Yuan, and J. Zhang, "Cloud theory-based simulated annealing algorithm and application," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 4-5, pp. 742–749, 2009.

[51] S. M. Johnson, "Optimal two- and three-stage production schedules with setup times included," *Naval Research Logistics Quarterly*, vol. 1, no. 1, pp. 61–68, 1954.

[52] I. Ribas, R. Companys, and X. Tort-Martorell, "An iterated greedy algorithm for the flowshop scheduling problem with blocking," *Omega* , vol. 39, no. 3, pp. 293–301, 2011.

[53] Q.-K. Pan and R. Ruiz, "An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem," *Omega* , vol. 44, pp. 41–50, 2014.

[54] M. K. Msakni, W. Khallouli, M. Al-Salem, and T. Ladhari, "Minimizing the total completion time in a two-machine flowshop problem with time delays," *Engineering Optimization*, vol. 48, no. 7, pp. 1164–1181, 2016.

[55] J. Dubois-Lacoste, F. Pagnozzi, and T. Stützle, "An iterated greedy algorithm with optimization of partial solutions for the makespan permutation flowshop problem," *Computers & Operations Research*, vol. 81, pp. 160–166, 2017.

[56] Q.-K. Pan, R. Ruiz, and P. Alfaro-Fernández, "Iterated search methods for earliness and tardiness minimization in hybrid flowshops with due windows," *Computers & Operations Research*, vol. 80, pp. 50–60, 2017.

[57] M. Nawaz, E. E. Enscore Jr., and I. Ham, "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," *Omega* , vol. 11, no. 1, pp. 91–95, 1983.

[58] F. Della Croce, V. Narayan, and R. Tadei, "The two-machine total completion time flow shop problem," *European Journal of Operational Research*, vol. 90, no. 2, pp. 227–237, 1996.

[59] J.-Q. Li, Q.-K. Pan, and F.-T. Wang, "A hybrid variable neighborhood search for solving the hybrid flow shop scheduling problem," *Applied Soft Computing*, vol. 24, no. 1, pp. 63–77, 2014.

[60] Y. Wang, X. Li, and R. Ruiz, "An Exact Algorithm for the Shortest Path Problem with Position-Based Learning Effects," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 11, pp. 3037–3049, 2017.

[61] Y. Wang, X. Li, R. Ruiz, and S. Sui, "An iterated greedy heuristic for mixed no-wait flowshop problems. IEEE Transactions on cybernetics," *IEEE Transactions on cybernetics*, 2017.

[62] S.-C. Liu, W.-L. Hung, and C.-C. Wu, "Note on a single-machine scheduling problem with Sum of processing times based learning and ready times," in *Mathematical Problems in Engineering*, vol. 2015, p. 9, 2015.

[63] G. M. Komaki, E. Teymourian, V. Kayvanfar, and Z. Booyavi, "Improved discrete cuckoo optimization algorithm for the three-stage assembly flowshop scheduling problem," *Computers & Industrial Engineering*, vol. 105, pp. 158–173, 2017.

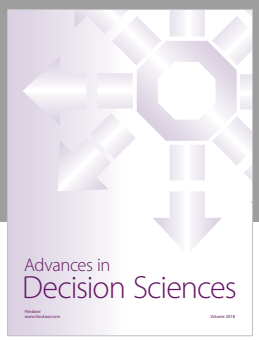
[64] D. E. Critchlow and M. A. Fligner, "On distribution-free multiple comparisons in the one-way analysis of variance," *Communications in Statistics—Theory and Methods*, vol. 20, no. 1, pp. 127–139, 1991.

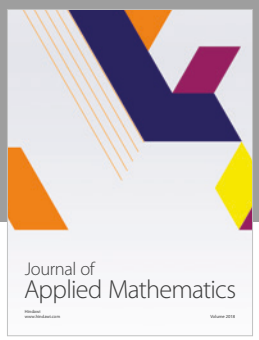
[65] M. D. Hollander, A. Wolfe, and E. Chicken, *Nonparametric Statistical Methods*, John Wiley & Sons, Inc., Hoboken, NJ, USA, 3rd edition, 2014.

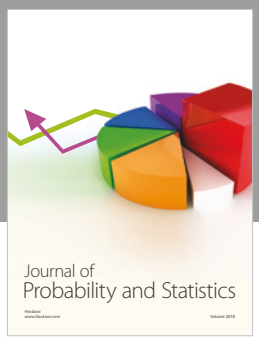
[66] R. Rudek, "Computational complexity and solution algorithms for flowshop scheduling problems with the learning effect," *Computers & Industrial Engineering*, vol. 61, no. 1, pp. 20–31, 2011.

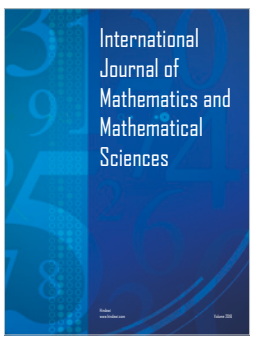
[67] R. a. Rudek, "The computational complexity analysis of the two-processor flowshop problems with position dependent job processing times," *Applied Mathematics and Computation*, vol. 221, pp. 819–832, 2013.

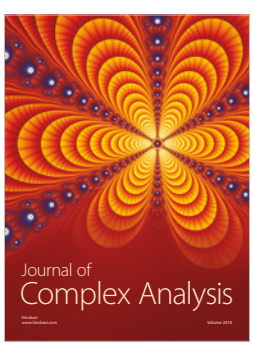
[68] R. a. Rudek, "Parallel machine scheduling with general sum of processing time based models," *Journal of Global Optimization*, vol. 68, no. 4, pp. 799–814, 2017.

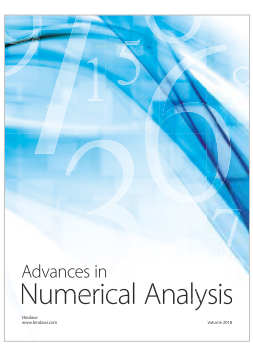


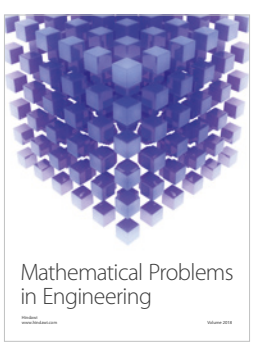


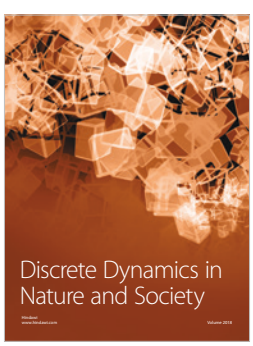


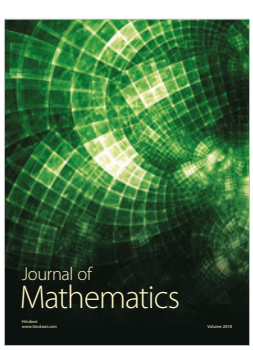


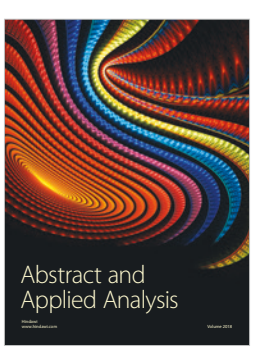


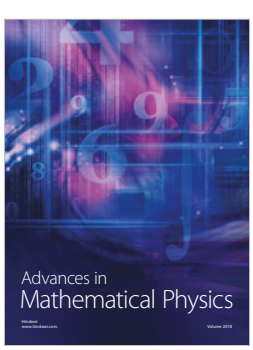












Hindawi

Submit your manuscripts at
www.hindawi.com